UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DATA BALANCING APPROACHES IN QUALITY, DEFECT, AND PATTERN ANALYSIS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

MD MANJURUL AHSAN
Norman, Oklahoma
2023

DATA BALANCING APPROACHES IN QUALITY, DEFECT, AND PATTERN ANALYSIS

A DISSERTATION APPROVED FOR THE
SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

BY THE COMMITTEE CONSISTING OF

Acknowledgements

I am deeply grateful to my advisors, Dr. Shivakumar Raman and Dr. Zahed Siddique, for their unwavering support and guidance throughout my doctoral journey. My sincerest thanks also go to my former supervisor, Dr. Pedro Huebner, who made a significant contribution to the early stages of this dissertation with his invaluable time and effort. I am also thankful for the constructive comments and guidance received from Dr. Theodore Trafalis and Dr. Charles D. Nicholson. I am grateful to Dr. Mohammed Atiquzzaman for his invaluable assistance in directing my academic path and ambitions. I am indebted to Dr. Farrokh Mistree and Dr. Janet Allen for their invaluable guidance, tips, and information that helped me to make new contributions to the scientific community. I also express my gratitude to Dr. Yongtao Liu for providing the data and guidelines required to conduct experiments on a portion of my research.

Finally, I would like to express my heartfelt thanks to my parents, siblings, lab mates, and friends from home and here for their unwavering support and encouragement throughout my Ph.D. journey. Without their support, my achievements would not have been possible.

Table of Contents

Abstract

The imbalanced ratio of data is one of the most significant challenges in various industrial domains. Consequently, numerous data-balancing approaches have been proposed over the years. However, most of these data-balancing methods come with their own limitations that can potentially impact data-driven decision-making models in critical sectors such as product quality assurance, manufacturing defect identification, and pattern recognition in healthcare diagnostics. This dissertation addresses three research questions related to data-balancing approaches: 1) What are the scopes of data-balancing approaches toward the major and minor samples? 2) What is the effect of traditional Machine Learning (ML) and Synthetic Minority Over-sampling Technique (SMOTE)-based data-balancing on imbalanced data analysis? and 3) How does imbalanced data affect the performance of Deep Learning (DL)-based models?

To achieve these objectives, this dissertation thoroughly analyzes existing reference works and identifies their limitations. It has been observed that most existing data-balancing approaches have several limitations, such as creating noise during oversampling, removing important information during undersampling, and being unable to perform well with multidimensional data. Furthermore, it has also been observed that SMOTE-based approaches have been the most widely used data-balancing approaches as they can create synthetic samples that are easy to implement compared to other existing techniques. However, SMOTE also has its limitations, and therefore, it is required to identify whether there is any significant effect of SMOTE-based oversampled approaches on ML-based data-driven models' performance. To do that, the study conducts several hypothesis tests considering several popular ML algorithms with and without hyperparameter settings. Based on the overall hypothesis, it is found that, in many cases based on the reference dataset, there is no significant performance improvement on data-driven ML models once the imbalanced data is balanced using SMOTE approaches.

Additionally, the study finds that SMOTE-based synthetic samples often do not follow the Gaussian distribution or do not follow the same distribution of the data as the original dataset. Therefore, the study suggests that Generative Adversarial Network (GAN)-based approaches could be a better alternative to develop more realistic samples and might overcome the limitations of SMOTE-based data-balancing approaches. However, GAN is often difficult to train, and very limited studies demonstrate the promising outcome of GAN-based tabular data balancing as GAN is mainly developed for image data generation. Additionally, GAN is hard to train as it is computationally not efficient. To overcome such limitations, the present study proposes several data-balancing approaches such as GAN-based oversampling (GBO), Support Vector Machine (SVM)-SMOTE-GAN (SSG), and Borderline-SMOTE-GAN (BSGAN). The proposed approaches outperform existing SMOTE-based data-balancing approaches in various highly imbalanced tabular datasets and can produce realistic samples. Additionally, the oversampled data follows the distribution of the original dataset.

The dissertation later examines two case scenarios where data-balancing approaches can play crucial roles, specifically in healthcare diagnostics and additive manufacturing. The study considers several Chest radiography (X-ray) and Computed Tomography (CT)-scan image datasets for the healthcare diagnostics scenario to detect patients with COVID-19 symptoms. The study employs six different Transfer Learning (TL) approaches, namely Visual Geometry Group (VGG)16, Residual Network (ResNet)50, ResNet101, Inception-ResNet Version 2 (InceptionResNetV2), Mobile Network version 2 (MobileNetV2), and VGG19. Based on the overall analysis, it has been observed that, except for the ResNet-based model, most of the TL models have been able to detect patients with COVID-19 symptoms with an accuracy of almost 99%. However, one potential drawback of TL approaches is that the models have been learning from the wrong regions. For example, instead of focusing on the infected lung regions, the TL-based models have been focusing on the non-infected regions. To address this issue, the study has updated the TL-based models to reduce the models' wrong localization.

Similarly, the study conducts an additional investigation on an imbalanced dataset containing defect and non-defect images of 3D-printed cylinders. The results show that TL-based models are unable to locate the defect regions, highlighting the challenge of detecting defects using imbalanced data. To address this limitation, the study proposes preprocessing-based approaches, including algorithms such as Region of Interest Net (ROIN), Region of Interest and Histogram Equalizer Net (ROIHEN), and Region of Interest with Histogram Equalization and Details Enhancer Net (ROIHEDEN) to improve the model's performance and accurately identify the defect region.

Furthermore, this dissertation employs various model interpretation techniques, such as Local Interpretable Model-Agnostic Explanations (LIME), SHapley Additive exPlanations (SHAP), and Gradient-weighted Class Activation Mapping (Grad-CAM), to gain insights into the features in numerical, categorical, and image data that characterize the models' predictions. These techniques are used across multiple experiments and significantly contribute to a better understanding the models' decision-making processes.

Lastly, the study considers a small mixed dataset containing numerical, categorical, and image data. Such diverse data types are often challenging for developing data-driven ML models. The study proposes a computationally efficient and simple ML model to address these data types by leveraging the Multilayer Perceptron and Convolutional Neural Network (MLP-CNN). The proposed MLP-CNN models demonstrate superior accuracy in identifying COVID-19 patients' patterns compared to existing methods.

In conclusion, this research proposes various approaches to tackle significant challenges associated with class imbalance problems, including the sensitivity of ML models to multidimensional imbalanced data, distribution issues arising from data expansion techniques, and the need for model explainability and interpretability. By addressing these issues, this study can potentially mitigate data balancing challenges across various industries, particularly those that involve quality, defect, and pattern analysis, such as healthcare diagnostics, additive manufacturing, and product quality. By providing valuable insights into the models' decision-making process, this research could pave the way for developing more accurate and robust

ML models, thereby improving their performance in real-world applications.

LIST OF ABBREVIATIONS

AdaBoost	Adaptive Boosting
Ada-CNN	Adaptive Condensed Nearest Neighbor
ADAM	Adaptive Momentum
AI	Artificial Intelligence
AM	Additive Manufacturing
ANN	Artificial Neural Network
API	Application Programming Interface
APPs	Anti-parasitic peptides
AUC	Area Under the Curve
AUPR	Area Under the Precision-Recall Curve
BB	Balanced Bagging
BC	Balance Cascade
BPNN	Back-Propagation Neural Network
BSMOTE	borderline SMOTE
CAD	Computer-Aided Design
CC	Cluster Centroids
CDC	Center for Disease Control and Prevention
Chest X-ray	Chest Radiography
CIP	Class Imbalance problem
CNN	Convolutional Neural Network

CNN	Condensed Nearest Neighbor
ConvNets	Convolutional Neural Networks
COVID-19	Coronavirus Disease 2019
CT	Computed Tomography
CUSBoost	Cost-Sensitive Boosting
CVDs	Cardiovascular Diseases
DL	Deep Learning
DNN	Deep Neural Networks
DT	Decision Tree
EE	Easy Ensemble
ENN	Edited Nearest Neighbors
ET	Extra Tree
EusBoost	Ensemble of Under-Sampling Boosting
FC	Fully Connected
GAN	Generative Adversarial Networks
GB0	GAN-based Oversampling
GPSC	Genetic Programming Symbolic Classifier
Grad-CAM	Gradient-weighted Class Activation Mapping
GRU	Gated Recurrent Unit
G-SMOTE	Geometric SMOTE
ICU	Intensive Care Units
IDS	Intrusion Detection System

IHT	Instance Hardness Threshold
InceptionResNetV2	Inception Residual Network Version 2
IoT	Internet of Things
KDF	kernel Density Function
KNN	k-Nearest Neighbor
LIME	Local Interpretable Model Agnostic Explanations
LR	Logistic Regression
LSTM	Long Short-Term Memory
MERS-CoV	Middle East Respiratory Syndrome
ML	Machine Learning
MLP-CNN	Multilayer Perceptron and Convolutional Neural Network
MLTL	Multi-Label Tomek Link
MNN	M Nearest Neighbors
MobileNetV2	Mobile Network Version 2
NB	Naive Bayes
NCL	Neighbor Cleaning Rule
NCR	Neighborhood Cleaning Rule
NM	Near-Miss
NN	Neural Network
NPNN	No-Propagation Neural Network
N-US	Neighbourhood based Under-Sampling
ODU	One-sided Dynamic Undersampling

OSS	One-Sided Selection
PE	Pre-eclampsia
PRAUC	Precision-Recall Area Under the Curve
PSO	particle Swarm Optimization
R2L	Remote to Local
ReLU	Rectified Linear Unit
ResNet	Residual Network
RF	Random Forest
RMSprop	Root Mean Square Propagation
RNN	Recurrent Neural Network
RT-PCR	Reverse Transcription Polymerase Chain Reaction
RUSBoost	Random Under-Sampling Boosting
SARS-CoV	Severe Acute Respiratory Syndrome
SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus 2
SD	Standard Deviation
SDP	Software Defect Prediction
SGD	Stochastic Gradient Descent
SHAP	Shapley Additive Explanations
SLFN	Single Hidden Layer Feed-Forward Neural Network
SLR	Systematic Literature Review
SMLS	Smart Materials and Intelligent Systems
SMOTE	Synthetic Minority Oversampling Technique

Chapter 1

Introduction & Dissertation Structure

1.1 Overview

The present doctoral dissertation focuses on developing data-balancing approaches in quality, defect, and pattern analysis. The primary objective is to identify the most widely used data-balancing approaches and their limitations. To achieve this, the study considered three types of data: numerical, categorical, and mixed data, which contains numerical, categorical, and image data. While excluding other data types, such as sequential data, speech data, and time series data, most of the datasets used in this study are from the University of California, Irvine (UCI) public repository, and one dataset was obtained from the Smart Materials and Intelligent Systems (SMLS) laboratory at the University of Oklahoma. Based on their characteristics, the datasets are categorized into quality, defect, and pattern analysis. These datasets are imbalanced and require data-balancing approaches to develop fair Machine Learning (ML) models. Among the data balancing techniques, the study chose to dig deeper into the Synthetic Minority Oversampling Technique (SMOTE)-based approach, which has shown promising results compared to other techniques, such as random oversampling and undersampling. The study involved four hypothesis conditions to evaluate the impact of traditional ML and SMOTE-based data-balancing on imbalanced data analysis. The study results demonstrated that some ML models had significant performance improvements while others did not.

Moreover, the study introduced novel oversampling approaches such as Generative Adversarial Network (GAN)-based oversampling and Support Vector Machine (SVM)-SMOTE-GAN, which generated more realistic samples aligning with the original data distributions. The dissertation also aimed to understand the impact of imbalanced data on Deep Learning (DL)-based models. The study evaluated two scenarios, healthcare diagnostics and additive manufacturing. In the healthcare diagnostics scenario, the study analyzed ML-based prediction models in Coronavirus Disease 2019 (COVID-19) diagnosis on balanced and imbalanced data, using explainable Artificial Intelligence (AI) techniques such as Local Interpretable Model Agnostic Explanations (LIME) and Gradient Weighted Class Activation Mapping (Grad-CAM). Similarly, the proposed approaches incorporated novel image processing techniques in Additive Manufacturing (AM) to identify defect regions. The proposed approach will have significant implications for various domains, such as healthcare, manufacturing, and transportation, where data-balancing approaches are required to develop robust ML-based decision models.

1.2 Data Driven Model

A data-driven model is a mathematical or statistical model that utilizes large amounts of data to identify patterns and relationships and make predictions or decisions (Ali et al., 2020). This approach has gained popularity in various fields, including finance, healthcare, and marketing, due to its ability to provide insights that may need to be apparent using traditional analytical methods (Y. Wang, Kung, & Byrd, 2018; J. Zhang, Xie, Li, Shen, & Xia, 2020). Data-driven models can be used for several tasks, such as predicting customer behavior, forecasting market trends, or diagnosing medical conditions (Bell & Mgbemena, 2018; Gabrielli, Wüthrich, Blume, & Sansavini, 2022; Ahsan, Gupta, et al., 2020). By leveraging big data and advanced analytics techniques, data-driven models can assist organizations in making more informed and accurate decisions, resulting in improved outcomes and increased efficiencies.

The system engineering V model for data-driven models comprises several major stages, such as data assessment, data normalization and cleaning, the structure of the ML inference engine, ML training and implementation, ML engine inference integration, test deployment, and statistical analysis and evaluation, as illustrated in Figure 1.1. This model provides a structured approach to developing and deploying data-driven models, starting from the initial assessment and preparation of the data to the final evaluation of the model's performance. Each stage in the model is crucial for ensuring the accuracy and reliability of the model's predictions and decisions. By following this model, organizations can ensure that their data-driven models are developed and deployed systematically and effectively, resulting in improved outcomes and increased efficiencies (Marina, Trasnea, & Grigorescu, 2018).

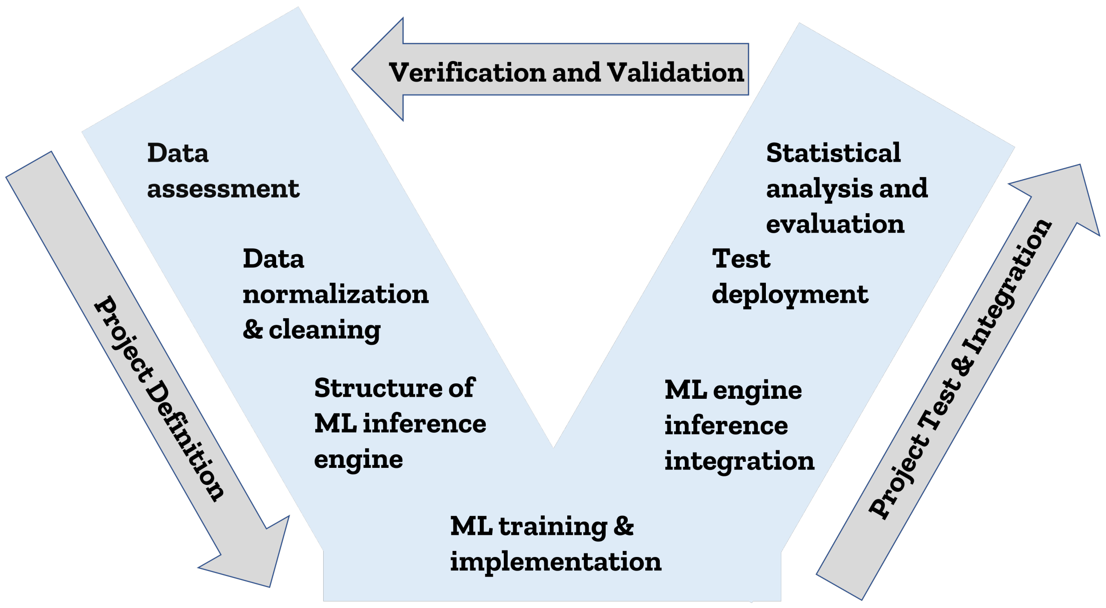

Verification and Validation
Data assessment
Data normalization & cleaning
Structure of ML inference engine
ML training & implementation
ML engine inference integration
Test deployment
Statistical analysis and evaluation
Project Definition
Project Test & Integration

Figure 1.1: System engineering V model for data driven model.

1.2.1 Data Assessment

Data assessment is an essential process that examines the quality, quantity, and completeness of the data collected for a study (Schmidt et al., 2021). This process is necessary to ensure that the data is reliable and valid and accurately represents the research question. Data assessment involves verifying that the data is complete, accurate, and consistent and checking for any errors or anomalies that may have been introduced during collection or storage (Karr, Sanil, & Banks, 2006). Additionally, data assessment includes evaluating the sample's representativeness and determining whether any biases or limitations may impact the study's outcomes. By conducting a thorough data assessment, researchers can identify and address any issues with the data, leading to more accurate and reliable results (Thabtah, Hammoud, Kamalov, & Gonsalves, 2020).

1.2.2 Data Normalization and Cleaning

Data normalization and cleaning are crucial steps in the data preparation process for any research study or analysis. Data normalization refers to transforming data to a standard scale or range to eliminate variations and make it comparable across different sources (Brownlee, 2020a). This process ensures that the data is consistent and accurate, making it easier to interpret and analyze. Cleaning involves identifying and correcting errors, inconsistencies, or missing data points. Ensuring that the data is complete, accurate, and reliable is essential. This process often involves identifying and removing outliers, filling in missing values, and standardizing data types (Rajkumar et al., 2022). These steps help ensure that the data is ready for analysis and can yield more accurate and reliable results. Researchers can minimize errors and inaccuracies by performing proper data normalization and cleaning, leading to better insights and conclusions (Brownlee, 2020a).

1.2.3 Structure of ML Inference Engine

The structure of a ML inference engine typically involves three main components: input layer, hidden layer(s), and output layer (Rubio-Solis, Panoutsos, Beltran-Perez, & Martinez-Hernandez, 2020). The input layer is where the raw data is fed into the system, and the hidden layer(s) is where the data is processed and transformed using mathematical functions and weights. The output layer presents the transformed data in a valid format, such as a prediction or classification. In addition to these three main components, many inference engines also include a bias term, an additional input that helps fine-tune the model's predictions. The structure of the inference engine can vary depending on the specific ML algorithm being used. However, in general, the input layer is responsible for receiving data, the hidden layer(s) are responsible for processing the data, and the output layer is responsible for presenting the results (Shim, Luo, Seo, & Yu, 2020). The inference engine's effectiveness is determined by its predictions' accuracy and precision, which can be evaluated using a variety of performance metrics such as precision, recall, and F1-score (Srivastava et al., 2021; Ahsan, E Alam, Trafalis, & Huebner, 2020).

1.2.4 ML Training and Implementation

ML training is the process of teaching an ML model to recognize patterns in data and make predictions based on those patterns. This involves feeding the model a large amount of data and using algorithms to identify patterns and relationships within that data. Once the model has been trained, it can be implemented to make predictions on new, unseen data (Yurochkin, Bower, & Sun, 2019; Brownlee, 2022).

Implementing an ML model involves deploying the trained model into a production environment, which can be used to make real-time predictions. This requires integrating the model with the application or system used, ensuring that the input data is appropriately formatted and processed, and setting up monitoring and evaluation mechanisms to ensure the model's accuracy and performance over time (Brownlee, 2022).

The implementation process also considers data security, privacy, and ethical concerns, mainly when dealing with sensitive data or making decisions that may significantly impact individuals or groups. Regular model maintenance and updating may also be necessary to ensure the model's continued accuracy and relevance (Watson, 2019).

1.2.5 ML Engine Inference Integration

ML engine inference integration is the process of integrating a trained ML model into a production system for real-time use. This involves deploying the model to an inference engine responsible for processing new data and making predictions based on the model's learned patterns. The inference engine is typically designed to optimize the model's performance for the specific use case, including latency, memory usage, and throughput factors (Jamil, Kahng, Kim, & Kim, 2021; Shafi, Rai, Sen, & Ananthanarayanan, 2021).

The integration process can vary depending on the specific use case and the requirements of the production system. In some cases, the inference engine may be hosted on the same system as the application, while in other cases, it may be deployed to a separate server or cloud platform. The integration process may also involve creating Application Programming Interface (API) or other interfaces to enable communication between the inference engine and the application (Wulf & Blohm, 2020).

Overall, successfully integrating an ML model into a production system is critical for ensuring that the model provides accurate and actionable real-time insights. Careful planning and testing are essential to ensure that the model operates as intended and that potential issues are identified and addressed promptly.

1.2.6 Test Deployment

ML test deployment is the process of deploying the trained ML model into a real-world production environment to evaluate its performance and ensure that it meets the desired requirements. This phase is critical in ensuring the model works as intended and is reliable. The ML test deployment phase includes several activities, such as creating test data sets, testing the model's accuracy and performance, evaluating its stability, and monitoring its behavior in the production environment (Shankar, Garcia, Hellerstein, & Parameswaran, 2022). The main goal of this phase is to identify and address any issues that may arise during model deployment and to ensure that the model performs optimally in the production environment. This phase also involves ensuring that the infrastructure and resources required to support the model are in place and that the model can be easily maintained and updated as needed (Paleyes, Urma, & Lawrence, 2022). Proper ML test deployment can significantly improve the chances of success for a ML project, leading to better outcomes and increased efficiency in real-world applications.

1.2.7 Statistical Analysis and Evaluation

ML Statistical analysis and evaluation is a critical step in the ML process, which involves assessing the performance of the trained model. This process involves various techniques and metrics, such as confusion matrix, precision, recall, accuracy, F1-score, and ROC curve. The confusion matrix is used to evaluate the performance of the classification model and provides information about true positive, true negative, false positive, and false negative predictions. Precision is a metric that measures the proportion of correctly predicted positive instances among all predicted positive instances. In contrast, recall measures the proportion of correctly predicted positive instances among all actual positive instances. In assessing the performance of a classification model, the accuracy metric provides a measure of its overall effectiveness, while the F1-score, being a harmonic mean of precision and recall, provides a more nuanced evaluation. The receiver operating characteristic (ROC) curve is a commonly used tool for assessing classification model performance considering the true positive rate against the false positive rate. By conducting statistical analysis and evaluation, ML models can be optimized and improved, leading to better performance and more accurate predictions (Ahsan, E Alam, et al., 2020; Jang, Kim, Harerimana, Kang, & Kim, 2020).

1.3 Study towards Data Balancing Approaches

ML has become a widely used technique in solving problems in various fields such as healthcare, finance, transportation, and more. ML algorithms require a large amount of data to learn and make accurate predictions. However, in many real-world scenarios, the data is imbalanced, meaning that one class has significantly more data points than the other(s) (Qian & Li, 2020; X. Wang, Liu, et al., 2020). For example, in fraud detection, the number of fraudulent transactions is often much lower than the number of legitimate ones. Similarly, in medical diagnosis, the number of patients with a rare disease is often much smaller than the number of healthy patients (X. Wang, Liu, et al., 2020).

Imbalanced data can lead to biased models, where the minority class is often misclassified (Tao et al., 2019). This can have serious consequences in real-world scenarios, where the cost of misclassifying the minority class is often much higher than the majority class. For example, misclassifying a fraudulent transaction as legitimate can have a severe financial impact, while misdiagnosing a rare disease can have life-threatening consequences.

1.4 Importance of Data Balancing

The importance of data-balancing approaches lies in their ability to address the class imbalance problem that often arises in ML applications. Imbalanced datasets have been a challenge in the field of ML, where the performance of the model is often biased towards the majority class due to its dominance in the dataset (Z. Chen, Duan, Kang, & Qiu, 2021). As a result, the minority class is usually ignored, leading to a high rate of false negatives and poor classification performance. This creates a need to study data-balancing approaches to improve the performance of the model and make accurate predictions for the minority class.

Despite the advancements in ML and the availability of different data-balancing approaches, the need to study data balancing is still relevant in the 21st century. This is because the imbalance problem is still challenging, even though new and improved algorithms and techniques are constantly being developed. The scientific contributions of this dissertation are based on the need to explore the effectiveness of different data-balancing approaches in the context of quality, defect, and pattern analysis.

The motivation behind conducting this research is to develop an efficient and reliable methodology for the effective analysis of imbalanced datasets and provide a useful reference for researchers and practitioners in the field. The importance of data-balancing approaches is not only limited to the academic world but also extends to various industrial sectors, including healthcare, finance, transportation, and manufacturing, where accurate and reliable predictions are crucial for decision-making.

By studying data-balancing approaches, this dissertation aims to contribute to the field of ML by identifying the best approach for different datasets and applications. The benefit of this research is that it will improve the performance of the model, reduce the rate of false negatives, and make accurate predictions for the minority class. This will have practical implications in areas such as defect, failure, and quality analysis, where accurate predictions are necessary to identify potential issues and prevent failures. Overall, this dissertation will contribute to developing effective and accurate predictive models for imbalanced datasets and improving the performance of ML algorithms.

1.5 Goal and Objectives

The current industrial landscape is characterized by increasingly complex mechatronics systems, uncertain and evolving environments, and a growing reliance on data-driven decision-making approaches (Brunton et al., 2021; Bibri, 2022). Over the years, data-driven decision-making systems have become popular for detecting faults or patterns that may impact industrial operations and for identifying their underlying causes (i.e., fault isolation). One of the major challenges facing these systems is the presence of imbalanced data or rare events. This can lead to poor decision-making and bias in the models. Despite some existing solutions, there remains a need for ongoing research to address this challenge effectively.

In this dissertation, the aim is to comprehensively examine the impact of imbalanced data on data-driven decision-making systems. To achieve this objective, the plan is to analyze the performance of these systems in the context of several industrial case studies. This research will contribute significantly to understanding the challenges that arise due to imbalanced data and aid in developing more effective solutions to address this issue in data-driven decision-making systems.

To do that, this research attempts to identify a successful solution for the following research question (RQ):

RQ1. What are the scopes of data-balancing approaches toward

the major and minor samples?

The purpose of RQ1 is to gain an understanding of the current state of the art and limitations of data-balancing approaches in handling imbalanced data. To achieve this, a thorough analysis of existing techniques and approaches to data balancing is conducted in the literature review section of the study. This review provides a comprehensive overview of the field's current state and highlights the challenges and limitations of existing approaches. In subsequent study sections, several proposed data balancing approaches are presented and evaluated in various industrial case scenarios. These proposed approaches are designed to address the limitations of current data balancing methods and offer more effective solutions for handling imbalanced data in industrial settings. By evaluating these proposed approaches, the study aims to gain insight into the scope and potential of data-balancing approaches for handling major and minor samples in imbalanced data analysis. Figure 1.2 illustrates an approach to address RQ1.

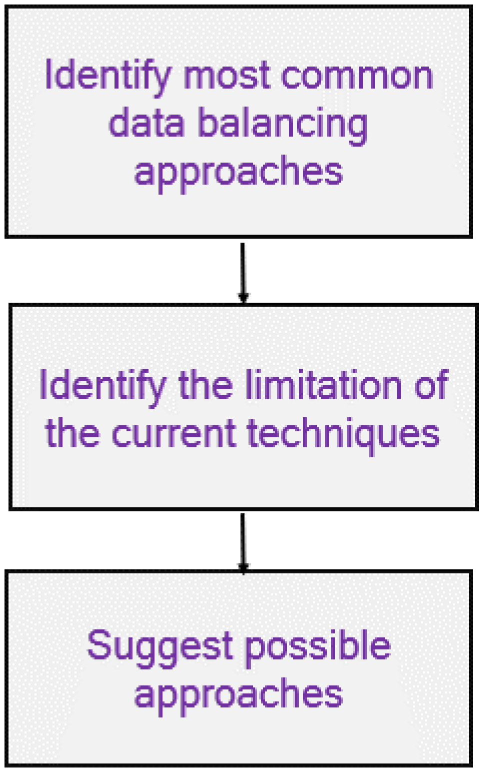

Identify most common data balancing approaches
Identify the limitation of the current techniques
Suggest possible approaches

Figure 1.2: Approach to address RQ1.

RQ2. What is the effect of traditional Machine learning and SMOTE based data balancing on imbalanced data analysis?

In my literature review, I discovered that SMOTE is widely used as an oversampling technique in the field of imbalanced data analysis. Despite its popularity, SMOTE and SMOTE-based approaches have been criticized for their potential to introduce multicollinearity issues. Despite this criticism, there is a lack of empirical studies evaluating the performance of SMOTE-based approaches when combined with various machine learning techniques on various imbalanced datasets.

During the literature review section, it was discovered that SMOTE is a widely used oversampling technique in the field of imbalanced data analysis. Despite their popularity, SMOTE and SMOTE-based approaches have been criticized for their potential to introduce multicollinearity issues. Therefore, there is a need for more empirical studies that evaluate the performance of SMOTE-based approaches when combined with various ML techniques on various imbalanced datasets.

In order to address this research gap and evaluate the effect of traditional ML and SMOTE-based data-balancing on imbalanced data analysis, a study was designed and conducted as part of RQ2. An experimental framework was devised to assess the performance of different ML-based approaches in handling imbalanced data, as illustrated in Figure 1.3. This research question aims to contribute to understanding the significant effect of SMOTE-based approaches incorporated with various ML models on handling imbalanced data.

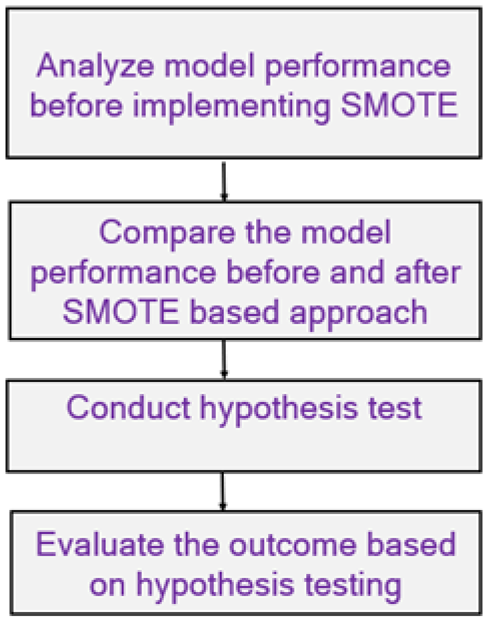

Analyze model performance before implementing SMOTE
Compare the model performance before and after SMOTE based approach
Conduct hypothesis test
Evaluate the outcome based on hypothesis testing

Figure 1.3: Approach to address RQ2.

RQ3. How does imbalanced data affect the performance of Deep Learning (DL)-based models?

The objective of this RQ3 is to evaluate the performance of DL approaches on imbalanced data analysis. Given the popularity of DL-based approaches in various domains, it is important to assess their potential limitations and biases. The significance of this research lies in understanding the limitations and potential of DL approaches in imbalanced data analysis, which is vital for improving the performance of these models. The results of this study will provide insights into the challenges of DL models on imbalanced datasets and suggest possible solutions. Figure 1.4 illustrates an approach to address the RQ3.

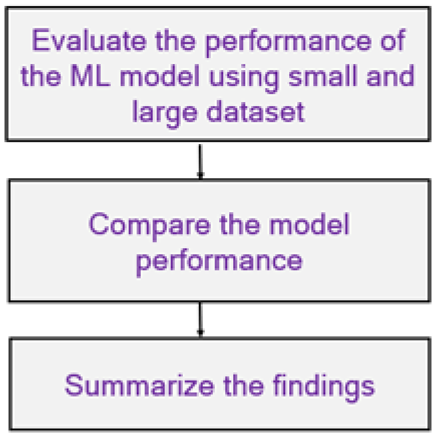

Evaluate the performance of the ML model using small and large dataset
Compare the model performance
Summarize the findings

Figure 1.4: Approach to address RQ3.

1.6 Outline of Dissertation

To enable a detailed discussion, a comprehensive overview of the dissertation's chapters is presented in Figure 1.5. The initial chapter of this dissertation provides a fundamental understanding of the research by presenting a general overview of the study's focus, significance, objectives, and research questions. This chapter lays the foundation for exploring the importance of data-balancing approaches and outlines the research's primary goal. Furthermore, this chapter presents various approaches that can be employed to address the research questions, and the subsequent chapter provides a literature review that examines different data-balancing approaches (refer to Chapter 2). The literature review aims to explore various data balancing techniques, such as oversampling, undersampling, and ensemble methods, and identify their strengths and limitations.

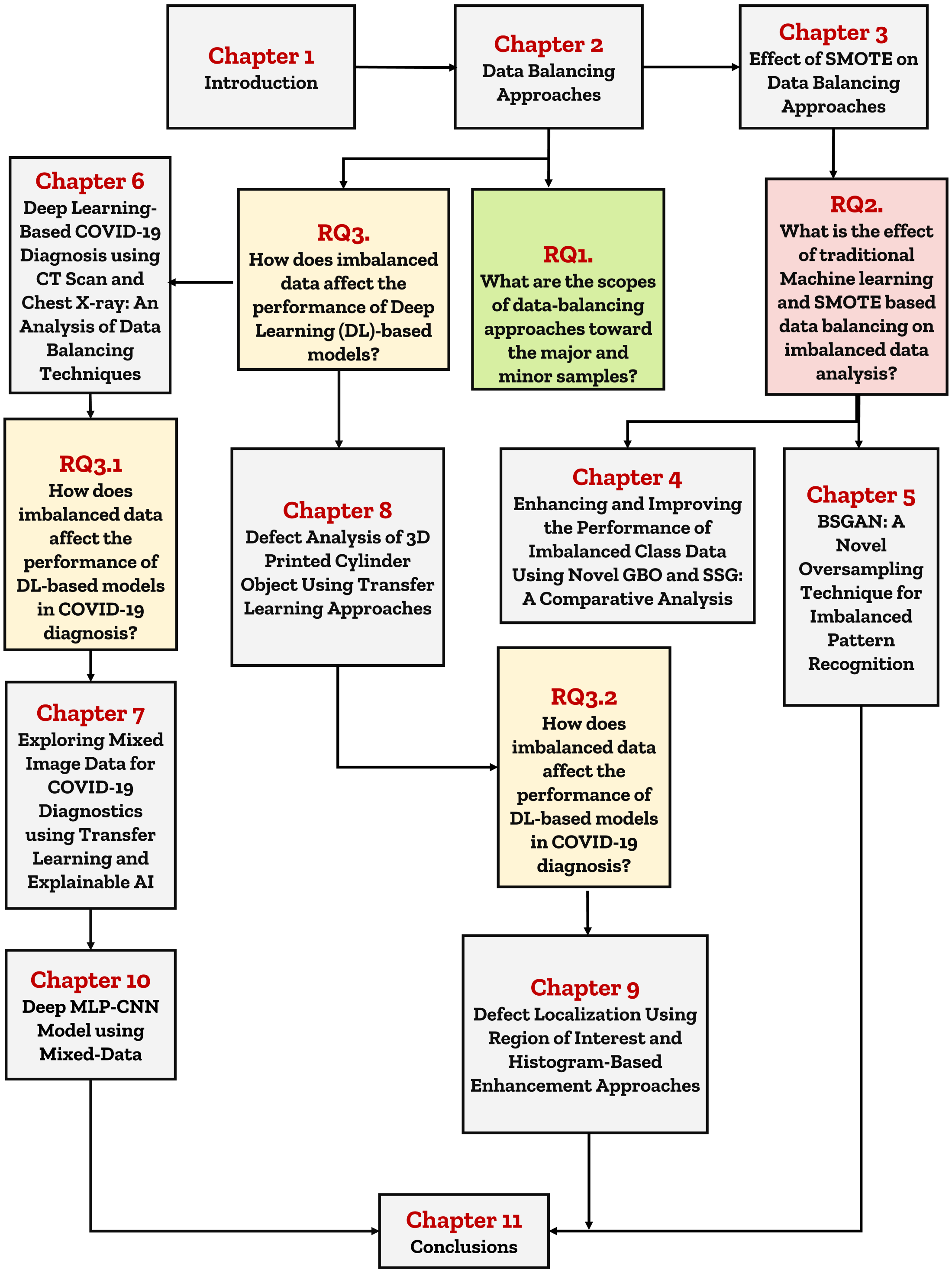

Chapter 1
Introduction
Chapter 2
Data Balancing Approaches
Chapter 3
Effect of SMOTE on Data Balancing Approaches
Chapter 6
Deep Learning-Based COVID-19 Diagnosis using CT Scan and Chest X-ray: An Analysis of Data Balancing Techniques
RQ3.
How does imbalanced data affect the performance of Deep Learning (DL)-based models?
RQ1.
What are the scopes of data-balancing approaches toward the major and minor samples?
RQ2.
What is the effect of traditional Machine learning and SMOTE based data balancing on imbalanced data analysis?
RQ3.1
How does imbalanced data affect the performance of DL-based models in COVID-19 diagnosis?
Chapter 8
Defect Analysis of 3D Printed Cylinder Object Using Transfer Learning Approaches
Chapter 4
Enhancing and Improving the Performance of Imbalanced Class Data Using Novel GBO and SSG: A Comparative Analysis
Chapter 5
BSGAN: A Novel Oversampling Technique for Imbalanced Pattern Recognition
RQ3.2
How does imbalanced data affect the performance of DL-based models in COVID-19 diagnosis?
Chapter 7
Exploring Mixed Image Data for COVID-19 Diagnostics using Transfer Learning and Explainable AI
Chapter 10
Deep MLP-CNN Model using Mixed-Data
Chapter 9
Defect Localization Using Region of Interest and Histogram-Based Enhancement Approaches
Chapter 11
Conclusions

Figure 1.5: Flow diagram of the each chapter of this dissertation.

In general, Chapter 2 aims to address RQ1 by providing a comprehensive literature review on data-balancing approaches, including their background, applications, limitations, and effectiveness in addressing Class Imbalance Problems (CIPs) in ML applications. The literature review includes a detailed explanation of the effects of undersampling, oversampling, hybrid approaches, SMOTE-based approaches, and GAN-based approaches, with an extensive discussion of overall review findings, limitations, and potential scope. The chapter endeavors to provide a clear understanding of the various data balancing approaches available and their effectiveness in dealing with class imbalance issues. Moreover, it seeks to contribute to the ongoing discussion on the benefits and drawbacks of these approaches and to offer insights into potential future research directions in this area. The aim is to provide an academically rigorous and in-depth analysis of state-of-the-art data-balancing techniques for ML applications.

Chapter 3 starts with an introduction discussing the motivation and providing a chapter outline, followed by an experimental setup section that details the datasets (Ionosphere, Pageblocks, Poker, Spambase, Wine Quality, and Yeast) and ML algorithms (AdaBoost, Decision Tree, Gradient Boosting, K-Nearest Neighbors, Logistic Regression, Random Forest, and SVM) employed in the study, along with the performance evaluation methodology. The computational results section presents the performance of each algorithm on the datasets, leading to a discussion of the results and any notable observations. A hypothesis testing section then compares the performance of the algorithms with and without SMOTE-based data-balancing, featuring separate discussions for each algorithm's performance and significant findings. The chapter concludes with an overview of the overall findings and implications, offering insights into the effectiveness of SMOTE-based data-balancing approaches in various imbalanced datasets.

Chapter 4 evaluates the effectiveness of oversampling techniques in highly imbalanced datasets, specifically focusing on the SMOTE and SVM-SMOTE algorithms. The study utilizes nine benchmark datasets to provide a comprehensive evaluation, employing statistical measures such as accuracy, precision, recall, and F1-score. The Kernel Density Function (KDF) is used to analyze oversampled data distribution, providing a better understanding of synthetic sample characteristics and their distribution within the dataset's feature space. Based on the analysis results, the study proposes two novel oversampling approaches, GAN-based Oversampling (GBO) and SVM-SMOTE-GAN (SSG), to address the limitations of SMOTE and improve oversampling techniques' performance in highly imbalanced datasets. The proposed oversampling techniques can potentially improve classification performance in highly imbalanced datasets and can be helpful in various real-world applications.

Chapter 5 aims to propose a more effective GAN model that can be trained on small or large datasets with limited iterations and to introduce a new oversampling method called BSGAN, which combines the advantages of Borderline-SMOTE and GAN to synthesize new samples along the borderline of classes with Borderline-SMOTE and generate realistic samples with GAN. The chapter then evaluates the performance of Borderline-SMOTE, GAN, and BSGAN on four highly imbalanced datasets: Ecoli, Yeast, Wine Quality, and Abalone, using metrics such as accuracy, precision, recall, and F1-score. Furthermore, the chapter compares the performance of the proposed BSGAN model with that of several reference works, demonstrating that BSGAN outperforms many existing GAN-based oversampling techniques and addresses sensitive data issues effectively.

Chapter 6 commences with an introduction and motivation for the research, followed by a chapter outline. The background section provides an overview of Computed Tomography (CT) scan-based and chest X-radiography (X-ray)-based screening methods in healthcare. The research methodology elaborates on DL algorithms, including Visual Geometry Group (VGG), Inception Residual Network Version 2 (InceptionResNetV2), Residual Network (ResNet), and Mobile Network Version 2 (MobileNetV2), as well as the utilization of pre-trained Convolutional Neural Networks (ConvNets). The results section is divided into three studies, each discussing the confusion matrix, model accuracy, and model loss of the respective DL algorithm. Furthermore, test results with confidence intervals are presented. A discussion section delves into the implications of the results, with a focus on feature selection. The chapter concludes by summarizing the overall findings and suggesting potential future works in the field of healthcare diagnostics and data-balancing approaches.

Chapter 7 begins with an introduction and motivation for the research, followed by a chapter outline. The background section delves into the current state of the art and the significance of CT scan-based screening in the medical field. The research methodology section presents the approach used in the study, including a detailed explanation of LIME as an explainable AI technique, which provides insights into the model's decision-making process. The results section reports on the experiments' findings, analyzing the proposed model's performance in the CT scan image classification context. A discussion section follows, where the implications of the results are explored, limitations are identified, and potential improvements are proposed. The study's overall findings are summarized, emphasizing the key contributions and implications of the research. The chapter concludes by drawing a conclusion and outlining future research directions, exploring the potential for further advancements in mixed image data analysis and the application of explainable AI techniques in medical imaging.

Chapter 8 presents an improved Deep Neural Network (DNN) model for defect analysis in the 3D printing process and Grad-CAM visualization. The chapter begins with an introduction and motivation for the research, followed by a chapter outline. The methodology section encompasses data collection and various DL algorithms. Additionally, the LIME technique is discussed. The results section is divided into two studies, each presenting the performance of the DL algorithms in defect analysis. The discussions section delves into the model predictions, interpreting their implications and providing insights into the research's findings. The chapter summarizes the contributions and implications of the proposed model and Grad-CAM visualization for defect analysis in the 3D printing process.

Chapter 9 explores defect localization using Region of Interest (ROI)-based Convolutional Neural Network (CNN) approaches. The chapter begins with an introduction and motivation for the research, followed by a chapter outline. The background section provides an overview of the current state of the art in defect localization and the importance of the region of interest-based techniques.

The research methodology details the proposed approaches, including the ROI, Histogram Equalization (HE), and Details Enhancer (DE), as well as methods for model interpretation. The results section is divided into two studies, each evaluating the performance of the proposed approaches for defect localization.

Additionally, the models' explainability is assessed. A discussion section delves into the implications of the results, overall findings, and potential improvements. The chapter concludes by summarizing the contributions of the research and suggesting future works in the field of defect localization using ROI-based CNN approaches.

Chapter 10 explores the development of a deep Multi-Layer Perceptron (MLP)-CNN model using mixed data for early COVID-19 diagnosis. The chapter covers motivation, background review, dataset, and methodology, including the proposed model, explanation, and experimental setup. Additionally, the chapter presents computational results, a discussion, and a conclusion. The study aims to offer a reliable alternative screening method for COVID-19 patients, contributing to reduced mortality rates by facilitating early diagnosis, improved treatment, and prevention of disease transmission amid the ongoing global public health crisis.

Finally, Chapter 11 provides a comprehensive summary of the research questions explored in the previous chapters and outlines the key contributions of the study. It presents an overall summary of the research questions investigated and summarizes the main findings, followed by a detailed summary of each research question and its corresponding findings. The scope and limitations of data-balancing techniques for addressing the class imbalance in major and minor samples are discussed. The impact of traditional ML algorithms and SMOTE-based data-balancing techniques on imbalanced data analysis and the effect of DL approaches, specifically CNN, on imbalanced data analysis is explored. The chapter concludes by outlining the study's key contributions, including identifying limitations in existing data-balancing approaches and proposing new techniques for addressing the class imbalance in imbalanced datasets.

Chapter 2

Data Balancing Approaches

2.1 Introduction

Data imbalance is one of the challenging issues in various fields, including health-care, finance, and telecommunication. This issue negatively affects the performance of the Machine Learning (ML) algorithm, resulting in inaccurate predictions and suboptimal decision-making (Ali et al., 2020; Bell & Mgbemena, 2018; Thabtah et al., 2020). Several data-balancing approaches have been introduced in the literature to address this issue, which aims to improve the classification performance of ML models by either oversampling the minority class or undersampling the majority class. The research question that this chapter will address is RQ1: " What are the scopes of data-balancing approaches toward the major and minor samples?" To answer this question, the chapter will consider the following approaches:

- First, a general overview of some of the most widely used data-balancing approaches will be presented.
- Second, the most relevant literature that employs several data-balancing approaches will be discussed.
- Finally, the chapter will discuss the general findings, limitations, and potential scope at the end of the chapter.

By addressing RQ1, this chapter aims to provide a comprehensive overview of data-balancing approaches and their potential findings. Additionally, the chapter highlights that data-balancing approaches can be applied to various real-world domains, such as disease diagnosis, fraud detection, and anomaly detection, to improve ML models' performance.

2.2 Motivation

The motivation behind this chapter is to address the challenges and limitations of various data-balancing approaches in different domains where imbalanced data is a significant problem. While researchers have proposed numerous data balancing approaches, each approach has been found to have its unique set of advantages and limitations, thereby warranting further research in this field. Despite numerous studies, a comprehensive summary of widely used data-balancing approaches still needs to be included in the literature.

The primary objective of this chapter is to provide a comprehensive overview of data-balancing approaches for major and minor samples, including the advantages, limitations, and potential applications of different techniques. By reviewing recent studies that employ data-balancing methods in various real-world problems, this chapter aims to provide insights into the efficacy of different data-balancing approaches. The information presented in this chapter will enable practitioners to select the most appropriate data-balancing approach for their specific use case, leading to more accurate predictions and better decision-making.

2.3 Chapter Outline

The chapter is organized as follows. First, a brief background of different data-balancing approaches is discussed in Section 2.4. Second, Section 2.5 outlines the systematic procedure employed to collect the literature for the study, providing a comprehensive summary of the process.. Third, a concise summary of the data-balancing approaches used in the referenced literature is provided in Section 2.6. Finally, in Section 2.7, the chapter presents the overall findings, study outcomes, and future scope.

2.4 Background

Data balancing is a process of adjusting the class distribution of imbalanced datasets. In an imbalanced dataset, the number of instances of one class is significantly higher than the other. This issue is common in real-world datasets, such as fraud detection or medical diagnosis, where the minority class represents the target class or the class of interest. The data imbalance leads to the biased performance of ML models towards the majority class, resulting in poor accuracy, precision, and recall for the minority class (Cui, Jia, Lin, Song, & Belongie, 2019; Y. Yang & Xu, 2020; S. Wang, Dai, Shen, & Xuan, 2021).

Several data-balancing approaches have been proposed in the literature to address the imbalanced dataset problem. Common data-balancing approaches include undersampling, oversampling, and hybrid methods (Galar, Fernandez, Barrenechea, Bustince, & Herrera, 2011; Caruana et al., 2015; Le, Hoang Son, Vo, Lee, & Baik, 2018; Laurikkala, 2001; Guyon & Elisseeff, 2003; Z. Zhang, Krawczyk, Garcia, Rosales-Pérez, & Herrera, 2016).

2.4.1 Undersampling

Undersampling involves reducing the number of instances of the majority class to balance the class distribution with the minority class. Random undersampling and TomekLinks are some of the widely used undersampling approaches (D. Devi, Biswas, & Purkayastha, 2020; A. Mukherjee, Mukhopadhyay, Panigrahi, & Goswami, 2019; H. Sharma & Gosain, 2023). Details of some of the popular undersampling techniques are discussed below:

- **Cluster Centroids** is an undersampling method that aims to balance imbalanced datasets by reducing the number of instances in the majority class using centroid-based clustering. In this method, the majority class is clustered, and the centroid of each cluster is calculated. Then, the majority of class instances that are farthest from the centroid are removed. This approach helps to reduce the redundancy in the majority class and preserve the information of the minority class. The mathematical formulation of the Cluster Centroids algorithm can be written as (Le et al., 2018):

  Let X be the set of feature vectors, and y be the set of corresponding labels. Let C be the number of centroids to be selected. The Cluster Centroids algorithm can be formulated as follows:

1. Compute the Euclidean distance between all the instances in the majority class.
2. Cluster the majority class using KMeans algorithm with C clusters.
3. Select the C centroids from the majority class.
4. Remove the samples that are not centroids from the majority class.
5. Combine the undersampled majority class with the minority class to obtain a balanced dataset.

- **Random Undersampling** is a simple yet widely used undersampling method introduced by Laurikkala in 2001 (Laurikkala, 2001). The idea behind this method is to randomly remove instances from the majority class until the desired class balance is achieved. The drawback of this method is that it may lead to the loss of important information and decrease the model's overall performance. The mathematical formulation of random undersampling can be represented as follows (Brownlee, 2020d):

  Let S be the original set of instances, where S_{maj} and S_{min} represent the majority and minority class instances, respectively. Let n_{maj} and n_{min} be the number of instances in the majority and minority class, respectively. The goal of random undersampling is to obtain a new set S' such that the number of instances in S_{maj} is reduced to a desired level n'_{maj}. This can be achieved by randomly selecting n'_{maj} instances from S_{maj} without replacement and combining them with S_{min} to form the new set S'. The equation for random undersampling can be written as follows:

$$S' = S_{min} \cup S^{'}_{maj}, \tag{2.1}$$

where $S^{'}_{maj}$ is a randomly selected subset of S_{maj} such that $|S^{'}_{maj}| = n'_{maj}$.

- **Tomek Links** is a well-known undersampling technique that removes the overlapping samples between the minority and majority classes. This method is based on the idea of identifying pairs of samples that are nearest neighbors but belong to different classes and removing the majority sample from the pair. This technique can be implemented using a simple equation, where d(x_i, x_j) is the Euclidean distance between samples x_i and x_j, y_i is the class label of sample x_i, and n is the number of samples in the dataset (Guyon & Elisseeff, 2003):

$$||\mathbf{x}_i - \mathbf{x}_j||^2 \leq ||\mathbf{x}_i - \mathbf{x}_k||^2 \tag{2.2}$$

where $\mathbf{x}_i$ is an instance from the majority class, $\mathbf{x}_j$ is an instance from the minority class, and $\mathbf{x}_k$ is the nearest neighbor of $\mathbf{x}_j$ from the majority class. The inequality states that if $\mathbf{x}_j$ is closer to $\mathbf{x}_i$ than to its nearest neighbor in the majority class, then $\mathbf{x}_j$ is a Tomek link and can be removed to improve the class separation.

- **Neighborhood Cleaning Rule** (NCR) is an undersampling technique that removes noisy and borderline instances from the majority class based on the distribution of the minority class. This technique removes the majority class samples that are far from the minority class samples in the feature space.

The algorithm first determines the K nearest neighbors of each minority class sample and removes the majority class samples that are K furthest from the minority samples. The value of K is usually set to 3, 5, or 7 (Faris, 2014; Brownlee, 2020d).

Let X be the feature matrix of the dataset, y be the corresponding class labels, and C be the majority class. The NCR algorithm can be represented as:

1. Identify the minority class samples and their K nearest neighbors in the feature space.
2. Compute the average distance between each minority class sample and its K nearest neighbors.
3. For each majority class sample, compute the minimum distance to the minority class samples.
4. Remove the majority class samples that have a minimum distance larger than the average distance of their corresponding minority class samples.

The NCR algorithm can effectively remove the noisy and borderline instances from the majority class, making it a useful technique for handling imbalanced datasets (Hoens & Chawla, 2013).

Apart from this, several other undersampling approaches have been proposed by researchers, including but not limited to Edited Nearest Neighbors (ENN), One-Sided Selection (OSS), and Instance Hardness Threshold (IHT) (Triguero et al., 2019; Prudêncio, 2020; Jia & Zuo, 2017). However, undersampling approaches are often not recommended as they can significantly affect the model performance by removing important information from the majority class. In addition, undersampling can lead to overfitting of the remaining data, reducing the model's generalization ability. Oversampling approaches such as Synthetic Minority Oversampling Technique (SMOTE) and its extensions are more commonly preferred as they generate synthetic examples that retain important features of the minority class while avoiding the loss of information from the majority class. These methods have been shown to improve the performance of ML models on imbalanced datasets (Brownlee, 2020c).

2.4.2 Oversampling

Oversampling is another approach to balance the class distribution, where the minority class is artificially increased by generating synthetic instances or replicating the existing ones. Random oversampling, SMOTE, and Adaptive Synthetic Sampling (ADASYN) are some popular methods. Details of some of the popular oversampling techniques are discussed below (Brownlee, 2020c; H. He, Bai, Garcia, & Li, 2008; Chawla, Bowyer, Hall, & Kegelmeyer, 2002):

- **Random Oversampling** is a technique that involves randomly replicating the minority class samples until the number of samples in the minority class is equal to that of the majority class. This approach is straightforward and can be easily implemented. However, it may result in overfitting and decreased generalization performance if the synthetic samples are too similar to the original minority class samples. The probability of each minority sample being selected for replication is usually uniform but can be adjusted based on the degree of imbalance in the dataset. The following equation can represent the random oversampling approach (Brownlee, 2020c, 2020b; Calo, Efendiev, Galvis, & Li, 2016):

Let X = x_1, x_2, $\ldots$, x_n be the set of minority class samples and N be the desired number of minority class samples. The random oversampling approach can be formulated as:

1. Calculate the number of minority class samples, M.
2. Calculate the oversampling rate, r, as r = N/M.
3. For each minority class sample xi, randomly select k-1 nearest neighbors from the minority class, where k is a user-defined parameter.
4. For each xi and its k-1 neighbors, generate r-1 synthetic samples by randomly interpolating between them.
5. Add the synthetic samples to the minority class to obtain a balanced dataset.

Overall, random oversampling is a simple and effective technique for balancing imbalanced datasets, but care should be taken to avoid overfitting and to ensure that the synthetic samples are representative of the minority class distribution.

- **Synthetic Minority Over-sampling Technique** (SMOTE) is one of the most commonly used oversampling methods. It was initially proposed by Chawla et al. (2002) to tackle the problem of class imbalance by generating synthetic instances of the minority class. The fundamental concept of SMOTE involves creating new minority class instances by interpolating between existing minority class instances. To generate the new instances, two minority class instances are selected and a new instance is created along the line connecting them. The oversampling rate can be controlled by specifying the number of new instances to be produced. The SMOTE equation can be written as: (Chawla et al., 2002):

$$X_{new} = X_i + \lambda(X_j - X_i) \tag{2.3}$$

where X_i is an instance of the minority class, X_j is one of its k nearest neighbors, and λ is a random number between 0 and 1. More details of SMOTE will be discussed in Chapter 5, Section 5.5.

- **K-means SMOTE** is an extension of the SMOTE algorithm that utilizes the k-means clustering algorithm to identify the synthetic samples to be generated. The algorithm involves the following steps (Douzas, Bacao, & Last, 2018):

  1. Cluster the minority class instances into k clusters using the k-means clustering algorithm.

  2. Calculate the centroid and the distance to its k nearest neighbors for

each cluster. For each minority class instance, randomly select a cluster to generate synthetic samples.

3. Generate synthetic samples by interpolating between the instance and its k nearest neighbors within the selected

$$Synthetic_i = x_i + rand(0,1) * (x_{nearest} - x_i) \tag{2.4}$$

where $Synthetic_i$ is the newly generated synthetic instance, x_i is the minority instance being oversampled, $x_{nearest}$ is the nearest minority instance to x_i within the selected cluster, and $rand(0,1)$ is a random number between 0 and 1.

K-means SMOTE has been shown to outperform the original SMOTE algorithm in certain scenarios, particularly when dealing with datasets with high dimensionality and a large number of minority class instances. However, it may be computationally expensive due to using the k-means clustering algorithm (Last, Douzas, & Bacao, n.d.; Velmurugan & Santhanam, 2010).

- **ADASYN** was proposed by He et al. (2008) to address the problem of generating noisy instances in SMOTE. ADASYN generates more synthetic instances near the decision boundary, where the classification is difficult. The idea is to add synthetic instances to the minority class close to the decision boundary between the minority and majority classes. The oversampling rate can be adjusted by specifying the number of synthetic instances to be generated. The ADASYN equation is given by (H. He et al., 2008):

$$X_{new} = X_i + \lambda_i(X_i - X_{zi}) \tag{2.5}$$

where X_i is an instance of the minority class, X_{zi} is a randomly selected instance from the minority class that is close to X_i, λ_i is a random number between 0 and 1, and X_{new} is the new synthetic instance.

In addition to SMOTE-based oversampling methods, several other oversampling approaches have been proposed in the literature, such as SMOTEN (Synthetic Minority Over-sampling Technique using Extrapolation of a Nominal variable) and SMOTENC (Synthetic Minority Over-sampling Technique for Nominal and Continuous Features). Various studies have introduced these extensions of SMOTE-based approaches, and a detailed description of their practical implementation can be found in the Scikit-learn imbalanced-learn (imblearn) API documentation (Lemaître, Nogueira, & Aridas, 2017).

2.4.3 Hybrid Methods

Hybrid methods, which are a combination of over- and under-sampling techniques, have been proposed to overcome the limitations of individual under and over-sampling approaches. These methods combine the strengths of both over- and under-sampling to achieve better performance in imbalanced datasets (Gazzah, Hechkel, & Amara, 2015). For instance, in the SMOTEBoost algorithm, a combination of SMOTE and boosting is used to increase the number of minority class instances while reducing the number of majority class instances. The SMOTE-Boost algorithm works by creating synthetic minority class instances using the SMOTE algorithm and then using boosting to reduce the number of majority class instances (Chawla, Lazarevic, Hall, & Bowyer, 2003). The hybrid approach can help to overcome the limitations of using only one approach and can lead to more accurate classification results (Cui, Ma, & Saha, 2014).

2.4.4 Ensemble Methods

Ensemble methods are another technique used to address the class imbalance problem in machine learning (Z. Zhang et al., 2016). Ensemble methods combine multiple base classifiers, which can be either under-sampled or over-sampled, to improve the model's performance. One such ensemble method is the Easy Ensemble (EE) algorithm, which is a type of bagging method that creates multiple training sets by under-sampling the majority class and then trains several classifiers on each set. The final prediction is made by combining the outputs of all the classifiers. Another ensemble method is the Balance Cascade (BC) algorithm, which is an iterative algorithm that trains multiple classifiers on under-sampled data and then removes the majority of class samples that are misclassified by each classifier (X.-Y. Liu, Wu, & Zhou, 2008). This process is repeated until the desired level of balance is achieved. Ensemble methods effectively improve the classification performance on imbalanced datasets, particularly when combined with appropriate base classifiers and sampling techniques(Z. Zhang et al., 2016).

2.5 Systematic Analysis

Several oversampling techniques have been proposed to handle class imbalanced problems (Xin et al., 2021; Moreo, Esuli, & Sebastiani, 2016; Wibowo & Fatichah, 2021; Douzas & Bacao, 2019; Ishaq et al., 2021; Gök & Olgun, 2021; M. Mukherjee

& Khushi, 2021). This research undertakes a systematic analysis to identify existing techniques, their limitations, and prospects. The Systematic Literature Review (SLR) is employed as a review method whereby explicit and systematic procedures are used to identify, select, and critically evaluate relevant research to gather and evaluate data from the included studies (Okoli & Schabram, 2010). This approach is favored because it provides an accurate and reliable way to synthesize academic literature and is widely accepted in many research domains. The PRISMA recommendations are followed to report the SLR in this study, as they provide a standardized framework for reporting and enhancing the quality of systematic reviews and meta-analyses (Tricco et al., 2018). Ethics approval is not required for this study. While PRISMA is not a quality assessment tool, its 27-item checklist and four-phase analysis are evidence-based and offer clarity and transparency in reporting the SLR.

2.5.1 Identification of the Data

The Scopus and Web of Science (WOS) integrated database, which includes major publishers such as Emerald, Taylor and Francis, Springer, IEEE, and Wiley, is being used to conduct a comprehensive search study. Scopus is considered a reliable database for conducting Systematic Literature Reviews (SLRs) due to its high-quality indexing contents, as noted by many researchers (Fahimnia, Sarkis, & Davarzani, 2015; Malviya & Kant, 2015). The search period extends from 2018 through May 15, 2022, and encompasses all relevant publications published within this timeframe. To maintain focus and relevance, the initial search is conducted using the Boolean operator with "imbalance*" and "machine" AND "oversampl*," which results in 1229 articles from Scopus and 643 articles from WOS. After restricting the articles to journal papers, peer-reviewed articles, and those written in English, the remaining articles are reduced to 291 after title, keyword, and abstract analysis.

After applying the inclusion-exclusion criteria specified in Figure 2.1, a total of 151 papers for qualitative synthesis have been identified. To provide additional clarity on the article selection procedure, Table 2.1 presents a detailed description of the inclusion and exclusion criteria.

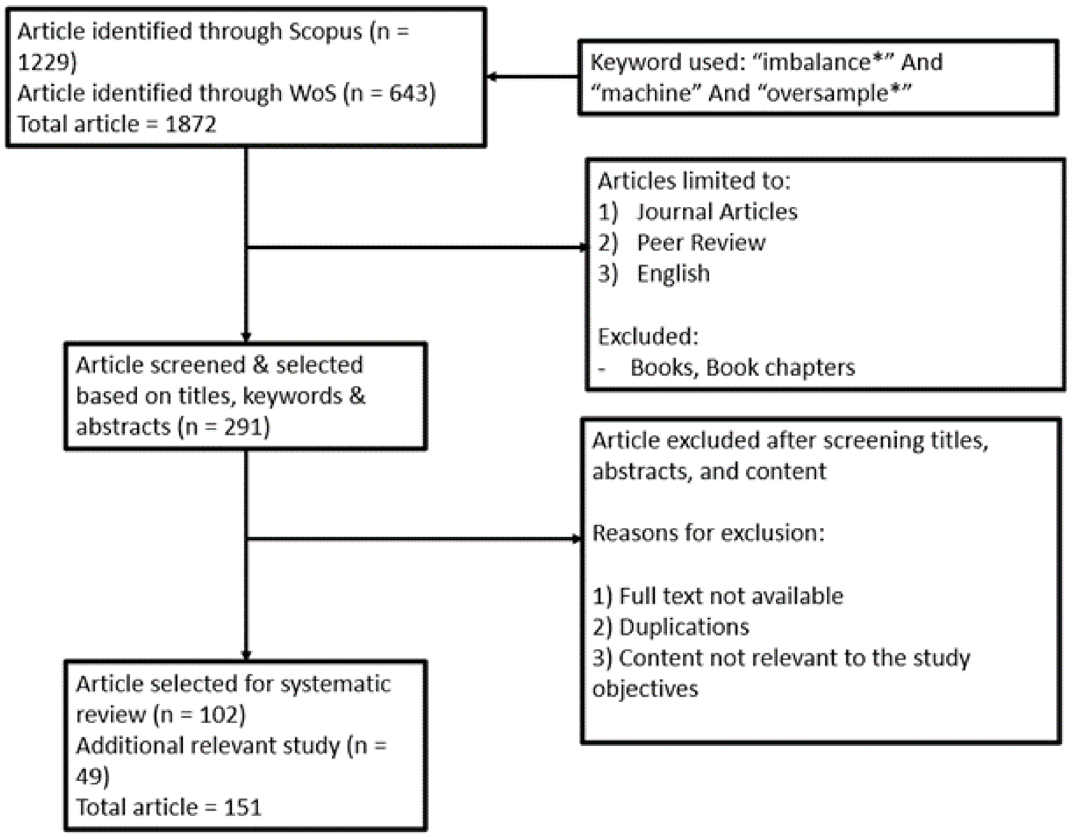

Article identified through Scopus (n = 1229)
Article identified through WoS (n = 643)
Total article = 1872
Keyword used: "imbalance*" And "machine" And "oversample*"
Articles limited to:
1) Journal Articles
2) Peer Review
3) English
Excluded:
- Books, Book chapters
Article screened & selected based on titles, keywords & abstracts (n = 291)
Article excluded after screening titles, abstracts, and content
Reasons for exclusion:
1) Full text not available
2) Duplications
3) Content not relevant to the study objectives
Article selected for systematic review (n = 102)
Additional relevant study (n = 49)
Total article = 151

Figure 2.1: The PRISMA 2015 flow diagram is utilized in this study to illustrate the article selection procedure (Moher et al., 2015).

Table 2.1: Inclusion-exclusion-based article selection procedure used during this study.

Screening Type	Criteria	Inclusion	Exclusion
Title screening	Is the title inclusive of keywords like "imbalance," "oversample," "oversampling," "Machine Learning," or "Deep Learning"?	291	1581
Abstract screening	Is the abstract mainly focused on the utilization of oversampling or ML algorithms to handle class imbalance?	190	101
Full text screening	Is the full text of the article accessible for analysis?	102	39
Additional screening	Is the article relevant to this study?	49	
Total article for final review		151	

Several types of research have been conducted over the years to deal with class imbalanced problems (CIP). After carefully screening, around 151 articles have been selected for Literature review. As CIP can significantly affect the ML models prediction, many sectors such as credit card fraud detection, fault analysis, traffic signal analysis, and rare cancer diagnosis are quite vulnerable to CIP. Most of the suggested methods are unable to provide a proper solution as most of the ML model's performance depends on various factors such as data size, algorithm types, data type, and imbalanced ratio (W.-J. Lin & Chen, 2013; S. Wang & Yao, 2013). This dissertation aims to investigate the techniques and approaches being used to handle CIPs and their current limitations.

2.6 Literature Review

2.6.1 Data Distribution

Data distribution on an imbalanced data set affects the ML model's performance during the prediction. If the data is skewed towards one class, then for the ML model, it is hard to perform the prediction without being biased towards the major class. To deal with those issues, several studies suggest data preprocessing steps before applying oversampling methods. For instance, Blagus and Lusa (2013), Gupta, Ahsan, Andrei, and Alam (2017), and Uyun and Sulistyowati (2020) have utilized data normalization techniques to improve the data distribution (Blagus & Lusa, 2013; Gupta, Ahsan, Andrei, & Alam, 2017; Uyun & Sulistyowati, 2020).

Pandey et al. (2019) used data normalization techniques to develop automatic arrhythmia detection. The author claimed that around 98.30% accuracy was achieved with data normalization and when the dataset is split as follows: train/test = 80/20 (Pandey & Janghel, 2019). Mohammed et al. (2020) used several data normalization techniques, such as min-max, Z-score, and L2 norm, along with SMOTE-based approaches to tackle the CIPs. However, it is not clear whether using such data normalization techniques has a significant effect on the model's overall performance or not (A. J. Mohammed, Hassan, & Kadir, 2020). Additionally, using several data normalization techniques before oversampling approach might change the entire minor class data distributions, which may not be helpful in predicting outlier or minor samples

Tesfahun and Bhaskari (2013) develop an intrusion detection model using Random Forest (RF) algorithms. The authors initially selected important features and then applied data Standardization methods and achieved more than 90% accuracy (Tesfahun & Bhaskari, 2013). However, none of the referenced literature demonstrates how their proposed method improves the data distribution after applying the oversampling techniques. Both of the studies use SMOTE-based approaches, which potentially create more noise in minor classes. Therefore, it can be assumed that the data distribution after the preprocessing and oversampling approach did not analyze properly and needed much attention.

2.6.2 Performance of ML Algorithms

The performance of ML algorithms on oversampled data set is not consistent. For instance, Rajesh and Dhuli (2018) used the Adaptive Boosting (AdaBoost) algorithm and achieved around 98.6% accuracy for detecting heart disease while trained and tested on the UCI heart disease dataset (Rajesh & Dhuli, 2018). Mahesh et al. (2022) achieved an accuracy of 95.47% using a similar dataset (Mahesh et al., 2022). Sisodia et al. (2017) used SMOTE oversampling techniques on a credit card dataset and achieved 95.33% accuracy with the Support Vector Machine (SVM) algorithm, while Mqadi et al. (2021) Achieved around 100% accuracy using the same algorithm and oversampling techniques on the same dataset (Sisodia, Reddy, & Bhandari, 2017; N. Mqadi, Naicker, & Adeliyi, 2021).

Sadgali et al. (2019) proposed a resampling approach to detect credit card fraud and achieved around 78.9% accuracy using Decision Tree (DT) (Sadgali, Nawal, & Benabbou, 2019). Ileberi et al. (2021) evaluated the performance of different ML models for credit card fraud detection. The author reported around 99% accuracy in detecting fraud using Decision Tree (DT) algorithms (Ileberi, Sun, & Wang, 2021). Sadgali et al. (2019) and Ileberi et al. (2021) both use SMOTE oversampling methods but reported different accuracy with DT approaches (Sadgali et al., 2019; Ileberi et al., 2021).

Therefore, one of the major concerns is that the performance of different ML algorithms varies with the similar SMOTE approaches. However, since none of the studies mentioned whether they have tuned the parameter of SMOTE or used the default parameters, it is difficult to conclude that the performance of ML algorithms differs solely based on oversampling approaches; indeed, a proper clarification is needed.

2.6.3 Effect of Undersampling

Undersampling has been widely used as a data-balancing approach to handle imbalanced datasets. The methods aim to balance the data distribution by removing the majority class instances to match the size of the minority class (Zeng, Zou, Wei, Liu, & Wang, 2016; W.-C. Lin, Tsai, Hu, & Jhang, 2017; Hira & Gillies, 2015). This approach has been shown to improve the classification performance of ML algorithms on imbalanced datasets. Several undersampling techniques have been proposed and used in the literature. One popular technique is Random Undersampling, which randomly selects instances from the majority class to balance the class distribution (Zuech, Hancock, & Khoshgoftaar, 2021). Another technique is the Near-Miss (NM) algorithm, which selects instances from the majority class based on their distance from instances from the minority class (N. M. Mqadi, Naicker, & Adeliyi, 2021). Additionally, TomekLinks and Cluster Centroids (CC) are undersampling techniques widely used in data balancing (Zeng et al., 2016; W.-C. Lin et al., 2017). A review of some popular and existing undersampling approaches and their effects on data balancing are discussed below.

CC is an undersampling technique that uses clustering algorithms to select centroids from the majority class. Several studies have evaluated the performance of CC in addressing the class imbalance. For instance, a study by Hira et al. (2015) compared CC with other undersampling techniques on several imbalanced datasets. The results showed that CC outperformed the other methods in terms of classification accuracy (89.2%), F1-score (0.74), and Area Under the Receiver Operating Characteristic Curve (AUC-ROC) (0.903) (Hira & Gillies, 2015). Lenka et al. (2022) introduced the issue of imbalanced binary classification in credit scoring and how it can affect the performance of traditional classification algorithms. To address this issue, undersampling and oversampling techniques have been proposed, and the paper introduces a new clustering-based undersampling technique called CUTE. The proposed technique computes the representatives of each member of the majority class subset. The model is evaluated using two credit-scoring datasets and is compared with four traditional resampling techniques. The performance of the proposed model is shown to improve consistency in various measures, such as accuracy, precision, recall, F1-score, and AUC (Lenka, Bisoy, Priyadarshini, & Nayak, 2022). However, the study does not compare the proposed technique with other state-of-the-art techniques, which could have provided more insights into their proposed approaches.

Zhang et al. (2022) developed a computational method for predicting Anti-Parasitic Peptides (APPs) using an ensemble of ML classifiers. The author generated a balanced dataset using CC undersampling to address class imbalance issues. Fifty-four classifiers were generated by combining nine groups of features and six ML algorithms, and their outputs were used to construct 54 feature representations. The best-performing feature representation in each group was selected for classification, and Logistic Regression (LR) was used to integrate them to construct the model. The proposed model achieved accuracy and AUC of 0.880 and 0.922 on an independent dataset compared to 0.739 and 0.873 for AMPfun (a state-of-the-art method for predicting antimicrobial peptides (AMPs) using ML classifiers), a state-of-the-art method (W. Zhang et al., 2022). While their proposed approaches showed superior performance, it is limited by the quality and diversity of the training data.

While CC has shown promising results in several studies, it may only be suitable for some datasets. For example, in a study by Ikotun et al. (2022), the authors found that CC was less effective on datasets with high-dimensional features and many classes (Ikotun & Ezugwu, 2022). Previously, in a study by Batista et al. (2004), the authors demonstrated that CC might not be suitable for datasets with overlapping classes as it may remove important samples from the majority class (Batista, Prati, & Monard, 2004).

Condensed Nearest Neighbor (CNN) is another popular undersampling technique introduced by Hart in 1968. CNN selects a subset of samples from the majority class by iteratively removing those samples that can be classified correctly using the nearest neighbor rule (Hart, 1968). Recently, Siddappa and Kampalappa (2019) proposed an Adaptive Condensed Nearest Neighbor (Ada-CNN) classifier to address the issues faced by k-Nearest Neighbor (kNN) and improve performance in imbalanced classification. Ada-CNN utilizes the distribution and density of the test point's neighborhood and uses artificial neural systems to learn an appropriate point-explicit k. The experimental results showed that Ada-CNN achieved nearly 94% accuracy for the diabetes dataset for imbalanced classification (Siddappa & Kampalappa, 2019). However, CNN may not be suitable for datasets with high-dimensional features and many classes, as shown in a study by Lemaître et al. (2017) (Lemaître et al., 2017).

Yang et al. (2022) proposed a hybrid sampling approach combining SMOTE and ENN to address the issue of sample imbalance in medical datasets. The proposed method is applied to missed abortion and diabetes datasets, and RF is used for modeling and prediction. The experimental results show that the SMOTE-ENN algorithm outperforms other sampling algorithms, with the best classification performance achieved by RF after SMOTE-ENN sampling. Pairwise and multiple comparisons demonstrate the effectiveness of the proposed approach in improving classification performance. The study highlights the importance of addressing sample imbalance in medical datasets to improve the performance of ML models for clinical diagnosis (F. Yang et al., 2022). Another study by García et al. (2010) evaluated the performance of ENN on imbalanced medical datasets, showing that it improved classification performance, particularly for the minority class (García, Fernández, Luengo, & Herrera, 2010). However, ENN may only be suitable for some datasets, as it may remove essential samples from the majority class, leading to decreased classification performance.

Mqadi et al. (2021) proposed a hybrid data-point approach combining feature selection with the NM undersampling technique to solve misclassification issues in credit card fraud transactions. The approach aimed to improve ML algorithm performance detecting fraud in imbalanced credit card datasets. The authors evaluated the proposed approach on two imbalanced credit card datasets using four algorithms: LR, SVM, DT, and RF. The experimental results showed that the proposed approach improved the predictive accuracy of the algorithms, with RF producing the best results (N. M. Mqadi et al., 2021). The study adds to the growing literature on the effectiveness of undersampling techniques in improving classification performance on imbalanced datasets.

Jia and Zuo (2017) proposed a novel One-sided Dynamic Undersampling (ODU) technique, which dynamically determines whether a majority sample should be used for classifier learning, thus minimizing information loss. This technique is incorporated into a No-Propagation Neural Network (NPNN) to create the ODU-NPNN classifier for imbalanced data classification. The results demonstrate that the ODU-NPNN outperforms NPNN-based algorithms and performs comparably to other well-known methods on real-world imbalanced datasets. The study's contribution is integrating the ODU technique into the classifier learning process, dynamically balancing the training data in each iteration (Jia & Zuo, 2017). In contrast, Nutthaporn and Tanasanee (2017) proposed a solution for imbalanced datasets by combining the Neighbor Cleaning Rule (NCL) and SMOTE techniques. The NCL technique removes outlier data in the majority class, while SMOTE is used to increase the sample data in the minority class to balance the dataset. The balanced dataset is classified by Naive Bayes (NB) and KNN algorithms. The experimental results show improved recall rates in models created from balanced datasets (Junsomboon & Phienthrakul, 2017). However, the study only evaluated the proposed method on a limited number of classifiers, and the performance of the proposed method could be further improved by incorporating other sampling techniques or classifiers.

Pereira, Costa, and Silla Jr (2020) proposed Multi-Label Tomek Link (MLTL), a novel resampling algorithm for addressing the class imbalance in multi-label datasets based on the standard Tomek Link resampling algorithm. In order to show that MLTL is a competitive methodology when compared to other multi-label resampling methods from the current literature, the research conducted tests on seven well-known datasets. This research contributes to multi-label classification by introducing a new resampling algorithm for handling class imbalance in multi-label datasets (Pereira, Costa, & Silla Jr, 2020). However, the study's potential limitations include evaluating the MLTL algorithm on only seven well-known datasets, which makes it unclear how effective it is on other datasets. Furthermore, the study did not compare the computational efficiency of MLTL to other resampling methods, which could be necessary for some practical applications.

Further research is needed to address these limitations and to investigate the effectiveness of MLTL on various datasets.

Goyal (2022) utilized KNN-based undersampling approaches for Software Defect Prediction (SDP). The author proposed a Neighbourhood-based Under-Sampling (N-US) algorithm to address the class imbalance issue in SDP and demonstrated its effectiveness in achieving high accuracy while predicting defective modules. The N-US algorithm under-sampled the dataset to maximize the visibility of minority data points while limiting the excessive elimination of majority data points to avoid information loss. The proposed SDP classifier with the N-US technique was compared with baseline models statistically using benchmark datasets from the NASA repository, and it outperformed the rest of the candidate SDP models with the highest AUC score (95.6%), the maximum Accuracy value (96.9%), and the ROC curve closest to the top left corner (Goyal, 2022). However, the proposed algorithm has not been compared with other oversampling techniques, and the experiments are conducted only on benchmark datasets from the NASA repository, limiting its generalization to other datasets.

In conclusion, several undersampling techniques, such as CC, CNN, NM, and OSDU, have been proposed to address the class imbalance in datasets, and they have shown promising results in several studies. However, these techniques may only be suitable for some datasets due to the presence of high-dimensional features and many classes or overlapping classes, which can lead to decreased classification performance.

2.6.4 Effect of Oversampling

In recent years, oversampling techniques have gained significant attention in various applications, including quality control, defect analysis, and pattern recognition, as they address the issue of class imbalance in datasets (Moreo et al., 2016; Wibowo & Fatichah, 2021; Douzas & Bacao, 2019; Ishaq et al., 2021). SMOTE, ADASYN, and Borderline-SMOTE are some of the most popular and widely used oversampling techniques that have been used by researchers and practitioners more often (Kabir & Ludwig, 2018; H. He et al., 2008; H. Han, Wang, & Mao, 2005). The increased number of minority class instances through oversampling techniques can enhance the classification performance of ML algorithms, making them more effective in detecting defects or recognizing patterns. However, the effectiveness of oversampling techniques can vary depending on the dataset and specific application. Therefore, careful evaluation and experimentation are necessary to select the appropriate oversampling technique for the given dataset and application. A review of some popular and existing oversampling approaches and their effects on data balancing are discussed below (Brownlee, 2020b).

Random oversampling is a commonly used oversampling technique in data balancing. It involves randomly duplicating instances of the minority class until the class distribution is balanced. Studies have shown the potential of random oversampling techniques in improving classification performance on imbalanced datasets (Xin et al., 2021; Calo et al., 2016). For example, Xin and Rashid (2021) used SMOTE as a random oversampling method and RF as a classifier to predict depression among women in Malaysia (Xin et al., 2021). The proposed model showed promising results, but the study is limited to a specific population and a small number of classifiers. Future research could explore the effectiveness of other oversampling techniques and classifiers on diverse populations.

Moreo et al. (2020) proposed a new oversampling method called Distributional Random Oversampling to address the issue of class imbalance in text classification, where the negative class outnumbers the positive class. The proposed method generates synthetic minority-class documents by utilizing the distributional properties of the terms in the collection. The effectiveness of the proposed method was evaluated on three datasets, Reuters-21578, OHSUMED-S, and RCV1-v2, and showed promising results (Moreo et al., 2016). However, the study only evaluated the proposed method on text classification datasets and did not compare its performance with other oversampling methods, which could be a potential limitation.

Wibowo and Fatichah (2021) conducted a comprehensive performance evaluation of various oversampling techniques, including random oversampling, to address the high-class imbalance in ML classification. The study aimed to provide unbiased evaluation results by balancing each class's data. The oversampling techniques, such as Random Oversampling, ADASYN, SMOTE, and Borderline-SMOTE, were combined with ML methods like RF, LR, and kNN. The results showed that RF with Borderline-SMOTE demonstrates the best performance, achieving an accuracy value of 0.9997, 0.9474 precision, 0.8571 recall, 0.9000 F1-score, 0.9388 ROCAUC, and 0.8581 Precision-Recall Area Under the Curve

(PRAUC) for the overall oversampling technique (Wibowo & Fatichah, 2021).

Douzas and Bacao (2019) proposed a new oversampling method called Geometric SMOTE (G-SMOTE) that generates synthetic samples in a geometric region of the input space around each selected minority instance. G-SMOTE is an enhancement of the SMOTE data generation mechanism and allows the deformation of the region from a hyper-sphere to a hyper-spheroid. The study evaluated G-SMOTE against baseline methods and SMOTE and found that G-SMOTE outperforms SMOTE and other baseline methods regarding the quality of the generated data (Douzas & Bacao, 2019).

Ishaq et al. (2021) used SMOTE and data mining techniques to improve the prediction of the heart failure patient survival. The study aimed to identify significant features and effective data mining techniques to increase the accuracy of cardiovascular patient survival prediction. The authors utilized several classification models such as DT, AdaBoost Classifier, and LR. The imbalance class problem was handled by SMOTE, and the ML models were trained on the highest-ranked features selected by RF. The experimental results demonstrated that Extra Tree Classifier outperformed other models, achieving an accuracy of 0.9262 with SMOTE in predicting a heart patient's survival (Ishaq et al., 2021).

Although SMOTE has been widely used to address the issue of class imbalance, it also has several limitations. One of the main limitations of SMOTE is that it can lead to overfitting, especially when generating synthetic samples for the minority class in areas that are densely populated by the majority class. Another limitation of SMOTE is that it may generate noisy samples that do not accurately represent the underlying data distribution, which can negatively impact the performance of ML models. Additionally, SMOTE is computationally expensive, mainly when dealing with large datasets, which limits its practical use. Therefore, it is important to consider these limitations when selecting oversampling techniques and to evaluate their effectiveness based on the specific characteristics of the dataset and the ML algorithm used.

A study conducted by Gök and Olgun (2021) developed an RF-based model using SMOTENC and a gradient-boosting to predict the severity level of Coronavirus disease 2019 (Covid-19) patients using blood samples. The study was conducted in two stages, with preprocessing methods implemented in the second stage for improved results. The model achieved an accuracy of 0.98 with the tuned RF algorithm. The study emphasized the importance of accurate and fast diagnosis in dealing with the pandemic towards the impact on health security, economic security, and social life (Gök & Olgun, 2021).

Mukherjee and Khushi (2021) introduce Synthetic Minority Over-sampling Technique with Edited Nearest Neighbors (SMOTE-ENC) as a new minority oversampling method that encodes nominal features as numeric values and reflects changes associated with the minority class. Experimental results demonstrate that SMOTE-ENC outperforms SMOTE-NC when datasets have a substantial number of nominal features and associations with categorical features and the target class. Additionally, SMOTE-ENC can address one of the major limitations of the SMOTE-NC algorithm by being applied to mixed and nominal datasets (M. Mukherjee & Khushi, 2021).

SMOTE-ENC has several limitations that should be considered when applied to a dataset. Firstly, it needs more support for high-dimensional categorical features, which can generate excessive synthetic examples, leading to overfitting. Secondly, SMOTE-ENC is sensitive to the scale of continuous features, requiring normalization or standardization, which can affect the distribution of nominal features. Thirdly, generating synthetic examples can reduce class separation, decreasing classification performance. Finally, the synthetic samples generated by SMOTE-ENC can introduce sampling bias, leading to overfitting and reduced generalization performance on new unseen data. Thus, researchers and practitioners should be cautious when using SMOTE-ENC and carefully evaluate its performance on a given dataset before using it in real-world applications (Brownlee, 2020b; M. Mukherjee & Khushi, 2021).

Ndichu et al. (2023) presented a novel AI-assisted security alert data analysis method that employed imbalanced learning techniques to address the severe class imbalance. The authors proposed an ensemble of three oversampling techniques to generate high-quality synthetic positive samples, along with a data subsampling algorithm to remove noisy negative samples. Results from experiments on an enterprise and benchmark dataset showed significantly improved recall and false positive rates compared to conventional oversampling techniques. These findings suggest the potential for more effective and efficient Artificial Intelligence (AI)-assisted security operations (Ndichu, Ban, Takahashi, & Inoue, 2023).

Garcia et al. (2023) conducted a research where oversampling techniques were applied to categorical data to discover risk variables connected to Cardiovascular

Diseases (CVDs). The authors wanted to analyze the impact of combining oversampling approaches and linear/nonlinear supervised ML algorithms in binary tasks and establish the most essential characteristics for predicting healthy and CVD patients. Questionnaire data from the Norwegian Centre for E-health Research connected with healthy and CVD patients were employed in the study. Experimental results suggested that oversampling and feature selection strategies boosted CVD prediction, with Generative Adversarial Networks (GAN) and linear models performing better than alternative oversampling techniques. The authors found that synthetic data production aids both identifying risk variables and constructing models with appropriate generalization capabilities in CVD prediction (García-Vicente et al., 2023).

In their study, WU and CAO (2021) investigated the issue of failure diagnosis in railway signal equipment. They collected data from a railway bureau for the period between 2016 and 2020, which was subjected to denoising and feature extraction. Additionally, minority class samples were synthesized using the ADASYN method. For failure diagnosis, the authors employed three different algorithms: Back-Propagation Neural Network (BPNN), Support Vector Machine (SVM), and C4.5. The experimental results demonstrated that these algorithms exhibited poor performance in diagnosing the original data. However, they performed significantly better when diagnosing the synthesized samples. Specifically, the BPNN algorithm achieved the best performance, with an average precision, recall rate, and F1-score of 0.94, 0.92, and 0.93, respectively. These findings confirmed the effectiveness of the BPNN algorithm for failure diagnosis and suggested its potential for practical application in railway signal equipment (Y. Wu & Cao, 2021).

Lu et al. (2020) propose a new model, ADASYN+RF, for identifying telecom frauds. The ADASYN algorithm is used to rebalance the original dataset, and the random forest algorithm is employed to train the new dataset to avoid overfitting. The results of two groups of comparative experiments show that the ADASYN algorithm used in their experiment is more advantageous than the traditional SMOTE algorithm for processing biased data. Furthermore, the accuracy, recall rate, and F1-score of the ADASYN+RF model are significantly improved compared to nonintegrated learning models (C. Lu, Lin, Liu, & Shi, 2020).

Despite its effectiveness, ADASYN has some limitations. One of the limitations of ADASYN is that it can produce synthetic samples that are highly similar to the existing samples, leading to overfitting. Another limitation is that ADASYN may generate noisy synthetic samples if the density distribution of the minority class is highly variable. Additionally, ADASYN may not be suitable for datasets with a small number of minority samples, as it may not produce enough synthetic samples to balance the classes effectively. Finally, ADASYN may perform poorly on datasets with highly overlapping classes or noisy features, affecting its ability to generate useful synthetic samples (Gnip, Vokorokos, & Drotár, 2021).

Fonseca, Douzas, and Bacao (2021) proposed a K-means SMOTE algorithm to address the imbalanced nature of most remotely sensed data and improve the quality of newly created artificial data for land cover classification. The proposed method is compared to three popular oversampling methods using seven remote sensing benchmark datasets, three classifiers, and three evaluation metrics using a five-fold cross-validation approach with three different initialization seeds. The statistical analysis of the results shows that the proposed method consistently outperforms the remaining oversampling approaches, producing higher-quality land cover classifications (Fonseca, Douzas, & Bacao, 2021).

Andelic et al. (2022) proposed a novel approach to detect malicious websites using the Genetic Programming Symbolic Classifier (GPSC) algorithm. The proposed approach utilizes several data sampling methods to balance the large class imbalance in the dataset, and a hyperparameter search method is employed to find the optimal combination of GPSC hyperparameters for high classification accuracy. The experimental results indicate that the best symbolic expression is obtained when the dataset is balanced with the KMeansSMOTE method. The proposed approach achieves high accuracy, AUC, precision, recall, and F1-score values, indicating its potential for effectively detecting malicious websites (Anelić, Baressi Šegota, Lorencin, & Glučina, 2022).

Devi et al. (2022) conducted a study on classifying different types of glass based on their oxide content using oversampling techniques. They preprocessed the glass-type dataset retrieved from the UCI ML repository by scaling and handling missing values and compared the performance of various classification algorithms with and without feature scaling. They applied oversampling methods such as Random, SMOTE, K-Means SMOTE, SVM-SMOTE, ADASYN, and Borderline to identify the target distribution of the dataset. The experimental results showed that SMOTE oversampling outperformed all other methods for the RF classifier, with an accuracy of 90% after feature scaling, compared to 72% accuracy before oversampling. The other oversampling methods also performed well, with the RF classifier achieving an 85-89% accuracy. The authors concluded that oversampling can significantly improve the accuracy of glass type identification, and SMOTE is the most effective oversampling technique for this task (M. S. Devi et al., 2022).

Like other oversampling methods, KMeansSMOTE also has some limitations. One of the primary limitations of KMeansSMOTE is that it may produce synthetic samples that are very similar to the original minority samples, leading to overfitting. Additionally, KMeansSMOTE may not work well on datasets with highly overlapping classes or noisy features. Furthermore, it may not be suitable for datasets with a small number of minority samples as it may not produce enough synthetic samples to balance the classes effectively. Finally, KMeansSMOTE requires setting the number of clusters beforehand, which can be challenging and may impact performance (Tao et al., 2020).

In summary, oversampling approaches have become popular for addressing imbalanced class problems in various domains. Promising oversampling techniques such as SMOTE, ADASYN, and KMeansSMOTE have enhanced classifier performance on imbalanced datasets. However, these techniques are not perfect and have limitations that can negatively impact their performance. For example, oversampling can lead to overfitting, create noisy samples, and may not work well on highly overlapping classes or noisy features. Additionally, no single oversampling technique can effectively work for all types of datasets (Batista et al., 2004; Mayabadi & Saadatfar, 2022). Therefore, additional research is necessary to address these limitations and develop new oversampling techniques that can effectively handle highly imbalanced datasets while addressing these challenges.

2.6.4.1 Effect of SMOTE

One of the primary contributions of this dissertation is the investigation of SMOTE-based data balancing approaches. Thus, this section includes an additional reference literature analysis that focuses specifically on SMOTE-based approaches.

Most of the referenced literature has widely used SMOTE as an oversampling technique in handling class imbalanced problems. One of the main reasons to use this oversampling method is its nature to create synthetic samples that are more diverse than traditional mean, mood-based oversampling techniques (Kabir & Ludwig, 2018; Almhaithawi, Jafar, & Aljnidi, 2020; Taneja, Suri, & Kothari, 2019). For instance, Kabir and Ludwig (2018) employed SMOTE to balance the breast cancer dataset and achieved an F1-score of 0.87, accuracy of 82.18%, and AUC score of 0.9284 (Kabir & Ludwig, 2018). Almhaithawi et al. (2020) used improved SMOTE to oversample the credit card dataset and achieved an F1-score of 1.0 and AUC score of 0.9710 (Almhaithawi et al., 2020). Manzano et al. (2022) utilized SMOTE to handle intrusion detection in cyber security and, using RF, achieved an accuracy of 87% and an AUC-ROC score of 0.95 (Manzano, Meneses, Leger, & Fukuda, 2022). However, none of those studies explain whether the statistical outcomes provide the true prediction of both major and minor classes without creating any biasness. Moreover, pre and post-prediction analyses are not provided considering interpretable ML systems, which ultimately creates more doubt regarding the model's actual performance.

The possibility of major and minor classes overlapping when using SMOTE as an oversampling technique is one of its potential drawbacks. Therefore, several studies suggest the improved version of SMOTE, such as SVM-SMOTE, Borderline-SMOTE, and ADASYN (Taneja et al., 2019; Qing et al., 2022; Shamsudin, Yusof, Jayalakshmi, & Khalid, 2020). Taneja et al. (2019) introduced SVM-SMOTE methods to handle CIP issues and demonstrated better performance on credit card fraud detection incorporating RF and achieved a 0.85 F1-score (Taneja et al., 2019). Shamsudi et al. (2020) proposed RF undersampling techniques based on ML models and achieved a precision of 0.80, recall of 0.83, and F1-score of 0.82 while balancing the minor class using SVM-SMOTE (Shamsudin et al., 2020). Obiedat et al. (2022) employed SVM-SMOTE and cost-sensitive approaches together and achieved an accuracy of 86.5% on the kidney disease diagnosis dataset (Obiedat et al., 2022).

By establishing a hyperplane between the main and minor classes, the SVM-SMOTE technique seeks to overcome the marginalization problem. Nevertheless, SVM is naturally sensitive to unbalanced data, which might have a negative effect on the distribution of the data. Therefore, evaluating the data distributions after data expansion is necessary using SVM-SMOTE (Shrinidhi, Kaushik Jegannathan, & Jeya, 2023; Nagra et al., 2022). Since none of the studies evaluate the data distribution after SVM-SMOTE based oversampling, it is necessary to investigate further. However, such limitations have been addressed in some of the referenced literature.

To overcome the limitations of SVM-SMOTE, many studies suggested an ADASYN-based approach. Khan et al. (2021) utilized ADASYN on a breast cancer dataset and showed that the proposed ADASYN-based ML model outperformed the existing SMOTE-based models by achieving an accuracy of 99.9%, precision of 1.0, recall of 0.995, and F-measure of 0.998. In this work, the author proposed computer-aided diagnosis systems that can automatically classify the characteristics of cancer patches in chest Radiography (X-ray) images (khan, Gupta, Kumar, & Venugopal, 2020). Mostafa et al. (2021) modified ADASYN and employed it to predict accident severity and demonstrated that their proposed Extra Tree (ET) ML algorithm model is capable of overcoming the limitations of existing SMOTE-based approaches by achieving better results as follows: 86.35% accuracy, 0.938 precision, 0.984 recall, and 0.960 F1-score (Mostafa, Salem, & Habashy, 2022).

A modified and updated SMOTE-based approach was introduced by Han et al. in 2005. The author demonstrates that the existing SMOTE did not pay much attention to the minor samples on the borderline. As an effect, the samples that share similar characteristics in major and minor classes often overlap during the data expansion process. In their study, the author introduces two algorithms: borderline-SMOTE1 and borderline-SMOTE2, in which only the minor sample near the borderline are over-sampled. Their preliminary computational result shows that, for the minor class, the proposed algorithms outperformed SMOTE and random-oversampling methods and achieved a better actual positive rate and F1-score (H. Han et al., 2005).

Many studies also considered Borderline SMOTE (BSMOTE) as the alternative solutions. For example, Majzoub et al. (2020) proposed a hybrid clustered affinitive BSMOTE-based approach to handle the imbalanced class data. During the study, the author identifies that existing SMOTE and borderline-based approaches create instances randomly, leading to the generation of useless new instances, which is time and memory-consuming. The authors demonstrate that the highest recall of 0.621 and F1-score of 0.728 is achieved on the amazon employee access dataset (Al Majzoub, Elgedawy, Akaydın, & Köse Ulukök, 2020). However, with their best performance, the recall and F1-score score is still poor and may not be appropriate to apply to real-world scenarios. Sun et al. (2022) introduced feature selection-based network anomalies detection strategies incorporating BSMOTE. The proposed techniques were tested with the three ML algorithms: KNN, DT, and RF. Their computation results show FR with around 91.46% accuracy detecting Web malware (Y. Sun et al., 2022). However, in the real world, the distribution of networks and attacks might differ from one system to another; therefore, considering only three ML algorithms and testing on a limited dataset might not provide the true outcomes of the proposed models.

Even though BSMOTE, ADASYN, and SVM-SMOTE have been used extensively over the years to overcome the noise created during the oversampling approaches, such existing techniques still reduce the noise during the oversampling approach is still a major challenge. Moreover, even after oversampling, it is hard to achieve a normally distributed data sample set most of the time. Considering these limitations, several studies introduced hybrid and improved SMOTE-based oversampling techniques. Song and Peng (2019) introduced ensemble-based SMOTE to handle CIP on credit card fraud detection dataset. The best performance was observed for DT- G-mean of 0.9193, F1-score of 0.9189, and AUC score of 0.9619. Itri et al. (2020) employed Threshold SMOTE (TH-SMOTE) along with the KNN-based model to improve insurance fraud detection performance. Their preliminary computational results illustrate the best performance on the car claim dataset by acquiring the precision of 0.8847, recall of 0.9894, AUC score of 0.93, and F1-score of 0.9341 (Song & Peng, 2019).

Figure 2.2 depicts the timeline of various SMOTE-based approaches that researchers have proposed over the years, as per the literature review.

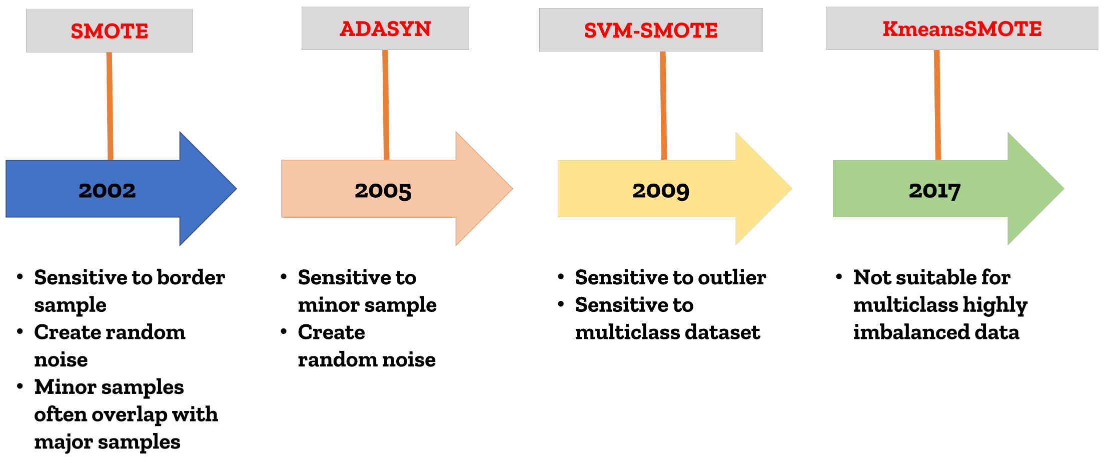

SMOTE
2002
Sensitive to border sample
Create random noise
Minor samples often overlap with major samples
ADASYN
2005
Sensitive to minor sample
Create random noise
SVM-SMOTE
2009
Sensitive to outlier
Sensitive to multiclass dataset
KmeansSMOTE
2017
Not suitable for multiclass highly imbalanced data

Figure 2.2: SMOTE and modified SMOTE-based approaches used in the referenced literature.

2.6.5 Effect of Hybrid Approaches

The combination of over- and under-sampling methods also known as hybrid methods, has been proposed as a potential solution to the limitations of individual oversampling or undersampling techniques (Lemaître et al., 2017; C.-L. Liu & Chang, 2022). By combining these methods, it is possible to reduce the impact of the limitations and achieve better performance on imbalanced datasets. The main idea behind combining over- and under-sampling methods is to first reduce the majority class using an undersampling technique and then to oversample the minority class using an oversampling technique (Brownlee, 2020b). This can help to address issues of overfitting and noise in synthetic samples, as well as improve the performance of classifiers on imbalanced datasets. Several studies have shown that combining over- and under-sampling methods can outperform individual oversampling or undersampling techniques on imbalanced datasets. However, further research is needed to investigate the optimal combination of techniques and their impact on different classifiers and performance metrics (C. Lin, Tsai, & Lin, 2023).

Manju and Nair (2019) proposed a method for classifying arrhythmia patients into ten classes, with one class representing normal conditions and the remaining classes representing different arrhythmia conditions. The authors preprocessed the highly imbalanced dataset and applied a combination of oversampling and undersampling using the Synthetic Minority Over-sampling Technique and Edited Nearest Neighbours (SMOTEENN), followed by feature reduction using Extreme Gradient Boosting (XGBoost). Finally, the feature-reduced dataset was classified using various supervised learning algorithms, resulting in an accuracy of 97.48%, which outperformed the state-of-the-art method (Manju & Nair, 2019). While the study demonstrated promising results, it is limited by using a single dataset and the absence of real-time data. Further research is needed to evaluate the proposed method's performance on multiple datasets, including real-time data, and to investigate the optimal combination of oversampling and undersampling techniques. The impact of different feature selection methods and classifiers on classification accuracy should also be examined.

A study conducted by Puri and Gupta (2022) proposed an improved hybrid bag-boost ensemble model for handling noisy class imbalanced data. The authors noted that data resampling techniques typically used for addressing class imbalance issues might be ineffective in the presence of noise. The authors introduced a new resampling technique consisting of K-Means SMOTE for oversampling and ENN for undersampling to remove noise. This technique was used to cluster the dataset using K-Means, oversample minority classes using SMOTE within each cluster, and then remove instances that created noise using ENN. The proposed model was evaluated using 11 binary imbalanced datasets with varying levels of attribute noise, with AUC-ROC as the performance metric. The experimental results showed that the proposed model outperformed other existing models and performed better with an increased noise level in the binary imbalanced datasets (Puri & Kumar Gupta, 2022).

Han et al. (2022) proposed an interpretable SMOTE-ENN-XGBoost to address imbalanced data issues for lodging detection. The SMOTE-ENN-XGBoost model achieved an F1-score of 0.930 and a recall of 0.899 on a testing set, indicating its potential for lodging detection. The authors also employed the Shapley Additive Explanations (SHAP) approach to interpreting the identification and prioritization of features that determine lodging classification and activity prediction. They found that canopy structure and textural features are relatively stable compared to spectral features, which are susceptible to the external environment. The study suggests that canopy structural, spectral, and textural information should be considered simultaneously when detecting crop lodging in a crop breeding program (L. Han et al., 2022). However, like many data balancing techniques, SMOTEENN is also limited in handling datasets with overlapping classes (Ding, Chen, Dong, Fu, & Cui, 2022). Additionally, it may also often remove relevant minority class samples during the cleaning process.

Several studies combined SMOTE and Tomek Links as hybrid methods to handle imbalanced data. For instance, a study conducted by Rana et al. (2021) aimed to predict the possibility of stroke using advanced ML and DL techniques using SmoteTomek. The authors proposed a final model using Artificial Neural Network (ANN), which achieved the best ROC score of 0.84 and compared the performance of other ML algorithms such as ensemble-based, tree-based, and NB-based algorithms. The study highlights the importance of using advanced techniques to predict stroke and provides valuable insights into the performance of different algorithms in predicting the condition (Rana, Chitre, Poyekar, & Bide, 2021). Another study conducted by Hairani, Anggrawan, and Priyanto (2023) aimed to improve the classification performance of imbalanced diabetes data using the SMOTETomek technique and the RF algorithm. The study used the Pima Indian Diabetes dataset. The performance was evaluated based on accuracy, sensitivity, precision, and F1-score using 10-fold cross-validation. The results showed that the RF algorithm with the SmoteTomek achieved the highest accuracy, sensitivity, precision, and F1-score compared to the RF algorithm with Smote. The study concluded that using the SMOTETomek link technique can improve the classification performance of the RF algorithm for imbalanced diabetes data (Hairani, Anggrawan, & Priyanto, 2023).

By combining SMOTETomek, a Convolutional Neural Network (CNN), and a Gated Recurrent Unit (GRU) with a software defect prediction (SDP) algorithm, Yang and Li (2022) proposed a unique technique for addressing class imbalance in SDP. In comparison to the original datasets, their solution showed an average accuracy gain of 5%, highlighting the usefulness of merging ML approaches with data balancing techniques in SDP (H. Yang & Li, 2022). However, further research is needed to investigate the optimal combination of techniques and their impact on different performance metrics and address the proposed approach's limitations.

While SMOTETomek has been shown to improve classification performance on moderately imbalanced datasets, it may have limited effectiveness on highly imbalanced datasets where the minority class makes up less than 10% of the data. In such cases, more advanced techniques may be necessary. Additionally, SMOTETomek involves a combination of oversampling and undersampling techniques, which can increase the computational time required to train a model. This can be a concern when dealing with large datasets or when the technique is applied repeatedly in an ML pipeline. Another potential limitation is the risk of overfitting, where the oversampling component of SMOTETomek can introduce bias in the data by creating synthetic samples that closely resemble existing minority class examples. Careful hyperparameter tuning and cross-validation can help mitigate this risk, but it remains an important consideration when using SMOTETomek (Brownlee, 2020b).

2.6.6 Effect of GAN-Based Approaches

Recently, it was observed that the existing SMOTE and hybrid SMOTE techniques cannot produce diverse data samples, especially on a highly imbalanced dataset with multiclass samples. To overcome such limitations, several studies recently suggested a GAN-based approach to creating a synthetic sample from the imbalanced dataset (Z. Lin, Khetan, Fanti, & Oh, 2018; A. Sharma, Singh, & Chandra, 2022; Xiao, Wu, & Lin, 2021; W. Jiang, Hong, Zhou, He, & Cheng, 2019). For instance, Jiang et al. (2019) proposed a GAN-based anomaly detection approach to detect fault from the highly imbalanced industrial time series data. Based on the author's assessment, the performance of SVM and CNN suffers from achieving high accuracy for the major class. Their proposed model contains three sub-network generators: encoder-decoder-encoder. Based on the result report, the study emphasized that their proposed model can detect abnormal samples from normal samples with 100% accuracy on both datasets. Based on the author's claim, their model can detect abnormal samples without having any prior knowledge of abnormalities (W. Jiang et al., 2019). Therefore, one of the major concerns is how the model performs on a test sample when the outlier's threshold value is almost close to the normal sample's threshold value.

In their research, Xiao et al. (2021) introduced the Wasserstein Generative

Adversarial Network (WGAN) model, a Deep Learning (DL)-based technique that was used to analyze data from three different cancer types: breast, stomach, and lung. The CIP ratio issue was addressed by generating new instances from the minor class using the WGAN approach. (Xiao et al., 2021). Kwon et al. (2021) used GAN based approach to creating minor samples from the cancer tissue images and demonstrated that in-corporate with Deep Neural Networks (DNN), their proposed model can detect the cancer cell successfully up to 61% accurately (Kwon, Park, Ko, & Ahn, 2021). Guo et al. (2021) used Traffic Augmentation GAN (TA-GAN) along with CNN to detect traffic signals and demonstrated promising results by achieving the AUC score of 0.9739 and F-measure of 0.8624 (Y. Guo et al., 2021).

GAN is widely utilized in computer vision domains for its ability to generate realistic images from random noise. This dynamic characteristic contributes to its appeal, as it can be applied to various data formats, including time-series, audio, and image data (Brownlee, 2020b; Ahsan, Ali, & Siddique, 2022; Rosolia & Osterrieder, 2021). Sharma et al. (2022) demonstrated that GAN could generate data with better Gaussian distribution, which is difficult to achieve using traditional imbalanced approaches. Their proposed GAN-based approaches show almost actual minor samples and outperform existing techniques by 10% (A. Sharma et al., 2022).

One of the potential drawbacks of GAN is that it is hard to train and often requires a lot of data. As GAN creates samples using random noise, it is time-consuming and requires many iterations to get optimal results (Ahsan, Ali, &

Siddique, 2022; A. Sharma et al., 2022; Yinka-Banjo & Ugot, 2020). Additionally, the parameters of the GAN are very sensitive, which ultimately makes GAN an unstable approach. However, many studies demonstrate that by using GAN, it is possible to overcome the multicollinearity and overlapping issues of the oversampling approach on a dataset. Therefore, if GAN-based algorithms are stable, computationally effective, and require less iteration to train will be the best alternative to overcome the existing challenges associated with SMOTE and GAN-based approaches.

2.6.7 Effect of Other Data Balancing Approaches

In addition to the data-balancing methods discussed previously, various other approaches have been proposed in the literature. For example, Zhang, Ran, and Mi (2019) proposed a new Intrusion Detection System (IDS) model based on CNN and SMOTE-ENN algorithm to identify network traffic intrusion. The model achieved an accuracy of 83.31% on the NSL-KDD dataset, with significantly improved detection rates for User to Root (U2R) and Remote to Local (R2L) attacks. The proposed SMOTE-ENN-based CNN IDS outperformed the previous IDS model based on traditional ML (X. Zhang, Ran, & Mi, 2019). However, the study's limitation lies in the dataset used, which may not represent all types of attacks and network environments.

Rahman and Zhu (2023) conducted a study to develop effective accounting fraud detection models for China A-Share listed firms using imbalanced ensemble learning algorithms. The study employed a sample of 33,544 Chinese firm-year instances from 1998 to 2017. It developed one LR and four ensemble learning classifiers, including AdaBoost, XGBoost, Cost-Sensitive Boosting (CUSBoost), and Random Under-Sampling Boosting (RUSBoost), based on 12 financial ratios and 28 raw financial data. The study evaluated the classifiers' out-of-sample performance using AUC and Area Under the Precision-Recall Curve (AUPR) metrics and divided the sample into train and test observations. The results showed that ensemble learning classifiers outperformed the LR model, and imbalanced ensemble learning classifiers (CUSBoost and RUSBoost) were more effective than standard ensemble learning models (AdaBoost and XGBoost) on average. Additionally, the study introduced an algebraic fused model in the supplement test, which achieved the highest average AUC and AUPR among all the employed algorithms (Rahman & Zhu, 2023). However, the study did not address the potential limitations of ensemble learning algorithms in accounting fraud detection or the generalizability of the results beyond China A-Share listed firms.

Mousavian, Haeri, and Moslehi (2022) proposed a new approach called strong balance-constrained clustering to enhance the accuracy and diversity of ensemble classification methods by creating strong, balanced data clusters. The method employs an ANN with more than one hidden layer and majority voting to make the final decision for the data class. The proposed method was evaluated on 16 datasets, and the results indicate that it is faster than other balancing methods while providing acceptable accuracy improvements for the ensemble classification method. The proposed method can enhance the performance of data mining and business objectives by adopting the appropriate budget, time, and energy assignment policies for various business domains (Mousavian Anaraki, Haeri, & Moslehi, 2022). However, the study did not explore the limitations and potential drawbacks of the proposed method, and further research is required to assess its applicability in different domains and datasets.

Gabriel et al. (2022) conducted a study to develop ML models for predicting whether surgery would be completed by the end of the operating room block time and if the patient would be discharged by the end of the recovery room nursing shift. The study compared several ML models, including Regression, RF, balanced RF, Balanced Bagging (BB), NN, and SVM, with LR, using features such as surgery, surgeon, service line, age, sex, weight, and scheduled case duration to make predictions. The authors used SMOTE to evaluate the model's performance. The study showed that the BB classifier performed the best, with an F1-score of 0.78, 0.80, 0.82, and 0.82 when predicting the outcome for start times of 1 pm, 2 pm, 3 pm, and 4 pm, respectively. The results indicated that ensemble learning models could significantly improve the accuracy of predicting the outcome at different start times, offering a more practical approach for operating room management to determine whether an add-on case at an outpatient surgery center could be appropriately booked (Gabriel et al., 2022).

Bennett et al. (2022) conducted a study to predict Pre-eclampsia (PE) using a Cost-Sensitive DNN (CSDNN) to address data imbalance and racial disparities. The study validated the model by using diverse data sources, such as Texas Public Use Data Files, Oklahoma PUDF, and the Magee Obstetric Medical and Infant databases. The CSDNN equipped with focal loss function outperformed other state-of-the-art techniques, with an AUC of 66.3% and 63.5% for the Texas and Oklahoma PUDF, respectively, and an AUC of 76.5% for the MOMI data (Bennett, Mulla, Parikh, Hauspurg, & Razzaghi, 2022). However, the study has some limitations, including retrospective and possibly missing data that may impact the model's accuracy. Nonetheless, the study provides valuable insights into the predictive power of clinical databases for PE among minority populations.

2.7 Overall Findings

This chapter explores the scope of data-balancing approaches toward major and minor classes by reviewing different oversampling and undersampling approaches introduced in the literature to handle CIPs. The study identified various factors affecting ML-based models' performance on different imbalanced datasets and case scenarios. Figure 2.3 presents a framework of data balancing approaches (DBA), where potential key findings have been combined using a fishbone diagram. The fishbone diagram starts with the type of data balancing approaches, followed by key challenges of imbalanced datasets, different ML algorithms, performance metrics, applications, and major limitations of the existing DBA based on the reference literature.

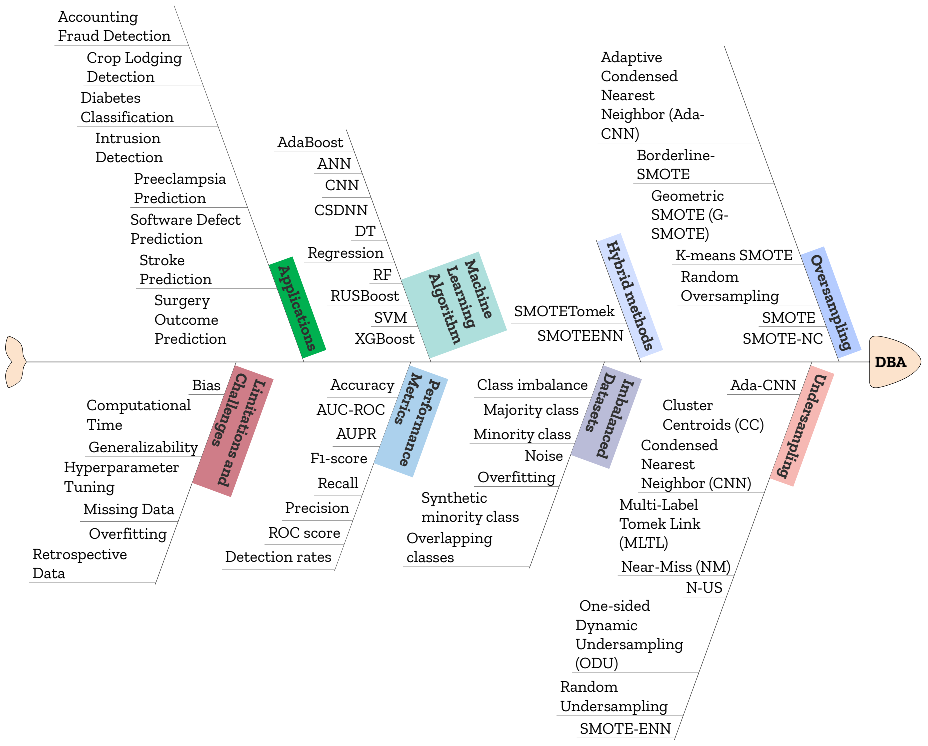


Figure 2.3: Framework of DBA based on referenced literature; DBA – Data balancing approaches.

The study found that data distribution on an imbalanced dataset significantly affects the performance of ML models during the prediction. If the data is skewed toward one class, then it is challenging for the ML model to perform the prediction without being biased toward the major class. To tackle these issues, several studies suggest using data preprocessing steps before applying data balancing approaches. The findings suggest that data normalization techniques, such as min-max, Z-score, and L2 norm, can improve the data distribution and enhance the performance of ML models (J. Wu, Zhao, Sun, Yan, & Chen, 2021; A. J. Mohammed et al., 2020;

Mustafa, 2019; Blagus & Lusa, 2013; Gupta et al., 2017; Uyun & Sulistyowati, 2020). However, it is unclear whether using such data normalization techniques has a significant effect on the model's overall performance or not. Additionally, using several data normalization techniques before data balancing may change the entire minor class data distribution, which may not help predict outlier or minor samples (N. Jiang & Li, 2021).

The performance of different ML algorithms varies with similar data-balancing approaches (Pawlicki, Choraś, Kozik, & Hołubowicz, 2020; Rajesh & Dhuli, 2018; Mahesh et al., 2022; Sisodia et al., 2017; N. Mqadi et al., 2021). However, the study found that none of the studies mentioned clearly whether they had tuned the parameter of the ML model or used default parameters. Thus, it is difficult to conclude that the performance of ML algorithms differs solely based on data balancing approaches. This highlights the need for proper clarification to understand better the impact of data balancing approaches on the performance of ML algorithms with or without parameter tuning.

Several studies evaluated the effectiveness of oversampling techniques on various applications, such as predicting depression, heart failure patient survival, the severity level of Covid-19 patients, failure diagnosis of railway signal equipment, identifying telecom frauds, and classifying different types of glass based on their oxide content (Sawangarreerak & Thanathamathee, 2020; Ishaq et al., 2021; Farahany, Wu, Islam, & Madiraju, 2022; M. S. Devi et al., 2022; Xin et al., 2021; Moreo et al., 2016; Wibowo & Fatichah, 2021). The results of these studies show that oversampling can significantly improve classification accuracy, and different oversampling techniques may perform better depending on the dataset and specific application. However, the effectiveness of oversampling techniques can vary depending on the dataset and specific application, making careful evaluation and experimentation necessary to select the appropriate ones for the given dataset and application.

SMOTE is the widely used oversampling technique that creates synthetic samples that are more diverse than traditional mean-based oversampling techniques (F. Shen, Zhao, Kou, & Alsaadi, 2021; Rivera & Xanthopoulos, 2016; Douzas & Bacao, 2019; M. Mukherjee & Khushi, 2021; Ndichu et al., 2023; García-Vicente et al., 2023). However, the study found that the performance of different ML algorithms on SMOTE-based oversampled datasets is inconsistent (Ishaq et al., 2021; Gök & Olgun, 2021). The accuracy of ML algorithms using similar datasets and oversampling techniques can vary depending on the tuning of SMOTE or the default parameters. Additionally, the major and minor classes overlapped due to oversampling, which is one of the potential drawbacks of applying SMOTE as an oversampling technique.

To overcome the limitations of SMOTE, several studies have introduced modified and updated SMOTE-based approaches, such as SVM-SMOTE, BSMOTE, and ADASYN (Y. Wu & Cao, 2021; Gnip et al., 2021; Fonseca et al., 2021; Anelić et al., 2022; M. S. Devi et al., 2022; Tao et al., 2020). The SVM-SMOTE algorithm reduces the marginalization by creating the hyperplane between major and minor classes. However, SVM is sensitive to imbalanced data by nature, which potentially affects data distribution (Ahsan, Ali, & Siddique, 2022; Brownlee, 2020b).

Therefore, evaluating the data distributions after data expansion is necessary using SVM-SMOTE. ADASYN-based approaches have been proposed to overcome the limitations of SVM-SMOTE (Tao et al., 2020). Borderline-SMOTE1 and Borderline-SMOTE2 algorithms are introduced in which only the minor sample near the borderline is over-sampled (Smiti & Soui, 2020). Several studies have also introduced hybrid and improved SMOTE-based oversampling techniques (W.-C. Lin et al., 2017; Manju & Nair, 2019; Puri & Kumar Gupta, 2022; L. Han et al., 2022).

In recent times, GAN-based approaches have been proposed as a viable option to generate synthetic samples from imbalanced datasets (Z. Lin et al., 2018; A. Sharma et al., 2022; Xiao et al., 2021; W. Jiang et al., 2019). One of the significant advantages of GAN is its capability to create realistic images from random noise, making it versatile for various data formats. Several studies have shown that GAN-based algorithms can effectively overcome multicollinearity and overlapping problems associated with oversampling techniques (Y. Guo et al., 2021; A. Sharma et al., 2022).

Despite the potential of GAN-based approaches, the training of GAN models can be challenging and may require a substantial amount of data. Moreover, the optimal selection of GAN parameters is vital as the sensitivity of the GAN parameters can significantly impact the stability of the approach. Hence, using GAN as a data balancing technique may require expertise in deep learning and additional computational resources (Ahsan, Ali, & Siddique, 2022; A. Sharma et al., 2022; Yinka-Banjo & Ugot, 2020).

In summary, GAN-based approaches have shown promise as a potential solution for addressing the class imbalance problem in datasets. However, the approach's sensitivity to training and the selection of optimal parameters should be taken into account before using it as a data balancing technique. Additional research is required to explore further and develop GAN-based approaches and make them more practical for real-world applications.

2.8 Conclusions

The present study addressed the research question: "What are the scopes of data-balancing approaches toward the major and minor samples?" The study sheds light on the potential of using data normalization techniques to enhance data distribution and on the impact of data balancing approaches on the performance of ML algorithms, which are critical components in ensuring data quality and accuracy.

However, there are still limitations and challenges in applying data balancing approaches, including the risk of overfitting, the need for hyperparameter tuning, and increased computational time. These challenges must be addressed to improve the effectiveness and applicability of data balancing approaches in quality, defect, and pattern analysis. Future research could focus on exploring new data balancing techniques, evaluating their performance on different ML algorithms, and determining optimal parameters for different oversampling approaches. Such research can contribute to developing better data-balancing approaches and improve the performance of ML models when dealing with imbalanced datasets.

In summary, data balancing approaches are essential in addressing the issue of imbalanced data in ML-based model development, particularly in quality, defect, and pattern analysis. Ongoing research is necessary to improve the effectiveness and applicability of these approaches in these fields, taking into consideration the unique challenges and limitations that exist.

Chapter 3

Effect of SMOTE on Data Balancing Approaches

3.1 Introduction

Based on the overall analysis and the referenced literature from Chapter Two, it is evident that Synthetic Minority Oversampling Technique (SMOTE)-based approaches are widely used for data balancing. However, there needs to be more research that measures the performance of SMOTE-based approaches in comparison to traditional Machine Learning (ML) algorithms with and without parameter tuning. This knowledge gap raises the question of whether SMOTE-based approaches have any effect on data balancing. Therefore, the research question that this chapter will address is RQ2: "What is the effect of traditional Machine Learning (ML) and SMOTE-based data balancing on imbalanced data analysis?" To answer this question, the chapter will consider the following approaches:

- Firstly, a general overview of the most widely used ML algorithms will be presented, and their performance on various datasets before and after SMOTE-based oversampling in terms of various statistical measures will be calculated.

- Secondly, a hypothesis test will be conducted to evaluate the significance of SMOTE-based approaches on data balancing. Four conditions will be considered for this purpose: the ML model's performance on the original dataset and the SMOTE-based oversampled dataset, the hyperparameter-tuned ML model's performance on the original dataset, and the SMOTE-based oversampled dataset.

- Finally, the chapter will discuss the general findings, limitations, and potential scope for future research.

By addressing RQ2, this chapter aims to provide a comprehensive overview of SMOTE-based data balancing approaches and their potential findings considering various imbalanced benchmark datasets.

3.1.1 Motivation

Traditional ML algorithms often struggle to perform well on imbalanced datasets due to their inherent bias towards the majority class.The resulting imbalanced class distribution in a dataset may lead to inadequate predictive performance for the minority class, which is often the focus of interest for researchers. Therefore, it is crucial to understand the impact of different data balancing techniques, such as SMOTE-based oversampling and hyperparameter tuning, on ML-based model performance in imbalanced datasets.

Therefore, the motivation behind the study is to investigate the effect of SMOTE-based data balancing on imbalanced data analysis by utilizing six benchmark datasets and testing with four designated conditions. The study's findings will contribute valuable insights to understanding the performance of ML models with and without hyperparameter tuning before and after balancing the data using SMOTE-based approaches.

3.1.2 Chapter Outline

The following is the structure of this chapter: Section 3.2 begins with a concise overview of the experimental setup, encompassing a comprehensive summary of the datasets employed, the ML algorithms utilized during the study, and the performance evaluation metrics applied throughout the research process. Subsequently, Section 3.3 presents the experimental results, followed by an overarching analysis and interpretation of the findings in Section 3.4. In Section 3.5, a hypothesis is formulated, considering four distinct conditions. The overall outcomes of the hypothesis testing are then discussed in Section 3.6. Lastly, Section 3.7 offers a general conclusion based on the comprehensive findings and highlights the study's limitations, thereby providing insights for further research scope.

3.2 Experimental Setup

The study was conducted using a Dell Inspiron 7579 laptop with a Windows 10 operating system. The hardware specifications of the laptop included an Intel Core I7-7500U 7th generation processor, Intel HD 620 integrated graphics, 16GB of system RAM, and 512GB of storage. The laptop also had a front VGA camera. The software used for data analysis included Python 3.9 as the primary programming language, and several packages were used to carry out the experiment, including Scikit learn, Numpy, Keras, PyTorch, and Matplotlib. These software tools and packages were selected to ensure that the study was carried out effectively and efficiently.

3.2.1 Datasets

The experiment used six benchmark datasets from the open-source UCI (the University of California Irvine) repository: Ionosphere, Pageblocks, Poker, Spambase, Winequality, and Yeast (Blake, 1998).

3.2.1.1 Ionosphere

The Ionosphere dataset is a widely used benchmark binary classification dataset that was collected by Johnson and Bresticker from the Johns Hopkins University Applied Physics Laboratory. The dataset aims to distinguish between good and bad radar returns from the ionosphere using 34 numeric attributes that are radar return characteristics measured by a phased array radar system. The dataset, which contains 351 instances, was donated by Dua and Taniskidou from the University of California, Irvine. Notably, the Ionosphere dataset has no missing values and has been extensively employed to evaluate and compare various ML algorithms and feature selection methods. As such, it has been cited in numerous research articles and ML textbooks as a standard dataset for binary classification tasks (Ganger, 2023; Sigillito, Wing, Hutton, & Baker, 1989).

3.2.1.2 Pageblocks

The PageBlocks dataset is a popular benchmark dataset used for evaluating the performance of classification algorithms in ML and data mining. The dataset comprises 5,473 web pages that are the same size and categorized into one of five groups according to their feature. Each example is represented by a feature vector that captures various attributes of the web page, such as height, width, and the number of horizontal and vertical lines on the page. The classes in the dataset are highly imbalanced, with most examples belonging to the “text” category, which contains 4,716 examples. The other categories, in descending order of frequency, are “horizontal line,” “picture,” “vertical line,” and “graphic” (Malerba, Esposito, & Semeraro, 1996).

3.2.1.3 Poker

The Poker dataset is a multivariate dataset used for classification tasks. The dataset comprises ten predictive features and a class attribute that characterizes the poker. The dataset comprises examples corresponding to a set of five playing cards selected from 52 cards. The order of cards is essential, resulting in a total of 9 possible classes, ranging from “nothing in hand” to “royal flush.” The dataset is frequently utilized to assess the effectiveness of classification algorithms concerning game-related data, which makes it a valuable asset for researchers studying ML domains (Radziukas, Maskeliūnas, & Damaševičius, 2019).

3.2.1.4 Spambase

The Spambase dataset is a binary classification dataset that contains email messages’ attributes and labels. The dataset contains 4601 instances, and each instance has 57 real attributes, including the frequency of specific words and characters, character capitalization, and punctuation marks, among others. The target variable indicates whether the email is spam (1) or not (0). The dataset has no missing values and has been widely used to evaluate and compare various ML algorithms and techniques for spam detection. The Spambase dataset is regarded as a standard benchmark for binary classification tasks, and it has been referenced in many research articles and ML textbooks (Hopkins, Reeber, Forman, & Suermondt, 1998).

3.2.1.5 Wine Quality

The Wine Quality dataset contains sensory data on wine quality and physicochemical attributes for red and white wine. The dataset has been used for regression and classification in ML domains. However, the dataset's classes are both ordered and imbalanced, making it a valuable resource for investigating the challenges of imbalanced class data analysis. Anomaly detection techniques can be employed to identify the rare instances of outstanding or inferior quality wines in the dataset. At the same time, feature selection methods can help identify which input variables are most relevant for predicting wine quality. The dataset was created by Paulo Cortez and his colleagues at the University of Minho, Portugal, and was initially published in the Decision Support Systems journal in 2009 (Cortez, Cerdeira, Almeida, Matos, & Reis, 2009).

3.2.1.6 Yeast

The Yeast dataset is a multivariate dataset with eight real attributes and 1484 instances. The dataset aims to predict the localization sites of proteins using various scoring and recognition methods for signal sequences, membrane-spanning regions, and amino acid contents. It includes information on the sequence name, scores for different protein regions, and the presence of specific substrings or targeting signals. The dataset has no missing values and was donated by Paul Horton to Kenta Nakai. The dataset is associated with classification tasks, and the references provided describe expert systems for predicting protein localization sites (Cortez et al., 2009).

3.2.2 ML Algorithms

ML algorithms represent computational methods capable of autonomously discovering and capturing intricate patterns and relationships inherent in data without necessitating explicit programming. The ML algorithms can be employed to evaluate and make predictions on various datasets, ranging from image and speech recognition to customer behavior and financial data (El Naqa & Murphy, 2015). In this research, six popular, most widely used ML techniques have been used to evaluate the effect of SMOTE-based approaches on nine highly imbalanced benchmark datasets. A summary of the ML algorithms used in this work is presented below:

3.2.2.1 Adaptive Boosting

Adaptive Boosting (AB) is a widely used classifier that Yoav Freund and Robert Schapire developed. The algorithm is based on ensemble learning, combining multiple weak classifiers to form a more robust classifier. AB assigns weights to each sample in the dataset based on its classification error rate. Samples that are difficult to classify receive greater weight, while those already well-categorized receive less weight. This approach helps the algorithm focus on the dataset's most challenging samples, which can improve its accuracy. AB can handle categorization and regression analysis tasks, making it a versatile tool for ML applications (Schapire, 2013). The formula to calculate AB is:

$$\hat{f}(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right)$$

where $\hat{f}(x)$ is the final boosted model, M represents weak classifiers number, α_m is the weight assigned to the m-th classifier, and $h_m(x)$ is the m-th weak classifier, which maps the input x to a binary label. The sign function outputs $+1$ or -1, depending on the sign of the sum. Adaboost iteratively updates the weights of misclassified samples, and trains a new weak classifier on the updated weights at each iteration. The resulting model can be represented as a weighted sum of the weak classifiers, where the weights depend on their performance in classifying the training data.

3.2.2.2 Decision Tree

The Decision Tree (DT) algorithm operates on the divide-and-conquer principle. It partitions the dataset into subsets recursively based on the input attribute values. In DT models, the attributes can take on discrete values, and the resulting tree structure reflects the combinations of attributes that lead to distinct class labels. This type of tree is known as a classification tree, where each leaf node corresponds to a unique class label.

In addition to classification trees, DT can be used to construct regression trees that handle continuous variables as input attributes. The two most famous and widely used DT algorithms are C4.5 and EC4.5. C4.5 is an extension of the ID3 algorithm that can handle both continuous and categorical variables. At the same time, EC4.5 is an enhanced version of C4.5 that addresses some of its limitations, such as overfitting and sensitivity to noisy data (Brijain, Patel, Kushik, & Rana, 2014). The formula to calculate DT model is:

$$h(x;\theta) = \sum_{i=1}^{K} c_i I(x \in R_i;\theta)$$

Where h is the decision tree model, x is the input vector, θ is the model's parameter, K is the number of leaves in the decision tree, R_i is the region associated with the i-th leaf, the function I returns a value of 1 as an indicator if x is in the region R_i, and c_i is the output value associated with the i-th leaf.

3.2.2.3 Gradient Boosting

Gradient Boosting (GB) is a powerful ensemble learning technique used to improve the performance of ML models. It is a sequential process that uses multiple decision trees to make predictions and minimize the error of the previous tree. In each iteration, the algorithm calculates the gradient of the loss function concerning the previous model's output, and the new tree is built to fit the negative gradient. GB was first proposed in 1999 by Jerome H. Friedman, who developed the algorithm to improve upon the performance of AB. The algorithm uses gradient descent to minimize the loss function and updates the weights of the weak learners in each iteration. The update formula can be represented as (Ayyadevara & Ayyadevara, 2018):

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{3.1}$$

where F_m is the current ensemble, h_m is the newly added weak learner, and γ_m is the step size for that iteration. The objective is to find the optimal values for h_m and γ_m that minimize the loss function.

3.2.2.4 K-Nearest Neighbors

The K-Nearest Neighbor (KNN) classification is a nonparametric technique used for classification and regression analysis. It was first introduced by Evelyn Fix and Joseph Hodges in 1951. In KNN classification, the class membership is determined by voting mechanisms. To compute the distance between two data samples, Euclidean distance techniques are commonly used. In regression analysis, the projected value is computed as the average of the values of the K nearest neighbors. The formula for the KNN algorithm can be expressed as follows: Given a query point x, we find the K closest points to x in the training set, T, denoted by $N_K(x)$. The class label for x is then determined by a majority vote of the labels of the K nearest neighbors. If the distance metric is Euclidean distance, then the distance between x and a training point x_i is given by (Kramer & Kramer, 2013; Brownlee, 2014):

$$d(x, x_i) = \sqrt{\sum_{j=1}^{p} (x_{ij} - x_j)^2} \tag{3.2}$$

where x_{ij} is the value of the jth feature for the ith training point, and p is the number of features. The KNN algorithm is a simple yet powerful algorithm that has been widely used in various fields, including computer vision, bioinformatics, and recommender systems.

3.2.2.5 Logistic Regression

Logistic Regression (LR) is a popular ML approach for solving classification problems. The model is based on a probabilistic framework, and its output values are generally between 0 and 1. The use of LR has been successful in a number of applications, including but not limited to the detection of spam emails, online fraud, and malignant tumors. The model uses a cost function known as the sigmoid function, which maps real numbers to a range between 0 and 1 (Ahsan, Luna, & Siddique, 2022; Dayton, 1992; Brownlee, 2016):

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.3}$$

where z is a linear combination of input variables and model parameters. The cost function aims to find the optimal set of model parameters that minimize the difference between the predicted and actual class labels. LR has been widely used in various fields due to its simplicity and effectiveness in solving classification problems wright1995logistic.

3.2.2.6 Random Forest

Random Forest (RF) is an ensemble learning algorithm in ML that builds multiple decision trees and aggregates their predictions to make a final prediction. The algorithm was proposed by Leo Breiman and Adele Cutler in 2001 and has since become a popular approach for classification, regression, and feature selection tasks. RF is particularly well-suited for high-dimensional data, missing values, and nonlinear relationships. However, it may not perform well with imbalanced class data, and evaluating its performance on SMOTE based datasets is important to ensure that it accurately classifies all classes, particularly minority classes. The RF model combines the DT's outputs by averaging them to get the final prediction. The model's hyperparameters, such as the number of trees, tree depth, and the number of features considered at each split, can be optimized to achieve better performance. The formula for calculating the output of a single decision tree is as follows (Saini, 2022; Brownlee, 2014; Ahsan & Siddique, 2021):

$$h(x_i, \theta_j) = \sum_{j=1}^{J} c_j I(x_i \in R_j) \tag{3.4}$$

where x_i is a sample, θ_j is the set of parameters of the j-th tree, c_j is a constant that is calculated using the training data and R_j is the region of the feature space that corresponds to the j-th leaf node of the tree. The final prediction of the Random Forest model is given by the majority vote of the output of all decision trees.

3.2.2.7 Support Vector Machine

Support Vector Machine (SVM) is a popular ML approach for classification and regression-related challenges. Vapnik introduced it in the late twentieth century (Drucker, Wu, & Vapnik, 1999). SVMs have found extensive use in multiple domains, such as disease diagnosis, speech recognition, text classification, facial expression recognition, protein fold, and distant homology discovery. SVM is a unique ML algorithm that can classify unlabeled data by using a hyperplane to identify clustering among the data. Unlike other algorithms, SVM has the ability to perform binary classification and multi-class classification, making it a powerful tool for ML applications. However, SVM output is not nonlinearly separable. Selecting appropriate kernels and parameters is crucial when applying SVM in data analysis to overcome such problems. The SVM optimization problem can be formulated as follows (L. Wang, 2005):

$$\min_{w,b,\xi} \frac{1}{2}w^T w + C\sum_{i=1}^{m} \xi_i$$

subject to:

$$y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i, \quad i = 1, 2, \ldots, m$$

$$\xi_i \geq 0, \quad i = 1, 2, \ldots, m$$

where w and b are the parameters of the model, ξ_i are slack variables, C is the regularization parameter, $\phi(x_i)$ is the feature space, and y_i is the label of the ith training sample. The optimization problem seeks to find the hyperplane with the largest margin that correctly classifies all training examples.

Table 3.1: Grid search hyperparameters of machine learning algorithms used to conduct the experiment.

Algorithm	Grid search parameters
AB	n_estimators: 50, 100, 150,learning_rate: 0.01, 0.1, 1
DT	criterion: 'gini', max_depth: 10, 50, min_samples_split: 3, 4, 5, max_features: 2, 5, 6, 8
GB	n_estimators: 50, 100, 150, learning_rate: 0.01, 0.1, 1, max_depth: 3, 5, 7
K-Nearest Neighbors	n_neighbors: 3, 5, 7, 9, weights: 'uniform', 'distance'
LR	penalty: 'l1', 'l2', C: 0.01, 0.1, 1, 10
RF	n_estimators: 50, 100, 150, max_depth: 10, 20, min_samples_split: 3, 4, 5, max_features: 2, 5, 6, 8
SVM	C: 0.1, 1, 10, kernel: 'linear', 'rbf', gamma: 'scale', 'auto'

3.2.3 Performance Evaluation

The experimental evaluation results are presented in terms of accuracy, precision, recall, F1-score, and AUC-ROC score (Ahsan & Siddique, 2021).

***Accuracy*:** The accuracy reflects the total number of instances successfully identified among all instances. The following formula can be used to calculate accuracy.

$$Accuracy = \frac{T_p + T_N}{T_p + T_N + F_p + F_N} \tag{3.5}$$

Precision Precision is defined as the percentage of accurately anticipated positive observations to all expected positive observations.

$$Precision = \frac{T_p}{T_p + F_p} \tag{3.6}$$

Recall: Recall measures how many total relevant results the algorithm accurately identified.

$$Recall = \frac{T_p}{T_n + F_p} \tag{3.7}$$

F1-score: The F1-score is a metric that combines accuracy and recall in a harmonic mean, serving as a comprehensive measure of the performance of an algorithm. The ideal F1-score is 1, which signifies a perfect balance between precision and recall.

$$F1 - score = 2 \times \frac{\text{Precision} \times \text{ Recall}}{\text{Precision+Recall}} \tag{3.8}$$

Area under curve (AUC): The area under the curve (AUC) is a metric that provides insight into the behavior of a model under different conditions. The AUC can be quantified using the following formula:

$$AUC = \frac{\sum R_i(I_p) - I_p((I_p + 1)/2}{I_p + I_n} \tag{3.9}$$

Where, l_p and l_n denotes positive and negative data samples and R_i is the rating of the i^{th} positive samples.

True Positive (T_p) = Positive instances classified as Positive

False Positive (F_p) = Negative instances classified as Positive

True Negative (T_n) = Negative instances classified as Negative

False Negative (F_n) = Positive instances classified as Negative

Geometric mean (G-mean): The geometric mean (G-mean) is a performance metric for classification models that is particularly useful when the classes are imbalanced. It is calculated as the square root of the product of the sensitivity and specificity of the model (Brownlee, 2016):

$$G - mean = \sqrt{\text{sensitivity} \times \text{specificity}} \tag{3.10}$$

The G-mean provides a balanced measure of the model's ability to correctly classify both the positive and negative instances. It takes into account both the true positive rate (sensitivity) and true negative rate (specificity) and is a useful measure when the data is imbalanced, as it does not give excessive weight to the majority class.

The performance of the model was assessed using a 5-fold cross-validation technique (refer to Figure 3.1), and the model's overall performance was determined by computing the average outcomes across all 5 folds.

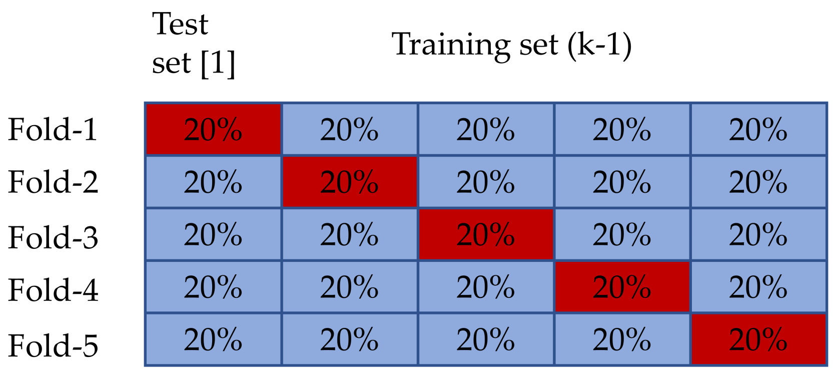


Figure 3.1: A graphical representation of the 5-fold cross-validation process for both the training and testing datasets (Ahsan, E Alam, et al., 2020).

The model performance of ML models with default parameters and hyperparameters is observed for original and SMOTE-based oversampled datasets. Table 3.2 summarizes the ML algorithms and their designated hyperparameters used in this study. The default parameters are based on the built-in function provided by Scikit learn open-source ML library.

Table 3.2: Grid search hyperparameters of ML algorithms used to conduct the experiment.

Algorithm	Grid search parameters
AB	n_estimatros: 50, 100, 150; learning_rate: 0.01, 0.1, 1
DT	criterion: 'gini'; max_depth: 10, 50; min_samples_split: 3, 4, 5; max_features: 2,5,6,8
GB	n_estimators: 50, 100, 150; learning_rate: 0.01, 0.1, 1; max_depth: 3, 4, 7
KNN	n_neighbors: 3, 5, 7, 9; weights: 'uniform', 'distance'
LR	penalty: 'l1', 'l2'; C: 0.01, 0.1, 1, 10
RF	n_estimators: 50, 100, 150; max_depth: 10, 20; min_samples_split: 3, 4, 5; max_features: 2, 5, 6, 8
SVM	C: 0.1, 1, 10; kernel: 'linear', 'rbf'; gamma: 'scale', 'auto'

The ML model performance was measured under four conditions:

- ***Condition one:*** ML performance on the original dataset – **(ML)**
- ***Condition Two:*** ML performance after oversampled SMOTE dataset –

(ML + SMOTE)

- ***Condition Three:*** Optimal ML-based performance on the original dataset – **(ML(HP))**

- ***Condition Four:*** Optimal ML performance on SMOTE-based oversampled dataset – **(ML(HP) + SMOTE)**

3.3 Computational Results

The performance has been measured in terms of accuracy, precision, recall, F1-score, ROC-AUC, and G-mean score using equations 6.1– 3.10.

3.3.1 Performance of AB

The results show that the model's performance under condition 2 (ML performance after oversampled SMOTE dataset) is the best among all conditions, with an average accuracy of 0.9356, average recall of 0.9688, average ROC-AUC of 0.975, and average G-mean of 0.9348. On the other hand, the performance of the model under condition 1 (ML performance on the original dataset) is the worst among all conditions, with an average accuracy of 0.9288, average recall of 0.9646, average ROC-AUC of 0.9388, and average G-mean of 0.9128. It can be concluded that oversampling the minority class using SMOTE might improve the performance of the AB ML model on the Ionosphere dataset. The best performance is achieved when the AB ML model is trained on the SMOTE-based oversampled dataset. The results show that this approach outperforms the other conditions regarding accuracy, recall, ROC-AUC, and G-mean.

Table 3.3: AB ML model performance evaluation on Ionosphere dataset under four different conditions; AB– Adaptive Boosting.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.873	0.957	0.914	0.943	0.957	0.9288
	Precision	0.909	0.938	0.898	0.918	0.977	0.928
	Recall	0.889	1	0.978	1	0.956	0.9646
	F1-score	0.899	0.968	0.936	0.957	0.966	0.9452
	ROC	0.92	0.936	0.884	0.961	0.993	0.9388
	G-mean	0.867	0.938	0.884	0.917	0.958	0.9128
Two	Accuracy	0.889	0.956	0.933	0.967	0.933	0.9356
	Precision	0.872	0.918	0.882	0.938	0.933	0.9086
	Recall	0.911	1	1	1	0.933	0.9688
	F1-score	0.891	0.957	0.938	0.968	0.933	0.9374
	ROC	0.935	0.988	0.985	0.985	0.982	0.975
	G-mean	0.889	0.955	0.931	0.966	0.933	0.9348
Three	Accuracy	0.873	0.957	0.914	0.943	0.957	0.9288
	Precision	0.909	0.938	0.898	0.918	0.977	0.928
	Recall	0.889	1	0.978	1	0.956	0.9646
	F1-score	0.899	0.968	0.936	0.957	0.966	0.9452

Table 3.3 ***Cont.***

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.92	0.936	0.884	0.961	0.993	0.9388
	G-mean	0.867	0.938	0.884	0.917	0.958	0.9128
Four	Accuracy	0.889	0.956	0.933	0.967	0.933	0.9356
	Precision	0.872	0.918	0.882	0.938	0.933	0.9086
	Recall	0.911	1	1	1	0.933	0.9688
	F1-score	0.891	0.957	0.938	0.968	0.933	0.9374
	ROC	0.935	0.988	0.985	0.985	0.982	0.975
	G-mean	0.889	0.955	0.931	0.966	0.933	0.9348

Similarly, the outcome of the AB was also evaluated on other referenced highly imbalanced datasets, including Pageblocks, Poker, Spambase, Winequality, and Yeast. Comprehensive results of the overall performance of each of the datasets can be found in Appendix .1.

Figure 3.2 presents the lowest and highest Receiver Operating Characteristic (ROC) scores for the AB ML algorithm. The worst performance was observed for AB on the Poker dataset (depicted in Figure 3.2 (a)) with default ML parameters. In contrast, the best performance was achieved on the oversampled Pageblocks dataset (depicted in Figure 3.2 (b)) with optimal ML parameters. The average of the five-fold lowest ROC score was 0.57, and the highest ROC score was 1.

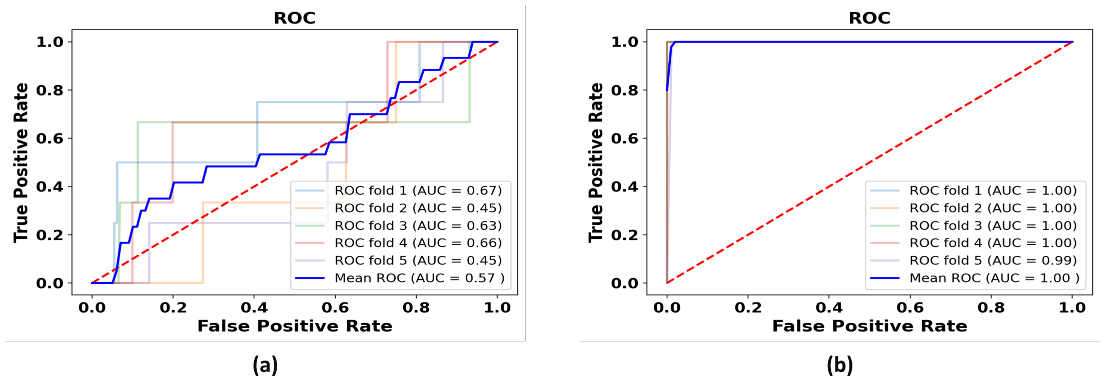


Figure 3.2: ROC score of Adaboost on (a) Poker dataset with default ML parameters and (b) oversampled Pageblocks dataset with optimal ML parameters.

3.3.2 Performance of DT

The DT ML model performance on the Pageblocks dataset was evaluated under four conditions, with results presented in Table 3.4. The highest performance of the DT ML model on the Pageblocks dataset was achieved under Conditions Two and Four, with an average accuracy of 0.9978, average precision of 0.9956, average recall of 1, average F1-score of 0.9978, average ROC of 1, and average G-mean of 0.9978. The lowest performance was observed under Condition Three, with an average accuracy of 0.9892, average precision of 0.9428, average recall of 0.7, average F1-score of 0.817, average ROC of 0.91, and average G-mean of 0.9596.

Table 3.4: DT ML model performance evaluation on Pageblocks dataset under four different conditions; DT–Decision Tree.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.968	0.989	0.989	1	0.979	0.985
	Precision	1	0.833	0.833	1	0.75	0.8832
	Recall	0.5	1	1	1	1	0.9
	F1-score	0.667	0.909	0.909	1	0.857	0.8684
	ROC	0.75	0.994	0.994	1	0.989	0.9454
	G-mean	0.707	0.994	0.994	1	0.989	0.9368
Two	Accuracy	1	1	1	1	0.989	0.9978
	Precision	1	1	1	1	0.978	0.9956
	Recall	1	1	1	1	1	1
	F1-score	1	1	1	1	0.989	0.9978
	ROC	1	1	1	1	0.989	0.9978
	G-mean	1	1	1	1	0.989	0.9978
Three	Accuracy	0.968	1	0.989	1	0.989	0.9892
	Precision	1	1	0.714	1	1	0.9428
	Recall	0.333	1	1	0.5	0.667	0.7
	F1-score	0.667	0.909	0.909	1	0.6	0.817

Table 3.4 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.75	0.994	0.889	0.917	1	0.91
	G-mean	0.816	0.994	0.994	1	0.994	0.9596
Four	Accuracy	1	1	1	1	0.989	0.9978
	Precision	1	1	1	1	0.989	0.9978
	Recall	1	0.989	1	1	1	0.9978
	F1-score	1	1	1	1	0.989	0.9978
	ROC	1	1	1	1	1	1
	G-mean	0.994	1	1	1	0.989	0.9966

Figure 3.3 presents the lowest and highest ROC scores for the DT ML algorithm. The worst performance was observed for DT on the Poker dataset (depicted in Figure 3.3 (a)) with default ML parameters. In contrast, the best performance was achieved on the oversampled Pageblocks dataset (depicted in Figure 3.3 (b)) with optimal ML parameters. The average of the five-fold lowest ROC score was 0.5, and the highest ROC score was 1.

3.3.3 Performance of GB

In the GB ML performance evaluation on the Poker dataset, conditions Two and Four showed the highest average performance with accuracy, precision, recall, F1-score, ROC, and G-mean scores of 0.9978. On the other hand, condition

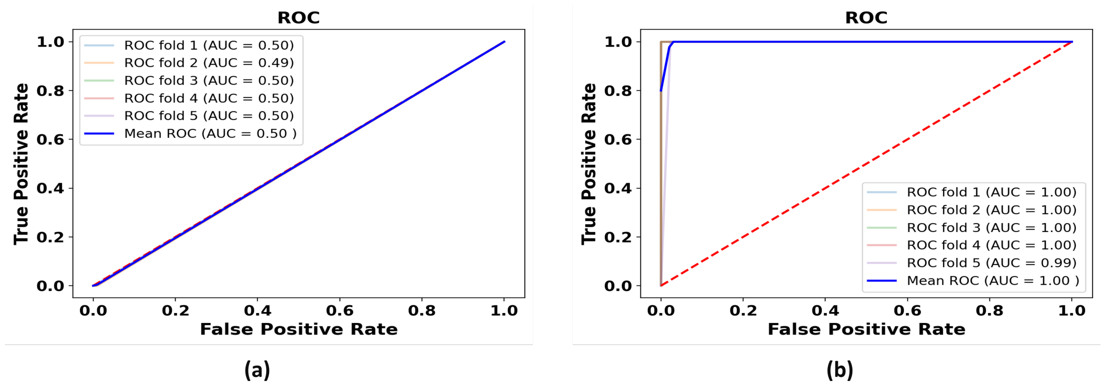


Figure 3.3: ROC score of DT on (a) Poker dataset with default ML parameters and (b) oversampled Pageblocks dataset with optimal ML parameters.

One showed the lowest average performance with accuracy, precision, recall, F1-score, ROC, and G-mean scores of 0.1732, 0.2, 0.15, 0.1714, 0.8164, and 0.1732, respectively. Condition Three showed an average performance with accuracy, precision, recall, F1-score, ROC, and G-mean scores of 0.9952, 0.6, 0.6, 0.7334, 0.989, and 0.7732, respectively.

Table 3.5: GB ML model performance evaluation on Poker dataset under four different conditions; GB–Gradient Boosting.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	Accuracy	0.986	0.99	0.99	0.99	0.997	0.9906
	Precision	0	0	0	0	1	0.2
One	Recall	0	0	0	0	0.75	0.15

Table 3.5 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	F1-score	0	0	0	0	0.857	0.1714
	ROC	0.805	0.445	0.837	0.995	1	0.8164
	G-mean	0	0	0	0	0.866	0.1732
Two	Accuracy	0.995	0.997	0.997	1	1	0.9978
	Precision	1	1	1	1	1	1
	Recall	0.99	0.993	0.993	1	1	0.9952
	F1-score	0.995	0.997	0.997	1	1	0.9978
	ROC	0.997	1	1	1	1	0.9994
	G-mean	0.995	0.997	0.997	1	1	0.9978
Three	Accuracy	0.986	0.99	1	1	1	0.9952
	Precision	0	0	1	1	1	0.6
	Recall	0	0	1	1	1	0.6
	F1-score	0.667	0	1	1	1	0.7334
	ROC	0.988	0.957	1	1	1	0.989
	G-mean	0.866	0	1	1	1	0.7732
Four	Accuracy	0.995	0.995	1	1	1	0.998
	Precision	1	1	1	1	1	1
	Recall	0.99	0.99	1	1	1	0.996
	F1-score	0.995	0.995	1	1	1	0.998

Table 3.5 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.998	1	1	1	1	0.9996
	G-mean	0.995	0.993	1	1	1	0.9976

Figure 3.4 presents the lowest and ROC scores for the GB ML algorithm. The worst performance was observed for GB on the oversampled Wine Quality dataset (depicted in Figure 3.4 (a)) with default ML parameters. In contrast, the best performance was achieved on the oversampled Yeast dataset (depicted in Figure 3.4 (b)) with optimal ML parameters. The average of the five-fold lowest ROC score was 0.70, and the highest ROC score was 1.

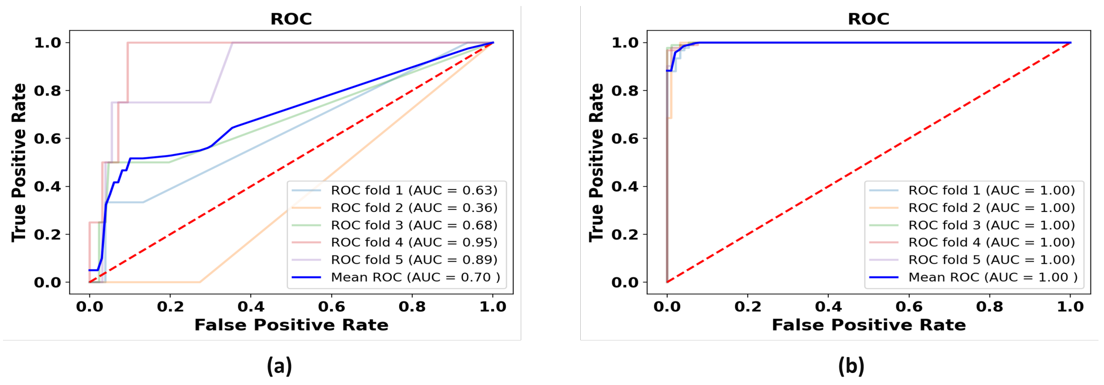


Figure 3.4: ROC score of GB on (a) oversampled Wine Quality dataset with default ML parameters and (b) oversampled Yeast dataset with optimal ML parameters.

3.3.4 Performance of KNN

Table 3.6 summarizes the performance evaluation of the KNN ML model on the Spambase dataset under four different conditions. The results indicate that the performance of the model varied under different conditions. The best average performance was achieved under condition Three, with an average accuracy of 0.923, an average precision of 0.9138, an average recall of 0.889, an average F1-score of 0.901, an average ROC of 0.9658, and an average G-mean of 0.9164. The worst average performance was observed under condition Two, with an average accuracy of 0.914, an average precision of 0.8978, an average recall of 0.9342, an average F1-score of 0.9158, an average ROC of 0.9624, and an average G-mean of 0.9136. Overall, the results suggest that the KNN model's performance is sensitive to the specific conditions and evaluation metrics.

Table 3.6: KNN ML model performance evaluation on Spambase dataset under four different conditions; KNN–K-Nearest Neighbors.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	Accuracy	0.912	0.897	0.911	0.929	0.897	0.9092
	Precision	0.915	0.866	0.927	0.914	0.878	0.9
	Recall	0.857	0.873	0.84	0.906	0.856	0.8664
One	F1-score	0.885	0.87	0.881	0.91	0.867	0.8826

Table 3.6 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.951	0.953	0.955	0.964	0.94	0.9526
	G-mean	0.901	0.892	0.896	0.925	0.889	0.9006
Two	Accuracy	0.893	0.924	0.919	0.917	0.917	0.914
	Precision	0.878	0.914	0.909	0.895	0.893	0.8978
	Recall	0.914	0.935	0.932	0.944	0.946	0.9342
	F1-score	0.896	0.925	0.92	0.919	0.919	0.9158
	ROC	0.95	0.967	0.964	0.968	0.963	0.9624
	G-mean	0.893	0.924	0.919	0.916	0.916	0.9136
Three	Accuracy	0.926	0.913	0.924	0.938	0.914	0.923
	Precision	0.925	0.888	0.935	0.92	0.901	0.9138
	Recall	0.884	0.893	0.867	0.923	0.878	0.889
	F1-score	0.904	0.89	0.9	0.921	0.89	0.901
	ROC	0.963	0.965	0.968	0.976	0.957	0.9658
	G-mean	0.918	0.909	0.913	0.935	0.907	0.9164
Four	Accuracy	0.911	0.943	0.927	0.935	0.922	0.9276
	Precision	0.887	0.922	0.909	0.909	0.894	0.9042
	Recall	0.943	0.968	0.95	0.966	0.957	0.9568
	F1-score	0.914	0.944	0.929	0.936	0.925	0.9296
	ROC	0.965	0.975	0.977	0.978	0.974	0.9738

Table 3.6 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.911	0.942	0.927	0.934	0.921	0.927

Figure 3.5 presents the lowest and highest ROC scores for the KNN ML algorithm. The worst performance was observed for KNN on the Wine Quality dataset (depicted in Figure 3.5 (a)) with default ML parameters. In contrast, the best performance was achieved on the oversampled Pageblocks dataset (depicted in Figure 3.5 (b)) with optimal ML parameters. The average of the five-fold lowest ROC score was 0.66, and the highest ROC score was 1.

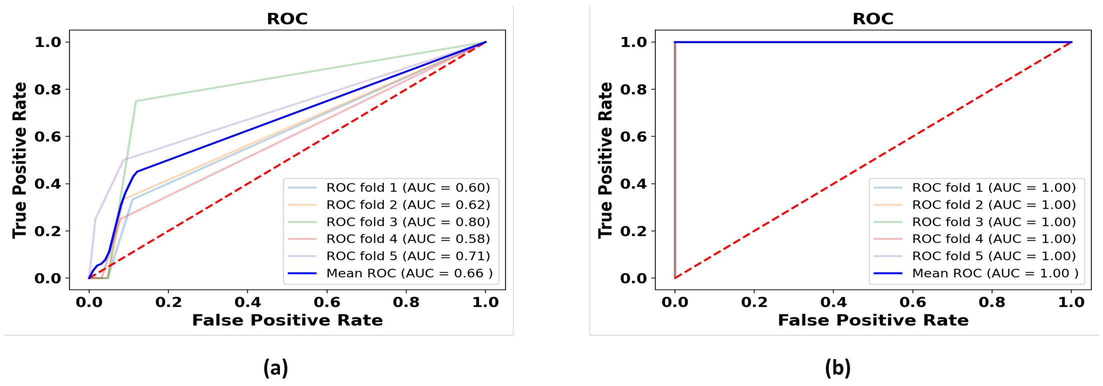


Figure 3.5: ROC score of KNN on (a) Wine Quality dataset with default ML parameters and (b) oversampled Pageblocks dataset with optimal ML parameters.

3.3.5 Performance of LR

The LR ML model performance on the Wine Quality dataset was evaluated under four different conditions, with the results summarized in Table 3.7. The best performance was achieved under conditions Four, with an average accuracy of 0.9018, an average precision of 0.8362, an average recall of 1, an average F1-score of 0.9106, an average ROC of 0.9346, and an average G-mean of 0.8966. The worst performance was observed under conditions One and Three, with an average accuracy of 0.9662 and 0.9722, respectively, and all other evaluation metrics having a value of 0.

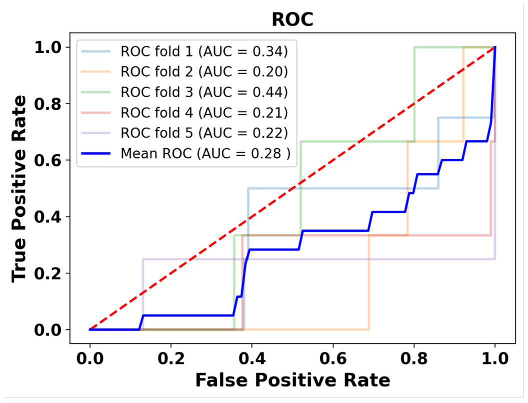
Table 3.7: LR ML model performance evaluation on Winequality dataset under four different conditions; LR–Logistic Regression.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.977	0.969	0.962	0.954	0.969	0.9662
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.862	0.794	0.864	0.974	0.874	0.8736
	G-mean	0	0	0	0	0	0
Two	Accuracy	0.933	0.933	0.906	0.925	0.941	0.9276
	Precision	0.893	0.882	0.847	0.871	0.9	0.8786
	Recall	0.984	1	0.992	1	0.992	0.9936
	F1-score	0.936	0.937	0.914	0.931	0.944	0.9324
	ROC	0.95	0.945	0.907	0.93	0.96	0.9384

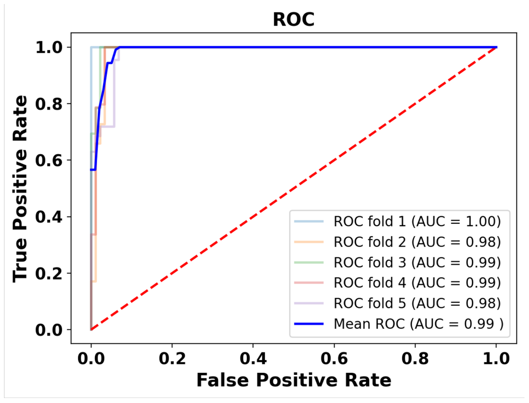
Table 3.7 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.932	0.931	0.901	0.922	0.94	0.9252
Three	Accuracy	0.977	0.977	0.969	0.969	0.969	0.9722
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.812	0.76	0.844	0.967	0.902	0.857
	G-mean	0	0	0	0	0	0
Four	Accuracy	0.898	0.914	0.89	0.898	0.909	0.9018
	Precision	0.83	0.852	0.821	0.831	0.847	0.8362
	Recall	1	1	1	1	1	1
	F1-score	0.907	0.92	0.901	0.908	0.917	0.9106
	ROC	0.931	0.951	0.914	0.925	0.952	0.9346
	G-mean	0.893	0.91	0.883	0.892	0.905	0.8966

Figure 3.6 presents the lowest and highest ROC scores for the LR ML algorithm. The worst performance was observed for LR on the Poker dataset (depicted in Figure 3.6 (a)) with optimal ML parameters. In contrast, the best performance was achieved on the oversampled Pageblocks dataset (depicted in Figure 3.6 (b)) with optimal ML parameters. The average of the five-fold lowest ROC score was 0.28, and the highest ROC score was 0.99.



(a)



(b)

Figure 3.6: ROC score of LR on (a) Poker dataset and (b) oversampled Pageblocks dataset with optimal ML parameters.

3.3.6 Performance of RF

The RF ML model was evaluated on the Yeast dataset under four different conditions. The best performance was seen in conditions Two and Four, where the average accuracy was 0.9784 and the average F1-score was 0.9798 and 0.9776, respectively. The worst performance was seen in conditions One and Three, where the average accuracy was 0.9572 and 0.9552, and the average F1-score was 0.7638 and 0.7638, respectively. The results indicate that conditions Two and Four were better performers, with high accuracy and F1-score, while conditions One and Three showed lower accuracy and F1-score.

Table 3.8: RF ML model performance evaluation on Yeast dataset under four different conditions; RF–Random Forest.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.981	0.922	0.981	0.941	0.961	0.9572
	Precision	0.833	0.625	0.833	0.714	0.875	0.776
	Recall	1	0.5	0.909	0.5	0.7	0.7218
	F1-score	0.909	0.556	0.87	0.706	0.778	0.7638
	ROC	0.996	0.957	0.99	0.978	0.978	0.9798
	G-mean	0.989	0.696	0.9	0.703	0.832	0.824
Two	Accuracy	0.973	0.973	0.984	0.978	0.984	0.9784
	Precision	0.978	0.948	1	0.958	0.979	0.9726
	Recall	0.967	1	0.989	0.978	0.989	0.9846
	F1-score	0.973	0.974	0.984	0.979	0.989	0.9798
	ROC	0.998	1	0.999	0.999	0.999	0.999
	G-mean	0.973	0.973	0.989	0.978	0.978	0.9782
Three	Accuracy	0.981	0.932	0.961	0.941	0.961	0.9552
	Precision	0.833	0.625	1	0.75	0.875	0.8166
	Recall	1	0.5	0.909	0.7	0.7	0.7618
	F1-score	0.909	0.556	0.87	0.706	0.778	0.7638

Table 3.8 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.998	0.956	0.992	0.979	0.979	0.9808
	G-mean	0.989	0.696	0.948	0.775	0.832	0.848
Four	Accuracy	0.968	0.973	0.989	0.978	0.984	0.9784
	Precision	0.978	0.939	0.989	0.979	0.978	0.9726
	Recall	0.967	1	0.989	0.989	0.989	0.9868
	F1-score	0.967	0.974	0.984	0.974	0.989	0.9776
	ROC	0.998	0.999	0.998	0.999	0.998	0.9984
	G-mean	0.967	0.973	0.989	0.967	0.989	0.977

Figure 3.7 presents the lowest and highest ROC scores for the RF ML algorithm. The worst performance was observed for RF on the Winequality dataset (depicted in Figure 3.7 (a)) with default ML parameters. In contrast, the best performance was achieved on the oversampled Yeast dataset (depicted in Figure 3.7 (b)) with optimal ML parameters. The average of the five-fold lowest ROC score was 0.84, and the highest ROC score was 1.

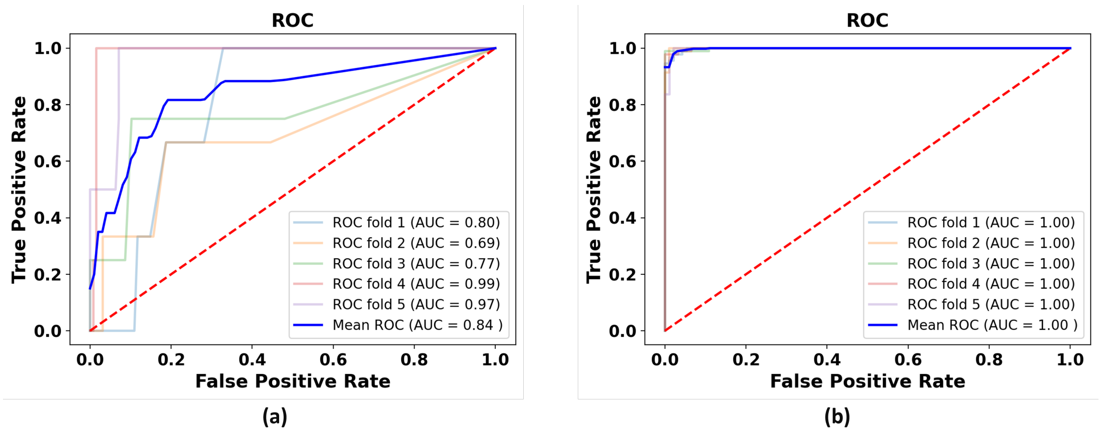


Figure 3.7: ROC score of RF on (a) Winequality dataset with default ML parameters and (b) oversampled Yeast dataset with optimal ML parameters.

3.3.7 Performance of SVM

The SVM model performed well on the Ionosphere dataset under four different conditions. The average accuracy across the five folds was 0.9512. The highest accuracy was achieved under condition two, with an average accuracy of 0.9512. Precision was also high under condition two, with an average precision of 0.9326. Condition two also showed the highest average recall of 0.9734. The average F1-score across all conditions was 0.9524. The highest average F1-score was achieved under condition two, with an average F1-score of 0.9524. The average ROC score was 0.9856, and the average G-mean score was 0.951.

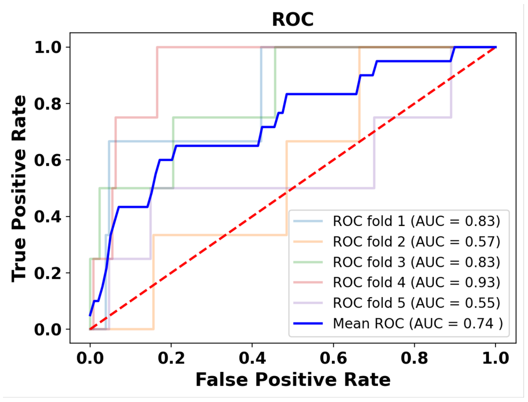
Table 3.9: SVM ML model performance evaluation on Ionosphere dataset under four different conditions; SVM– Support Vector Machine.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.93	0.943	0.886	0.957	0.986	0.9404
	Precision	0.955	0.918	0.863	0.938	0.978	0.9304
	Recall	0.933	1	0.978	1	1	0.9822
	F1-score	0.944	0.957	0.917	0.968	0.989	0.955
	ROC	0.984	0.998	0.96	0.965	0.996	0.9806
	G-mean	0.928	0.917	0.839	0.938	0.98	0.9204
Two	Accuracy	0.878	0.989	0.944	0.978	0.967	0.9512
	Precision	0.854	0.978	0.917	0.957	0.957	0.9326
	Recall	0.911	1	0.978	1	0.978	0.9734
	F1-score	0.882	0.989	0.946	0.978	0.967	0.9524
	ROC	0.969	0.997	0.978	0.986	0.998	0.9856
	G-mean	0.877	0.989	0.944	0.978	0.967	0.951
Three	Accuracy	0.93	0.943	0.886	0.957	0.986	0.9404
	Precision	0.955	0.918	0.863	0.938	0.978	0.9304
	Recall	0.933	1	0.978	1	1	0.9822
	F1-score	0.944	0.957	0.917	0.968	0.989	0.955

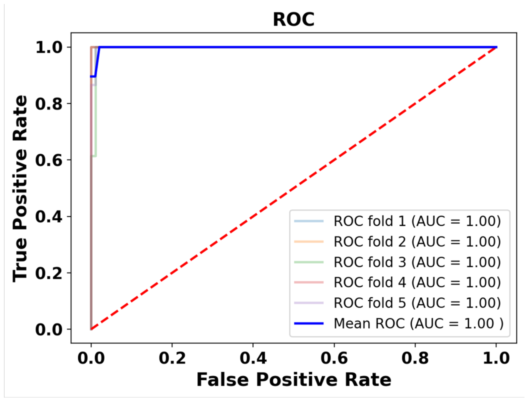
Table 3.9 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.984	0.998	0.96	0.965	0.996	0.9806
	G-mean	0.928	0.917	0.839	0.938	0.98	0.9204
Four	Accuracy	0.878	0.989	0.944	0.978	0.967	0.9512
	Precision	0.854	0.978	0.917	0.957	0.957	0.9326
	Recall	0.911	1	0.978	1	0.978	0.9734
	F1-score	0.882	0.989	0.946	0.978	0.967	0.9524
	ROC	0.969	0.997	0.978	0.986	0.998	0.9856
	G-mean	0.877	0.989	0.944	0.978	0.967	0.951

Figure 3.8 presents the lowest and highest ROC scores for the SVM ML algorithm. The worst performance was observed for SVM on the Wine Quality dataset (depicted in Figure 3.8 (a)) with default ML parameters. In contrast, the best performance was achieved on the oversampled Pageblocks dataset (depicted in Figure 3.8 (b)) with optimal ML parameters. The average of the five-fold lowest ROC score was 0.74, and the highest ROC score was 1.



(a)



(b)

Figure 3.8: ROC score of SVM on (a) Wine Quality dataset with default ML parameters and (b) oversampled Pageblocks dataset with optimal ML parameters.

3.4 Discussion of the Results

The study evaluated the performance of various ML algorithms on six different datasets and found that the accuracy, precision, recall, F1-score, ROC, and G-mean scores varied depending on the specific dataset and conditions used in the experiments. The study observed that in some cases, precision, recall, F1-score, and G-mean score became almost zero for some folds, indicating that the model was overly biased. However, the ROC score was still able to be calculated, and the study concluded that selecting an appropriate evaluation metric and model conditions is crucial for achieving better performance.

The study also demonstrated the importance of selecting the appropriate evaluation metric, mainly when working with highly imbalanced datasets. The ROC score was found to be a suitable metric for hypothesis testing, particularly when comparing the performance of different ML models before and after SMOTE-based oversampling.

Possible directions for future work include exploring other evaluation metrics or optimizing the model conditions to improve the overall performance of the ML algorithms. In Section 3.5, the study will consider the ROC score of each fold to conduct hypothesis testing to evaluate significant performance improvements of popular ML models regarding before and after SMOTE-based oversampling.

3.5 Hypothesis Testing

This study aims to investigate the impact of the Synthetic Minority Over-sampling Technique (SMOTE) on data balancing in imbalanced benchmark datasets. Hypothesis testing is used to evaluate the significance of SMOTE-based approaches on data balancing. Four conditions are considered for this purpose:

- The performance of a ML model on the original dataset,
- The performance of an ML model on the SMOTE-based oversampled dataset,
- The performance of a hyperparameter tuned ML model on the original dataset, and
- The performance of a hyperparameter tuned ML model on the SMOTE-based oversampled dataset.

The six hypotheses that has been developed based on these four conditions are as follows:

Hypothesis 1:

- H_0: There is no significant difference in the performance of the ML model with hyperparameter tuning on the original dataset and the SMOTE-based oversampled dataset.

- H_1: There is a significant difference in the performance of the ML model on the original dataset and the SMOTE-based oversampled dataset.

Hypothesis 2:

- H_0: There is no significant difference in the performance of the ML model with hyperparameter tuning on the original dataset and the SMOTE-based oversampled dataset.

- H_1: There is a significant difference in the performance of the ML model with hyperparameter tuning on the original dataset and the SMOTE-based oversampled dataset.

Hypothesis 3:

- H_0: There is no significant difference in the performance of the ML model with and without hyperparameter tuning on the original dataset.

- H_1: There is a significant difference in the performance of the ML model with and without hyperparameter tuning on the original dataset.

Hypothesis 4:

- H_0: There is no significant difference in the performance of the ML model with and without hyperparameter tuning on the SMOTE-based oversampled dataset.

- H_1: There is a significant difference in the performance of the ML model with and without hyperparameter tuning on the SMOTE-based oversampled dataset.

Hypothesis 5:

- H_0: There is no significant difference in the performance of the ML model on the original dataset and the original dataset with hyperparameter tuning. oversampled dataset.

- H_1: There is a significant difference in the performance of the ML model on the original dataset and the original dataset with hyperparameter tuning.

Hypothesis 6:

- H_0: There is no significant difference in the performance of the ML model on the SMOTE-based oversampled dataset and the SMOTE-based oversampled dataset with hyperparameter tuning. oversampled dataset.

- H_1: There is a significant difference in the performance of the ML model on the SMOTE-based oversampled dataset and the SMOTE-based oversampled dataset with hyperparameter tuning.

3.5.1 Performance of AB

The hypothesis testing results for the AB Algorithm on the Ionosphere dataset are presented in Table 3.10. The testing aimed to evaluate the effect of applying the SMOTE on the performance of the AB algorithm. The conditions compared in the hypothesis testing were ML, ML+SMOTE, ML(HP), and ML(HP)+SMOTE, based on their mean and Standard Deviation (SD).

A paired t-test has been conducted to compare the conditions, and the significance level was set at 0.05. The results showed that the mean difference between ML and ML+SMOTE and between ML(HP) and ML(HP)+SMOTE was -0.0362, with a standard deviation difference of 0.016774. The t-value was -2.3246, and the p-value was 0.06766, which was greater than 0.05. This result failed to reject the null hypothesis, indicating that there was no significant difference between the two conditions.

Similarly, the results showed that the mean difference between ML and ML(HP) and between ML+SMOTE and ML(HP)+SMOTE was 0, with a standard deviation difference of 0. The t-value was not computable (nan), and the p-value was also not computable (nan), failing to reject the null hypothesis.

In conclusion, the hypothesis testing results indicated that there was no statistically significant impact on the performance of the AB algorithm on the Ionosphere dataset when SMOTE was applied to balance the dataset. These results suggest that the use of SMOTE does not have a significant effect on the AB algorithm's performance.

Table 3.10: Results of hypothesis testing for AB algorithms on Ionosphere dataset. ML– Machine Learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9388	0.036864	ML	ML + SMOTE	-0.0362	0.016774	-2.32466	0.06766	Fail to reject H0
ML + SMOTE	0.975	0.02009	ML	ML(HP)	0	0			Fail to reject H0
ML(HP)	0.9388	0.036864	ML	ML(HP) + SMOTE	-0.0362	0.016774	-2.32466	0.06766	Fail to reject H0
ML(HP) + SMOTE	0.975	0.02009	ML + SMOTE	ML(HP)	0.0362	-0.01677	2.324656	0.06766	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0	0	nan	nan	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0362	0.016774	-2.32466	0.06766	Fail to reject H0

Similarly, the performance of the AB algorithm was also evaluated on other referenced highly imbalanced datasets, including Pageblocks, Poker, Spambase, Wine Quality, and Yeast. Comprehensive results of the overall performance of each of the datasets can be found in Appendix .3.

Table 3.11 presents the overall hypothesis testing outcome of AB algorithms on six different benchmark imbalanced datasets. For all six datasets, the null hypotheses were not rejected, indicating that the performance of AB algorithms did not significantly differ from the baseline performance. However, for the Wine

Quality dataset, the null hypothesis was rejected for hypothesis five, indicating that the performance of AB algorithms significantly differed from the baseline performance.

The overall outcome of the hypothesis testing suggests that AB algorithms did not significantly improve the performance of the classifiers in the six benchmark imbalanced datasets. However, further investigations are needed to determine the potential reasons for this outcome and to identify the scenarios where AB algorithms might be effective in improving classifier performance in imbalanced datasets with or without oversampling.

Table 3.11: Overall hypothesis outcome of AB algorithms on referenced six benchmarks imbalanced dataset used in this study.

Dataset	Hypothesis					
	One	Two	Three	Four	Five	Six
Ionosphere	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Pageblocks	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Poker	Reject H_0	Reject H_0	Reject H_0	Reject H_0	Reject H_0	Reject H_0
Spambase	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Winequality	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Reject H_0	Fail to reject H_0
Yeast	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0

3.5.2 Performance of DT

Table 3.12 shows the results of hypothesis testing for DT algorithms on the Pageblocks dataset. The table reports the mean ROC, standard deviation, mean difference, standard difference, t-value, and p-value for each pair of algorithm conditions. For all pairs of conditions, the null hypothesis was not rejected, indicating that there was no statistically significant difference between the performance of the decision tree algorithms. Specifically, the mean ROC of the ML algorithm was 0.9454 with a standard deviation of 0.0978, and the mean ROC of the ML+SMOTE algorithm was 0.893 with a SD of 0.0934. The mean ROC of the ML(HP) algorithm was 0.9634 with a standard deviation of 0.0069, and the mean ROC of the ML(HP)+SMOTE algorithm was 0.9122 with a standard deviation of 0.0864. These results suggest that the decision tree algorithms performed similarly on the Pageblocks dataset, regardless of the algorithm conditions tested.

Table 3.12: Results of hypothesis testing for DT algorithms on Pageblocks dataset. ML– Machine Learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9454	0.097762	ML	ML + SMOTE	-0.0524	0.093362	-1.29864	0.250731	Fail to reject H0
ML + SMOTE	0.9978	0.0044	ML	ML(HP)	0.0354	0.006937	1.787271	0.133935	Fail to reject H0
ML(HP)	0.91	0.090825	ML	ML(HP) + SMOTE	-0.0546	0.097762	-1.36804	0.229576	Fail to reject H0
ML(HP) + SMOTE	1	0	ML + SMOTE	ML(HP)	0.0878	-0.08643	2.310452	0.068868	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.0022	0.0044	-1.22474	0.27522	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.09	0.090825	-2.42724	0.059583	Fail to reject H0

Table 3.13 presents the overall hypothesis testing outcome of DT algorithms on six different benchmark imbalanced datasets. The study failed to reject H0

in all six hypotheses for the Ionosphere and Pageblocks datasets, indicating that oversampling approaches did not significantly affect the DT algorithm's performance. For the Spambase and Yeast datasets, the algorithm's performance was not significantly impacted for all six hypotheses, as the study failed to reject H0 in all cases.

In the case of the Poker dataset, the study rejected H0 for hypotheses One, Three, Four, Five, and Six, indicating that the oversampled data significantly impacted the performance of the DT algorithm. For the Wine Quality dataset, the study rejected H0 for hypotheses One, Three, Four, and Six, suggesting that the oversampled data in the dataset had a significant impact on the algorithm's performance in those cases.

Table 3.13: Overall hypothesis outcome of DT algorithms on referenced six benchmarks imbalanced dataset used in this study.

Dataset	Hypothesis					
	One	Two	Three	Four	Five	Six
Ionosphere	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Pageblocks	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Poker	Reject H_0	Fail to reject H_0	Reject H_0	Reject H_0	Reject H_0	Reject H_0
Spambase	Reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Winequality	Reject H_0	Fail to reject H_0	Reject H_0	Reject H_0	Fail to reject H_0	Reject H_0
Yeast	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0

Similarly, the performance of the DT algorithm was also evaluated on other referenced highly imbalanced datasets, including Ionosphere, Poker, Spambase, Wine Quality, and Yeast. Comprehensive results of the overall performance of each of the datasets can be found in Appendix .3.

3.5.3 Performance of GB

Table 3.14 displays the results of hypothesis testing for GB algorithms on the Poker dataset. The table reports the mean ROC results, SD, mean difference, standard difference, t-value, and p-value for each pair of algorithm conditions. After conducting the hypothesis testing for all pairs of conditions, it was observed that the null hypothesis cannot be rejected with a significance level of 0.05, as the p-values were greater than 0.05. Specifically, the p-values for all conditions were 0.018284, 0.071443, 0.009755, 0.014172, and 0.049007. Since all these p-values are greater than the significance level of 0.005, we can conclude that there is no statistically significant difference in performance between the GB algorithms in the given conditions. This result suggests that these algorithms perform similarly, and one algorithm does not outperform the other significantly. After conducting the hypothesis testing for all pairs of conditions, we found that the null hypothesis cannot be rejected with a significance level of 0.05, as the p-values were greater than 0.05. Specifically, the p-values for all conditions were 0.018284, 0.071443, 0.009755, 0.014172, and 0.049007. Since all these p-values are greater than the significance level of 0.005, we can conclude that there is no statistically significant difference in performance between the GB algorithms in the given conditions. This result suggests that these algorithms perform similarly, and one algorithm does not outperform the other significantly. Specifically, the mean ROC result for the ML algorithm was 0.8746 with a SD of 0.0598, and the mean ROC result for the ML+SMOTE algorithm was 0.783 with a SD of 0.0472. The mean ROC result for the ML(HP) algorithm was 0.8382 with a SD of 0.0752, and the mean ROC result for the ML(HP)+SMOTE algorithm was 0.977 with a standard deviation of 0.0055. These findings suggest that the GB algorithms performed similarly on the Poker dataset, regardless of the algorithm conditions tested.

Table 3.14: Results of hypothesis testing for GB algorithms on Poker dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.8746	0.059758	ML	ML + SMOTE	-0.0916	0.047214	-3.44764	0.018284	Fail to reject H0
ML + SMOTE	0.9662	0.012544	ML	ML(HP)	0.0364	-0.01543	2.281067	0.071443	Fail to reject H0
ML(HP)	0.8382	0.075184	ML	ML(HP) + SMOTE	-0.1024	0.054245	-4.05732	0.009755	Fail to reject H0
ML(HP) + SMOTE	0.977	0.005514	ML + SMOTE	ML(HP)	0.128	-0.06264	3.688184	0.014172	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.0108	0.007031	-2.58712	0.049007	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.1388	0.06967	-4.32801	0.007513	Fail to reject H0

Table 3.15 presents the overall hypothesis testing outcome of GB algorithms on six different benchmark imbalanced datasets. For the Ionosphere, Pageblocks, Poker, and Yeast datasets, all six hypotheses failed to reject the null hypothesis, indicating that there was no statistically significant difference between the performance of the GB algorithms on before and after oversampled data. For

the Spambase dataset, the GB algorithm performed significantly better according to the second and fifth hypotheses, but no significant difference was observed according to the other four hypotheses. For the Wine Quality dataset, the GB algorithm performed significantly better according to the fifth hypothesis, but no significant difference was observed according to the other five hypotheses. Overall, the results indicate that the performance of the GB algorithm was not significantly improved for most datasets, with a few exceptions.

Table 3.15: Overall hypothesis outcome of GB algorithms on referenced six benchmarks imbalanced dataset used in this study.

Dataset	Hypothesis					
	One	Two	Three	Four	Five	Six
Ionosphere	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Pageblocks	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Poker	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Spambase	Fail to reject H_0	Reject H_0	Fail to reject H_0	Fail to reject H_0	Reject H_0	Fail to reject H_0
Winequality	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Reject H_0	Fail to reject H_0
Yeast	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0

Similarly, the performance of the GB algorithm was also evaluated on other referenced highly imbalanced datasets, including Ionosphere, Pageblocks, Spambase, Wine Quality, and Yeast. Comprehensive results of the overall performance of each of the datasets can be found in Appendix .3.

3.5.4 Performance of KNN

Table 3.16 shows the results of hypothesis testing for KNN algorithms on the Spambase dataset. The table presents the mean, SD, mean difference, standard difference, t-value, and p-value for each pair of algorithm conditions. The null hypothesis for each test was that there was no significant difference in performance between the KNN algorithms. The p-values for the pairwise comparisons were used to determine whether to reject the null hypothesis. For the ML algorithm, the mean ROC was 0.9526, with a standard deviation of 0.00771. The mean AUC for the ML+SMOTE algorithm was 0.9428, with a standard deviation of 0.006468. The mean AUC for the ML(HP) algorithm was 0.9658, with a standard deviation of 0.006242, and the mean AUC for the ML(HP)+SMOTE algorithm was 0.9738, with a standard deviation of 0.004622.

The hypothesis testing results indicate that the null hypothesis cannot be rejected for the comparisons of ML vs. ML+SMOTE, ML(HP)+SMOTE vs. ML(HP), and ML+SMOTE vs. ML(HP)+SMOTE. The p-values for these comparisons were 0.033825, 0.017546, and 0.27522, respectively, greater than the significance level of 0.05. Therefore, there is no statistically significant difference in performance between these algorithm conditions. However, the null hypothesis was rejected for comparing ML+SMOTE vs. ML(HP) and ML(HP)+SMOTE vs. ML+SMOTE. The p-values for these comparisons were 1.42E-05 and 8.52E-05, respectively, smaller than the significance level of 0.05.

Table 3.16: Results of hypothesis testing for KNN algorithms on Spambase dataset. ML– Machine Learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9526	0.00771	ML	ML + SMOTE	-0.0098	0.001241	-2.89912	0.033825	Fail to reject H0
ML + SMOTE	0.9624	0.006468	ML	ML(HP)	-0.0132	0.001468	-16.6746	1.42E-05	Reject H0
ML(HP)	0.9658	0.006242	ML	ML(HP) + SMOTE	-0.0212	0.003088	-7.08242	0.000869	Reject H0
ML(HP) + SMOTE	0.9738	0.004622	ML + SMOTE	ML(HP)	-0.0034	0.000227	-1.22474	0.27522	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.0114	0.001847	-11.5551	8.52E-05	Reject H0
			ML(HP)	ML(HP) + SMOTE	-0.008	0.00162	-3.48596	0.017546	Fail to reject H0

Table 3.17 summarizes the overall hypothesis outcome of the KNN algorithms on the referenced six imbalanced benchmark datasets. The table indicates that the null hypothesis is rejected for most cases, indicating that oversampling significantly impacts the performance of KNN algorithms. Specifically, the null hypothesis is rejected for all six hypotheses for the Ionosphere, Poker, Spambase, and Winequality datasets. For the Pageblocks and Yeast datasets, the null hypothesis is rejected for some of the hypotheses, indicating that the oversampled dataset significantly positively impacts the KNN algorithm's performance for some cases but not for

others.

Table 3.17: Overall hypothesis outcome of KNN algorithms on referenced six benchmarks imbalanced dataset used in this study.

Dataset	Hypothesis					
	One	Two	Three	Four	Five	Six
Ionosphere	Reject H_0	Fail to reject H_0	Fail to reject H_0	Reject H_0	Fail to reject H_0	Reject H_0
Pageblocks	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Poker	Reject H_0	Fail to reject H_0	Reject H_0	Reject H_0	Fail to reject H_0	Reject H_0
Spambase	Fail to reject H_0	Reject H_0	Reject H_0	Fail to reject H_0	Reject H_0	Fail to reject H_0
Winequality	Reject H_0	Reject H_0	Reject H_0	Reject H_0	Fail to reject H_0	Reject H_0
Yeast	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0

Similarly, the performance of the Adaboost algorithm was also evaluated on other referenced highly imbalanced datasets, including Ionosphere, Pageblocks, Poker, Winequality, and Yeast. Comprehensive results of the overall performance of each of the datasets can be found in Appendix .3.

3.5.5 Performance of LR

Table C72 presents the results of hypothesis testing for LR algorithms on the Wine Quality dataset. The table displays the mean ROC results, standard deviation, mean difference, standard difference, t-value, and p-value for each pair of algorithm conditions. The null hypothesis was failed to rejected for all pairs of conditions, with p-values greater than 0.05. Specifically, the p-values were 0.056874, 0.185516, 0.07304, 0.046798, and 0.377733. As all these p-values are higher than the significance level of 0.005, we can conclude that there is no statistically significant difference in performance between the LR algorithms in the given conditions. This finding suggests that these algorithms perform similarly, and one algorithm does not outperform the other significantly.

Regarding the algorithm conditions tested, the mean ROC result for the ML algorithm was 0.8736 with a standard deviation of 0.0577, and the mean ROC result for the ML+SMOTE algorithm was 0.8088 with a standard deviation of 0.0392. The mean ROC result for the ML(HP) algorithm was 0.857 with a standard deviation of 0.0717, and the mean ROC result for the ML(HP)+SMOTE algorithm was 0.9346 with a standard deviation of 0.0148. These findings suggest that the LR algorithms performed similarly on the Winequality dataset, regardless of the algorithm conditions tested.

Table 3.18: Results of hypothesis testing for LR algorithms on Winequality dataset. ML– Machine Learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.8736	0.057694	ML	ML + SMOTE	-0.0648	0.039249	-2.46506	0.056874	Fail to reject H0
ML + SMOTE	0.9384	0.018446	ML	ML(HP)	0.0166	-0.01404	1.534408	0.185516	Fail to reject H0
ML(HP)	0.857	0.071733	ML	ML(HP) + SMOTE	-0.061	0.042854	-2.2634	0.07304	Fail to reject H0
ML(HP) + SMOTE	0.9346	0.01484	ML + SMOTE	ML(HP)	0.0814	-0.05329	2.625297	0.046798	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.0038	0.003605	0.967492	0.377733	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0776	0.056892	-2.46641	0.05678	Fail to reject H0

Table 3.19 summarizes the overall hypothesis outcome of the LR algorithms on the referenced six imbalanced benchmark datasets. The table shows the rejected null hypothesis outcomes for LR algorithms in some cases, including Ionosphere, Pageblocks, and Poker datasets, which indicates that oversampling improves the performance of the LR model on these datasets. Contrarily, on Spambase, Winequality, and Yeast datasets, LR model performance fails to reject the null hypothesis implying that oversampling does not significantly improve performance in these cases.

Table 3.19: Overall hypothesis outcome of LR algorithms on referenced six benchmarks imbalanced dataset used in this study.

Dataset	Hypothesis					
	One	Two	Three	Four	Five	Six
Ionosphere	Reject H_0	Fail to reject H_0	Fail to reject H_0	Reject H_0	Fail to reject H_0	Reject H_0
Pageblocks	Fail to reject H_0	Reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Poker	Reject H_0	Fail to reject H_0	Reject H_0	Reject H_0	Reject H_0	Reject H_0
Spambase	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Winequality	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Yeast	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0

Similarly, the performance of the LR algorithm was also evaluated on other referenced highly imbalanced datasets, including Ionosphere, Pageblocks, Poker, Spambase, and Yeast. Comprehensive results of the overall performance of each of the datasets can be found in Appendix .3.

3.5.6 Performance of RF

Table 3.20 presents the results of hypothesis testing for RF algorithms on the Yeast dataset. The table reports the mean, standard deviation, mean difference, standard difference, t-value, and p-value for each pair of algorithm conditions. After conducting hypothesis testing for all pairs of conditions, the null hypothesis was not rejected with a significance level of 0.05, indicating that there was no statistically significant difference in performance between the RF algorithms. The p-values for all conditions were greater than 0.05, ranging from 0.019853 to 0.075587. This result suggests that these algorithms perform similarly, and one algorithm does not outperform the other significantly. Specifically, the mean ROC result for the ML algorithm was 0.9798 with a standard deviation of 0.0134, and the mean ROC result for the ML+SMOTE algorithm was 0.999 with a standard deviation of 0.0006. The mean ROC result for the ML(HP) algorithm was 0.9808 with a standard deviation of 0.0144, and the mean ROC result for the ML(HP)+SMOTE algorithm was 0.9984 with a standard deviation of 0.0005. These findings suggest that the RF algorithms performed similarly on the Yeast dataset, regardless of the algorithm conditions tested.

Table 3.20: Results of hypothesis testing for RF algorithms on Yeast dataset. ML– Machine Learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9798	0.013363	ML	ML + SMOTE	-0.0192	0.01273	-3.37171	0.019853	Fail to reject H0
ML + SMOTE	0.999	0.000632	ML	ML(HP)	-0.001	-0.00108	-2.23607	0.075587	Fail to reject H0
ML(HP)	0.9808	0.014442	ML	ML(HP) + SMOTE	-0.0186	0.012873	-3.3172	0.021073	Fail to reject H0
ML(HP) + SMOTE	0.9984	0.00049	ML + SMOTE	ML(HP)	0.0182	-0.01381	2.967041	0.031264	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.0006	0.000143	3	0.030099	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0176	0.013952	-2.91025	0.03339	Fail to reject H0

Table 3.21 summarizes the overall hypothesis outcome of the RF algorithms on the referenced six imbalanced benchmark datasets. The table shows that for all six datasets, the RF model failed to reject the null hypothesis in all six hypotheses tested, indicating that there is no significant difference in the performance of the RF model before and after oversampling the imbalanced datasets.

Table 3.21: Overall hypothesis outcome of RF algorithms on referenced six benchmarks imbalanced dataset used in this study.

Dataset	Hypothesis					
	One	Two	Three	Four	Five	Six
Ionosphere	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Pageblocks	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Poker	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Spambase	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Winequality	Reject H_0	Fail to reject H_0	Reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Yeast	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0

Similarly, the performance of the RF algorithm was also evaluated on other referenced highly imbalanced datasets, including Ionosphere, Pageblocks, Poker, Spambase, and Wine Quality. Comprehensive results of the overall performance of each of the datasets can be found in Appendix .3.

3.5.7 Performance of SVM

Table 3.22 presents the results of hypothesis testing for SVM algorithms on the Ionosphere dataset. The results suggest that there is no statistically significant difference between the performance of the SVM algorithm and the SVM algorithm with SMOTE oversampling technique. Similarly, there is no significant difference between the SVM algorithm with hyper-parameter optimization and the SVM algorithm with hyper-parameter optimization and SMOTE. The mean ROC score for all four conditions is high, with values ranging from 0.9806 to 0.9856. All the p-values are greater than the significance level of 0.05, leading to a “fail to reject H0” conclusion, indicating that there is insufficient evidence to support the alternative Hypothesis.

Table 3.22: Results of hypothesis testing for SVM algorithms on Ionosphere dataset. ML– Machine Learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9806	0.015615	ML	ML + SMOTE	-0.005	0.004505	-0.92848	0.395774	Fail to reject H0
ML + SMOTE	0.9856	0.01111	ML	ML(HP)	0	0			Fail to reject H0
ML(HP)	0.9806	0.015615	ML	ML(HP) + SMOTE	-0.005	0.004505	-0.92848	0.395774	Fail to reject H0
ML(HP) + SMOTE	0.9856	0.01111	ML + SMOTE	ML(HP)	0.005	-0.00451	0.928477	0.395774	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0	0			Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.005	0.004505	-0.92848	0.395774	Fail to reject H0

Table 3.23 summarizes the overall hypothesis outcome of the SVM model on the six referenced imbalanced benchmark datasets. The table demonstrates that all six hypotheses failed to be rejected for the Ionosphere, Pageblocks, Poker, and Yeast datasets. In contrast, for the Spambase dataset, hypotheses Two and Five were rejected, while the other hypotheses failed to be rejected. For the Wine Quality dataset, hypotheses One, Three, and Five were rejected, while hypotheses Two, Four, and Six failed to be rejected. Based on these results, it can be concluded that the null hypothesis (H0) failed to be rejected in most cases,

suggesting that there is no significant difference between the performance of the SVM algorithms on the original and oversampled data.

Table 3.23: Overall hypothesis outcome of SVM algorithms on referenced six benchmarks imbalanced dataset used in this study.

Dataset	Hypothesis					
	One	Two	Three	Four	Five	Six
Ionosphere	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Pageblocks	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Poker	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0
Spambase	Fail to reject H_0	Reject H_0	Fail to reject H_0	Fail to reject H_0	Reject H_0	Fail to reject H_0
Winequality	Reject H_0	Fail to reject H_0	Reject H_0	Fail to reject H_0	Reject H_0	Fail to reject H_0
Yeast	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0

3.6 Overall Findings

Based on the results of Section 3.5, the following is a summary of the overall outcomes for each hypothesis:

Hypothesis One Findings

The purpose of the hypothesis was to investigate whether there is a significant difference in the performance of the ML model without hyperparameter tuning on the original dataset and the SMOTE-based oversampled dataset. The findings from Section 3.5.1 to Section 3.5.7 suggest that the null hypothesis cannot be rejected for all algorithms except KNN and LR on Ionosphere and Spambase and KNN on the Poker dataset. This indicates that there is no significant difference in performance between the original and oversampled datasets. However, for KNN and LR on Ionosphere and Spambase and KNN on Poker, the null hypothesis can be rejected, suggesting that there is a significant difference in performance between the original and oversampled datasets.

In summary, the hypothesis results suggest that the SMOTE-based oversampled dataset does not significantly affect the performance of the ML models for most of the algorithms and benchmarks evaluated in this study. However, the performance of some algorithms on specific datasets may be impacted by the oversampling technique.

Hypothesis Two Findings

This hypothesis aims to examine whether there is a significant difference in the performance of the ML model with hyperparameter tuning on the original dataset and the SMOTE-based oversampled dataset. The findings from Section 3.5.1 to Section 3.5.7 suggest that for all algorithms, except for LR on Pageblocks, the null hypothesis cannot be rejected, indicating that there is no significant difference in performance between the original and oversampled datasets. However, for

LR on Pageblocks, the null hypothesis can be rejected, suggesting that there is a significant difference in performance between the original and oversampled datasets.

Hypothesis Three Findings

Hypothesis Three tests whether there is a significant difference in the performance of the ML model with and without hyperparameter tuning on the original dataset. The overall findings, as reported in Section 3.5.1 to Section 3.5.7, indicate that the null hypothesis cannot be rejected for all algorithms and datasets. This implies that there is no significant difference in performance between the models with and without hyperparameter tuning on the original dataset, irrespective of the algorithm used. Therefore, hyperparameter tuning does not necessarily lead to a significant improvement in performance on the original dataset, at least not for the algorithms tested in this study. It is important to note that the results might differ for other algorithms and datasets, and additional research is required to establish whether this conclusion is generalizable beyond the tested scenarios.

Hypothesis Four Findings

Hypothesis Four tests whether there is a significant difference in the performance of the ML model with and without hyperparameter tuning on the SMOTE-based oversampled dataset. The result in Section 3.5.1 to Section 3.5.7 shows that for all algorithms and datasets, the null hypothesis cannot be rejected, indicating that there is no significant difference in performance between the models with and without hyperparameter tuning on the oversampled dataset.

Hypothesis Five Findings

Hypothesis Five tests whether there is a significant difference in the performance of the ML model on the original dataset and the original dataset with hyperparameter tuning. The overall findings suggest that, for most datasets and algorithms, the null hypothesis cannot be rejected, indicating that there is no significant difference in performance between the original dataset and the original dataset with hyperparameter tuning. However, for DT on Winequality, LR on Poker, and SVM on Winequality, the null hypothesis can be rejected, indicating that there is a significant difference in performance between the original dataset and the original dataset with hyperparameter tuning.

Hypothesis Six Findings

Hypothesis Six tests whether there is a significant difference in the performance of the ML model on the SMOTE-based oversampled dataset and the SMOTE-based oversampled dataset with hyperparameter tuning. The overall findings suggest that, for most datasets and algorithms, the null hypothesis cannot be rejected, indicating that there is no significant difference in performance between the oversampled dataset and the oversampled dataset with hyperparameter tuning. However, for DT and KNN on Winequality and LR on Poker, the null hypothesis can be rejected, indicating that there is a significant difference in performance between the oversampled dataset and the oversampled dataset with hyperparameter tuning.

Table 3.24 presents the outcomes of the six hypotheses on the performance of various ML algorithms across different imbalanced benchmark datasets.

Table 3.24: Hypothesis outcome from one to six of all algorithms on referenced six benchmarks imbalanced dataset used in this study.

Dataset	Algorithms						
	AB	DT	GB	KNN	LR	RF	SVM
Ionosphere	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0 except hypothesis One, Four, and Six	Fail to reject H_0 except hypothesis One, Four, and Six	Fail to reject H_0	Fail to reject H_0
Pageblocks	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0 except hypothesis Two	Fail to reject H_0	Fail to reject H_0
Poker	Reject H_0	Reject H_0 except hypothesis Two	Fail to reject H_0	Reject H_0 except hypothesis Two and Five	Reject H_0 except hypothesis Two	Fail to reject H_0	Fail to reject H_0
Spambase	Fail to reject H_0 except hypothesis One	Reject H_0	Fail to reject H_0 except hypothesis Two and Five	Fail to reject H_0 except hypothesis Two, Three, and Five	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0 except hypothesis Two and Five
Winequality	Fail to reject H_0 except hypothesis Five	Reject H_0 except hypothesis Two and Five	Fail to reject H_0 except hypothesis Five	Reject H_0 except hypothesis Five	Fail to reject H_0	Fail to reject H_0 except hypothesis One and Three	Fail to reject H_0 except hypothesis One, Three, and Five
Yeast	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0	Fail to reject H_0

3.7 Conclusion

The primary objective of the hypothesis testing was to address the research question posed in RQ2, which is 'What is the effect of traditional Machine learning and SMOTE-based data balancing on imbalanced data analysis?' To achieve this, six imbalanced benchmark datasets were used. Four conditions were tested: ML on the original dataset, ML on the SMOTE-based oversampled dataset, ML with hyperparameter tuning on the original, and ML with hyperparameter tuning on the SMOTE-based oversampled dataset. The study tested six hypotheses, and the results suggest that the Adaboost algorithm did not significantly improve the performance of the ML models in the six benchmark imbalanced datasets. However, further research is required to identify the scenarios where Adaboost algorithms might be effective in improving classifier performance in imbalanced datasets with or without oversampling.

The study found that the SMOTE-based oversampled dataset did not significantly affect the performance of the ML models for most of the algorithms and benchmarks evaluated in this study. However, the performance of some algorithms on specific datasets may be impacted by the oversampling technique. The results suggest that hyperparameter tuning does not necessarily lead to a significant improvement in performance on the original dataset, at least not for the algorithms tested in this study.

The results indicate that oversampling significantly impacts KNN algorithms' performance for most datasets. In contrast, the performance of the LR model is significantly improved by oversampling on the Ionosphere, Pageblocks, and Poker datasets. The RF model was found to be unaffected by oversampling for all six datasets. The SVM model's performance was relatively unaffected by oversampling for most of the datasets and algorithms tested.

The study has some limitations, such as the use of a limited number of imbalanced datasets and the use of a limited number of algorithms. Further research is required to evaluate other oversampling techniques' impact and investigate the effect of hyperparameter tuning on other imbalanced datasets and algorithms.

In conclusion, the findings suggest that SMOTE-based oversampling may only sometimes lead to a significant improvement in ML model performance in imbalanced datasets. However, oversampling can significantly impact the performance of some algorithms on specific datasets. Hyperparameter tuning may not necessarily lead to a significant improvement in performance on the original dataset, but it may still be beneficial in certain scenarios. The findings can be helpful for researchers and practitioners working in the field of imbalanced data analysis, particularly those dealing with selecting appropriate ML algorithms and data balancing techniques for imbalanced datasets.

Chapter 4

Enhancing and Improving the Performance of Imbalanced Class Data Using Novel GBO and SSG: A Comparative Analysis

4.1 Introduction

An imbalanced class ratio with datasets is a potential challenge in Machine Learning (ML)-based model development systems (Y. Yang & Xu, 2020). Class imbalance occurs when the total number of samples from one class is significantly higher than the other classes (Cui et al., 2019). In both binary and multiclass classification situations, this inequality can be observed (S. Wang, Dai, et al., 2021). The data class with the lowest sample calls the minor class, and the data class with the highest sample calls the major class. Major class frequently refers to a negative class in binary classification problems, whereas minor class refers to a positive class. It is currently a significant issue in various domains such as biology, health, finance, telecommunications, and disease diagnosis. As an effect, it is considered one of the most severe problems in data mining (Ahsan, Luna, & Siddique, 2022). Figure 5.1 depicts a two-dimensional representation of the major and minor classes.

Most ML algorithms are built in such a way that they perform well with balanced data but cannot perform on an imbalanced dataset. Therefore, several conditions, such as detecting credit card fraud or identifying malignant tumor cells, are difficult to accomplish using typical ML algorithms, where the primary goal is to identify the positive samples (rare samples) (Sevastyanov & Shchetinin, 2020). A well-known example is Caruana et al. (2015)'s study, which sought to determine which pneumonia patients might be hospitalized and which might be discharged home (Caruana et al., 2015). Unfortunately, their proposed approach generated misleading results for asthma or chest pain patients by estimating a lower likelihood of dying.

Numerous ML algorithms have been proposed, and their performance remains biased toward the major class. For instance, consider an imbalanced dataset comprising 10 924 non-cancerous (majority class) and 260 malignant (minority class) cell image data. Using traditional ML algorithms, there is a greater chance that the classification will exhibit a 100% accuracy for the major class and 0% -10% accuracy for the minor class, resulting in the probability of classifying 234 minor class as the major class (Fletcher, Nakeshimana, & Olubeko, 2021). Therefore, 234 patients with cancer would be misdiagnosed as non-cancerous. Such an error is more costly in medical treatment, and a misdiagnosis of a malignant cell has significant health repercussions and may result in a patient's death.

Synthetic Minority Oversampling Technique (SMOTE) is a popular method for dealing with imbalanced data in ML, but it does have some limitations, such as (Qadrini, 2022; C. Meng, Zhou, & Liu, 2020):

1. SMOTE is usually applied to two-class problems: It is not designed to handle imbalanced data with more than two classes.

2. SMOTE may not work well if the minority class is very small: If the minority

class is very small, the synthetic minority instances created by SMOTE may not represent the true minority class distribution.

3. SMOTE may over-emphasize the importance of the minority class: By generating many synthetic minority instances, SMOTE can potentially over-emphasize the importance of the minority class, leading to a model that is overly sensitive to the minority class at the expense of the majority class.

4. SMOTE may not be suitable for high-dimensional data: The interpolation process used by SMOTE can be sensitive to noise in high-dimensional data. This can have negative consequences for the performance of a classifier trained on such synthetic data, as the classifier may not be able to generalize effectively to the true minority class distribution.

Overall, SMOTE may not be the best choice to address imbalanced data in ML on many occasions. Therefore, it is crucial to carefully evaluate its limitations and consider how they may influence the ML model's performance. In this work, we have empirically evaluated the effectiveness of SMOTE and SVM-SMOTE on a range of imbalanced datasets.

The technical contribution of this study can be summarized as follows:

1. The effectiveness of the SMOTE and SVM-SMOTE was evaluated using nine benchmarks of highly imbalanced datasets–Pageblocks, Ecoli, Poker, Winequality, Yeast, Ionosphere, Spambase, shuttle, and Abalone. These benchmarks were chosen in order to accurately assess the performance of

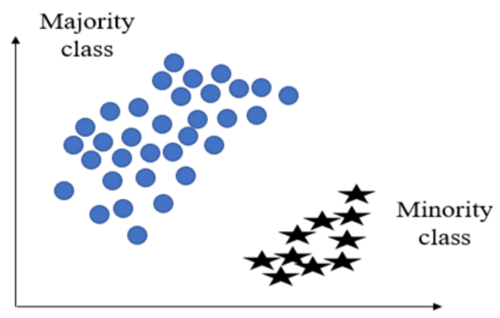


Figure 4.1: Hypothetical example of majority and minority class

SMOTE and SVM-SMOTE in scenarios where the majority class significantly outnumbers the minority class.

2. Various statistical measures were used to evaluate the performance, including accuracy, precision, recall, and F1-score. These measures provide a comprehensive assessment of the ability of different oversampling approaches to classify samples in the minority class correctly and their overall performance.

3. In addition to evaluating the performance of SMOTE, the study also included an analysis of the oversample data distribution using the kernel Density Function (KDF). This analysis was conducted to understand better the characteristics of the synthetic samples generated by various sampling strategies used in this study and how they are distributed within the feature space of the dataset.

4. Based on the results of this analysis, two novel oversampling approaches were proposed: Generative Adversarial Network (GAN)-based Oversampling (GBO) and Support Vector Machine (SVM)-SMOTE-GAN (SSG). These

approaches were designed to address some of the limitations of SMOTE and to improve the performance of oversampling techniques in highly imbalanced datasets.

4.2 Motivation

The motivation of the study is driven by the limitations of two popular data oversampling techniques: SVM-SMOTE and GAN. SMOTE is known to create synthetic samples that may contain noise and are not always realistic, while GAN has the potential to generate highly realistic samples but is difficult to train and requires a large number of training samples and iterations. The aim of this study is to combine the advantages of both SVM-SMOTE and GAN approaches to develop a novel oversampling technique that can create more diverse and realistic samples while also being computationally efficient compared to traditional GAN. The expected outcome of the study is to provide researchers with insights into using GAN beyond its limitations to only create realistic image data but also generate other types of data such as text, numerical and multi-feature data. This could potentially play a crucial role in improving data-balancing approaches in various fields where imbalanced data is a common challenge.

4.3 Chapter Outline

The remainder of the chapter is organized as follows: Section 4.4 reviews the relevant literature and outlines the motivation and limitations of the referenced literature. Section 4.5 presents the methodology for the study, while the experimental setup is described in Section 4.6. Section 4.7 presents the simulation results. In Section 4.8, the experimental results are presented and discussed. Section 4.9 highlights the limitations and scope of the work. Finally, Section 5.8 presents the conclusions and suggests potential avenues for future research.

4.4 Background

Algorithms and techniques to address CIPs are usually classified into three main categories: data level, cost-sensitive, and ensemble algorithms (as shown in Figure 4.2) (Brownlee, 2020b).

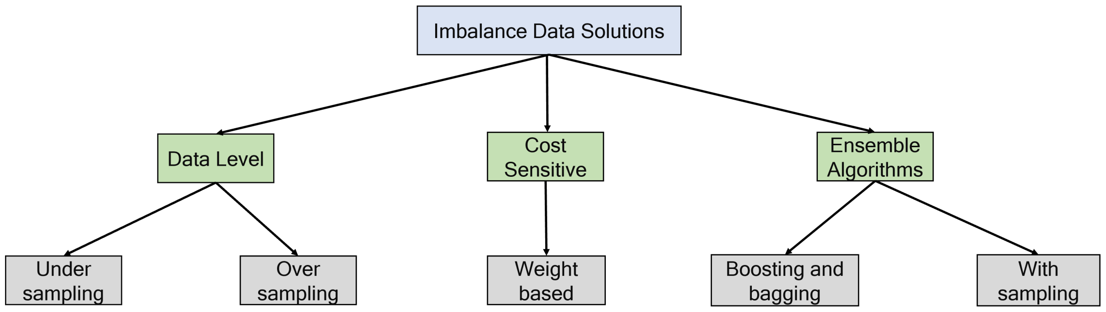


Figure 4.2: Major approaches to handle CIP in the machine learning domain.

In data-level solutions, Oversampling approaches are mostly used where minor class data is oversampled by applying different techniques. Oversampling techniques often employed include Adaptive Synthetic (ADASYN), Random Oversampling, SMOTE, and Borderline-SMOTE (Kaur, Pannu, & Malhi, 2019). Chawla's SMOTE is the most popular and commonly utilized among all Oversampling approaches. However, traditional SMOTE produces more noise and is unsuitable for high-dimensional data. To resolve these issues, Wang et al. (2021) proposed active learning-based SMOTE (S. Wang, Dai, et al., 2021). Zhang et al. (2022) proposed SMOTE-reverse k-Nearest Neighbors-(RkNN), a hybrid oversampling technique to identify the noise instead of local neighborhood information (A. Zhang et al., 2022). Maldonado et al. (2022) demonstrated that traditional SMOTE faces major difficulties when it comes to defining the neighborhood to generate additional minority samples. To overcome these concerns, the authors proposed a feature-weighted oversampling, also known as (FW-SMOTE) (Maldonado, Vairetti, Fernandez, & Herrera, 2022). Aside from that, SMOTE is often computationally costly, considering the time and memory usage for high-dimensional data. Berando et al. (2022) pioneered the use of C-SMOTE to address time complexity issues in binary classification problems (Bernardo & Della Valle, 2022). Obiedat et al. (2022) presented SVM-SMOTE combined with particle Swarm Optimization (PSO) for sentiment analysis of customer evaluations; however, the proposed algorithms remained sensitive to multidimensional data (Obiedat et al., 2022).

On the other hand, Undersampling procedures reduce the sample size of the major classes to create a balanced dataset. Near-miss Undersampling, Condensed Nearest Neighbour (CNN), and Tomek Links are three popular Undersampling methods used more frequently (Kaur et al., 2019). Once the data sample is reduced using Under-sampling techniques, there is a higher chance that it will also eliminate many crucial pieces of information from the major class. As a result, Oversampling is generally preferred over Undersampling by researchers and practitioners (Z. Sun, Zhang, Sun, & Zhu, 2020).

Most ML algorithms consider that all the misclassification performed by the model is equivalent, which is a frequently unusual case for CIP, wherein misclassifying a positive (minor) class is considered the worse scenario than misclassifying the negative (major) class. Therefore, in the cost-sensitive approach higher penalty is introduced to the model for misclassifying the minor samples. In this process, the cost is assigned based on the error made by the model. Suppose the algorithm fails to classify the minor class. The penalty will be higher (i.e., 10 for each misclassification), and if the algorithm fails to classify the major class, then the penalty will be lower (i.e., 1 for each misclassification). Shon et al. (2020) proposed hybrid Deep Learning (DL) based cost-sensitive approaches to classify kidney cancer (Shon, Batbaatar, Kim, Cha, & Kim, 2020). Wang et al. (2020) used multiple kernel learning-based cost-sensitive approaches to generate synthetic instances and train the classifier simultaneously using the same feature space (L. Wang, Wang, & Fu, 2020). One of the potential drawbacks of the cost-sensitive approach is that no defined protocol can be used to set the penalty for misclassification. Therefore, adjusting weight is less preferred due to its complexity of use. The cost (weight) for misclassification is set by the expert's opinions or by manually experimenting until the appropriate cost is identified, which is very time-consuming. Further, determining the penalty requires measuring the impact of the features and considering various criteria. However, such a procedure becomes more complex with multidimensional and multiclass label data.

Several algorithm-based solutions have been proposed to improve the effect of ML classification on the imbalanced dataset. Shi et al. (2022) proposed an ensemble resampling-based approach considering sample concatenation (ENRe-SC). According to the author, the proposed strategy can mitigate the adverse effect of removing the major class caused by Undersampling approaches (H. Shi, Zhang, Chen, Ji, & Dong, 2022). Muhammad et al. (2021) proposed an evolving SVM decision function that employs a genetic method to tackle class imbalanced situations (M. Mohammed, Mwambi, Mboya, Elbashir, & Omolo, 2021). In Taneja et al. (2019), various data balancing techniques were compared for their effectiveness in detecting fraud in a credit card transaction dataset from a European bank, which was highly imbalanced, with fraud comprising less than 1% of transactions. The most effective combination was the SVM-SMOTE and Random Forest classifier, with an F1-score of 0.85 (Taneja et al., 2019). Qaddoura et al. (2020) proposed a three-stage approach for intrusion detection in Internet of Things (IoT) networks involving clustering with data reduction, oversampling, and Single Hidden Layer Feed-Forward Neural Network (SLFN) classification. The combination of SVM-SMOTE oversampling and k-means++ clustering with SLFN classification was the most effective configuration, according to accuracy, precision, recall, and G-mean evaluations. However, further research is needed to validate these findings and explore additional methods for improving intrusion detection in IoT environments (Qaddoura, Al-Zoubi, Almomani, & Faris, 2021).

Jiang et al. (2019) proposed GAN based approaches to handle the imbalanced class problem in time series data (W. Jiang et al., 2019). Xiao et al. (2021) introduced a DL-based method known as the Wasserstein-GAN model and applied it to three different datasets: lung, stomach, and breast cancer. WGAN can generate new instances from the minor class and solve the CIP ratio problem (Xiao et al., 2021).

GAN has become a widely utilized technique in computer vision domains. GAN's capacity to generate real images from random noise is one of its potential benefits (Z. Lin et al., 2018). This dynamic characteristic contributes to GAN's appeal, as it has been used in nearly any data format (i.e., time-series data, audio data, image data) (Frid-Adar, Klang, Amitai, Goldberger, & Greenspan, 2018). Sharma et al. (2022) showed that by using GAN, it is possible to generate data in which the sample demonstrates better Gaussian distribution, which is often difficult to achieve using traditional imbalanced approaches. Their proposed GAN-based approaches show comparatively 10% higher performance than any other existing techniques while producing minor samples that are almost real (A. Sharma et al., 2022). However, one major drawback of their suggested approach is that the model is hardly stable and very time-consuming to generate new samples. Therefore, an updated stable GAN-based oversampling technique might be crucial in tackling class imbalanced problems.

Most GAN-based approaches demonstrate promising results in generating realistic image samples, while the performance on tabular data (i.e., numerical data and categorical data) is still minimal (Y. Lu, Wu, Tai, & Tang, 2018). Considering the opportunity, this work shows that a GAN-based approach can be used to handle imbalanced tabular data apart from image data as well.

As an effect, this study presents two GAN-based Oversampling strategies:

GBO and SSG. The experimental results reveal that the Neural Network (NN) classification on expanded data by the proposed GBO and SSG algorithm performs better than the traditional SMOTE on many occasions and is able to generate almost realistic samples than the traditional SMOTE-based approaches.

The proposed GAN-based oversampling technique is designed specifically for generating tabular data instead of image datasets. One of the primary objectives of this method is to design a stable GAN model that can generate realistic minority class samples. This is crucial, as synthetic samples' quality may substantially impact an ML model's overall performance. By producing high-quality synthetic samples, we can limit the amount of noise in the oversampled dataset, which may be an issue with classic oversampling algorithms like SMOTE. The proposed GAN-based oversampling approach intends to improve the stability and performance of ML models by producing high-quality synthetic samples that can be used to balance class distributions in tabular datasets.

4.5 Methods

4.5.1 SVM-SMOTE Algorithm

SVM-SMOTE is one of the several variations of SMOTE algorithms that have been introduced over the years. It creates new observations from the sample of the minority class that are harder to classify (Zheng, 2020). SVM-SMOTE uses instances and observations from the minor class, which is the support vector of an SVM. There are two methods to create the data. If neighbours from the minority class surround the support vector, it will create the data by extrapolation. However, if the support vector is surrounded mainly by neighbours from the majority class, it will create the data by interpolation. The following is a detailed description of the SVM-SMOTE process (Gu, Angelov, & Soares, 2020).

- Step 1: SVM is trained on the entire dataset to find the support vectors from the minor and the major class. SVM constructs hyperplanes with the highest distance to the closest training points (the margins) to separate classes. The sample on the margin (as shown in Figure 4.3) are the support vectors.

- Step 2a: Combined with the nearest neighbour algorithm, is trained on the entire dataset (minority and majority classes together). Then it will look at the closest neighbours of each support vector in the minor class. If most of the neighbours from the support vectors belong to the minority class, then it will perform extrapolation; in other words, it will expand the boundaries by creating a sample out of the boundary.

  In the case of extrapolation, In Figure 4.3(a), if line S_v is the support vector line and X_{nb} is the nearest neighbours value of instance x_i, then the artificial instance generated by SVM-SMOTE is X_{sy}.

- Step 2b: If most of the neighbours of the support vector belong to the majority class, then the algorithm will perform the interpolation. In the case of interpolation (refer to Figure 4.3(b)), if line S_v is the support vector line and X_{nb} is the nearest neighbors value of instance x_i, then the artificial instance generated by SVM-SMOTE is X_{sy}. The idea is that the new example will fall within the two existing observations from the minority. Therefore, the SVM-SMOTE will not expand the boundaries of the minority class; instead, it will create samples within the boundary (Brownlee, 2020b).

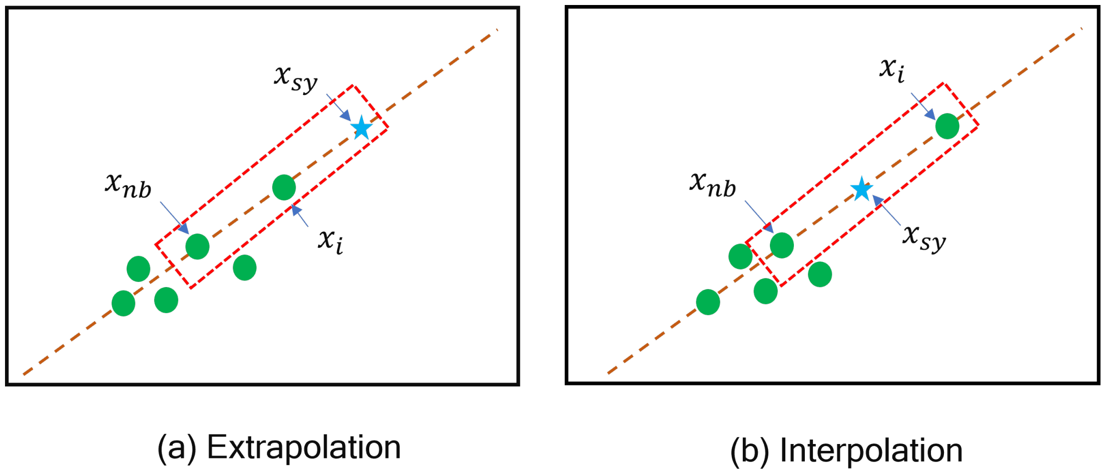


Figure 4.3: Process of generating synthetic sample using (a) extrapolation and (b) interpolation.

A pseudocode of the SVM-SMOTE-based approach is presented in Algorithm 1 (V. Kumar et al., 2022).

4.5.2 Proposed Approaches

Major and minor classes overlapped due to oversampling, one of the potential drawbacks of applying SMOTE as oversampling techniques (S. Wang, Dai, et al., 2021). The SVM-SMOTE algorithm reduces the marginalisation, creating the hyperplane between major and minor classes. However, SVM is sensitive to imbalanced data by nature, which potentially often affects data distribution (Brownlee,

Algorithm 1 : Pseudocode for SVM-SMOTE

```
1: Input: X training sample, N sampling level (100, 200,.......%), K nearest
   neighbors, m number of nearest neighbor to decide sampling type, $SV^+$ set of
   positive support vectors (SVs), T number of artificial instances to be created,
   amount array contains the number of artificial instances, nn array contains k
   positive nearest neighbors of each positive SV.
2: initialize parameters
3: $T \leftarrow (N/100) \times |X\ |$
4: Compute $SV^+$ by training SVMs with $X$
5: Compute amount by evenly distributing $T$ among $SV^+$
6: Compute $nn$
7: For each $sv_i^+ \in SV^+$, compute $m$ nearest neighbors on $X$.
8: Create ammount[$i$] artificial positive instances.
9: $X_new = X \cup X_{new}^+$
10: Output : $X_{new}$ : New oversampled instances
```

2020b). Therefore, GAN-based two Oversampling approaches are proposed–GBO and SSG– to reduce the marginalization and improve the data distribution between major and minor samples.

4.5.2.1 Modified GAN

GAN is conventionally designed to create images that look almost similar to the real images; therefore, it is not ideal for oversampling approaches in other data types (i.e., numerical, categorical, and text data) (A. Sharma et al., 2022). However, since GAN has the powerful ability to augment the images, this concept can be used to create a new but real sample which can also be used as an oversampling approach in CIP-related problems. Since original GAN requires a good amount of data, a lack of data from the minor sample may not be helpful in creating enough samples using GAN. GAN has two neural networks where the generator's goal is to generate fake samples that confuse the discriminator to classify as "real." To maximize its performance, optimising the discriminator's loss (when data comes from the generator) to force the discriminator to classify the fake samples as real. Contrarily, the goal of the discriminator is to distinguish between original and fake data by minimizing the loss when given batches of both original and generated fake data samples (Creswell et al., 2018). The discriminator's loss can be calculated as follows (A. Sharma et al., 2022):

$$\max_{D} \mathbb{E}_x[logD(x)] + \mathbb{E}_z[log(1 - D(G(z)))] \tag{4.1}$$

Here, $D(x)$ denotes the probability output of real data, x of the discriminator, and $D(G(z))$ denotes the probability output of generated samples by z. The generator loss can be calculated as follows (A. Sharma et al., 2022):

$$\min_{G} -\mathbb{E}_z[logD(G(z))] \tag{4.2}$$

A pseudocode of the GAN-based approach is presented in Algorithm 6.

The proposed SSG is developed by modifying and combining SVM-SMOTE and GAN.

Here, the random sample of GAN is replaced with the collection of Oversample minority instances from SVM-SMOTE.

The updated Discriminators loss can be expressed as follows:

$$\max_{D} \mathbb{E}_{x^*}[logD(x^*|x)] + \mathbb{E}_u[log(1 - D(G(u)))] \tag{4.3}$$

and the generator loss can be expressed as:

$$\min_{G} \mathbb{E}_z[logD(G(u))] \tag{4.4}$$

Where x^* is the training sample of the minor class, and u is the Oversampled data of the same classes generated by the SVM-SMOTE.

The SSG model, as depicted in Figure 4.4, is composed of two sections. The first section employs SVM-SMOTE to generate a collection of oversamples, which are then utilized to replace random samples. The second section of the model continues the GAN process by incorporating these newly generated samples

Algorithm 2 : Pseudocode for GAN

```
1: // Input: data sample x and noise samples z (generated randomly).
   // initialize parameters
   // $m_i$ is minibatch indices for $i^{th}$. $n_{fake}$ denotes to number of fake samples
   needed and $T$ is total iterations.
2: GAN $(x, z, n_{fake})$
3: for t=1:T do
4:     // step size $S = 1$
5:     // subscript $d$ and $g$ refers to discriminator and generator
6:     for $s = 1 : S$ do
7:         $g_d \leftarrow$
8:         $SGD(-\log D(x) - \log(1 - D(G(z)), W_D, m_i)$
9:         $W_d \leftarrow weights(g_d, W_d)$
10:        $W_g \leftarrow weights(g_g, W_g)$
11:    end for
12: end for
13: $x' \leftarrow$ FinalOutput $(Model_d(W_d, x, z), Model_g(W_g, x, z), n_{fake})$
14: return $x'$
```

from SVM-SMOTE (inspired by (A. Sharma et al., 2022)). One key distinction between the proposed SSG and GBO models is the use of pre-generated samples from SVM-SMOTE in the SSG model rather than random samples as in the GBO model. This helps the SSG model produce more accurate oversamples by providing high-quality input samples. By leveraging the complementary nature of SVM-SMOTE and GAN, the SSG model can effectively guide the GAN toward producing realistic samples even before further fine-tuning is applied.

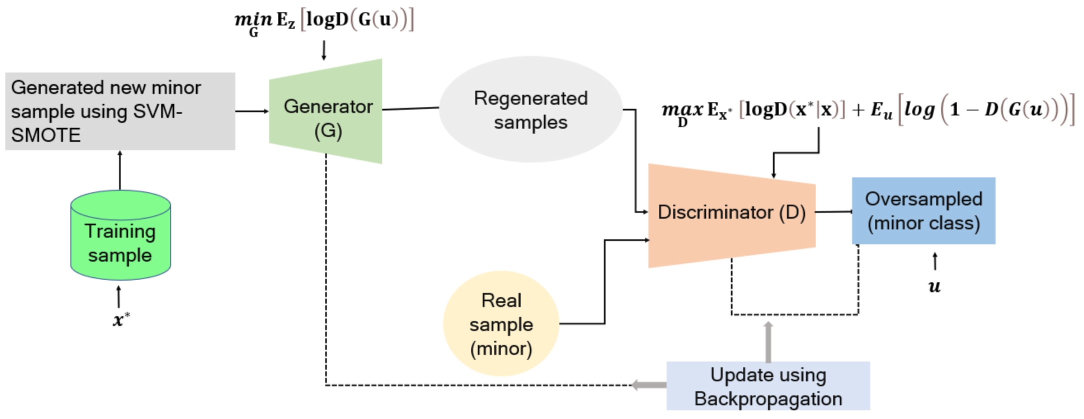

$\min_G E_z[\log D(G(u))]$
Generated new minor sample using SVM-SMOTE
Generator (G)
Regenerated samples
$\max_D E_{x^*}[\log D(x^*|x)] + E_u[\log(1 - D(G(u)))]$
Training sample
x^*
Discriminator (D)
Oversampled (minor class)
u
Real sample (minor)
Update using Backpropagation

Figure 4.4: Process of generating "fake" samples using SSG.

A pseudocode for the proposed GAN-based approach is presented in Algorithm 6.

4.6 Classification Methods and Evaluation Index

With advancements in computational power and ML tools, NN-based approaches are now most widely used on small and large datasets due to their robust and higher accuracy than other existing ML algorithms (Ahsan, E Alam, et al., 2020). One of the potential advantages of using the NN approach is that NN-based

Algorithm 3 : Pseudocode for Proposed modified GAN based approaches

```
1: Input: minor samples $X^*$ from the training sample $x$ of size $N$ with $N - n$ over-samples;
2: parameter $k$ for KNN to find the nearest neighbour
3: Execute SVM-SMOTE given in Algorithm 1 and then GAN given in Algorithm 6
4: $u \leftarrow$ call Algorithm 1 $(x^*, k)$// generate over-sampled minor data $u$.
5: $u \leftarrow$ call Algorithm 6 ($x^*$, $u$, $N$ - $n$).
```

methods are less sensitive compared to traditional ML algorithms (i.e., KNN, SVM, Logistic Regression) (Ahsan, Luna, & Siddique, 2022). The proposed NN approach was developed based on tuning the following parameters: batch size, learning rate, epochs, number of hidden layers, and optimization algorithms. The parameter settings for the proposed NN are illustrated in Figure 4.5.

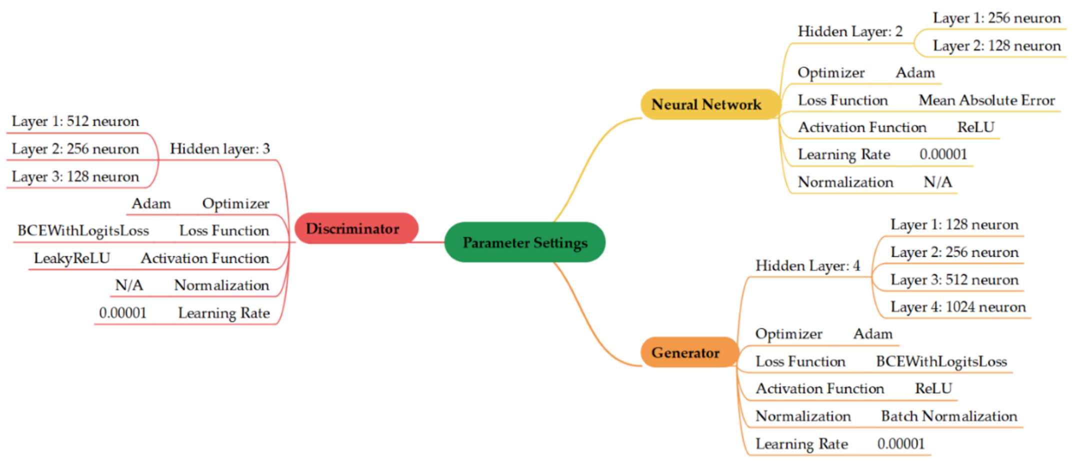


Figure 4.5: Parameter settings used in this study.

To carry out the whole experiment procedure following steps are used as follows:

1. Initially, the dataset was split into two sets where 80%, 70%, and 60% of data were used for the training, and 20%, 30%, and 40% of the data was used for testing purposes in three separate experiments.

2. Dataset was fit to different algorithms such as SMOTE, GBO, and SSG to create Oversample data separately and save it as a new dataset.

3. The newly created balanced dataset was trained and tested with the designed NN models.

4. The simulation was run ten times, and the result was presented by averaging all the results.

4.6.1 Experimental Evaluation Index

The results are presented regarding the accuracy, precision, recall, and F1-score with standard deviation. Suppose the dataset is classified into two types: positive and negative samples. Then the evaluation matrix can be expressed as follows (Ahsan, E Alam, et al., 2020):

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{4.5}$$

$$Precision = \frac{T_p}{T_p + F_p} \tag{4.6}$$

$$Recall = \frac{T_p}{T_n + F_p} \tag{4.7}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.8}$$

Where, True Positive (T_p)= Positive sample classified as Positive

False Positive (F_p)= Negative samples classified as Positive

True Negative (T_n)= Negative samples classified as Negative

False Negative (F_n)= Positive samples classified as Negative.

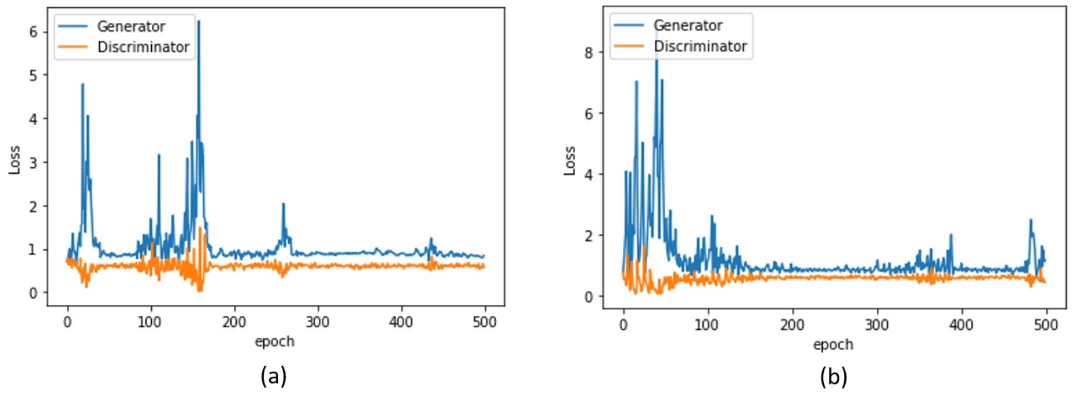


Figure 4.6: Loss per epoch during the training with (a) GBO and (b) SSG.

4.7 Simulation Experiment

4.7.1 Experimental Environment

The experiment was carried out using nine benchmark datasets obtained from the open-source UCI (University of California Irvine) repository: Pageblocks, Ecoli, Poker, Winequality, Yeast, Ionosphere, Spambase, Shuttle, and Abalone dataset. Detailed information regarding those datasets is summarized in Table 7.1 (A. Sharma et al., 2022).

These datasets were chosen as they are commonly used to evaluate the performance of oversampling techniques and represent a wide range of different types of

Table 4.1: Properties of Imbalanced dataset utilized in this study.

Dataset	Total sample	Minor class	Major class	Total features	Minority class(%)	Description
Page-blocks	471	28	443	10	5.94	Classify blocks from page layout
Ecoli	335	20	315	7	5.97	Protein localization
Winequality	655	18	637	10	2.74	Classify the wine quality
Abalone	4177	840	3337	8	20.1	Predict the age of abalone
Ionosphere	351	126	225	34	35.71	Classify radar returns
Spambase	4601	1812	2788	57	39.39	Classify spam and non-spam email
Shuttle	58000	170	57830	9	0.294	80% sample belongs to major class
Yeast	513	51	462	8	9.94	Predicting protein localization cite.

data and class imbalance ratios. It is worth noting that the datasets consist of both numerical and categorical data. The proposed oversampling models were implemented and experimented with using the Anaconda modules with Python 3.8 on a traditional laptop with typical configurations (Windows 10, 16 GB of RAM, and Intel Core I7-7500U). For validation purposes, the dataset was split into the following training set/testing set ratios: 60:40, 70:30, and 80:20. Split ratios of this type are frequently employed in machine learning approaches to evaluate and validate models. By using different split ratios, researchers can get a sense of how the model's performance is affected by the size of the training set (Ahsan, E Alam, et al., 2020).

4.7.2 Numerical Experiment

The experiment was carried out with nine different datasets (as shown in Table 7.1), considering without sampling the data and Oversampling the data using four different methods— SMOTE, SVM-SMOTE, GBO, and SSG. Then, to conduct the appropriate comparison, a NN is used to classify the expanded and original data. As previously stated, each experiment was repeated ten times to ensure scientifically valid and appropriate experimental results. The GBO model and SSG are tested with different parameter settings. Due to the significant effort required to tune those parameters manually, only three primary parameters were optimized: batch size, epochs, and learning rate. Grid search is used to determine the best parameters (Ahsan, E Alam, et al., 2020). The following parameters were used for grid search methods:

$$\text{Batch size} = [32, 64,128]$$

$$\text{Number of epochs} = [40, 50, 100, 200, 500]$$

$$\text{Learning rate} = [0.00001, 0.0001, 0.001, 0.01]$$

Figure 4.4 illustrates the optimal parameters obtained using grid search methods for discriminator, generator, and NN. The generator network contains four hidden layers with 128, 256, 512, and 1024 neurons respectively. The discriminator network has three hidden layers with 512, 256, and 128 neurons. The training procedure is carried out with the Adam optimizer and a binary cross-entropy function with a training data batch size of 32 and an initial learning rate of

0.00001.

Figure 4.6 illustrates the loss per epoch on Abalone datasets for the generator and discriminator during the training. From the Figure, it is clear that using a minor sample generated by SVM-SMOTE shows more stability (refer to Figure 4.6b) than the sample created from random noise (Figure 4.6a).

4.8 Results and Discussion

The performance of different oversampling techniques was evaluated using a 20% test dataset on nine highly imbalanced benchmark datasets, and the results are summarized in Table 4.2. The table reveals that GBO performed best on the Pageblocks, Ecoli, Poker, Winequality, Yeast, Ionosphere, Shuttle, and Abalone datasets, while SSG demonstrated the best performance on all of the datasets in terms of performance improvement on minor classes (highlighted with bold font).

Table 4.2: Performance evaluation of NN on different Oversampling techniques used in this study on nine highly imbalanced benchmarks datasets with a 20% test data set.

Dataset	**Sampling strategy**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Accuracy (minor-class)**
Pageblocks	Without-oversampling	93.68%	0.95	0.95	0.94	37.50%
	SMOTE	98.95%	0.99	0.99	0.99	**100%**
	SVM-SMOTE	97.89%	0.98	0.98	0.98	**100%**
	GBO	98.94%	0.99	0.99	0.99	**100%**

Table 4.2 continued from previous page

Dataset	**Sampling strategy**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Accuracy (minor-class)**
	SSG	98.94%	0.99	0.99	0.99	**100%**
Ecoli	Without-oversampling	95.52%	0.91	0.96	0.93	0%
	SMOTE	94%	0.96	0.94	0.95	50%
	SVM-SMOTE	92.53%	0.93	0.93	0.93	33%
	GBO	94.02%	0.96	0.94	0.95	**100%**
	SSG	94.20%	0.94	0.93	0.95	**100%**
Poker	Without-oversampling	99.32%	0.99	0.99	0.99	33%
	SMOTE	100%	1	1	1	**100%**
	SVM-SMOTE	100.00%	1	1	1	**100%**
	GBO	100.00%	1	1	1	**100%**
	SSG	100.00%	1	1	1	**100%**
Winequality	Without-oversampling	95.42%	0.91	0.95	0.93	0%
	SMOTE	90%	0.94	0.9	0.92	**50%**
	SVM-SMOTE	90.07%	0.94	0.9	0.92	**50%**
	GBO	95.42%	0.91	0.95	0.93	**50%**
	SSG	95.42%	0.91	0.95	0.93	**50%**
Yeast	Without-oversampling	97.08%	0.97	0.97	0.97	72.72%
	SMOTE	90.29%	0.93	0.9	0.91	83.33%
	SVM-SMOTE	89.32%	0.95	0.89	0.91	0%
	GBO	93.20%	0.95	0.93	0.94	**90.90%**

Table 4.2 continued from previous page

Dataset	Sampling strategy	Accuracy	Precision	Recall	F1-score	Accuracy (minor-class)
	SSG	91.26%	0.94	0.91	0.91	**90.90%**
Ionosphere	Without-oversampling	90.14%	0.92	0.9	0.9	77.78%
	SMOTE	95.77%	0.92	0.92	0.92	85%
	SVM-SMOTE	92.95%	0.93	0.93	0.93	**85.71%**
	GBO	91.54%	0.93	0.92	0.93	**85.71%**
	SSG	91.54%	0.93	0.92	0.93	**85.71%**
Spambase	Without-oversampling	91.01%	0.91	0.91	0.91	86%
	SMOTE	92.61%	0.92	0.92	0.92	88%
	SVM-SMOTE	91.23%	0.91	0.91	0.91	88%
	GBO	91.47%	0.92	0.92	0.92	87.67%
	SSG	91.67%	0.92	0.92	0.92	**88.56%**
Shuttle	Without-oversampling	99.95%	1	1	1	80.00%
	SMOTE	99.99%	1	1	1	**100.00%**
	SVM-SMOTE	99.00%	1	1	1	**100.00%**
	GBO	99.93%	1	1	1	**100.00%**
	SSG	99.00%	1	1	1	**100.00%**
Abalone	Without-oversampling	97.08%	0.97	0.97	0.97	72.72%
	SMOTE	88.03%	0.9	0.88	0.89	85.97%
	SVM-SMOTE	93.20%	0.94	0.93	0.94	81.81%
	GBO	96.11	0.97	0.96	0.96	**90.90%**

Table 4.2 continued from previous page

Dataset	Sampling strategy	Accuracy	Precision	Recall	F1-score	Accuracy (minor-class)
	SSG	95.15%	0.96	0.95	0.95	**90.90%**

Table 4.3 compares the performance of different Oversampling methods using 30% of data for testing. From the overall observation, it was revealed that the proposed SSG model significantly improved the accuracy of the minor class by 21% and 7.15% compared to without-oversampling and SMOTE-based approaches on the Yeast dataset. Overall, the proposed SSG method showed the best results among all of the datasets except Winequality, while the GBO method performed the best on the Paegeblocks, Ecoli, Poker, Ionosphere, and Shuttle datasets. In contrast, NN models trained without oversampling performed poorly on all datasets (except the shuttle dataset) compared to any of the approaches used in the study. The results of the shuttle dataset demonstrate a lack of variability in performance across the various Oversampling techniques. One possible explanation for this observation is the presence of relatively large amounts of data within both the major and minor classes.

Table 4.3: Performance evaluation of NN on different Oversampling techniques used in this study on nine highly imbalanced benchmarks datasets with a 30% test data set.

Dataset	Sampling strategy	Accuracy	Precision	Recall	F1-score	Accuracy (minor-class)
Pageblocks	Without-oversampling	93%	0.93	0.94	0.92	33%
	SMOTE	99.29%	0.99	0.99	0.99	**100%**
	SVM-SMOTE	98.59%	0.99	0.99	0.99	**100%**

Table 4.3 continued from previous page

Dataset	**Sampling strategy**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Accuracy (minor-class)**
	GBO	98.59%	1	1	1	**100%**
	SSG	100%	1	1	1	**100%**
Ecoli	Without-oversampling	95%	0.48	0.5	0.49	0%
	SMOTE	97%	0.98	0.97	0.97	75%
	SVM-SMOTE	89.11%	0.33	0.31	0.32	60%
	GBO	93.00%	0.92	0.95	0.94	**100%**
	SSG	93.00%	0.95	0.93	0.94	**100%**
Poker	Without-oversampling	99.32%	0.5	0.5	0.5	0%
	SMOTE	100%	1	1	1	**100%**
	SVM-SMOTE	100%	1	1	1	**100%**
	GBO	100%	1	1	1	**100%**
	SSG	100%	1	1	1	**100%**
Winequality	Without-oversampling	98%	0.49	0.5	0.5	0%
	SMOTE	89%	0.93	0.89	0.91	50%
	SVM-SMOTE	98%	0.98	0.99	0.87	**100%**
	GBO	98%	0.97	0.98	0.98	**0%**
	SSG	98%	0.97	0.98	0.98	**0%**
Yeast	Without-oversampling	94%	0.97	0.97	0.97	79.00%
	SMOTE	92.85%	0.95	0.93	0.93	87.50%
	SVM-SMOTE	94.81%	0.96	0.95	0.95	92.85%

Table 4.3 continued from previous page

Dataset	Sampling strategy	Accuracy	Precision	Recall	F1-score	Accuracy (minor-class)
	GBO	98.05%	0.98	0.98	0.98	92.85%
	SSG	98.70%	0.99	0.99	0.99	**100.00%**
Ionosphere	Without-oversampling	93.39%	0.94	0.93	0.93	100.00%
	SMOTE	92.45%	0.93	0.92	0.92	95.71%
	SVM-SMOTE	97.16%	0.97	0.97	0.97	94.87%
	GBO	99.05%	0.99	0.99	0.99	**97.44%**
	SSG	99.05%	0.99	0.99	0.99	**97.44%**
Spambase	Without-oversampling	93.26%	0.93	0.93	0.93	89.94%
	SMOTE	92.75%	0.93	0.93	0.93	89.60%
	SVM-SMOTE	94.42%	0.95	0.94	0.94	95.66%
	GBO	94.78%	0.95	0.95	0.95	95.32%
	SSG	96.52%	0.97	0.97	0.97	**97.40%**
Shuttle	Without-oversampling	99.98%	1	1	1	**97.20%**
	SMOTE	91.63%	0.92	0.92	0.92	**97.20%**
	SVM-SMOTE	99.98%	1	1	1	**97.20%**
	GBO	99%	1	1	1	**97.20%**
	SSG	99%	1	1	1	**97.20%**
Abalone	Without-oversampling	90%	0.9	0.9	0.9	66.29%
	SMOTE	88.75%	0.89	0.89	0.89	81.11%
	SVM-SMOTE	82.85%	0.89	0.83	0.84	79.5%

Table 4.3 continued from previous page

Dataset	Sampling strategy	Accuracy	Precision	Recall	F1-score	Accuracy (minor-class)
	GBO	89.47%	0.9	0.89	0.9	81.85%
	SSG	88.28%	0.9	0.88	0.89	**86.29%**

Table 4.4 presents the evaluation metrics of accuracy, precision, recall, and F1-scores for different oversampling techniques, with 40% test data. The results show that SSG demonstrated the best performance among all datasets except Ecoli and Spambase. On the other hand, GBO demonstrated better results on all datasets apart from Ecoli, Spambase, and Abalone. This limited performance on these datasets may be due to the use of a smaller amount of data for training, as 40% of the data was used for testing.

Table 4.4: Performance evaluation of NN on different Oversampling techniques used in this study on nine highly imbalanced benchmarks datasets with a 40% test data set.

Dataset	Sampling strategy	Accuracy	Precision	Recall	F1-score	Accuracy (minor-class)
Pageblocks	Without-oversampling	97.88%	0.98	0.98	0.98	63.63%
	SMOTE	99.47%	100%	0.99	0.99	**100%**
	SVM-SMOTE	98.94%	0.99	0.99	0.99	**100%**
	GBO	98.94%	0.99	0.99	0.99	**100%**
	SSG	99.47%	0.99	1	0.99	**100%**

Table 4.4 continued from previous page

Dataset	**Sampling strategy**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Accuracy (minor-class)**
Ecoli	Without-oversampling	96%	0.96	0.96	0.94	14.28%
	SMOTE	98%	0.98	0.98	0.98	**86%**
	SVM-SMOTE	97.01%	0.98	0.97	0.97	85.71%
	GBO	97.01%	0.97	0.97	0.97	71.42%
	SSG	97.01%	0.97	0.97	0.97	42.85%
Poker	Without-oversampling	99.15%	0.99	0.99	0.99	16.67%
	SMOTE	100%	1	1	1	**100%**
	SVM-SMOTE	100%	1	1	1	**100%**
	GBO	100%	1	1	1	**100%**
	SSG	100%	1	1	1	**100%**
Winequality	Without-oversampling	95.42%	0.91	0.95	0.93	0%
	SMOTE	89%	0.93	0.89	0.9	33%
	SVM-SMOTE	90.46%	0.91	0.9	0.91	8.33%
	GBO	95.42%	0.91	0.95	0.93	**50%**
	SSG	94.27%	0.91	0.94	0.93	**50%**
Yeast	Without-oversampling	95.14%	0.95	0.95	0.95	57.14%
	SMOTE	91.74%	0.94	0.92	0.93	75.95%
	SVM-SMOTE	95.63%	0.96	0.96	0.96	76.19%
	GBO	95.14%	0.95	0.95	0.95	**80.95%**
	SSG	95.14%	0.95	0.95	0.95	**80.95%**

Table 4.4 continued from previous page

Dataset	Sampling strategy	Accuracy	Precision	Recall	F1-score	Accuracy (minor-class)
Ionosphere	Without-oversampling	90.78%	0.91	0.91	0.91	97.73%
	SMOTE	90.07%	0.91	0.9	0.9	95.45%
	SVM-SMOTE	90.78%	0.91	0.91	0.91	95.45%
	GBO	90.78%	0.91	0.91	0.91	**97.73%**
	SSG	92.91%	0.93	0.93	0.93	**97.73%**
Spambase	Without-oversampling	93.04%	0.93	0.93	0.93	91.25%
	SMOTE	92.88%	0.93	0.93	0.93	91.52%
	SVM-SMOTE	92.77%	0.93	0.93	0.93	**92.05%**
	GBO	93.15%	0.93	0.93	0.93	90.17%
	SSG	92.88%	0.93	0.93	0.93	90.17%
Shuttle	Without-oversampling	99.87%	1	1	1	97.60%
	SMOTE	99.87%	1	1	1	**100%**
	SVM-SMOTE	99.87%	1	1	1	**100%**
	GBO	99.00%	0.99	1	1	0.00%
	SSG	99.87%	1	1	1	**100%**
Abalone	Without-oversampling	89.88%	0.9	0.9	0.9	70.72%
	SMOTE	87.97%	0.89	0.88	0.88	83.47%
	SVM-SMOTE	85.45%	0.89	0.85	0.86	79.97%
	GBO	89.64%	0.9	0.9	0.9	77.97%
	SSG	89.47%	0.9	0.89	0.9	**83.76%**

To understand the classification effect, the overall misclassification is presented in Table 4.5, where the total number of false positives and false negatives predicted by each oversampling method are counted for each dataset. Here, the algorithm with the lowest misclassification is highlighted with bold fonts and is the primary concern that will help identify the true potential of the algorithm's performance on imbalanced datasets. From the table, it can be observed that SSG demonstrates better performance on all of the imbalanced datasets for 20% of test data. On the other hand, with 30% of test data, except the Abalone dataset, SSG demonstrated the lowest misclassification rate on the remaining imbalanced dataset. However, with 40% test data, we found that the proposed SSG could not perform well on Winequality, Yeast, Shuttle, and Abalone datasets. The potential reason is that increasing the test size also reversely reduces the training data sample, and any GAN-based approach requires a large amount of training data. Therefore, reducing the training sample will not be a good choice. The performance of GBO-based approaches also shows promising results compared to SMOTE and SVM-SMOTE-based approaches for all three case scenarios where 20%, 30%, and 40% of data is used for testing purposes.

Table 4.5: Misclassification occurred by different oversampling techniques used in various datasets; WS–Without Oversampling, S–SMOTE, SVM-SMOTE–SS.

Dataset	**Misclassification = False positive (F_p) + False negative (F_n)**														
	20% test data					**30% test data**					**40% test data**				
	WS	**S**	**SS**	**GBO**	**SSG**	**WS**	**S**	**SS**	**GBO**	**SSG**	**WS**	**S**	**SS**	**GBO**	**SSG**
Pageblock	6	**1**	2	**1**	**1**	9	1	2	**0**	**0**	4	1	2	2	**1**
Ecoli	5	3	3	3	**2**	5	4	8	**2**	**2**	6	**3**	**4**	**4**	**4**
Poker	2	**0**	**0**	**0**	**0**	3	**0**	**0**	**0**	**0**	5	**0**	**0**	**0**	**0**
Winequality	10	13	13	7	**6**	3	22	2	3	**0**	**12**	30	25	**12**	15
Yeast	12	10	9	8	**6**	4	11	8	3	**2**	10	**17**	**9**	10	10
Ionosphere	18	8	8	7	**6**	7	8	3	**1**	**1**	13	14	13	13	**10**
Spambase	85	68	77	73	**62**	93	100	77	72	**48**	128	133	133	**126**	**126**
Shuttle	**4**	6	6	6	**4**	15	15	15	**9**	**9**	**15**	21	21	48	21
Abalone	122	100	122	85	**80**	126	141	215	**132**	147	**169**	252	243	173	176

The optimal results revealed that the proposed SSG shows higher accuracy, precision, recall, and F1-score in most of the highly imbalanced benchmark datasets used in this study. However, during the experiment, it was noticed that even though the accuracy result between the proposed Oversampling techniques and SMOTE techniques may not significantly differ, the benefit of the proposed model can be easily identified once the total misclassification is calculated (as shown in Table 4.5).

Apart from this, on some occasions, it was found that data without oversampling demonstrated the best performance, such as on the Shuttle dataset with 20% and 40% test data where only four samples and nine samples are misclassified;

On the Abalone dataset with 40% test data where 169 samples are misclassified.

Figure 4.7 displays the oversampling effect on the Abalone dataset. The figure shows that the sample created using the proposed SSG (Figure 4.7b) follows better Gaussian distribution than the sample generated by the original SVM-SMOTE (see Figure 4.7a).

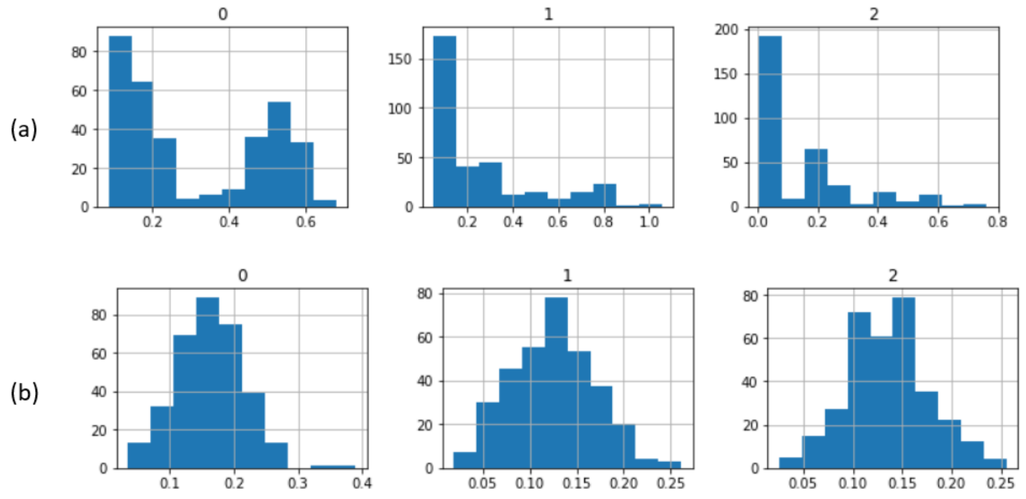


Figure 4.7: Performance observation using histogram of the attribute of Abalone dataset using (a) SVM-SMOTE and (b) SSG.

The experimental results for standard deviation are displayed in Table 4.6, with the best experimental results in bold font. According to Table 4.6, when the proposed SSG is used, the inter-class distance between categories for all the benchmark datasets, except the Yeast dataset, is closest to the raw/original data. On the other hand, when the GBO approach is used on the Pageblocks, Poker, Spambase, and Yeast datasets, the interclass distance across categories after Oversampling are closest to the original data.

Table 4.6: Effect of SMOTE and proposed sampling techniques on the imbalanced dataset in terms of standard deviation.

Dataset	Standard deviation				
	Without-oversampling	SMOTE	SVM-SMOTE	GBO	SSG
Pageblock	0.1782	0.22	0.1927	**0.1630**	**0.1578**
Ecoli	0.1653	**0.2062**	0.1141	0.0737	**0.2062**
Poker	0.345	0.33	0.334	**0.332**	**0.332**
Winequality	0.1549	0.1809	0.1268	0.1872	**0.1607**
Yeast	0.0915	0.1305	0.0655	**0.0723**	0.1677
Ionosphere	0.4710	0.45	0.4036	0.4498	**0.4545**
Spambase	0.0544	0.0772	0.556	0.0547	**0.0504**
Shuttle	44.63	31.2841	36.83	47.34	**41.32**
Abalone	0.2631	0.2821	0.2606	0.1681	**0.2628**

To evaluate the efficiency of the proposed models, the overall process time was measured during the training phase, and the results are presented in Table 4.7. The table shows that the proposed approach has no significant effect on the newly Oversampled dataset during the training phase in terms of computational complexity. Therefore, the proposed approach can be considered efficient as existing SMOTE and SVM-SMOTE.

Table 4.7: Comparison of computational complexity in terms of overall process time for different oversampling approaches used in this study.

Sampling strategy	NN	
	Process time (seconds)	Process time/epochs (seconds)
SMOTE	3.5	0.12
SVM-SMOTE	3.1	0.103
GBO	9.16	0.3
SSG	3.05	**0.1**

KDF estimation, a non-parametric technique, was utilized on both the original and synthetic data to demonstrate the comparability of the data distributions. The resulting visualization, shown in Figure 4.8, presents the original data distributions as the red area. The SMOTE, SVM-SMOTE, GAN, and SSG-generated sample distributions are depicted as green, pink, gold, and blue lines, respectively. By examining this figure, it can be inferred that the proposed SSG oversampling approach can generate nearly realistic samples for the six benchmark datasets: (a) Pageblocks, (b) Ecoli, (d) Winequality, (e) Yeast, (g) Spambase, (h) Shuttle, and (i) Abalone.

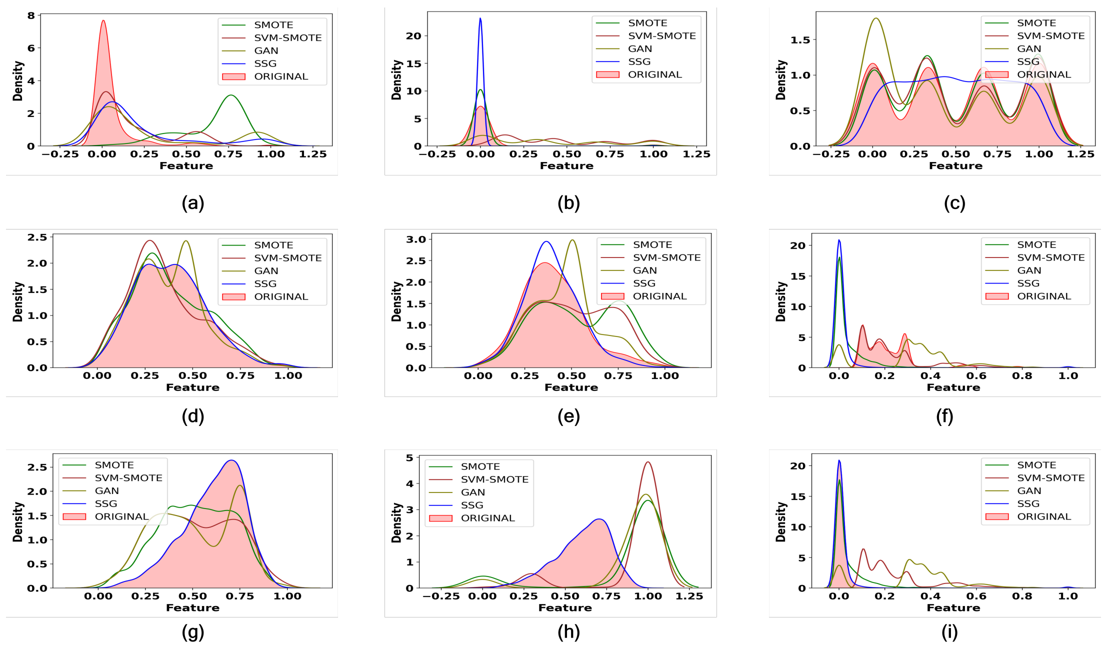


Figure 4.8: Kernel density estimation of original and synthetic data (a) Pageblocks, (b) Ecoli, (c) Poker, (d) Winequality, (e) Yeast, (f) Ionosphere, (g) Spambase, (h) Shuttle, and (i) Abalone.

During this study, the Local Interpretable Model-Agnostic Explanations (LIME) approach was employed to evaluate the black box behavior of the proposed models. LIME is a valuable tool for model interpretability, enabling us to gain insight into the rationales behind the predictions made by the model. This is achieved by analyzing and visualizing the individual contributions of the features, as demonstrated in Figure 5.8. The figure displays the contribution of various features in determining the predicted Wine Quality. The model is 99% confident that the selected Wine Quality is bad and the variables Sulphates, Sulfur dioxide, volatile acidity, and chlorides have the highest influence on the predicted wine quality.

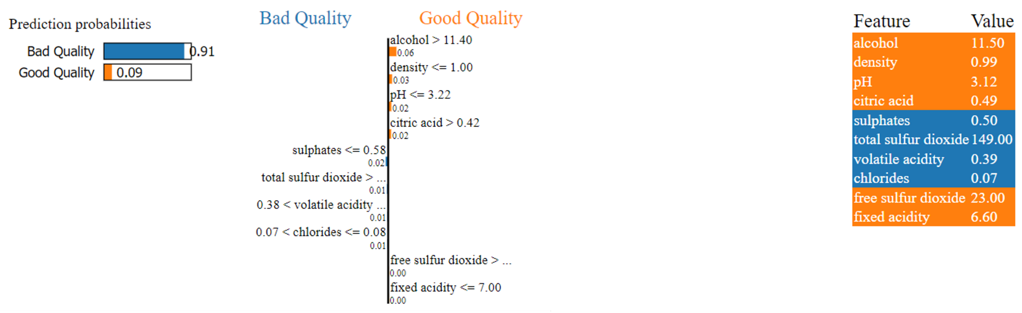


Figure 4.9: Model interpretation using LIME on Winequality dataset.

To obtain more profound insights into the predictions of the oversampled dataset by the proposed model, the current study employed the SHapley Additive exPlanations (SHAP) framework. The illustration in Figure 4.10 presents the force plot of the first observation of the Abalone dataset. The force plot provides a graphical representation of the impact of each feature on the prediction of the model output. As depicted in the figure, the base value is 0.3 and the final value, f(x) = 0.76, represents the predicted value of the abalone.

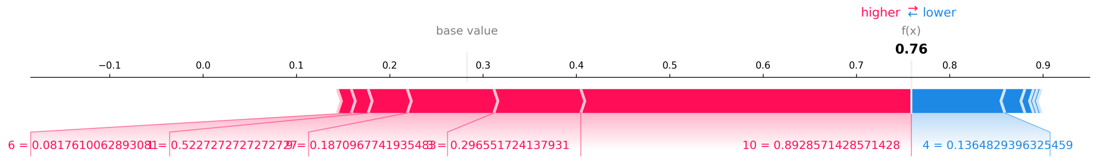


Figure 4.10: Force Plot of the first Observation of the Abalone dataset using SHAP

Figure 4.11 presents the expanded data distribution of various features of the Wine Quality dataset using the proposed SSG model through a scatter plot representation. The upper portion of the figure displays the original data distributions, while the lower portion showcases the data distribution generated by the proposed SSG model. The visualization demonstrates the capability of the proposed SSG model to generate almost realistic data distributions.

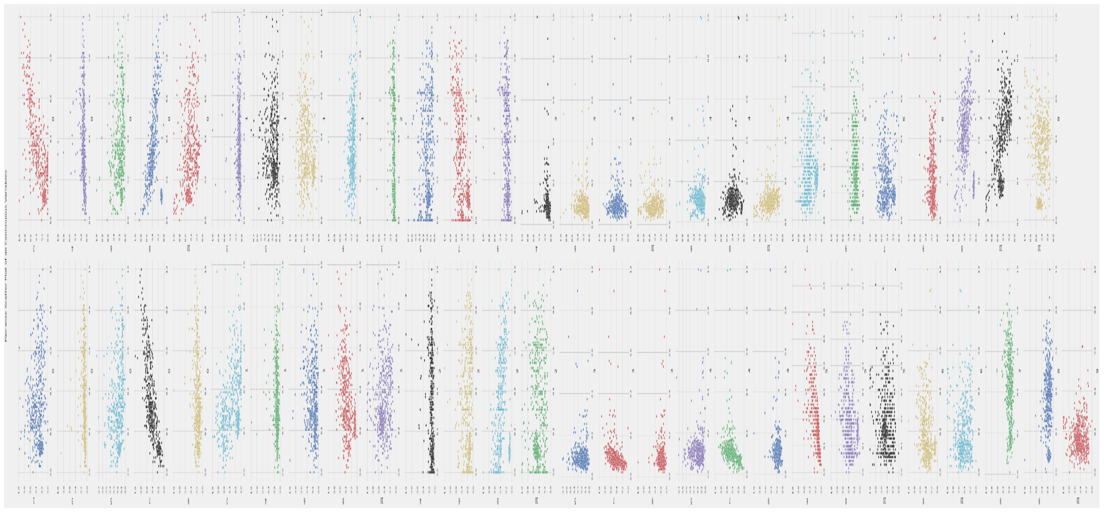


Figure 4.11: Expanded Wine Quality dataset's feature distributions using the proposed SSG Model.

4.9 Limitations of the Study

The study presents the following limitations, which will be addressed in future works that consider the choice of tools and methods utilized:

1. The study was limited to utilizing only NN models, and future research should investigate the performance of other complex algorithms, including Deep Neural Networks (DNN) and Recurrent Neural Networks (RNN).

2. The study did not include time series or graph-based datasets; thus, further experiments are needed to assess the performance of the proposed models on sequential data. This research direction will be a focus of future investigations.

3. The study did not evaluate the performance of the proposed algorithms with advanced ML algorithms like Long Short-Term Memory Networks (LSTM) due to the absence of sequential data. However, in the near future, the proposed approaches will be integrated with LSTM to evaluate the model's performance on time series data.

4. Additionally, future work will aim to verify the overall findings of the proposed approach by applying several ML approaches, such as Random Forest, Decision Tree, and XGBoost, to similar benchmark datasets that are highly imbalanced. The results will be compared with the outcomes of the proposed GBO and SSG algorithms.

4.10 Conclusion

The study aims to address the classification problem of the imbalanced dataset and the limitations of the existing SMOTE approaches by proposing two new Oversampling techniques based on the GAN. The study findings suggest that the effect of the proposed models demonstrates better Gaussian distribution with less marginalization than the original SMOTE on nine highly imbalanced benchmark datasets. The preliminary computation results show that the proposed GAN-based Oversampling (GBO) and SVM-SMOTE-GAN (SSG) can produce synthetic samples. A Neural network (NN) can achieve a higher classification in terms of accuracy, precision, recall, and F1-score. The misclassification rate by the proposed algorithm–GBO and SSG–is lower than the original SMOTE on benchmark datasets used in this study (refer to Table 4.6). Future studies should apply these methods in other datasets, apply mixed-data analysis using kernel methods and combine the proposed approaches with CNN or RNN to deal with the time series data. Further, it would be an exciting opportunity to investigate the combination of GAN with other popular Oversampling techniques, such as Borderline-SMOTE and Adaptive Synthetic (ADASYN).

Chapter 5

BSGAN: A Novel Oversampling Technique for Imbalanced Pattern Recognition

5.1 Introduction

Imbalanced data classification is a problem in data mining domains where proportion of data class of a dataset differ relatively by a substantial margin. In this situation, one class contains a few numbers of samples (known as the minor class) whereas on the other class contains majority of the samples (Ahsan, Ali, & Siddique, 2022; Longadge & Dongre, 2013). Such an imbalanced ratio produces biased results towards major class (majority classes). The issue of imbalanced data is a prevalent problem in many real-world scenarios, such as detecting fraudulent financial transactions, identifying rare medical conditions, or predicting equipment failures in manufacturing (Sahu, Harshvardhan, & Gourisaria, 2020; Jalali et al., 2019).

There are three major ways to handle Class imbalanced problems (CIP) problems (Gosain & Sardana, 2017; Geng & Luo, 2019) :

- Data level solutions (i.e., random undersampling, random oversampling, one-sided selection)
- Cost sensitive (i.e., cost-sensitive resampling, cos sensitive ensembles)
- Ensemble algorithms (i.e., boosting and bagging, random Forest)

Among different data level solutions, oversampling techniques are the most widely used, and the Synthetic Minority Oversampling Technique (SMOTE) is the most often adopted by the researcher and practitioners to CIPs. Chawla et al. (2002) initially proposed SMOTE based solutions, and it became popular due to its capability to produce synthetic samples, ultimately leading to the opportunity to reducing the biasness of the Machine Learning (ML) models (Chawla et al., 2002). However, the existing SMOTE has two potential drawbacks:

1. The synthetic instances generated by the SMOTE often are in the same direction. As an effect, for some of the ML classifiers, it is hard to create a decision boundary between the major and minor classes.

2. SMOTE tends to create a large number of noisy data, which often overlaps with major class (as shown in Figure 5.1)

To overcome the noise generated by the SMOTE, several expansion of SMOTE has been proposed, such as Support Vector Machine (SVM)-SMOTE and Borderline-SMOTE. However, SVM-SMOTE is known for its sensitivity issues with multiclass data samples, while Borderline-SMOTE can only focus on the minor samples that are close between the boundaries and major class (H. Han et al., 2005).

Therefore, both SVM-SMOTE and Borderline-SMOTE has limitation in creating diverse and normally distributed data with less marginalization after data expansion. Considering these challenges, this study proposes a hybrid method of oversampling that exploits the diverse sets of samples, which will be helpful for the ML-based model to differentiate between major and minor classes. The hybrid

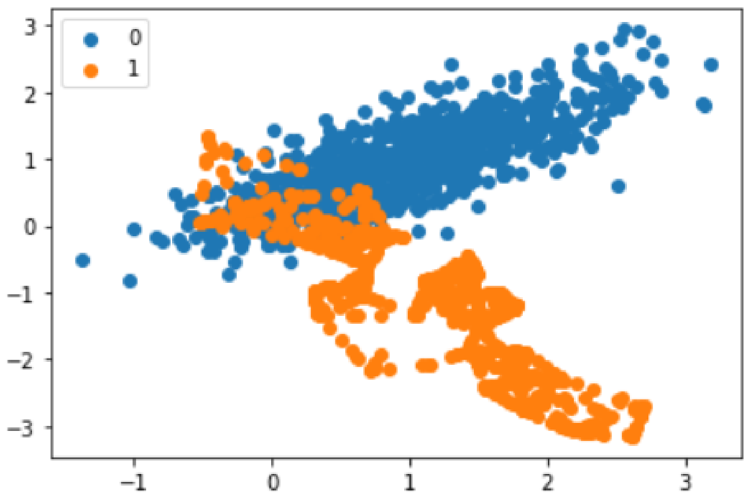


Figure 5.1: The Oversampling effect of SMOTE often creates noisy samples and, therefore, major and minor samples overlap. Here 0 indicates the initial major samples, and 1 indicates minor samples after oversampling.

approach combines two popular oversampling techniques: Borderline-SMOTE and Generative adversarial Network (GAN). First, a novel approach that integrates two Neural Networks (NN) architectures, generator and discriminator, with Borderline-SMOTE, into a unified architecture trained end-to-end is proposed, and the final prediction is obtained through averaging all individual predictions. Second, the efficacy of this approach is evaluated on four datasets with highly imbalanced class distributions. The main contributions of this study can be summarized as follows:

- The generator and discriminator networks are modified and constructed in the study to propose a more effective GAN model that can be trained on a small or large dataset with limited iteration.

- Subsequently, a new oversampling method, Borderline-SMOTE based GAN (BSGAN), is introduced, which combines the advantages of Borderline-SMOTE and GAN. New samples are synthesized along the borderline of classes with Borderline-SMOTE, and realistic samples are generated with GAN, resulting in a highly effective oversampling technique.

- The performance of Borderline-SMOTE, GAN, and BSGAN is examined on four highly imbalanced datasets: Ecoli, Yeast, Wine quality, and Abalone. Later, the effectiveness of these three algorithms is evaluated by comparing their findings to the performance of the datasets without oversampling, using metrics such as accuracy, precision, recall, and F1-score.

- Finally, the performance of the proposed BSGAN model is compared with several reference literature. The preliminary results indicate that the model outperforms many existing GAN-based oversampling techniques, and issues related to sensitive data are effectively addressed. Furthermore, a more diverse dataset is created by incorporating Gaussian distributions in the approach instead of generating the extreme outliers prevalent in many current methods.

5.2 Motivation

The motivation of this study is to further improve the performance of data oversampling techniques by proposing a new approach that combines the advantages of Borderline-SMOTE and GAN. Building upon the success of two novel oversampling techniques proposed in Chapter 4 that utilized Support Vector Machine (SVM)-SMOTE and GAN, this study aims to develop a new approach using similar concepts to determine if the performance of oversampled datasets can be further enhanced. By exploring new ways to balance imbalanced datasets, this study seeks to provide valuable insights into improving the accuracy and effectiveness of machine learning models in a range of fields where imbalanced data is a common challenge.

5.3 Chapter Outline

The remainder of the chapter is organized as follows: Section 5.4 reviews previously published research focusing on different approaches to handling CIPs. Section 5.5 briefly describes SMOTE, Borderline-SMOTE, GAN, and the architecture of the proposed BSGAN technique. In Section 5.5.6, the performance of the different oversampling techniques is evaluated by considering various statistical measurements. An overall discussion and comparison with the current work is presented in Section 5.6. Finally, Section 5.8 concludes the chapter by summarizing the study's contributions and providing potential avenues for future research.

5.4 Related Work

CIPs are one of the existing and ongoing research in data science domains. As the imbalanced ratio potentially affects the models' prediction, several approaches have been proposed to balance the dataset in a way that can be used to develop an unbiased prediction model (Lango & Stefanowski, 2022; Fern, Amir, & Azemi, 2022). Among them, oversampling approaches are most widely used as they provide data-level solutions with less complexity and computational issues (Goodman, Sarkani, & Mazzuchi, 2022). Therefore, the focus of the study has been mainly on popular oversampling methods such as SMOTE, Borderline-SMOTE, and SVM-SMOTE and their modified, adopted versions that have been proposed during the last few years.

A number of recent studies have addressed the limitations and challenges of current SMOTE, and Borderline-SMOTE, particularly regarding their marginalization and noise sensitivity. These limitations refer to the tendency of SMOTE and Borderline-SMOTE to generate synthetic samples that are far from the real data distribution and their vulnerability to the impact of noisy samples in the minority class. For instance, Li et al. (2022) introduced cluster-Borderline-SMOTE, a hybrid method to classify rock groutability (K. Li et al., 2022). Ning et al. (2021) combined SMOTE with Tomek links techniques for identifying glutarylation sites (Ning, Zhao, & Ma, 2021). Zhang et al. (2020) proposed a modified Morderline-SMOTE by combining it with the Relief algorithms for intrusion detection (J. Zhang, Zhang, & Li, 2020). Sun et al. (2020) applied ensemble techniques by combining Adaboost-SVM with SMOTE. The empirical experiments are carried out based on the financial data of 2628 Chinese listed companies (J. Sun, Li, Fujita, Fu, & Ai, 2020). Liang et al. (2020) introduced hybrid oversampling techniques by combining k-means and SVM. The authors claim that the proposed models can generate samples without considering the outlier samples (X. Liang, Jiang, Li, Xue, & Wang, 2020). However, none of the experiments justifies how their proposed model creates a normally distributed dataset.

Recently, GAN have demonstrated the potential to create real samples using random noise (Ahsan, Ali, & Siddique, 2022). For instance, the exiting GAN can be utilized to create real images of any objects from random noise with several neural network iterations. While GAN is generally extensively applied in computer vision domains, the adoption of GAN can be observed in handling CIPs. For instance, Ali Gombe et al. (2019) proposed MFC-GAN, where multiple fake samples is used to create synthetic data to develop a balanced dataset (Ali-Gombe & Elyan, 2019). Kim et al. (2020) used GAN-based approaches to detect anomalies from publicly available datasets like MNIST and Fashion MNIST (Kim, Jeong, Choi, & Seo, 2020). Rajabi et al. (2022) present a novel approach for generating synthetic data that balances the trade-off between accuracy and fairness, through their proposed method, TabFairGAN. Their approach specifically focuses on complex tabular data, and has been empirically evaluated on various benchmark datasets, including UCI Adult, Bank Marketing, COMPAS, Law School, and DTC dataset. The results of the experiments reveal that TabFairGAN demonstrates promising performance, achieving an average accuracy of $78.3 \pm 0.001\%$ and an F1-score of 0.544 ± 0.002 (Rajabi & Garibay, 2022). Engelmann and Lessmann (2021) proposed the cWGAN approach for generating tabular datasets containing both numerical and categorical data. The effectiveness of this approach was evaluated on several highly imbalanced benchmark datasets, including the German credit card, HomeEquity, Kaggle, P2P, PAKDD, Taiwan, and Thomas datasets. The results showed that the cWGAN approach achieved an overall rank of 4.1 for Logistic Regression (Engelmann & Lessmann, 2021). Jo et al. (2022) presented the OBGAN method for generating data from the minority region close to the border. The performance of the OBGAN method was evaluated on several UCI imbalanced datasets. The results indicated that the OBGAN method achieved the highest recall and F1-score of 0.54 and 0.65, respectively (Jo & Kim, 2022).

However, most of the existing GAN-based approaches are computationally expensive and often hard to train due to their instability.

Taking this opportunity into account, a novel hybrid approach is proposed in this work by combining Borderline SMOTE and GAN, which is named BSGAN. The BSGAN is tested along with borderline SMOTE, GAN, and without oversampling on four highly imbalanced datasets— Ecoli, Wine quality, Yeast, and Abalone. The empirical, experimental results demonstrate that BSGAN outperformed most of the existing tested techniques regarding various statistical measures on most of the datasets used in this study.

5.5 Methodology

In this section, a detailed discussion of the algorithms utilized in the study is presented. The algorithms under consideration include SMOTE, Borderline-SMOTE, GAN, and the proposed approach, BSGAN.

5.5.1 SMOTE

SMOTE is one of the most widely used oversampling techniques in ML domains, proposed by Chawla chawla2002smote. The SMOTE algorithm has the following input parameters that can be controlled and changed: K as the number of nearest neighbors (default value, k = 5), oversampling percentage parameters (default value 100%).

In SMOTE, a random sample is initially drawn from the minor class. Then k-Nearest Neighbors (KNN) are identified to observe the random samples. After that, one of the neighbors is taken to identify the vector between the instant data point and the selected neighbors. In order to create additional instances from the first minor instance on the line, the newly discovered vector is multiplied by a random value between 0 and 1. Once it achieves the percentage ratio specified by the user, SMOTE repeats the procedure with different small samples. Algorithm 4 displays the pseudocode of SMOTE, where the appropriate function is introduced for each step of SMOTE process.

As mentioned earlier, SMOTE generates randomly new samples on the datasets, which increases the noise in the major class area, or within the safe minor region far from the borderline area and overfitting it, therefore not efficiently increasing the classification accuracy in order to classify the minor samples. As an effect, SMOTE has several derivatives, such as Borderline-SMOTE, SMOTEBOOST, Safe-level-SMOTE, and others, which were introduced to limit or reduce these problems. This research primarily focuses on utilizing and modifying the Borderline-SMOTE

Algorithm 4 : Pseudocode for SMOTE

```
Input: P number of minor class sample; $S\%$ of synthetic to be generated; K
Number of nearest neighbors
Output: $N_s = (S/100) * P$ synthetic samples
1. Create function ComputKNN ($i \leftarrow 1toP, P_i, P_j$)
for $i \leftarrow 1toP$ do
    Compute K nearest neighbors of each minor instance $P_i$ and other minor
    instance $P_j$.
    Save the indices in the nnaray.
    Populate $(N_s, i, nnarray)$ to generate new instance.
end for
$N_S = (S/100) * P$
while $N_s \neq 0$ do
    Create function GenerateS ($P_i$, $P_j$)
    Choose a random number between 1 and K, call it nn.
    for $att \leftarrow 1$ to numattrs do
        dif= $P_i[nnarray[nn]][attr] - P_j[i][attr]$
        $gap = randomnumberbetween0and1$
        $Synthetic[newindex[attr] = P_i[i][attr] + gap * dif$
    end for
    newindex = newindex + 1
    $N_s = N_s - 1$
end while
4. Return $(*EndofPopulate.*)$
End of Pseudo-Code.
```

to overcome the existing limitations mentioned in section 5.4.

5.5.2 Borderline-SMOTE

Borderline-SMOTE is a popular extension of the SMOTE that is designed to handle imbalanced datasets in ML domains. Borderline-SMOTE was proposed to address some of the limitations of SMOTE for imbalanced dataset classification. Unlike SMOTE, which randomly interpolates between minority samples, Borderline-SMOTE specifically focuses on synthesizing new samples along the borderline between the minority and majority classes. This approach helps to improve the class balance in the dataset and prevent the model from overfitting to the majority class (H. Han et al., 2005). The Borderline-SMOTE algorithm extends the traditional SMOTE by differentiating between minority samples by utilizing the M' number of majority instances within the M-Nearest Neighbors (MNN) of a given minority instance P_i. The default value of M is set to 5. The minority instance is considered safe if the number of majority instances within its MNN is within the range of 0 to M/2. On the other hand, if all of the MNN of a minority instance consist of majority instances, with $M' = M$, the instance is considered to be noise and is eliminated from the computation function to reduce oversampling near the border. Finally, a minority instance is considered a danger instance P' if the number of majority instances within its MNN falls within the range of M/2 to M. After that, Borderline-SMOTE measure KNN between borderline instance and minor instances and generates a new instance using the following equations:

$$\text{New instance} = P'_i + \text{gap} * (\text{distance}(P'_i, P_j)) \tag{5.1}$$

Where P'_i is the borderline minor instance, P_j is the randomly chosen KNN minor instance, and a gap is a random number between 0 and 1. Algorithm 5 displays the pseudocode of Borderline-SMOTE. One of the potential drawbacks of Borderline-SMOTE is that it focuses on the borderline region; therefore, widening the region might confuse the classifier.

5.5.3 GAN

GAN is a class of ML frameworks that contains two Neural Networks (NN). The goal of this framework is to train both networks simultaneously and improve their performance while reducing their loss function as well. Following true data distribution, a new sample is generated with the same statistics as the training set (A. Sharma et al., 2022). The pseudocode for the GAN algorithm is presented in Algorithm 6, where Stochastic Gradient Descent (SGD) and weights are defined functions that determine mini-batch gradient or any other variant such as Adaptive Momentum (ADAM) or Root Mean Square Propagation (RMSprop) and update the weights respectively (Nugroho & Yuniarti, 2022; Hameed, Karlik, & Salman, 2016; Ketkar & Ketkar, 2017). Once the algorithm terminates, 'good' fake samples are collected with accumulateFakeEx based on classification accuracy.

GAN typically contains two NN: generator (G) and discriminator (D). The goal of the G is to create fake samples that look almost real. A random noise between 0 and 1 is used initially to create fake samples. On the other hand, D is trained with the real sample from the dataset. A random sample created by G is then passed to D so that D can distinguish between the real and the fake

Algorithm 5 : Pseudocode for Borderline-SMOTE

```
Input: P number of minor sample; s% of synthetic to generate; M number of nearest neighbors to create the borderline subset; k Number of nearest neighbors
Output: $(s/100)^{*}P'$ synthetic samples
1. Creating function MinDanger ()
for i ← 1 to P do
  Compute M nearest neighbors of each minor instance and other instances from the dataset,
  Check number of Major instance M’ within the Mnn
  if $M/2 < M' < M$ then
    Add instance P to borderlines subset P’
  end if
end for
2. ComputeKNN (i ← 1 to P’, $\mathbf{P}_i, P_j$)
3. $N_s = (S/100) * P'$
while $N_s \neq 0$ do
  4. $GenerateS(P'_i, P_j)$
  $N_s = N_s - 1$
end while
5. Return (*End of populate.*)
End of Pseudo-Code.
```

samples. The goal of the G is to fool the D by creating fake samples which look like reals. Conversely, the goal of the D is not to get fooled by G. During this process, both D and G optimize their learning process. The loss function for D can be calculated as follows:

$$\max_{D} \mathbb{E}_x[logD(x)] + \mathbb{E}_z[log(1 - D(G(z)))] \tag{5.2}$$

Where, the notation $D(x)$ represents the probability distribution obtained from a real data sample x, while $D(G(z))$ refers to the probability distribution produced by a generated sample z.

The loss function of G can be calculated as follows:

$$\min_{G} -\mathbb{E}_z[logD(G(z))] \tag{5.3}$$

5.5.4 Proposed BSGAN

The proposed approach combined Borderline-SMOTE and Naïve GAN to handle CIPs. The Borderline-SMOTE starts by classifying the minor class observations. If all the neighbors are close to the major class, it classifies any minor samples as a noise point. Further, it classifies a few points as border points with major and minor classes close to the neighborhood and resamples from them. In the proposed BSGAN, the loss function of GAN is updated and combined with the Borderline-SMOTE algorithms. Instead of passing random noise to G, a sample created by Borderline-SMOTE is used. The loss for D is updated and can be expressed as follows

$$\max_{D} \mathbb{E}_{x^*}[logD(x^*|x)] + \mathbb{E}_u[log(1 - D(G(u)))] \tag{5.4}$$

Algorithm 6 Pseudocode for GAN

```
// Input: Consists of a training dataset with sample x and noise samples generated from an appropriate random number generator, represented as z. There is an optional parameter, the size of the desired fake sample, indicated as $\boldsymbol{n}_f ake$.
// Initialize the parameters.
// Determine the mini batch indices, represented as $m_i$, for i, as well as the total number of iterations, represented as T.
GAN $(x, z, n_{fake})$
for t=1:T do
    // generally step size S is 1
    // subscript d and g refers to discriminator and generator entity respectively

    for $s = 1 : S$ do
        $g_d \leftarrow$
        $SGD(-\log D(x) - \log(1 - D(G(z)), W_d, m_i)$
        $W_d \leftarrow weights(g_d, W_d)$
        $W_g \leftarrow weights(g_g, W_g)$
    end for
end for
$x' \leftarrow$ accumulateFakeEx $(Model_d(W_d, x, z), Model_g(W_g, x, z), n_{fake})$
return $x'$
```

The updated loss for the G can be express as follows:

$$\min_{G} -\mathbb{E}_z[logD(G(u))] \tag{5.5}$$

Where, the variable x^* represents a training sample from the minority class, while U refers to the data that has been oversampled using the Borderline-SMOTE method.

Figure 5.2 demonstrates the overall flow diagram of the proposed BSGAN algorithms.

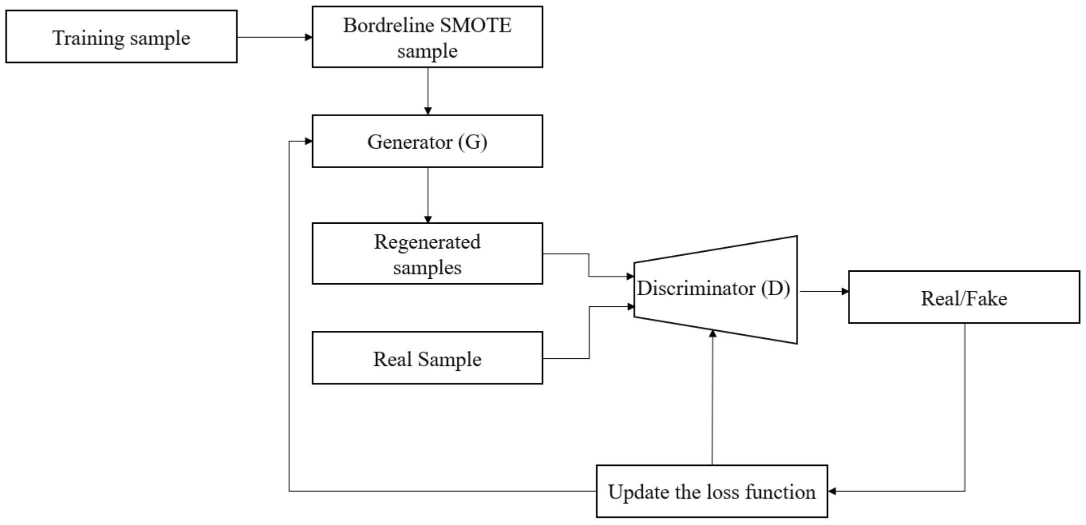


Figure 5.2: Flow diagram of Proposed Borderline-SMOTE-Generative Adversarial Networks (BSGAN) models.

The pseudocode of the proposed BSGAN is described in Algorithm 7. As illustrated in Algorithm 7, there are two sections of BSGAN. The first one replaces the random number sample from the sample generated by Borderline-SMOTE. The subsequent sections continue to utilize GAN in conjunction with the Borderline-

SMOTE algorithm. Algorithm 7 presents a systematic approach to this integration through a two-step procedure. Firstly, it executes the Borderline-SMOTE function, as described in Algorithm 5, and then proceeds to execute the modified GAN function, as outlined in Algorithm 6, employing the generated sample $'u'$ instead of the conventional random noise $'z.'$ This methodology enhances the accuracy and effectiveness of the GAN, thereby providing a more sophisticated solution to the issue at hand.

Algorithm 7 : Pseudocode for proposed BSGAN

```
Step 1 → Input: A set of minority samples $X^*$ from the training data $x$ with a size of $N$ that requires generating $N - n$ over-sampled instances;
Step 2 → A user-defined parameter $k$ for the K-nearest neighbors is executed
Step 3 → Execution of Borderline-SMOTE and GAN: The Borderline-SMOTE and GAN algorithms are executed in the following manner:
1 $u \leftarrow$ . Call Algorithm 1 ($x$·$k$) to generate over-sampled minority examples $u$.
2 Call Algorithm 2 ($x$,$u$,$N$ - $n$) to generate additional instances.
```

5.5.5 Proposed Neural Network

A NN model is used to train and test the model on a different dataset. Parameters such as batch size, number of epochs, learning rate, and the hidden layer are tuned manually by trial and error process. Table 7.1 presents the details of the optimized parameters obtained throughout the experiment to achieve the best experimental outcomes for the discriminator, generator, and NN. The number of epochs varies for each dataset as each dataset differs due to different features and sample sizes.

Table 5.1: Parameter settings used to develop the discriminator, generator, and Neural Network.

Parameters	**Discriminator**	**Generator**	**Neural Network**
Number of hidden layer	4	3	3
Number of neurons	64,128,256,512	512, 256,128	256, 128,1
Batch size	32	32	32
Learning rate	0.00001	0.00001	0.00001
Optimizer	Adam	Adam	Adam
Loss function	Binary cross entropy	Binary cross entropy	Binary cross entropy
Activation function	ReLU	ReLU	ReLU & Sigmoid

5.5.6 Performance Evaluation

5.5.6.1 Datasets

The proposed model is evaluated and compared on four highly imbalanced datasets, namely Ecoli, Yeast, Winequality, and Abalone, which feature class imbalance, as presented in Table 5.2. The datasets are primarily adopted from the UCI ML repository, which researchers and practitioners commonly use to evaluate the model performance for CIPs. Some datasets, such as Winequality and Ecoli, are highly imbalanced and contain only 2.74% and 5.97% minority classes.

Table 5.2: Characteristics of imbalanced dataset utilized for the experiment.

Dataset	# of sample	Minor sample	Major sample	Total features	Minority class(%)	Description
Ecoli	335	20	315	7	5.97	Protein localization
Yeast	513	51	462	8	9.94	Predicting protein localization cite.
Wine quality	655	18	637	10	2.74	Classify the wine quality
Abalone	4177	840	3337	8	20.1	Predict the age of abalone

5.5.6.2 Experimental Setup

The entire experiment was conducted utilizing an office-grade laptop with standard specifications, including a Windows 10 operating system, an Intel Core I7-7500U processor, and 16 GB of RAM. Initially, the dataset is split into the following ratios— train set/test set: 80/20. The experimental evaluation results are presented in terms of accuracy, precision, recall, F1-score, AUC-ROC, and interclass distance which have been calculated using the following formulas:

$$Accuracy = \frac{T_p + T_N}{T_p + T_N + F_p + F_N} \tag{5.6}$$

$$Precision = \frac{T_p}{T_p + F_p} \tag{5.7}$$

$$Recall = \frac{T_p}{T_n + F_p} \tag{5.8}$$

$$F1 - score = 2 \times \frac{\text{Precision} \times \text{ Recall}}{\text{Precision+Recall}} \tag{5.9}$$

Here, False Positive (F_p) = Negative instances classified as Positive

True Negative (T_n) = Negative instances classified as Negative

False Negative (F_n) = Positive instances classified as Negative

$$AUC = \frac{\sum R_i(I_p) - I_p((I_p + 1)/2}{I_p + I_n} \tag{5.10}$$

Where, l_p and l_n denotes positive and negative data samples and R_i is the rating of the i^{th} positive samples.

True Positive (T_p) = Positive instances classified as Positive

$$\text{Interclass distance} = \frac{|\mu_1 - \mu_2|}{\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{5.11}$$

This assumes that there are two classes with means μ_1 and μ_2, and sample sizes of n_1 and n_2, respectively.

5.6 Results

The overall performance for data with and without oversampling was measured using equations 1,2,3 and 4 and presented in Table 10.4. The best results are highlighted with bold fonts. From the table, it can be seen that Proposed BSGAN outperformed all of the techniques across all measures in all datasets. However, on the Wine Quality dataset, GAN and BSGAN both demonstrated similar performance on the train set by achieving an accuracy of 99.17%. The highest F1-score was achieved using BSGAN (0.9783) on the Yeast dataset. The lowest F1-score was achieved on the Abalone dataset when tested without oversampling techniques (0.9041). The highest recall score of 1.0 was achieved on the Wine Quality dataset using BSGAN. On the other hand, the lowest recall score of

0.9055 was achieved on the Abalone dataset when the dataset was tested without oversampling techniques. A maximum precision score of 0.9768 was achieved on the Ecoli dataset using BSGAN, while the lowest precision score of 0.9036 was observed on the Abalone dataset. In Table 10.4, the bold font is indicated the best result acquired with different oversampling techniques.

Table 5.3: Performance evaluation of different oversampling techniques used in this study on highly imbalanced benchmark datasets.

Dataset	**Oversampling Strategy**	**Train**	**Test**			
		Accuracy	Accuracy	Precision	Recall	F1-score
Ecoli	Without-oversampling	93.22%	91.67%	0.9167	0.9167	0.9095
	Borderline-SMOTE	98.84%	95.11%	0.9661	0.9523	0.9572
	GAN	98.33%	97.61%	0.9767	0.9761	0.9703
	BSGAN	99.29%	**97.85%**	**.9786**	**0.9785**	**0.9783**
Yeast	Without-oversampling	92.61%	90.72%	0.9043	0.9072	0.9042
	Borderline-SMOTE	87.89%	92.32%	0.9347	0.9232	0.9274
	GAN	97.11%	94.18%	0.9396	0.9418	0.9351
	BSGAN	97.17%	**94.65%**	**0.9441**	**0.9465**	**0.9412**
Wine quality	Without-oversampling	98.37%	**93.90%**	**0.9390**	**1.0**	**0.9685**
	Borderline-SMOTE	99.03%	92.68%	0.9068	0.9268	0.9150
	GAN	99.17%	93.84%	0.9332	0.9932	0.9623
	BSGAN	99.17%	**93.90%**	**0.9390**	**1.0**	**0.9685**
Abalone	Without-oversampling	90.37%	90.55%	0.9036	0.9055	0.9041
	Borderline-SMOTE	87.17%	84.21%	0.8945	0.8421	0.8539
	GAN	94.09%	90.54%	0.9032	0.9054	0.9037
	BSGAN	94.18%	**90.64%**	**0.9049**	**0.9064**	**0.9052**

The confusion matrix was calculated on the test set to simplify the understanding of the performance of different oversampling techniques on different imbalanced datasets. Figure 5.3 displays the confusion matrix for different sampling techniques on a given Ecoli test dataset. On the Ecoli dataset, maximum misclassification occurred for the dataset without oversampling techniques, up to 7.46% (5 samples). On the other hand, minimum misclassification occurred for BSGAN, up to 1.49% (only one sample).

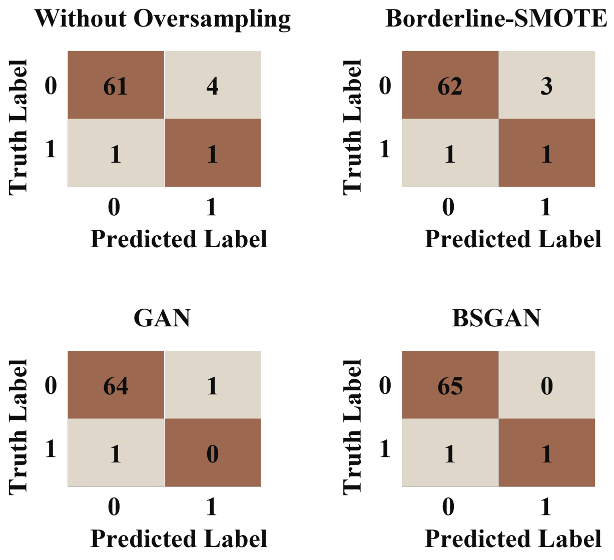


Figure 5.3: Confusion matrix of with and without oversampling techniques on Ecoli test dataset.

Figure 5.4 displays the confusion matrix for different sampling techniques on a wine quality test dataset. From the figure, it can be observed that the NN model performance on the Wine quality dataset without oversampling demonstrated the worst classification by misclassifying 13 out of 131 samples (9.9%). In comparison, BSGAN showed the best performance by misclassifying only 4 out of 131 samples (3.05%).

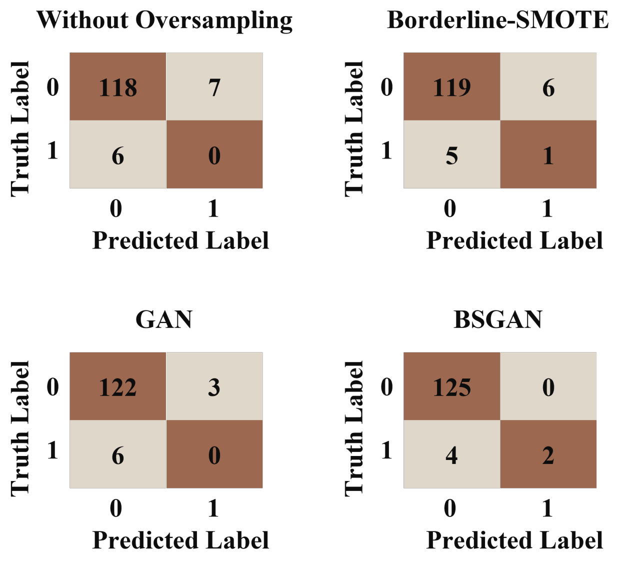


Figure 5.4: Performance measurement of without and with oversampling techniques on Wine quality test dataset using confusion matrices.

Figure 5.5 displays the confusion matrix for different sampling techniques on a given Yeast test dataset. From the figure, it can be observed that NN model performance on yeast dataset Borderline-SMOTE demonstrated the worst performance by misclassifying 8 out of 131 samples (7.77%), while BSGAN showed the best performance by misclassifying only four samples (3.88%).

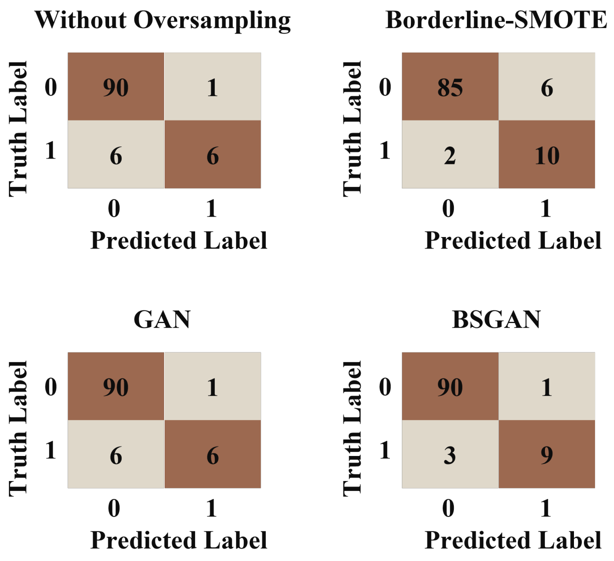


Figure 5.5: Performance measurement of without and with oversampling techniques on Yeast test dataset using confusion matrices.

Figure 5.6 illustrates the confusion matrix for different sampling techniques on a given Abalone test dataset. From the figure, it can be observed that NN model performance on the Abalone dataset Borderline-SMOTE demonstrated the worst performance by misclassifying 122 out of 836 samples (14.59%), while BSGAN showed the best performance by misclassifying 73 samples (8.73%).

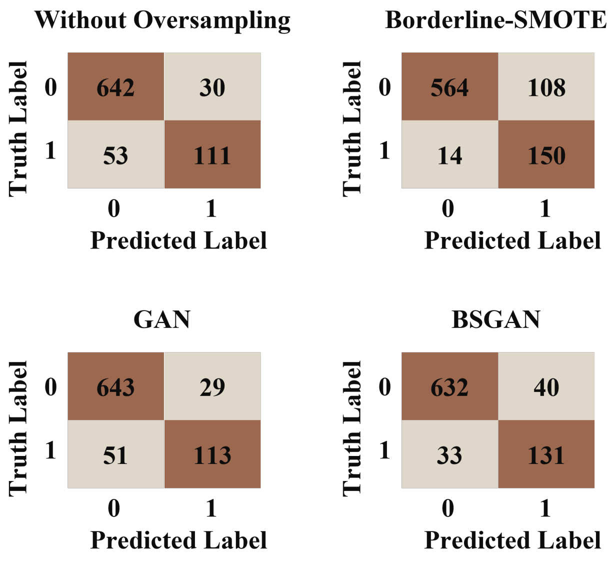


Figure 5.6: Performance measurement of without and with oversampling techniques on Abalone test dataset.

To understand the data distribution after expanding the dataset using different oversampling techniques inter-class distance have been measured using equation 5.11. The closer the inter-class distance between the dataset and the expanded data, the better the classification effect ultimately demonstrates better gaussian distributions. From Table 7.5, it can be observed that the interclass distance between the BSGAN and the dataset without oversampling is the closest compared to any other oversampling techniques used in this study. On the Abalone dataset, Borderline-SMOTE also demonstrates the closest inter-class distance with original datasets. Unfortunately, data expansion after applying GAN shows the worst performance on three out of four imbalanced datasets— Ecoli, Wine quality, and Abalone.

Table 5.4: The inter-class distance between original datasets and the datasets after the expansion using different oversampling techniques; WS–Without oversampling, BS–Borderline SMOTE, GBO–GAN based Oversampling, BSGAN–Borderline-SMOTE based GAN.

Dataset	WS	BS	GBO	BSGAN
Ecoli	0.1650	0.1352	0.0893	0.150
Yeast	0.093	0.079	0.083	0.10
Wine quality	0.1541	0.1531	.0871	0.158
Abalone	0.2633	0.25	0.1856	0.25

5.7 Discussion

As a means of comparing the results with those available in the literature, the performance of the proposed methods on Yeast datasets is contrasted in Table 5.5 in terms of accuracy, precision, recall, and F1-score. The table shows that BSGAN outperformed all of the referenced literature across all measures except the performance of accuracy. While Jadhav et al. (2020) achieved the highest accuracy (98.42%), their precision score is relatively very low, and their F1-score is 0, which hinders a direct comparison of all reported performance measures.

Table 5.5: Comparison with the previous study on Yeast datasets.

References	Techniques	Accuracy	Precision	Recall	F1-score
(A. Sharma et al., 2022)	SMOTified-GAN	96.11%	0.91	0.83	0.873
(Siddappa & Kampalappa, 2019)	LMDL	56.87%	.57	.57	.55
(Karia, Zhang, Naeim, & Ramezani, 2019)	GenSample	70%	0.47	0.50	0.48
(Jo & Kim, 2022)	OBGAN	-	-	0.6135	0.5556
(Jadhav, 2020)	svmradial	98.42%	0.8	-	0
Proposed study	BSGAN	97.17%	0.9441	0.9465	0.9412

On Ecloi datasets, our proposed BSGAN demonstrates consistent performance and outperformed all of the referenced literature in terms of accuracy by achieving an accuracy of 99.29%. Sharma et al. (2022) claimed 100% precision, recall, and F1-score while the accuracy is only 90.75% (as shown in Table 5.6). Therefore, there is some discrepancy in the results reported by the authors.

Table 5.6: Comparison with the previous study on Ecoli datasets.

References	Techniques	Accuracy	Precision	Recall	F1-score
(A. Sharma et al., 2022)	SMOTified-GAN	90.75%	1	1	1
(Siddappa & Kampalappa, 2019)	LMDL	80.95%	.80	.81	.79
(Xiaolong, Wen, & Yanfei, 2019)	DSMOTE	0.706	0.936	0.805	
(Mohamad, Selamat, Subroto, & Krejcar, 2021)	PCA-Ranker	77.68%	0.44	0.37	0.38
(Jadhav, 2020)	svmradial	0.99	1	0.9231	
Proposed study	BSGAN	99.29%	0.9786	0.9785	0.9783

On the Abalone dataset, BSGAN becomes the second-best algorithm in terms of precision, recall, and F1-score, while the question raised as SMOTified-GAN demonstrates nearly perfect precision, recall, and F1-score (as shown in Table 5.7).

Table 5.7: Comparison with the previous study on Abalone datasets.

References	Techniques	Accuracy	Precision	Recall	F1-score
(A. Sharma et al., 2022)	SMOTified-GAN	98.61%	1	1	0.9222
(Jo & Kim, 2022)	-	-	-	0.5960	0.4908
(Mohamad et al., 2021)	PCA-Ranker	99.23%	0.5	0.5	0.5
(Jadhav, 2020)	svmradial	97.70%	0.00	-	0.00
Proposed study	BSGAN	94.18%	0.9049	0.9064	0.9052

On Wine quality datasets, BSGAN became the second-best algorithm in terms of precision, recall, and F1-score, while PCA-Ranker showed the best results(as shown in Table 5.8). Again with 97.19% accuracy achieving a nearly perfect score of precision, recall, and F1-score is hardly feasible.

Table 5.8: Comparison with the previous study on Wine quality datasets.

References	Techniques	Accuracy	Precision	Recall	F1-score
(Jo & Kim, 2022)	OBGAN	-	-	0.5389	0.6508
(Siddappa & Kampalappa, 2019)	LMDL	71.11%	.72	.71	.71
(Mohamad et al., 2021)	PCA-Ranker	97.19%	1	1	1
(A. Sharma et al., 2022)	SMOTified-GAN	95.58%	0.53	0.69	0.5274
Our study	BSGAN	93.90%	0.9390	1.0	0.9685

In Figure 5.7, measures of the Area Under the Curve (AUC) of the Receiver Characteristics Operator (ROC) are plotted for each oversampling technique applied to the test set of different datasets. Our proposed BSGAN shows the best performance on all datasets, and the highest AUC score (0.89) is achieved on the yeast dataset. The worst performance (AUC = 0.5) is achieved on Wine quality and Ecoli datasets without applying any oversampling techniques.

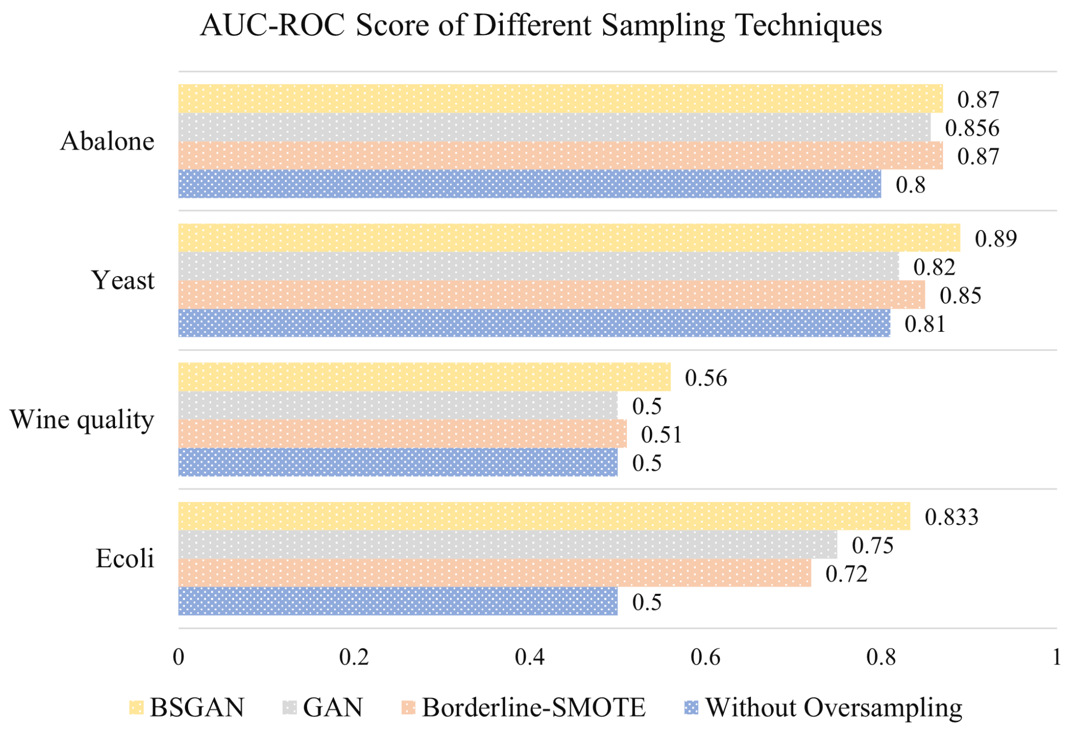
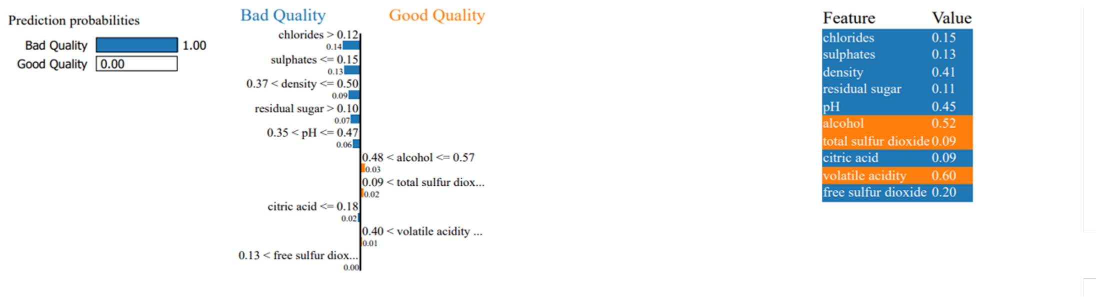


Figure 5.7: AUC-ROC scores for different sampling techniques on referenced imbalanced datasets used in this study.

During the study, Local Interpretable Model-Agnostic Explanations (LIME) were employed to assess the black box behavior of our proposed models. LIME, a valuable tool for model interpretability, affords us an understanding of the rationales behind the predictions made by the model through analysis and visualization of the individual feature contributions. This is illustrated in Figure 5.8, which shows the contribution of various features to the prediction of Wine quality. The model is 99% confident that the predicted Wine Quality is poor, and the variables with the greatest impact on the predicted wine quality are Sulphates, Sulfur dioxide, volatile acidity, and chlorides.



Feature	Value
chlorides	0.15
sulphates	0.13
density	0.41
residual sugar	0.11
pH	0.45
alcohol	0.52
total sulfur dioxide	0.09
citric acid	0.09
volatile acidity	0.60
free sulfur dioxide	0.20

Figure 5.8: Interpreting the model using LIME on the Wine quality dataset.

Additionally, the Shapley Additive Explanations (SHAP) framework was employed to comprehend the prediction outcomes of the model on the oversampled dataset with more clarity. As depicted in Figure 4.10, the illustration presents a force plot of the first observation in the Wine quality dataset. This force plot graphically illustrates the influence of each feature on the prediction made by the model. The figure shows that the baseline value is 0.3 and the final value, f(x) = 0.76, signifies the predicted value of the abalone.

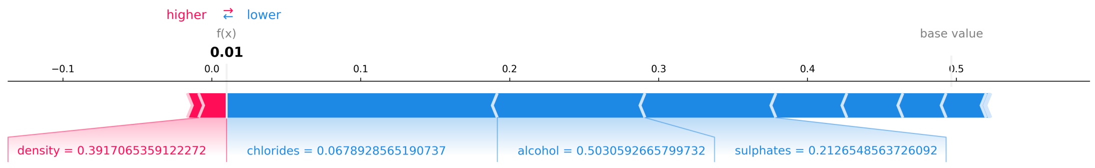


Figure 5.9: Force Plot Observation of the Wine quality data using SHAP.

Figure 5.10 presents a SHAP explanation for the second observation in the test data from the Wine quality dataset. The actual outcome reflects a poor wine quality, which was accurately predicted by the model. The figure displays the average predicted score of the dataset, represented by E(f(x)), at the bottom and is equal to -0.194. The prediction score for the specific instance, represented by f(x), is shown at the top and equals 3.825. The waterfall plot sheds light on the contribution of each feature in the prediction process, leading to a change in the prediction from E(f(x)) to f(x). The feature 'pH' is seen to have the largest impact and plays a crucial role in the prediction by decreasing the prediction value. Conversely, the feature 'density' has a negative impact on the prediction outcome.

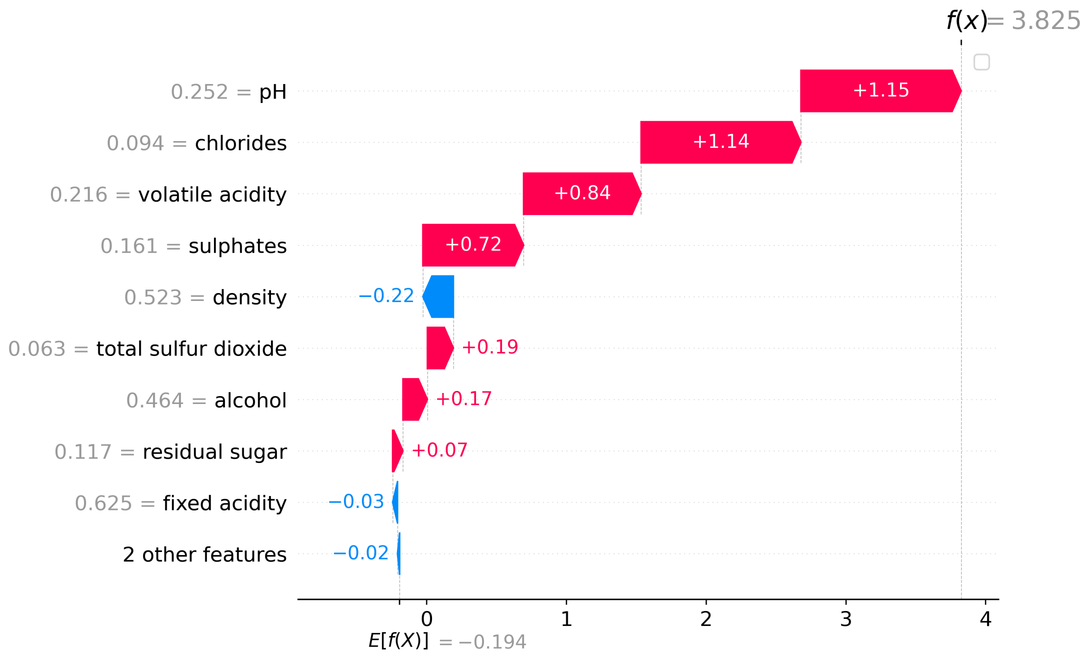


Figure 5.10: A Waterfall plot example for the median predicted wine quality in the Wine quality dataset.

Figure 5.11 presents a SHAP explanation of the 15th observation in the test data from the Wine Quality dataset. The actual outcome depicts a wine of good quality, which was correctly predicted by the model. As seen in the figure, the expected value is near 1, indicating that factors such as pH and citric acid played a significant role in the model's determination of the wine as being of good quality.

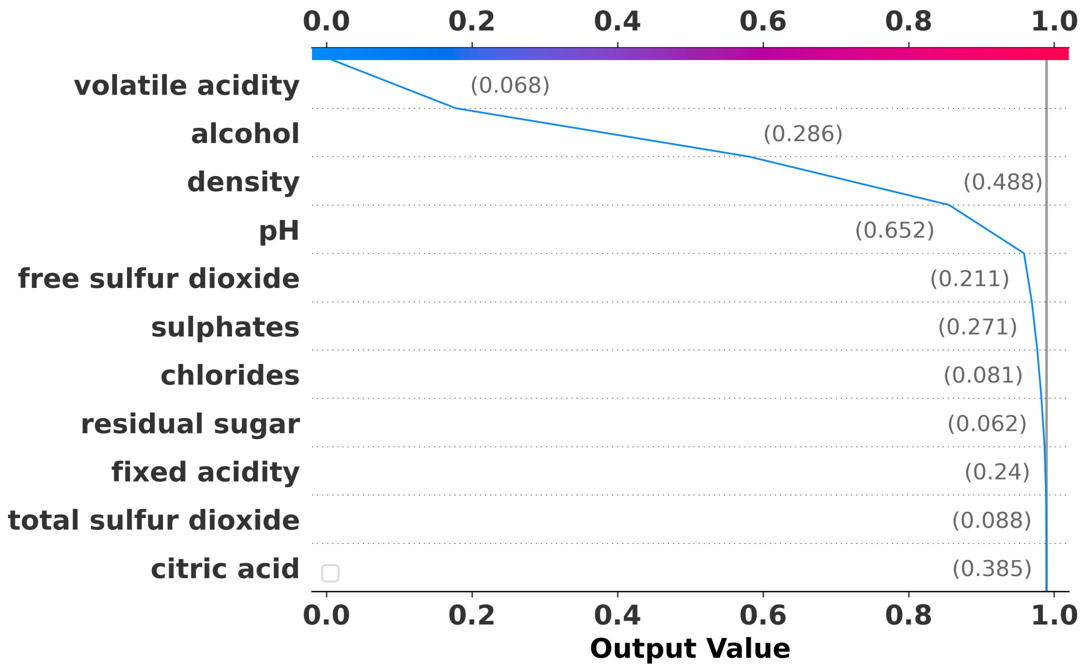


Figure 5.11: Model interpreation with expected value using SHAP on Wine quality dataset.

5.8 Conclusions

The study proposed and evaluated the performance of BSGAN approaches to handle class imbalance problems using four highly imbalanced datasets. The proposed BSGAN method outperformed Borderline-SMOTE and GAN-based oversampling techniques in various statistical measures. Furthermore, comparing the proposed state-of-the-art techniques using Neural Network approaches demonstrated better performance than many of the existing recent reference literature, as highlighted in Tables 5.5– 5.8. The inter-class distance measurement ensures that the data distribution follows Gaussian distribution after data expansion using BSGAN, as referred to in Table 7.5. The findings of the proposed techniques should provide some insights to researchers and practitioners regarding the advantage of GAN-based approaches and help to understand how they can potentially minimize the marginalization and sensitivity issues of the existing oversampling techniques. Future works include but are not limited to applying BSGAN on other high imbalance and big datasets, experimenting with mixed data (numerical, categorical, and image data), changing the parameters of the proposed models, and testing it for multiclass classification.

Chapter 6

Deep Learning-Based COVID-19 Diagnosis using Chest X-ray: An Analysis of Data Balancing Techniques

6.1 Introduction

In recent years, Deep Learning (DL) based approaches have shown promising results in healthcare diagnostics, particularly for critical diseases such as heart disease, cancer, and diabetes (Ahsan, Luna, & Siddique, 2022; Krawczyk, Galar, Jeleń, & Herrera, 2016; J. Shen et al., 2021). However, imbalanced datasets pose a common challenge in this field. Several studies have proposed various solutions to address this issue, including ensemble-based solutions, preprocessing techniques, resampling approaches, and DL-based approaches. For instance, Krawczyk et al. (2016) proposed Ensemble of Under-Sampling Boosting (EusBoost), an ensemble-based solution, to diagnose malignancy in a breast cancer dataset, achieving high sensitivity and Area Under the Curve (AUC) (Krawczyk et al., 2016). Majid et al. (2014) employed K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) in a two-step process to detect human breast and colon cancer, achieving an accuracy of 96.71% (Majid, Ali, Iqbal, & Kausar, 2014). Shen et al. (2021) proposed a hybrid improved SMOTE and adaptive SVM method to predict the postoperative survival of lung cancer patients, achieving an accuracy of 95.11% (J. Shen et al., 2021). Rath et al. (2021) used Generative Adversarial Networks (GAN) and Long Short-Term Memory (LSTM) to detect heart disease, achieving 99.2% accuracy (Rath, Mishra, Panda, & Satapathy, 2021). Reddy et al. (2020) employed resampling approaches to address the imbalance issue in stroke detection, achieving 99.8% accuracy (Reddy, Seshadri, Bothra, Suhas, & Thundiyil, 2020). Rai et al. (2021) used Synthetic Minority Over-sampling Technique (SMOTE) and Tomek Link data sampling approaches to automate Myocardial Infarction (MI) prognosis, achieving 99.8% accuracy (Rai, Chatterjee, Dubey, & Srivastava, 2021). Satu et al. (2020) used various ML approaches, with Random Forest outperforming the others, obtaining an accuracy of 99.067% in detecting type 2 diabetes (Shahriare Satu, Atik, & Moni, 2020). Roy et al. (2021) proposed an improved ML framework for classifying type 2 diabetes, achieving an accuracy of almost 98% (Roy et al., 2021). Dagliati et al. (2018) used Random Forest (RF) and Over-Sampling to detect type 2 diabetes mellitus, predicting the occurrence of retinopathy, neuropathy, or nephropathy in different time scenarios (Dagliati et al., 2018).

While these studies show promising results, they also have limitations, such as the need for manual parameter reconfiguration, missing value, and data noise issues. Further research is needed to address these limitations and improve the accuracy and effectiveness of Machine Learning (ML) and DL-based models in healthcare diagnostics.

6.2 Motivation

This study focuses on the impact of DL on the analysis of imbalanced data in healthcare diagnostics. The research question at the center of this study is RQ3, which is “How does imbalanced data affect the performance of DL-based models?”. Here, the study aims to address this question in the context of healthcare diagnostics. Due to time constraints, evaluating the performance of imbalanced data for all existing diseases was impossible. However, given the recent outbreak of Coronavirus Disease 2019 (COVID-19) and the need for scientific contributions, this study focused on DL-based chest radiograph (X-ray) image analysis for patients with COVID-19 symptoms. Therefore, this chapter aims to address the research question more precisely as follows: “How does imbalanced data affect the performance of DL-based models in COVID-19 diagnosis?” To address the research question, the present study:

- First, analyzed existing research works, their contributions, and limitations.
- Subsequently, utilized an updated transfer learning-based approach to improve performance and overcome the limitations of previous studies.
- Finally, compared the study’s performance with the referenced literature to provide a broader view of the study’s overall contributions.

6.3 Chapter Outline

The following chapter is organized as follows: Section 10.4 provides a brief background on COVID-19 and DL-based diagnosis. Section 10.5 outlines the methodology of the experiment in detail. The results of the experiment are presented in Section 10.6, followed by a discussion of the study's findings in Section 10.7. Finally, Section 10.8 summarizes the overall conclusions of the research and suggests potential avenues for future research.

6.4 Background

The Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2), previously known as the Novel Coronavirus, was first reported in Wuhan, China and rapidly spread around the world, pushing the World Health Organization (WHO) to declare the outbreak of the virus as a global pandemic and health emergency on March 11, 2020. According to official data, 19 million people have been infected worldwide, with the number of deaths surpassing $700,000$, and 12 million recovery cases reported by August 6, 2020 (Dashbord, June,2020). In the United States, the first case was reported on January 20, 2020, which evolved into a current number of confirmed cases, deaths, and recovered patients reaching more than 5 million, $162,000$, and 2.5 million, respectively (August 6, 2020 data) (Dashbord, June,2020).

COVID-19 can be transmitted in several ways. The virus can spread quickly among humans via community transmission, such as close contact between individuals, and the transfer of respiratory droplets produced via coughing, sneezing, and talking. Several symptoms have been reported so far, including fever, tiredness, and dry cough as the most common. Additionally, aches, pain, nasal congestion, runny nose, sore throat, and diarrhea have also been associated with the disease (Rothan & Byrareddy, 2020; Pan et al., 2020). Several methods can be followed to detect SARS-CoV-2 infection (M. Shen et al., 2020), including:

- Real-Time Reverse Transcription Polymerase Chain Reaction (RT-PCR)-based methods
- Isothermal nucleic acid amplification-based methods
- Microarray-based methods.

Health authorities in most countries have chosen to adopt the RT-PCR method, as it is regarded as the gold-standard in diagnosing viral and bacterial infections at the molecular level (Tahamtan & Ardebili, 2020). However, due to the rapidly increasing number of new cases and limited healthcare infrastructure, rapid detection or mass testing is required to lower the curve of infection. Recent studies claimed that chest Computed Tomography (CT) has the capability to detect the disease promptly. Therefore, in China, to deal with many new cases, CT scans were used for the initial screening of patients with COVID-19 symptoms (Hu et al., 2020; Bai et al., 2020; Y. Li & Xia, 2020; Bernheim et al., 2020). Similarly, chest radiograph (X-ray) image-based diagnosis may be a more attractive and readily available method for detecting the onset of the disease due to its low cost and fast image acquisition procedure. In our study, we investigate recent literature on the topic and tackle the opportunity to present an effective deep learning-based screening method to detect patients with COVID-19 from chest X-ray images.

In the recent past, the adoption of Artificial Intelligence (AI) in the field of infectious disease diagnosis has gained notable prominence, which led to the investigation of its potential in the fight against the novel coronavirus(Santosh, 2020; Ghoshal & Tucker, 2020; Narin, Kaya, & Pamuk, 2020). Current AI-related research efforts on COVID-19 detection using chest CT and X-ray images are discussed below to provide a brief insight on the topic and highlight our motivations to research it further.

6.4.1 Chest X-ray Based Screening

Preliminary studies have used Transfer Learning (TL) techniques to evaluate COVID-19 and pneumonia cases in the early stages of the COVID-19 pandemic (Khalifa, Taha, Hassanien, & Elghamrawy, 2020; Minaee, Kafieh, Sonka, Yazdani, & Soufi, 2020; Afshar et al., 2020; Ahsan, Gupta, et al., 2020). However, data insufficiency also hinders the ability of such proposed models to provide reliable COVID-19 screening tools based on chest X-ray (Narin et al., 2020; Hemdan, Shouman, & Karar, 2020; Sethy & Behera, 2020). For instance, Hemdan et al. (2020) proposed a CNN-based model adapted from Visual Geometry Group (VGG)19 and achieved 90% accuracy using 50 images (Hemdan et al., 2020). Ahsan et al. (2020) developed a COVID-19 diagnosis model using Multilayer Perceptron and Convolutional Neural Network (MLP-CNN) for mixed-data (numerical/categorical and image data). The model predicts and differentiates between 112 COVID-19 and 30 non-COVID-19 patients, with a higher accuracy of 95.4% (Ahsan, E Alam, et al., 2020). Sethy and Behera (2020) also considered only 50 images and used Residual Network (ResNet)50 for COVID-19 patients classification, and ultimately reached 95% accuracy (Sethy & Behera, 2020). Also, Narin et al. (2020) used 100 images and achieved 86% accuracy using Inception Residual Network Version 2 (InceptionResNetV2) (Narin et al., 2020). As noted, these studies use relatively small datasets, which does not guarantee whether their proposed models would perform equally well on larger datasets. Also, the possibility of a model overfitting is another concern for larger CNN-based networks when trained with small datasets.

In view of these issues, recent studies proposed model training with larger datasets and reported a better performance compared to smaller ones (Brunese, Mercaldo, Reginelli, & Santone, 2020a; Apostolopoulos & Mpesiana, 2020; Ozturk et al., 2020; Khan, Shah, & Bhat, 2020). Chandra et al. (2020) developed an automatic COVID screening system to detect infected patients using 2088 (696 normal, 696 pneumonia, and 696 COVID-19) and 258 (86 images of each category) chest X-ray images, and achieved 98% accuracy (T. B. Chandra, Verma, Singh, Jain, & Netam, 2020). Sekeroglu et al. (2020) developed a DL-based method to detect COVID-19 using publicly available X-ray images (1583 healthy, 4292 pneumonia, and 225 confirmed COVID-19), which involved the training of DL and ML classifiers (Sekeroglu & Ozsahin, 2020). Pandit et al. (2020) explored pre-trained VGG-16 using 1428 chest X-rays with a mix of confirmed COVID-19, common bacterial pneumonia, and healthy cases (no infection). Their results showed an accuracy of 96% and 92.5% in two and three output class cases (Pandit, Banday, Naaz, & Chishti, 2020). Ghosal and Tucker (2020) used 5941 chest X-ray images and obtained 92.9% accuracy (Ghoshal & Tucker, 2020). Brunese et al. (2020) proposed a modified VGG16 model and achieved 99% accuracy with a dataset of 6505 images. However, they have used fairly balanced data with a 1 : 1.17 ratio; 3003 COVID-19 and 3520 other patients. It is not immediately clear how their model would perform on an imbalanced dataset (Brunese, Mercaldo, Reginelli, & Santone, 2020b). On the other hand, Khan et al. (2020) developed a model based on Xception CNN techniques considering 284 COVID-19 patients and 967 other patients (data ratio 1 : 3.4). Partially as an effect of a more imbalanced dataset, their reported accuracy was comparatively low, reaching 89.6% (Khan et al., 2020). On imbalanced datasets, there is a higher chance that the model may be biased on significant classes and might affect the overall performance of the model.

Developing DL models using small image datasets often results in the incorrect identification of regions of interest in those images, an issue not often addressed in the existing literature. Therefore, in the present work, the models' performance is analyzed layer by layer, and only the best-performing ones are selected based on the correct identification of infectious regions on X-ray images. Previous works often do not demonstrate how their proposed models perform with imbalanced datasets, which is often challenging. Here, small, imbalanced, and large datasets are considered, and a comprehensive description of the results is presented with statistical measures, including 95% confidence intervals, p-values, and t-values. A summary of the technical contributions is presented below:

- Modification and evaluation of six different deep CNN models (VGG16, InceptionResNetV2, ResNet50, Mobile Network Version 2 (MobileNetV2), ResNet101, VGG19) for detection of COVID-19 patients using X-ray image data on both balanced and imbalanced datasets; and
- Verify the possibility to locate affected regions on chest X-rays incorporated with heatmaps, including a cross-check with a medical doctor's opinion.

6.5 Research Methodology

Three separate studies are being evaluated and proposed, and each of them uses a distinct dataset. The details of these studies and the corresponding datasets are presented as follows:

1. Study One – smaller, balanced dataset: chest X-ray images of 25 patients with COVID-19 symptoms, and 25 images of patients with diagnosed pneumonia, obtained from the open-source repository shared by Dr. Joseph Cohen (Cohen, Morrison, & Dao, 2020).
2. Study Two – larger, imbalanced dataset: chest X-ray images of 262 patients with COVID-19 symptoms, and 1583 images of patients with diagnosed pneumonia, obtained from the Kaggle COVID-19 chest X-ray dataset (*COVID-19 chest xray*, June, 2020).

3. Study Three – multiclass dataset: chest X-ray images of 219 patients with COVID-19 symptoms, 1345 images of patients with diagnosed pneumonia and 1073 images of normal patients, also obtained from the Kaggle COVID-19 chest X-ray dataset (*Chest X-Ray Images (Pneumonia)*, n.d.).

Figure 6.1 presents a set of representative chest X-ray images of both COVID-19 and pneumonia patients from the aforementioned datasets. Table 6.1 details the overall assignment of data for training and testing of each investigated CNN model. In both studies, six different Deep Learning approaches were investigated: VGG16 (Simonyan & Zisserman, 2014), InceptionResNetV2 (Szegedy, Ioffe, Vanhoucke, & Alemi, 2017), ResNet50 (Akiba, Suzuki, & Fukuda, 2017), MobileNetV2 (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018), ResNet101 (K. He, Zhang, Ren, & Sun, 2016) and VGG19 (Simonyan & Zisserman, 2014).

Table 6.1: Assignment of data used for training and testing of Deep Learning models.

Study	Label	Train	Test
One	COVID-19	20	5
	Normal	20	5
Two	COVID-19	210	52
	Normal	1266	317
Three	COVID-19	176	43
	Normal	1073	268
	Pneumonia	1076	269

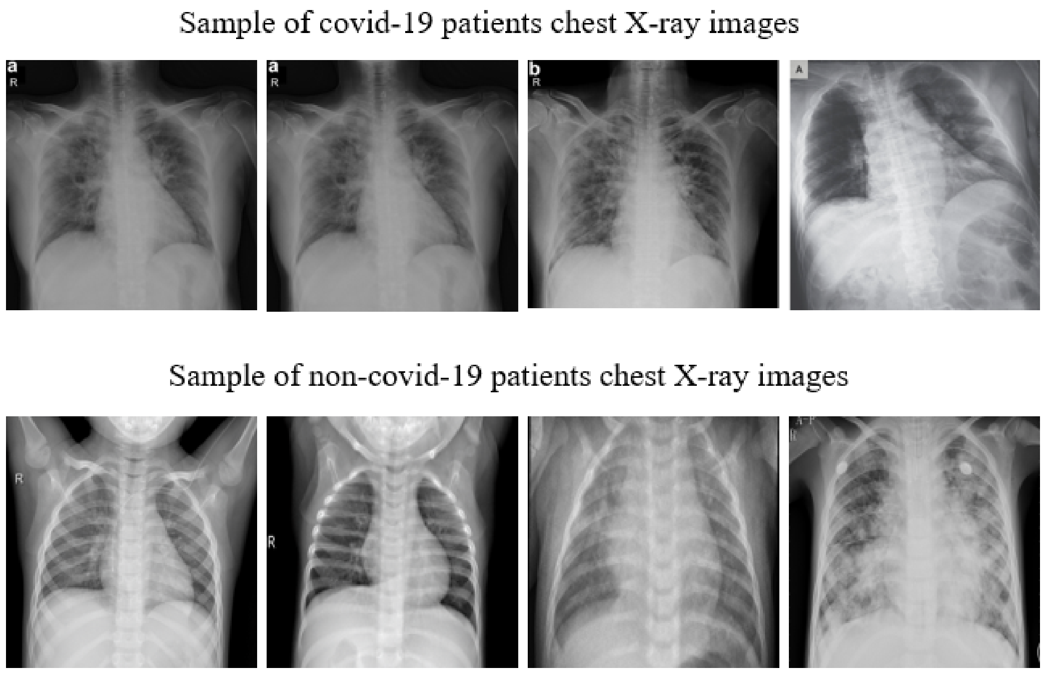

Sample of covid-19 patients chest X-ray images
Sample of non-covid-19 patients chest X-ray images

Figure 6.1: Representative samples of chest X-ray images from the open source data repositories(Cohen et al., 2020) used in our proposed studies.

6.5.1 Deep Learning Algorithms

6.5.1.1 VGG

VGG, or Visual Geometry Group, proposed two CNN models that are extremely deep in their architecture. These models are referred to as VGG16 and VGG19, respectively, and consist of 16 and 19 layers. The models were trained on a dataset containing one million samples from the ImageNet dataset. The VGG16 model is designed to take input images of size 224 by 224. Images are first passed through a series of Convolutional layers containing filters ranging from 64 to 512. The pooling layer is implemented using a 2-by-2 filter with max-pooling. Rectified

Linear Unit (ReLU) is employed as the activation function with a stride size of 2. The pooling layer is then converted to a 1D vector in the dense layer. Finally, the Softmax activation function is applied to the output layer to classify the samples into one of the 1000 available categories (Simonyan & Zisserman, 2014).

6.5.1.2 InceptionResNetV2

InceptionResNetv2 is a DL architecture proposed by Szegedy et al. in 2017. It is an extension of the Inception and ResNet architectures, which combines the benefits of both architectures for image classification tasks. IRv2 has 164 layers and merges 1x1, 3x3, and 5x5 convolutions with max pooling to extract features from the input image. The residual connections allow for the training of deeper networks while preventing the vanishing gradient problem. The architecture also incorporates batch normalization and the ReLU activation function for improved training speed and accuracy. IRv2 achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge dataset in 2016, with a top-5 accuracy of 95.3% (Szegedy et al., 2017).

6.5.1.3 ResNet

The ResNet DNN, which stands for Residual Network, was developed by Microsoft researchers in 2015 to address the problem of disappearing gradients in very deep neural networks. One layer of maximum pooling, one layer of average pooling, and one layer of regular pooling are used in ResNet50, a variation of ResNet. The ResNet50 design starts with a 7x7 convolution layer with 64 kernels and a stride size of 2, followed by a max-pooling layer also with a stride size of 2. The network then applies nine convolutional layers using, successively, 64, 64, and 256 kernel filters. Each of the network's last nine layers has 512, 512, and 2048 kernel filters. The Softmax function is then used as the activation function for the output layer, which is followed by an FC layer with 1000 nodes that incorporates an average pooling layer. ResNet101's model design is similar to ResNet50's, with the addition of a third layer made up of 256, 256, and 1024 filters in the fourth block (K. He et al., 2016; Ahsan et al., 2023; Ahsan, Abdullah, et al., 2022).

6.5.1.4 MobileNetV2

MobileNetV2 is a state-of-the-art module that implements inverted residuals and a linear bottleneck. This simple model can be extended for use in mobile phone applications, where it can accept low-dimensional input, reduce the number of operations, and consume less memory while maintaining a higher degree of accuracy. This model uses depthwise separable convolution, which divides the convolution operation into two distinct layers. The first layer is depth-wise convolution, which is highly efficient because it performs just one filtering operation. The second layer acquires additional features due to the linear calculation of the inputs. MobileNetV2 has significantly reduced the computational cost of standard model layers by a factor of k2, often saving 8 or 9 times the computational resources required for 3×3 depth-wise separable convolution compared to other conventional models (Sandler et al., 2018).

6.5.2 Using Pre-Trained Convet

A pre-trained network is a network that was previously trained on a larger dataset which, in most cases, is enough to learn a unique hierarchy to extract features from. It works more effectively on small datasets. A prime example is the VGG16 architecture, developed by Simoyan and Zisserman (2014) (Simonyan & Zisserman, 2014). Figure **??** shows a sample architecture of the pre-trained model procedure. All models implemented in this study are available as a pre-package within Keras (Chollet, 2017).

Figure 6.2 demonstrates a fine-tuning sequence on the VGG16 network. The modified architecture follows the steps below:

1. Firstly, the models were initiated with a pre-trained network without a Fully Connected (FC) layer.

2. Then, an entirely new connected layer added a pooling layer and "softmax" as an activation function, appended it on top of the VGG16 model.

3. Finally, the convolution weight was frozen during the training phase so that only the FC layer should train during the experiment.

The same procedure was followed for all other DL techniques. In this experiment, the additional modification of the model for all CNN architectures was constructed as follows: $AveragePooling2D(Poolsize = (4,4)) \rightarrow Flatten \rightarrow Dense \rightarrow Dropout(0.5) \rightarrow Dense(Activation = \text{``}softmax\text{''})$.

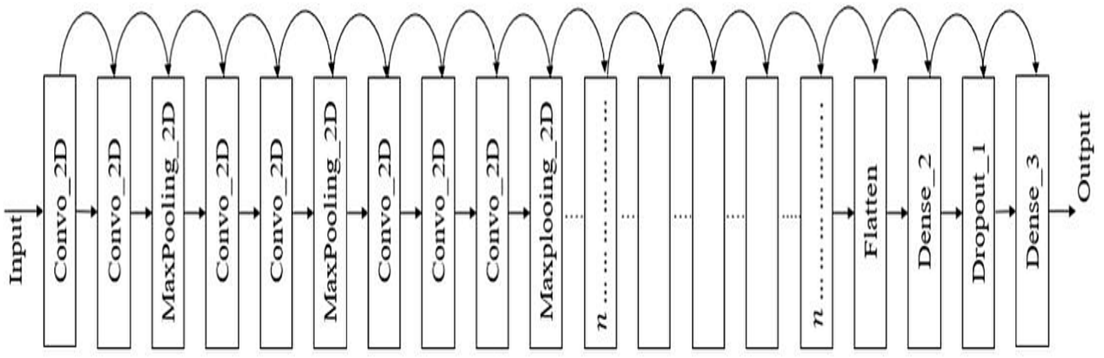


Figure 6.2: VGG16 architecture used during this experiment.

As it is known, most pre-trained models contain multiple layers which are associated with different parameters (i.e., number of filters, kernel size, number of hidden layers, number of neurons) (Denil, Shakibi, Dinh, Ranzato, & De Freitas, 2013). However, manually tuning those parameters is considerably time consuming (Hutter, Lücke, & Schmidt-Thieme, 2015; Qolomany, Maabreh, Al-Fuqaha, Gupta, & Benhaddou, 2017). With that in mind, three parameters of the DL model are optimized during this study: batch size [1], epochs [2], and learning rate [3] (inspired by (L. N. Smith, 2018; S. L. Smith, Kindermans, Ying, & Le, 2017)). The grid search method, which is commonly utilized for parameter tuning, was employed in this study (Bergstra & Bengio, 2012). Initially, the following parameters are randomly selected:

$$\text{Batch size} = [4, 5, 8, 10]$$

[1]Batch size characterizes the number of samples to work through before updating the internal model parameters (Brownlee, 2018)

[2]It defines how many times the learning algorithm will work through the entire dataset (Brownlee, 2018)

[3]It is a hyper-parameter that controls the amount to change the model in order to calculate the error each time the model weights are updated (B. Chandra & Sharma, 2016)

$$\text{Number of epochs} = [10, 20, 30, 40]$$

$$\text{Learning rate} = [.001, .01, 0.1]$$

For Study One, better results were achieved using the grid search method with the following parameters:

$$\text{Batch size} = 8$$

$$\text{Number of epochs} = 30$$

$$\text{Learning rate} = .001$$

Similarly, for Study Two, the best results were achieved with:

$$\text{Batch size} = 50$$

$$\text{Number of epochs} = 50$$

$$\text{Learning rate} = .001$$

Finally, during Study Three, best performance was achieved with:

$$\text{Batch size} = 50$$

$$\text{Number of epochs} = 100$$

$$\text{Learning rate} = .001$$

The adaptive learning rate optimization algorithm (Adam) was used as an optimization algorithm for all models due to its robust performance on binary image classification (Perez & Wang, 2017; Filipczuk, Fevens, Krzyżak, & Monczak, 2013). As commonly adopted in data mining techniques, this study used 80% data for training, whereas the remaining 20% was used for testing (Mohanty, Hughes, & Salathé, 2016; Menzies, Greenwald, & Frank, 2006; Stolfo, Fan, Lee, Prodromidis, & Chan, 2000). Each study was conducted twice, and the final result was represented as the average of those two experiment outcomes, as suggested by Zhang et al. (2020) (J. Zhang, Xie, Pang, et al., 2020). Performance results were presented as model accuracy, precision, recall, and F1-score (Ahsan, Li, et al., 2020).

$$\text{Accuracy} = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \tag{6.1}$$

$$\text{Precision} = \frac{t_p}{t_p + f_p} \tag{6.2}$$

$$\text{Recall} = \frac{t_p}{t_n + f_p} \tag{6.3}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{ Recall}}{\text{Precision+Recall}} \tag{6.4}$$

Where,

- True Positive (t_p)= COVID-19 patient classified as patient
- False Positive (f_p)= Healthy people classified as patient
- True Negative (t_n)=Healthy people classified as healthy
- False Negative (f_n)= COVID-19 patient classified as healthy.

6.6 Results

6.6.1 Study One

The overall model performance for all CNN approaches was measured both on the training (40 images) and test (10 images) sets using equation 6.1– 6.4. Table 6.2 presents the results of the training set. In this case, VGG16 and MobileNetV2 outperformed all other models in terms of accuracy, precision, recall, and F1-score. In contrast, the ResNet50 model showed the worst performance across all measures.

Table 6.2: Study One model performance on train set.

Model	Accuracy	Precision	Recall	F1-score
VGG16	100%	100%	100%	100%
InceptionResNetV2	97%	98%	97%	97%
ResNet50	70%	81%	70%	67%
MobileNetV2	100%	100%	100%	100%
ResNet101	80%	83%	80%	80%
VGG19	93%	93%	93%	92%

Table 6.3 presents the performance results for all models on the test set. Models VGG16 and MobileNetV2 showed 100% performance across all measures. On the other hand, ResNet50, ResNet101, and VGG19 demonstrated significantly worse results.

Table 6.3: Study One model performance on test set.

Model	Accuracy	Precision	Recall	F1-score
VGG16	100%	100%	100%	100%
InceptionResNetV2	97%	98%	97%	97%
ResNet50	70%	81%	70%	67%
MobileNetV2	100%	100%	100%	100%
ResNet101	70%	81%	70%	67%
VGG19	70%	71%	70%	70%

6.6.1.1 Confusion Matrix

Confusion matrices were used to better visualize the overall performance of prediction. The test set contains 10 samples (5 COVID-19 and 5 other patients). In accordance with the performance results previously presented, Figure 6.3 shows that the VGG16, InceptionResNetV2, and MobileNetV2 models correctly classified all patients. In contrast, models ResNet50, and ResNet101 incorrectly classified 3 non-COVID-19 patients as COVID-19 patients, and models VGG19 classified 2 non-COVID patients as COVID-19 patients while also classifying 1 COVID-19 patient as non-COVID-19.

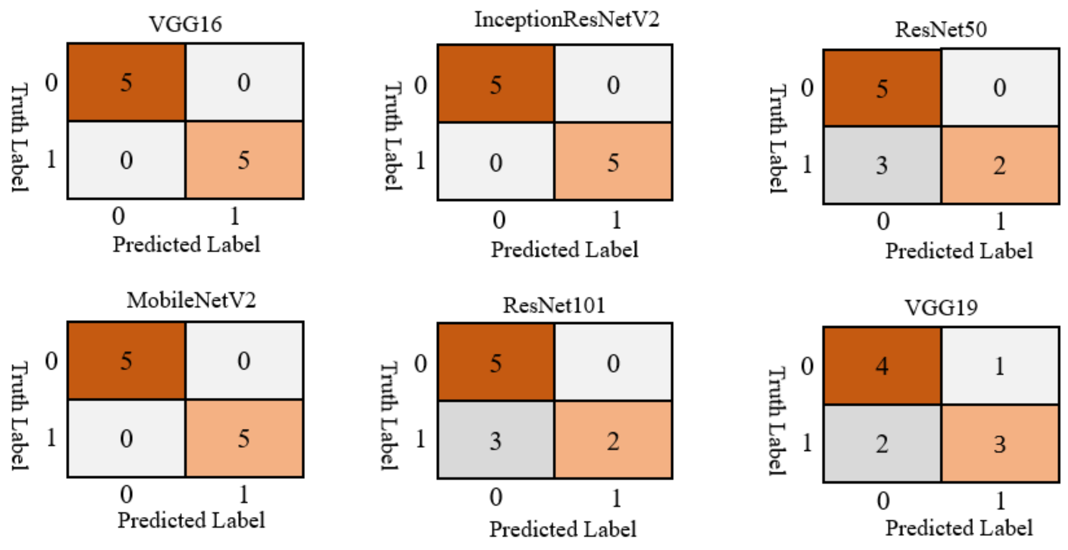


Figure 6.3: Study One confusion matrices for six different deep learning models applied on the test set.

6.6.1.2 Model Accuracy

Figure 6.4 shows the overall training and validation accuracy during each epoch for all models. Models VGG16 and MobileNetV2 demonstrated higher accuracy at epochs 25 to 30, while VGG19, ResNet50, and ResNet101 displayed lower accuracy which sporadically fluctuated between epochs 10.

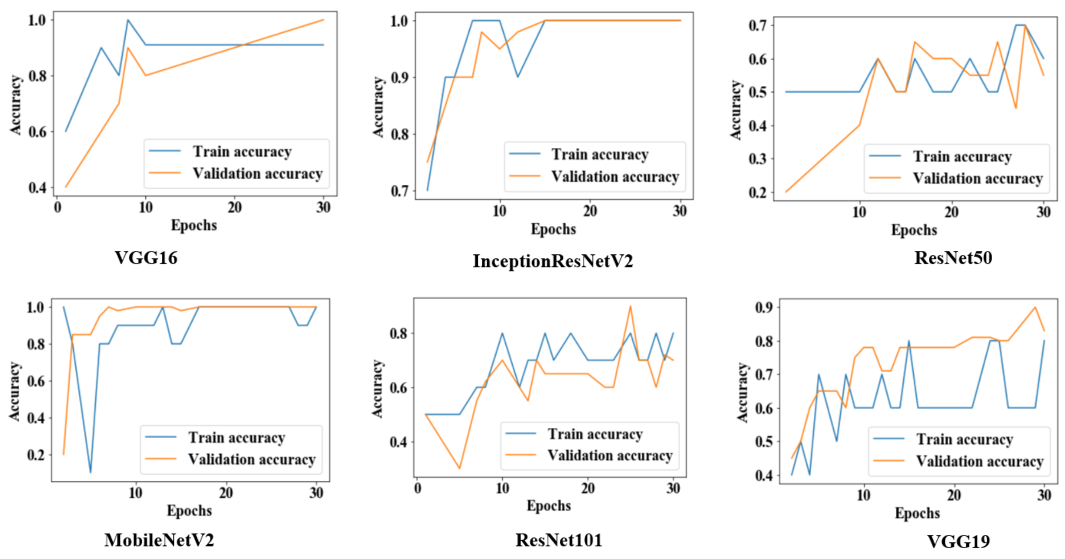


Figure 6.4: Training and validation accuracy throughout the execution of each model in Study One.

6.6.1.3 Model Loss

Figure 6.5 shows that both training loss and validation loss were reduced following each epoch for VGG16, InceptionResNetV2, and MobileNetV2. In contrast, for VGG19, both measures are scattered over time, which is an indicative of poor performance.

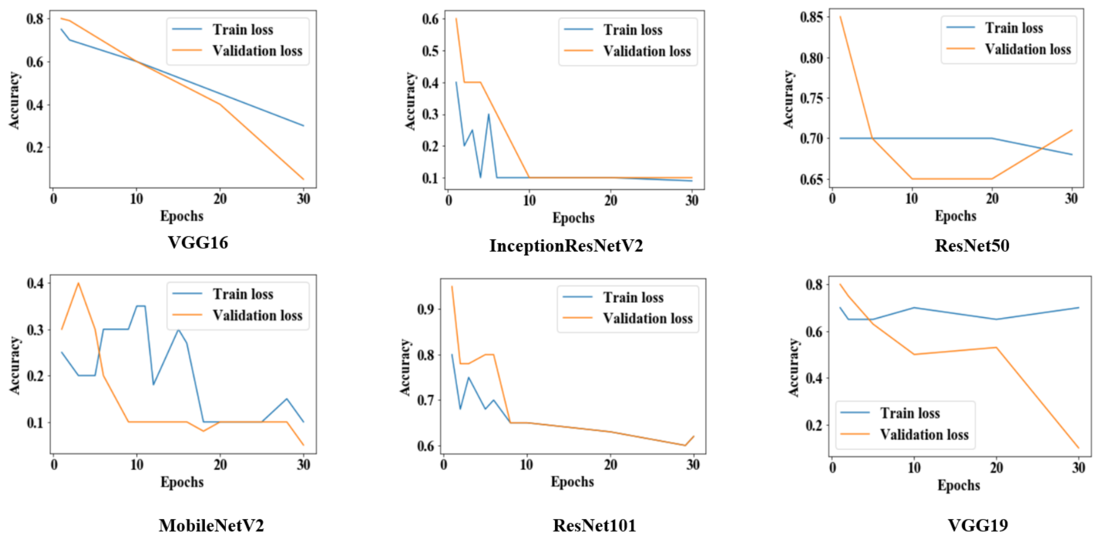


Figure 6.5: Training and validation loss throughout the execution of each model in Study One.

6.6.2 Study Two

For Study Two, on the training set, most model accuracies were measured above 90%. Table 6.4 shows that 100% accuracy, precision, recall, and F1-score were achieved using MobileNetV2. Among all other models, ResNet50 showed the worst performance across all measures.

Table 6.4: Study Two model performance on train set.

Model	Accuracy	Precision	Recall	F1-score
VGG16	99%	99%	99%	99%
InceptionResNetV2	99%	100%	99%	99%
ResNet50	93%	96%	93%	93%
MobileNetV2	100%	100%	100%	100%
ResNet101	96%	97%	88%	92%
VGG19	99%	98%	96%	97%

Table 6.5 presents the performance results for all models on the test set. Models VGG16, InceptionResNetV2, and MobileNetV2 showed 99% accuracy; however, the precision, recall, and F1-score were distinct for each model, yet all above 97%. On the lower end, ResNet50 demonstrated relatively lower performance across all measures.

Table 6.5: Study Two model performance on test set.

Model	Accuracy	Precision	Recall	F1-score
VGG16	99%	100%	97%	98%
InceptionResNetV2	99%	99%	98%	98%
ResNet50	93%	94%	93%	92%
MobileNetV2	99%	99%	99%	99%
ResNet101	96%	97%	88%	92%
VGG19	97%	97%	91%	94%

6.6.3 Confusion Matrix

Figure 6.6 shows that most of the models performance is satisfactory on the test set. In Study Two, classification accuracy for ResNet50 and ResNet101 is significantly better compared to Study One, possibly as an effect of the models being trained with more data and more epochs. In general, MobileNetV 2 performed better among all the models and misclassified only 2 images out of 369 images, while ResNet50 showed lower performance and misinterpreted 25 images out of 369 images.

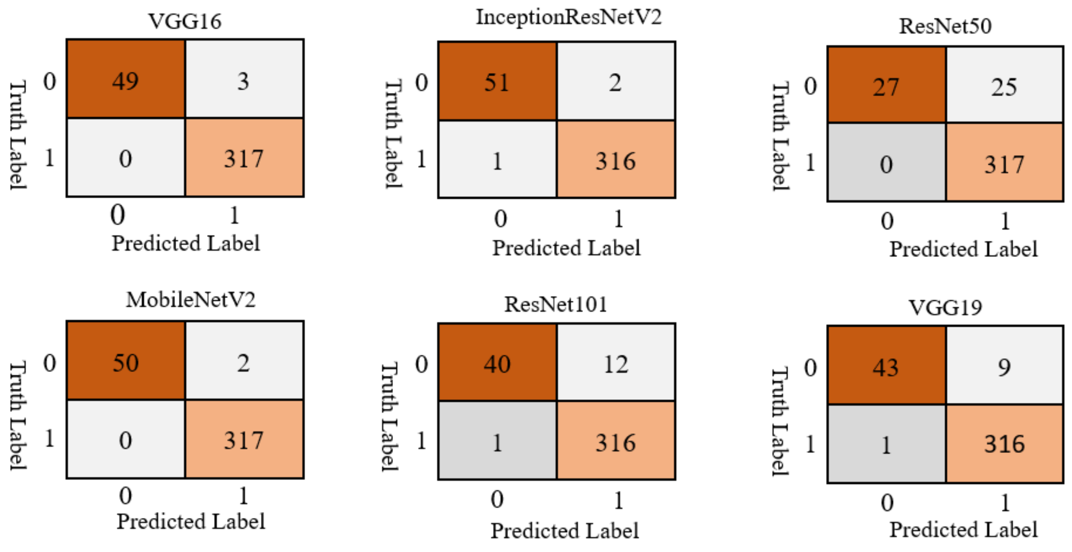


Figure 6.6: Study Two confusion matrices for six different deep learning models applied on the test set.

6.6.3.1 Model Accuracy

Figure 6.7 suggests that the overall training and validation accuracy were more steady during Study Two than Study One. The performance of ResNet50 and

ResNet101 significantly improved once trained with more data (1845 images) and more epochs (50 epochs).

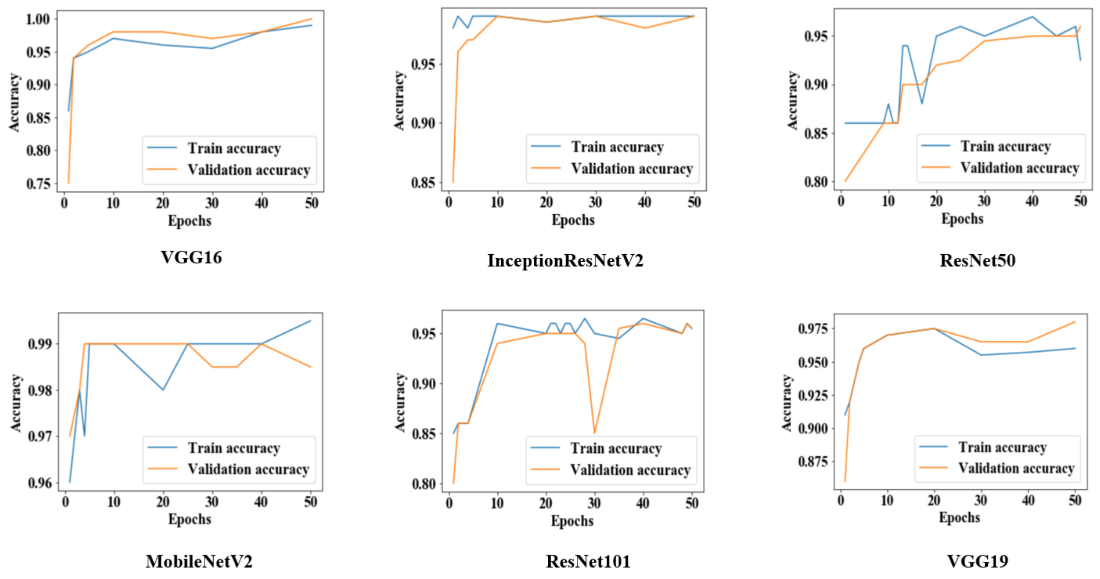


Figure 6.7: Training and validation accuracy throughout the execution of each model in Study Two.

6.6.3.2 Model Loss

Figure 6.8 provides evidence that both training and validation losses were minimized following each epoch for all models, potentially as an effect of the increased batch size, number of epochs, and data amount.

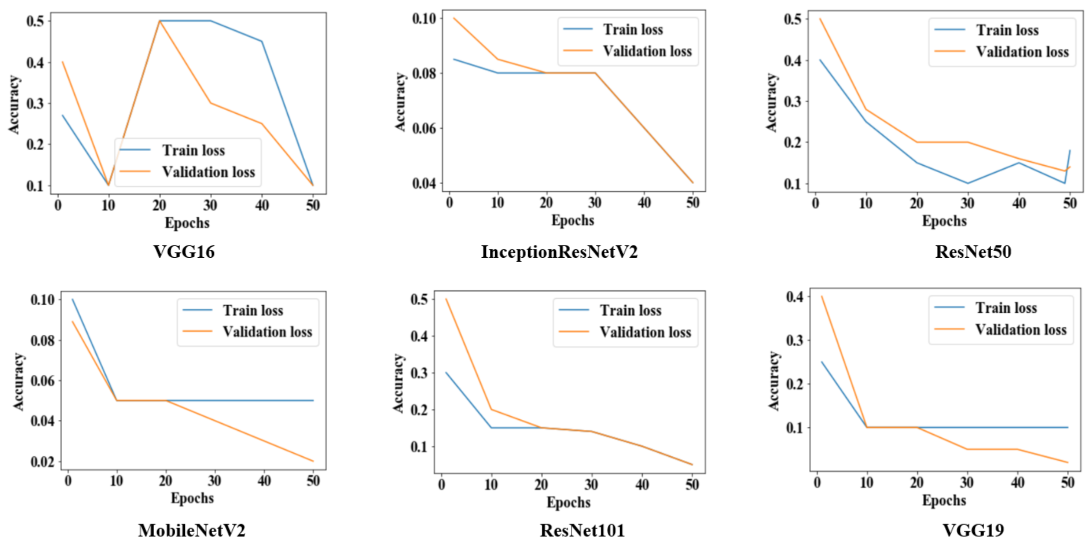


Figure 6.8: Training and validation loss throughout the execution of each model in Study Two.

6.6.4 Study Three

As a means of highlighting the potential of the proposed models with more complex classifications, a small-scale pilot study was conducted to assess the performance of the VGG16 model on a multi-class dataset. The performance outcomes for the train and test runs are presented in Table 6.6. The accuracy remained above 90% on both runs, which suggests a notably high performance of our model with either binary or multi-class datasets.

Table 6.6: VGG16 model performance on train and test datasets of Study Three.

VGG16	Accuracy	Precision	Recall	F1-score
Train	93%	93%	88%	91%
Test	91%	90%	91%	86%

6.6.5 Test Results with Confidence Intervals

Table 6.7 presents 95% confidence intervals for model accuracy on the test sets for Studies One and Two. For instance, in Study One, the average accuracies for VGG16 and MobileNetV2 were found to be 100%; however, the Wilson score and Bayesian interval show that the estimated accuracies lie between 72.2% to 100% and 78.3% to 100%, respectively. On the other hand, Study Two reported relatively narrower interval ranges.

Table 6.7: Confidence Interval ($\alpha = 0.05$) for studies One and Two on test accuracy.

Study	Model	Actual accuracy	*Methods*	
			Wilson Score	Bayesian Interval
One	VGG16	100%	72.2% – 100%	78.3% – 100%
	InceptionResNetV2	97%	68.1% – 99.8%	72.5% – 99.9%
	ResNet50	70%	39.7% – 89.2%	39.4% – 90.7%
	MobileNetV2	100%	72.2% – 100%	78.3% – 100%
	ResNet101	70%	39.7% – 89.2%	39.4% – 90.7%
	VGG19	70%	39.7% – 89.2%	39.4% – 90.7%
Two	VGG16	99%	97.6% – 99.7%	97.8% – 99.8%
	InceptionResNetV2	99%	98.0% – 99.9%	98.3% – 99.9%
	ResNet50	93%	90% – 95.4%	90.3% – 95.5%
	MobileNetV2	99%	97.6% – 99.7%	97.8% – 99.8%
	ResNet101	96%	94.1% – 97.9%	94.2% – 98.0%
	VGG19	97%	95.1% – 98.5%	95.2% – 98.6%

A paired t-test was conducted to compare model accuracies on both studies as shown in Table 6.8. There was no significant difference identified within the scores for Study One (M = 84.50, SD = 15.922) and Study Two (M = 97.39, SD = 2.38); $t(5) = 2.251$, $p = .074$. These results suggest that model accuracy is competent on both datasets and makes no statistically significant differences

($p > 0.05$).

Table 6.8: Descriptive statistics of paired t-test for Study One and Study Two. M – Mean; SD – Standard deviation; SEM – Standard error mean; DF – Degree of freedom.

Study	M	SD	SEM
One	84.50	15.922	6.50
Two	97.3967	2.38362	.9731
Paired Difference	-12.89	14.03	5.72
		Results	
t		-2.251	
DF		5	
$\alpha = .05$		.074	

6.7 Discussion

As a means of comparing the results with those available in the literature, Table 6.9 contrasts the accuracies of the three best performing CNN models on small datasets as part of Study One. It is relevant to emphasize that none of the referenced studies presents their results as confidence intervals, which hinders a direct comparison, but still allows for a higher-level assessment of the reported performance measures.

During this study, an accuracy range of 68.1% to 99.8% was achieved using InceptionResNetV2 with 50 chest X-ray images, while Narin et al. (2020) obtained 86% accuracy with 100 images (Narin et al., 2020). Hemdan et al. (2020) and

Sethy and Behera (2020) used small datasets of 50 images and acquired 90% and 9% accuracy using VGG19 and ResNet50 + SVM, respectively (Hemdan et al., 2020; Sethy & Behera, 2020).

Table 6.9: Different deep CNN models performance on small chest X-ray image dataset.

References	Model	Datasize	Accuracy
(Hemdan et al., 2020)	VGG19	50	90%
(Sethy & Behera, 2020)	ResNet50+SVM	50	95%
(Narin et al., 2020)	InceptionResNetV2	100	86%
Best model from	VGG16		72.2% – 100%
Study One with	InceptionResNetV2	50	68.1% – 99.8%
95% CI	MobileNetV2		72.2% – 100%

Additionally, In Study Two, some of the models—VGG16, InceptionResNetV2, MobileNetV2,VGG19— demonstrated almost similar accuracy while considering a highly imbalanced dataset than referenced literature (Ozturk et al., 2020; Khan et al., 2020) that also used imbalanced datasets (Table 6.10). For the imbalanced dataset, chest X-ray images of 262 COVID-19 patients and 1583 non-COVID-19 patients (1 : 6.04) have been used. Apostolopoulos and Mpesiana (2020) used 1428 chest X-ray images where the data ratio was 1 : 5.4 (224 COVID-19: 1208 others) and achieved 98% accuracy (Apostolopoulos & Mpesiana, 2020). Similarly, Khan et al. (2020) used 1251 chest X-ray images, data proportion 1 : 3.4 (284 COVID-19:967 others), and acquired 89.6% accuracy (Khan et al., 2020). In Study Two, some of the best models were acquired, including VGG16, VGG19, InceptionResNetV2, and MobileNetV2, with accuracies ranging from 97% to around 100%.

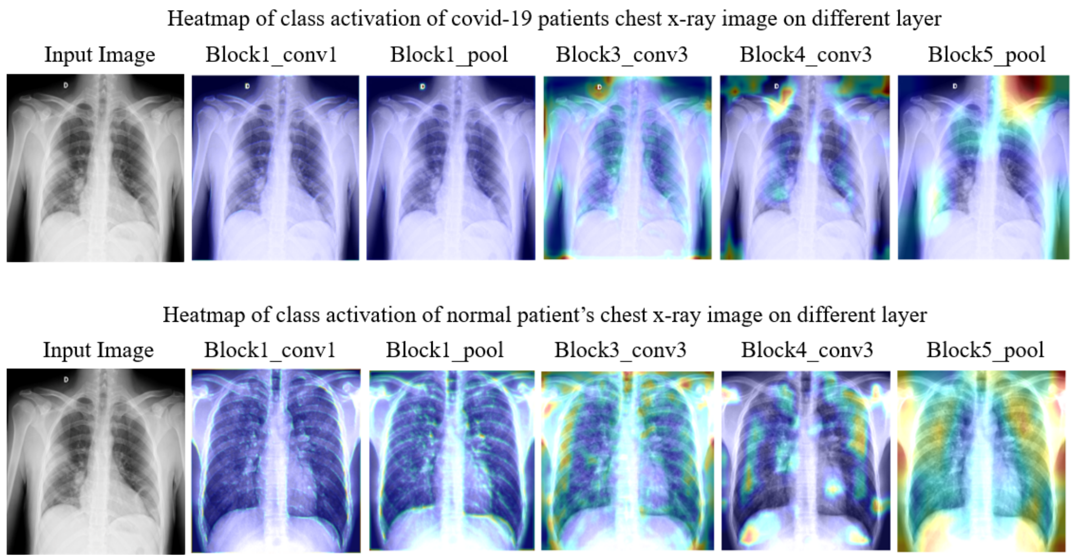

Heatmap of class activation of covid-19 patients chest x-ray image on different layer
Input Image
Block1_conv1
Block1_pool
Block3_conv3
Block4_conv3
Block5_pool
Heatmap of class activation of normal patient's chest x-ray image on different layer
Input Image
Block1_conv1
Block1_pool
Block3_conv3
Block4_conv3
Block5_pool

Figure 6.9: Heatmap of class activation on different layers.

Table 6.10: Comparison of models performance on imbalanced datasets.

References	Model	Datasize	Data ratio	Accuracy
(Brunese et al., 2020a)	VGG16	6505	3003 : 3520 ≈1:1.17	99%
(Apostolopoulos & Mpesiana, 2020)	CNN	1428	224 : 1208 ≈1:5.4	98%
(Ozturk et al., 2020)	DarkNet	1750	250 : 1500 = 1 : 6	98%
(Khan et al., 2020)	Xception	1251	284 : 967 ≈1:3.4	89.6%
(Lee et al., 2020)	VGG-16	1821	607 : 1214 = 1 : 2	95.9%
(Shelke et al., 2020)	DenseNet-VGG16	2271	500 : 1771 ≈1:3.54	95%
Best model from Study Two with 95% CI	VGG16	1845	262 : 1583≈1:6.04	97.6% – 99.7%
	InceptionResNetV2			97.6% – 99.7%
	MobileNetV2			97.6% – 99.7%
	VGG19			95.2% – 98.6%

6.7.1 Feature Selection

Figure 6.9 highlights extracted features as an effect of different CNN layers of VGG16 models applied on chest X-ray images from Study One. For instance, in block1_conv1 and block1_pool1, the extracted features were slightly fuzzy, while in block4_conv3 and block5_pool, those features become more visible/prominent. The heatmap also demonstrates a considerable difference in both COVID-19 and other patient images corresponding to each layer. For instance, as shown in Figure 6.10 (left), two specific regions were highlighted by heatmap for the COVID-19 patient's X-ray image, whereas for other patients' images, the areas were found to be haphazard and small.

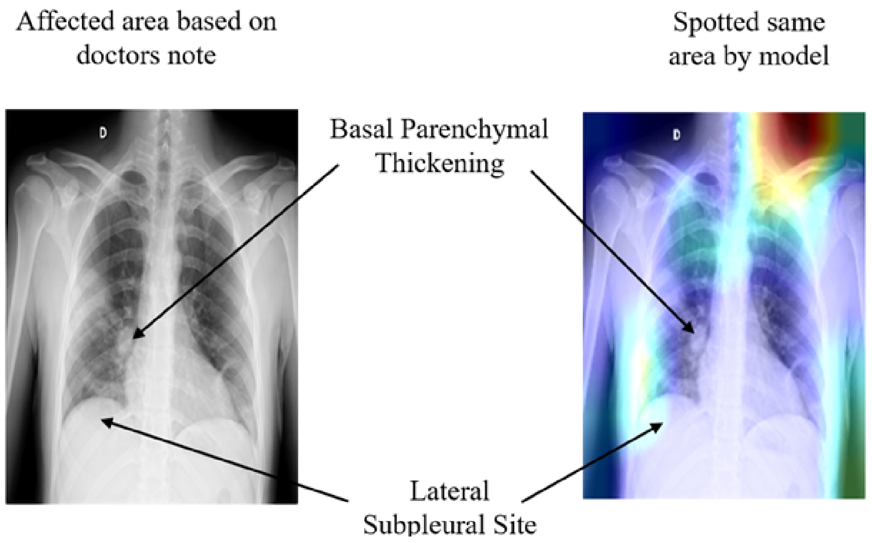

Affected area based on doctors note
Spotted same area by model
Basal Parenchymal Thickening
Lateral Subpleural Site

Figure 6.10: Model's ability to identify important features on chest X-ray using VGG16.

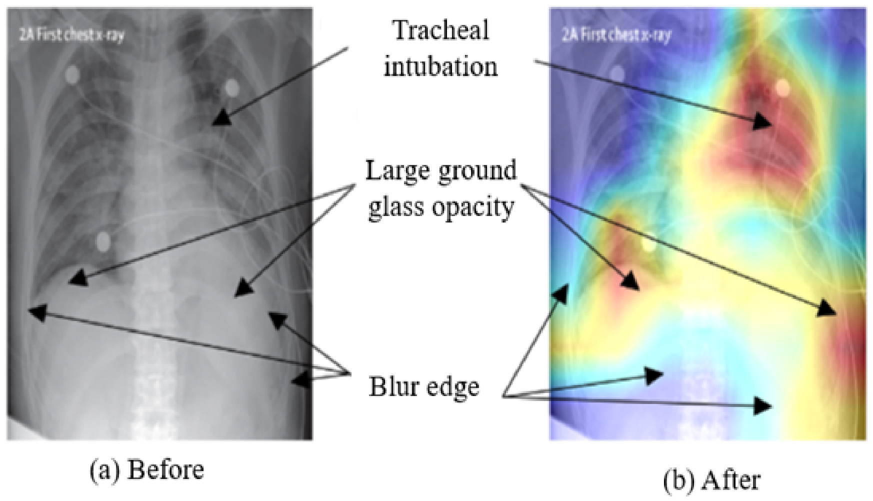


Figure 6.11: Model's competency to identify essential features on chest X-ray using MobileNetV2.

During the experiment, each layer plays a significant role in identifying essential features from the images in the training phase. As a result, it is also deemed possible to see which features are learned and play a crucial role in differentiating between the two classes. In Figure6.10, the left frame represents a chest X-ray image of a COVID-19 patient, and the right one highlights infectious regions of that same image, as spotted by the VGG16 model during Study One. The highlighted region on the upper right shoulder, which resulted from the individual layer of the VGG16 model (Study One), can be considered an irrelevant and therefore unnecessary feature identified by the network. The following topics extend the discussion on this issue:

1. The models attained unnecessary details from the images since the dataset is small compared to the model architecture (contains multiple CNN layers).
2. The models extracted features beyond the center of the images, which might not be essential to differentiate the COVID-19 patients from the non-COVID-19 patients.
3. The average age of COVID-19 patients in the first case study is 55.76 years. Therefore it is possible that individual patients might have age-related illnesses (i.e. weak/damaged lungs, shoulder disorder) , apart from complications related to COVID-19, which are not necessarily considered by the doctor's notes.

Interestingly, the these irrelevant regions spotted by our models decreased significantly when trained with larger datasets (1845 images) and increased epochs (50 epochs). For instance, Figure 6.11, presents the heatmap of the Conv-1 layer of MobileNetV2, acquired during the Study Two. The heatmap verifies that the spotted regions are very similar and match closely with the doctor's findings.

6.8 Overall Findings

This study aimed to investigate the impact of imbalanced data on the performance of DL-based models in COVID-19 diagnosis using chest X-ray images. The study's findings suggest that imbalanced data can affect the performance of DL-based models in COVID-19 diagnosis, highlighting the importance of addressing this issue in the development of diagnostic tools. The study's best-performing models (VGG16 and MobileNetV2) achieved high accuracy ranges from 97.6% to almost 100% on a highly imbalanced dataset. However, some models extracted irrelevant details from the images, which might affect the models' performance.

To address this issue, the study suggests training the models on larger datasets with increased epochs to improve the models' performance and reduce the models' extraction of irrelevant details from the images. The study's outcomes provide a solution to the issue of imbalanced data in DL-based models' performance and contribute to developing more effective diagnostic tools in healthcare, particularly in COVID-19 diagnosis.

In conclusion, this study contributes to the understanding of the effect of imbalanced data on DL-based models' performance in healthcare diagnosis, specifically in COVID-19 diagnosis, using chest X-ray images. The study's findings and proposed solutions can guide future research in the development of more accurate and effective diagnostic tools.

6.9 Conclusion and Future Works

The study evaluated six DL-based approaches to detect SARS-CoV-2 infection from chest X-ray images. The modified VGG16 and MobileNetV2 models accurately distinguished patients with COVID-19 symptoms on both balanced and imbalanced datasets, achieving nearly 99% accuracy. Healthcare professionals cross-checked the model outputs to ensure validation. However, the study had some limitations, including limited data availability and the exclusion of categorical patient data. Future works should address these limitations and explore opportunities to bridge the gap between proposed models and existing Computer-Aided Design (CAD) systems. To address the limitations identified in this study, future research directions should focus on the following:

- Utilizing rapidly expanding open databases of COVID-19 patient records, especially those containing chest X-ray images, to assess the performance of deep learning models confidently.

- Investigating more robust classification models that include categorical patient data to achieve higher performance levels.

- Bridging the gap between proposed models and existing CAD systems to develop more efficient diagnosis tools.

Chapter 7

Exploring Mixed Image Data for COVID-19 Diagnostics using Transfer Learning and Explainable AI

7.1 Introduction

In Chapter 6, the impact of imbalanced data on developing Deep Learning (DL) based diagnostic models for Coronavirus Disease 2019 (COVID-19) patients using chest X-ray images are analyzed. The preliminary results indicate that the proposed Transfer Learning (TL) approach can detect patients with COVID-19 symptoms. However, recent literature suggests that Computed Tomography (CT)-scan could be a promising alternative to chest Radiography (X-ray) for developing Artificial Intelligence (AI) diagnostic tools (Chua et al., 2020; Narin et al., 2020; Barstugan, Ozkaya, & Ozturk, 2020; F. Shi et al., 2020; Gozes et al., 2020; Fang et al., 2020). Given this opportunity, this chapter aims to evaluate the performance of previously proposed six TL models - VGG16, InceptionResNetV2, ResNet50, MobileNetV2, ResNet101, and VGG19 - on a mixed dataset containing both chest X-ray and CT-scan images.

7.2 Motivation

The primary motivation behind this study is to enhance the accuracy and reliability of DL-based models for diagnosing COVID-19 patients using chest X-ray and

CT-scan images. While Chapter 6 focused solely on the impact of imbalanced data on DL-based models using chest X-ray images, it is important to develop models that can learn from both chest X-ray and CT-scan images to improve the effectiveness of healthcare diagnostics.

Furthermore, this chapter presents a unique contribution to the literature as it is the only chapter in the dissertation considering imbalanced mixed image data. By addressing the primary research question RQ3, which is "How does imbalanced data affect the performance of DL-based models?" in the context of COVID-19 diagnosis, this chapter aims to provide valuable insights into the development and performance of DL models using mixed image datasets.

Moreover, this chapter aims to enhance the interpretability of the proposed DL models by utilizing Local Interpretable Model-Agnostic Explanations (LIME) to provide model interpretation and detailed information on the learning procedure. Incorporating LIME also addresses one of the limitations outlined in Chapter 6 and improves the transparency and interpretability of the proposed DL models.

Overall, this chapter will serve as an important contribution to the field of DL-based medical diagnosis, providing insights into developing models that can learn from mixed datasets and improving the interpretability of DL models using LIME.

7.3 Chapter Outline

The forthcoming chapter is structured as follows: Section 10.4 provides a brief and concise overview of COVID-19 and the utilization of DL for diagnosis using CT-scan images. Section 10.5 expounds on the research methodology in a comprehensive and detailed manner. The experimental results are presented in Section 10.6, followed by an in-depth and meticulous discussion of the study's findings in Section 10.7. Finally, Section 10.8 summarizes the research's overall conclusions and offers potential avenues for future research.

7.4 Background

The previous chapter (Chapter 6) provided an overview of COVID-19 disease diagnosis using chest X-rays and the relevant literature. Consequently, to prevent redundancy, this section focuses solely on literature that explores DL-based models employing CT-scan images for detecting COVID-19 patients.

7.4.1 CT Scan-based Screening

To date, several efforts in detecting COVID-19 from CT images have been reported. A recent study by Chua et al. (2020) suggested that the pathological pathway observed from the pneumonic injury leading to respiratory death can be detected early via chest CT, especially when the patient is scanned two or more days after the development of symptoms (Chua et al., 2020). Related studies proposed that DL techniques could be beneficial for identifying COVID-19 disease from chest

CT (Narin et al., 2020; Barstugan et al., 2020). For instance, Shi et al. (2020) introduced a machine learning-based method for the COVID-19 screening from an online COVID-19 CT dataset (F. Shi et al., 2020). Similarly, Gozes et al. (2020) developed an automated system using artificial intelligence to monitor and detect patients from chest CT (Gozes et al., 2020). Chua et al. (2020) focused on the role of Chest CT in the detection and management of COVID-19 disease from a high-incidence region (United Kingdom) (Chua et al., 2020). Ai et al. (2020) also supported CT-based diagnosis as an efficient approach compared to RT-PCR testing for COVID-19 patients detection with a 97% sensitivity (Ai et al., 2020; Fang et al., 2020).

Due to data scarcity, most preliminary studies considered minimal datasets (J. Chen et al., 2020; Ardakani, Kanafi, Acharya, Khadem, & Mohammadi, 2020; X. Wang, Deng, et al., 2020). For example, Chen et al. (2020) used a U-Shaped Convolutional Neural Network (UNet)++ DL model and identified 51 COVID-19 patients with a 98.5% accuracy (J. Chen et al., 2020). However, the authors did not mention the number of healthy patients used in the study. Ardakani et al. (2020) used 194 CT images (108 COVID-19 and 86 other patients) and implemented ten deep learning methods to observe COVID-19 related infections and acquired 99.02% accuracy (Ardakani et al., 2020). Moreover, a study conducted by Wang et al. (2020) considered 453 CT images of confirmed COVID-19 cases, from which 217 images were used as the training set, and obtained 73.1% accuracy, using the inception-based model. The authors, however, did not explain the model network and did not show the mark region of interest of the infections (S. Wang, Kang, et al., 2021). Similarly, Zheng et al. (2020) introduced a DL-based model with 90% accuracy to screen patients using 499 3D CT images (X. Wang, Deng, et al., 2020). Despite promising results, very high performance on small datasets often raises questions about the model's practical accuracy and reliability. Therefore, a better way to represent model accuracy is to present it with an associated confidence interval (Brownlee, 2014). However, none of the work herein referenced expressed their results with confidence intervals, which should be addressed in future studies.

As larger datasets become available, DL-based studies taking advantage of their potential have been proposed to detect and diagnose COVID-19. Xu et al. (2020) investigated a dataset of 618 medical images to detect COVID-19 patients and acquired 86.7% accuracy using ResNet23 (X. Xu et al., 2020). Li et al. (2020) utilized an even larger dataset (a combination of 1296 COVID-19 and 3060 Non-COVID-19 patients CT images) and achieved 96% accuracy using ResNet50 (L. Li et al., 2020b). With larger datasets, it is no surprise that DL-based models predict patients with COVID-19 symptoms with accuracies ranging from 85% to 96%. However, obtaining a chest CT scan is a notably time-consuming, costly, and complex procedure. Despite allowing for comparatively better image quality, its associated challenges inspired many researchers to propose X-ray-based COVID-19 screening methods as a reliable alternative way (Apostolopoulos & Mpesiana, 2020; Rachna, 2020).

7.5 Research Methodology

Table 7.1 summarizes the adopted dataset (*Chest X-Ray Images (Pneumonia)*, n.d.), which contains both CT scans (200 COVID-19 and 200 Non-COVID-19) and chest X-rays (1583 COVID-19 and 608 Non-COVID-19) images of patients expressing pneumonia symptoms. During the study, 80% of the data was dedicated to training and the remaining 20% to testing. Figure 7.1 presents a set of representative images used in the analysis.

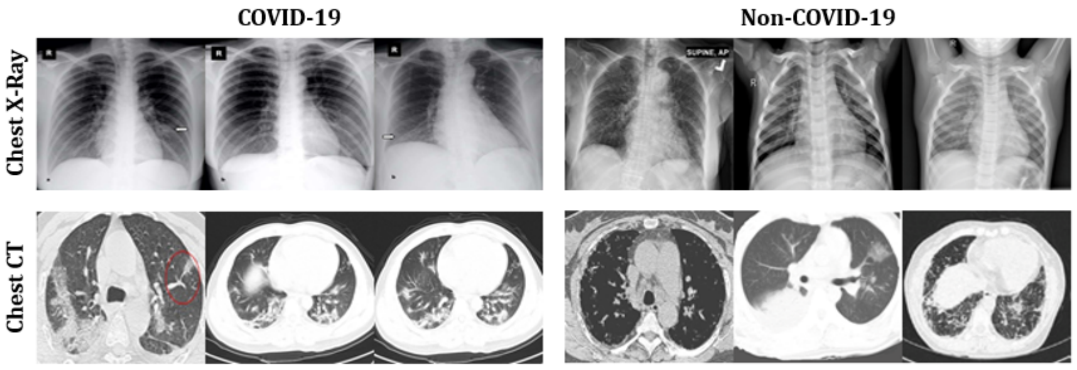


Figure 7.1: Representative sample images of chest X-rays and CT scans used in the mixed dataset adopted for analysis.

Table 7.1: Summary of the mixed dataset used in the analysis, including training and test sets.

Dataset	Label	Train			Test		
		Chest X-ray	CT scan	Total	Chest X-ray	CT scan	Total
Mixed Data	COVID-19	486	160	646	122	40	162
	Non-COVID-19	1266	160	1426	317	40	357
Total		1752	320	2072	439	80	519

Six different pre-trained Convolutional Neural Networks (ConvNets) were used, namely VGG16, MobileNetV2, ResNet50, ResNet101, InceptionResNetV2, and VGG19. A detailed explanation of the architecture of each network can be found in Chapter 6. Each model was developed with the advantages of TL in mind. Three parameters, specifically batch size, epochs, and learning rate (as suggested by (L. N. Smith, 2018; S. L. Smith et al., 2017)), are considered for model optimization. During the study, the grid search method is employed to fine-tune parameters (Bergstra & Bengio, 2012). At first, the following parameters were chosen at random:

$$\text{Batch size} = [20, 30, 40, 50, 60];$$
$$\text{Number of epochs} = [20, 25, 30, 35, 40];$$
$$\text{Learning rate} = [0.001, 0.01, 0.1].$$

The best results were obtained with the following:

$$\text{Batch size} = 50;$$
$$\text{Number of epochs} = 35;$$
$$\text{Learning rate} = 0.001.$$

The statistical analysis was assessed using metrics such as accuracy, precision, recall, and F1-score, which were adopted from the methodology presented in Chapter 6. Aside from this, the remaining experimental setup was maintained consistent with the methodology presented in Chapter 6.

7.5.1 LIME as Explainable AI

The overall prediction was interpreted using LIME, a procedure that allows the understanding of the input features of the DL models, which affect its predictions. LIME is regarded as one of the few methodologies that work well with tabular data, text, and images and is extensively employed for its reliability in explaining the intricacies of image classification (?, ?). For image classification, LIME creates superpixels. Superpixels are the result of image over-segmentation. Superpixels store more data than pixels and are more aligned with image edges than rectangular image patches for the primary prediction (?, ?). Table 7.2 shows the parameters used to calculate the superpixel during this experiment.

Table 7.2: Superpixel calculation parameters.

Function	Value
Kernel size	200
Maximum distance	200
Ratio	0.2

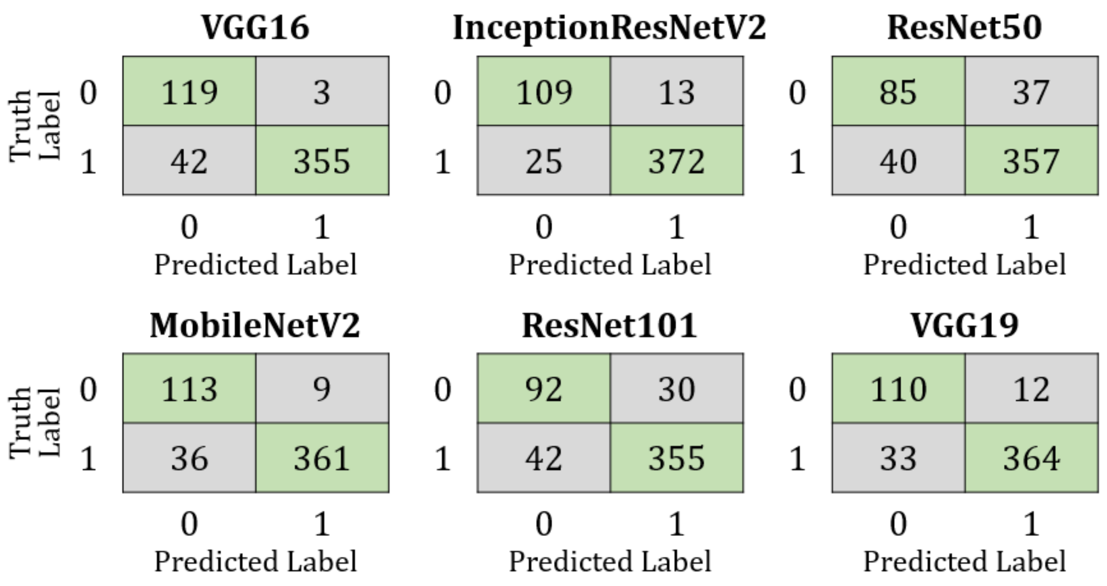
7.6 Results

Table 10.4 presents a summary of the performance of all models on the training and test sets along with a 95% confidence interval. MobileNetV2 outperformed all models in terms of accuracy, precision, recall, and F1-score. Contrarily, the ResNet50 model showed the worst performance considering all measures.

Table 7.3: COVID-19 screening performance of all models using a mixed dataset, presented with 95% confidence intervals (CI, $\alpha = 0.05$). T_a—Training Set; T_s—Test Set.

Algorithm	Accuracy (%)			Precision (%)			Recall (%)			F-1 score (%)		
	$\mathbf{T}_a$	$\mathbf{T}_s$	**CI**	$\mathbf{T}_a$	$\mathbf{T}_s$	**CI**	$\mathbf{T}_a$	$\mathbf{T}_s$	**CI**	$\mathbf{T}_a$	$\mathbf{T}_s$	**CI**
VGG16	95	91	93 ± 1.4	95	93	94 ± 1.3	95	91	93 ± 1.4	95	92	93.5 ± 1.34
InceptionResNetV2	94	93	93.5 ± 1.34	95	93	94 ± 1.3	94	93	93.5 ± 1.34	94	93	93.5 ± 1.35
ResNet50	88	85	86.5 ± 1.86	87	85	86 ± 1.89	88	85	86.5 ± 1.86	87	85	86 ± 1.89
MobileNetV2	99	91	95 ± 1.2	99	92	95.5 ± 1.13	99	91	95 ± 1.2	99	91	95 ± 1.2
ResNet101	88	86	87 ± 1.83	88	87	87.5 ± 1.80	88	86	87 ± 1.83	88	86	87 ± 1.83
VGG19	94	91	92.5 ± 1.43	94	92	93 ± 1.4	94	91	92.5 ± 1.43	94	92	93 ± 1.4

To better understand the overall performance of each model during the prediction stage on the test set, Figure 7.2 presents a set of confusion matrices. The test set contained a combination of 519 chest X-ray and CT scan images (122 COVID-19 and 397 Non-COVID-19). It can be detected that MobileNetV2 and VGG19 correctly classified the maximum number of COVID-19 and non-COVID-19 patients, whereas ResNet50 expressed the worst performance with the maximum number of misclassified samples compared to any other model.



VGG16

Truth Label \ Predicted Label	0	1
0	119	3
1	42	355

InceptionResNetV2

Truth Label \ Predicted Label	0	1
0	109	13
1	25	372

ResNet50

Truth Label \ Predicted Label	0	1
0	85	37
1	40	357

MobileNetV2

Truth Label \ Predicted Label	0	1
0	113	9
1	36	361

ResNet101

Truth Label \ Predicted Label	0	1
0	92	30
1	42	355

VGG19

Truth Label \ Predicted Label	0	1
0	110	12
1	33	364

Figure 7.2: Confusion matrices of all models applied to the mixed test dataset.

The performance of all models during training and testing, per each epoch, are presented in Figure 7.3. In this case, the accuracy of VGG16, MobileNetV2, and VGG19 models reached 100% while loss decreased by nearly 100% at epoch 35.

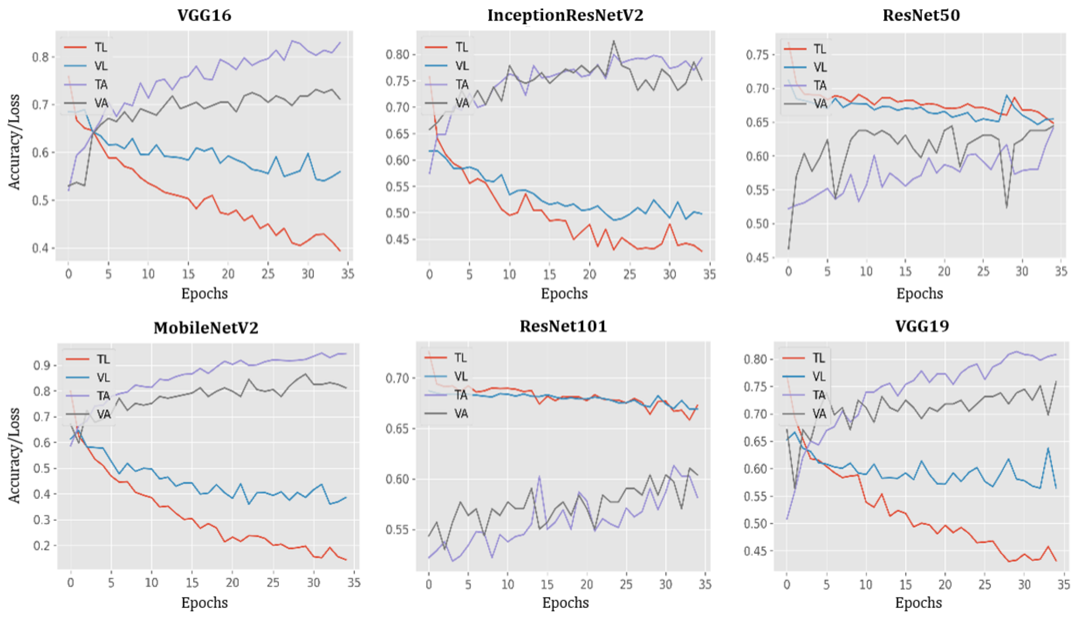


Figure 7.3: Plots of model accuracy and loss following each epoch applied to both training and testing datasets; TL = training loss; VL = validation loss; TA = training accuracy; VA = validation accuracy.

7.6.1 AUC-ROC Curve

In Figure 7.4, measures of the Area Under the Curve (AUC) of the Receiver Characteristic Operator (ROC) are plotted for each model with the True Positive Rate (TPR) in the vertical axis and False Positive Rate (FPR) in the horizontal axis, applied to the test set. MobileNetV2 shows the best performance (AUC=0.816), while ResNet101 shows the worst (AUC=0.590).

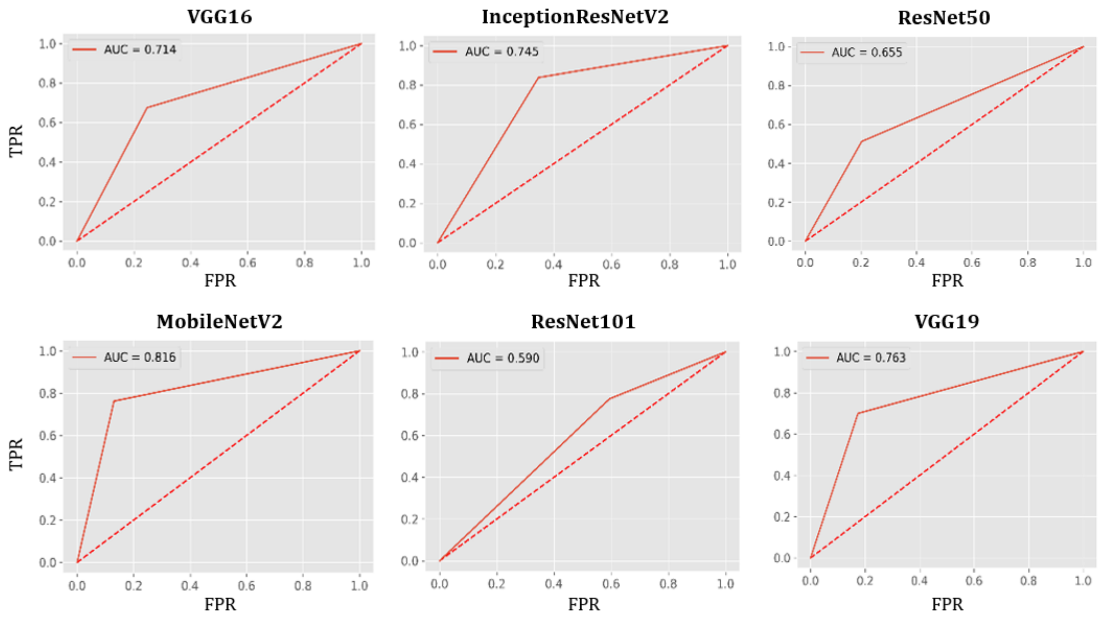


Figure 7.4: AUC-ROC curves for all models using the test set.

Figure 7.5 shows the output after computing the superpixels on sample CT scan and chest X-ray images.

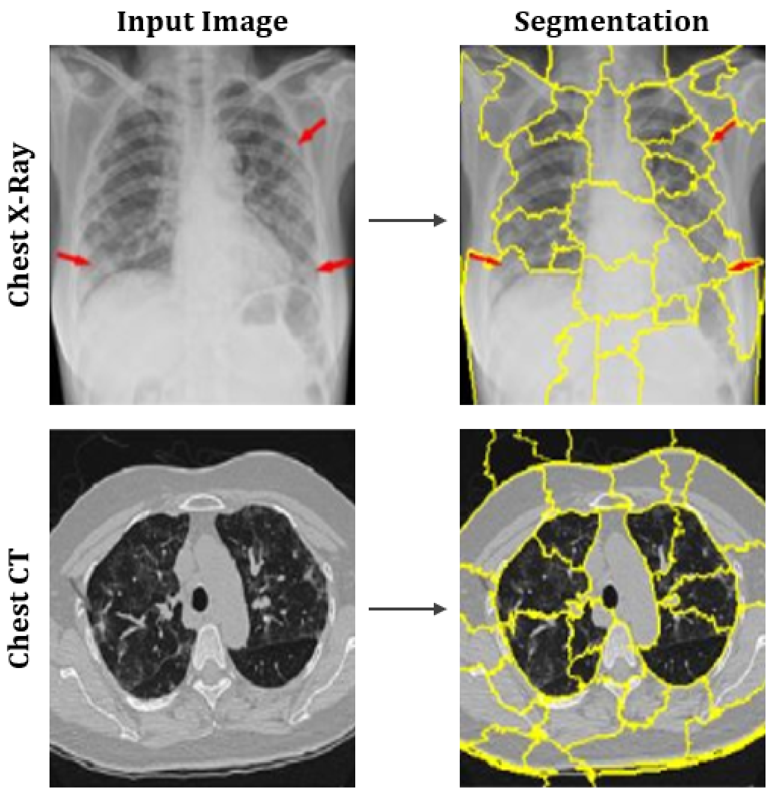


Figure 7.5: Representation of superpixels on sample images of chest X-rays and CT scans.

Additionally, Figure 7.6 shows different image conditions in terms of perturbation vectors and perturbation images. Figure 7.6 illustrates that the number of features varies with the number of perturbations.

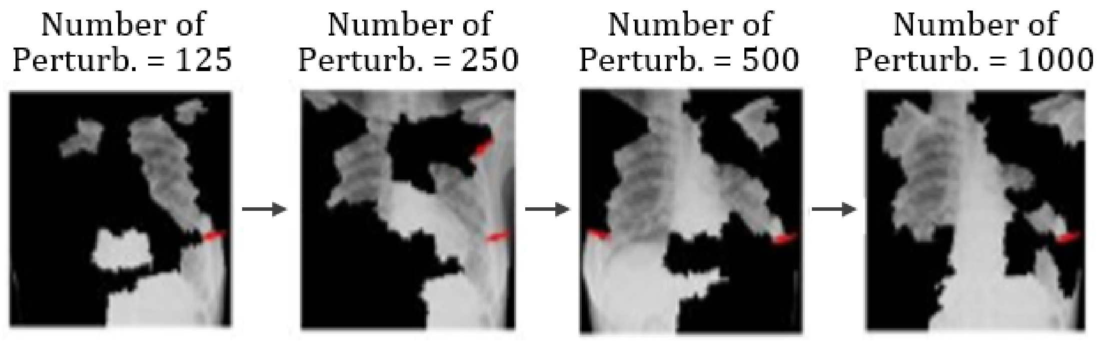


Figure 7.6: Example of the varying number of features as the number of perturbations changes.

The distance metric or cosine metric with a kernel width of 0.25 is used to understand the distance difference between each perturbation and the original image. A linear model is used for the proposed model's explanations. Additionally, the coefficient was found for every superpixel in the picture, which represents the strength of a superpixel's impact on predicting COVID-19 patients. Finally, the top features (only four features are considered for the purposes of this study) are sorted to determine the most essential superpixel, as shown in Figure 7.7. The features and the prediction were addressed together during this study. As shown in Figure 7.7, models, such as VGG16, MobileNetV2, and VGG19 trained with CT scan images incorrectly classified COVID-19 patients as Non-COVID-19 patients. On the other hand, while analyzing combined models, ResNet50 shows the worst performance by misclassifying both CT and chest X-ray images.

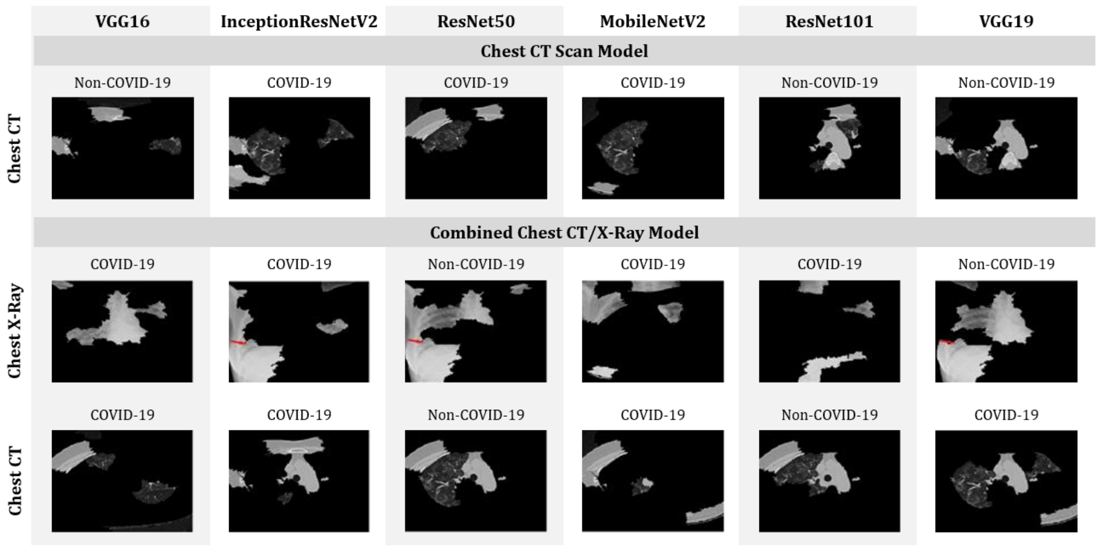


Figure 7.7: Top four features that enabled the identification of COVID-19 patients from CT-scan-only and mixed datasets.

7.7 Discussion

In this study, six different DL-based models were proposed and evaluated for their ability to distinguish between patients with and without COVID-19, with demonstrated advantages of tests conducted on combined datasets, comprising both CT scan and X-ray images (as opposed to a singular point of reference with only CT scans or X-rays). Among all proposed models, MobileNetV2 achieved an accuracy ranging between 94–99% depending on the dataset applied. A summary of the accuracy of all six models, considering the CT scan, chest X-ray, and the mixed dataset, is presented in Table 7.4. Other than MobileNetV2, the VGG16 model demonstrates higher performance on the chest X-ray dataset by achieving an accuracy of 98.5% ± 1.19%, which outperforms many studies in the current literature. For example, Wang and Wong (2020) and Khan et al. (2020) used CNN-based approaches to detect the onset of the COVID-19 disease using chest X-ray images and achieved an accuracy of 83.5% and 89.6%, respectively (L. Wang & Wong, 2020; Khan et al., 2020). In comparison, as mentioned earlier, the proposed VGG16 and MobileNetV2 models achieved an accuracy of approximately 98.5% ± 1.19%.

Table 7.4: Top-performing models in terms of accuracy and different datasets adopted.

Dataset	Datasize	Model	Accuracy (%)
Chest X-ray	400	VGG16	98.5 ± 1.191
		MobileNetV2	98.5 ± 1.191
CT-Scan	400	MobileNetV2	94 ± 2.327
Mixed-data	2591	MobileNetV2	95 ± 1.12

In Table 7.5, the accuracy of different DL models used in previous studies (where CT scan images were utilized for the experiment) was compared to the models of this study, taking into account various database sizes. Here, an accuracy of 98.5% ± 1.19% was achieved using 400 images with the MobileNetV2 model. These results outperform the referenced literature, which used large datasets containing 4356 and 1065 images, respectively (L. Li et al., 2020a; S. Wang, Kang, et al., 2021). In contrast, Butt et al. (2020) used a CNN-based approach, specifically a ResNet23 model to detect the onset of COVID-19 disease using CT scan images and achieved an accuracy of around 86.7% (Z. Xu et al., 2020). Jin et al. (2020) used 1882 CT-scan images and achieved an accuracy of 94.1% (C. Jin et al., 2020).

Table 7.5: Comparison between previous studies found in the literature and our present study.

References	Model	Dataset Size	Accuracy
(L. Li et al., 2020a)	ResNet50	4356	90%
(S. Wang, Kang, et al., 2021)	Inception-M	1065	74%
(J. Zhang, Xie, Li, et al., 2020)	ResNet50	1531	90%
(Song et al., 2020)	ResNet50	274	86%
(J. Chen et al., 2020)	UNet^{++}	133	98.5%
(C. Jin et al., 2020)	CNN	1882	94.1%
(Butt, Gill, Chun, & Babu, 2020)	ResNet23	618	86.7%
This study	MobileNetV2	400	98.5% ± 1.19%

It is relevant to emphasize that none of the referenced literature considered a mixed dataset, which hinders a direct comparison with the results of this study. However, preliminary computational results on a mixed dataset indicated that a modified MobileNetV2 model is capable of differentiating between patients with COVID-19 symptoms with an accuracy of 95% ± 1.12%. Additionally, analyzing the proposed models with LIME illustrated MobileNetV2's contribution to successfully characterizing the onset of COVID-19 by recognizing essential features in CT/X-ray images.

The primary goal of this study was to develop an integrated system that can detect patients with COVID-19 symptoms from a dataset containing CT scan, chest X-ray, or a combination of CT scan and chest X-ray images of potential COVID-19 patients. At this stage, the scope of the current literature in this field of work remains narrow and often does not consider combined CT-scan and chest X-ray image datasets with explainable AI. Here, predicted features were identified with LIME to understand the models' decision-making process.

Going forward, results of studies such as the one herein presented must be verified in consultation with healthcare experts. In addition, future work can take advantage of evaluating how other interpretable models could be used with mixed datasets in an attempt to validate the overall predictions presented here.

7.8 Overall Findings

This study aimed to investigate the performance of DL-based models on imbalanced data in healthcare diagnostics, specifically addressing how imbalanced data affect the performance of DL-based models. The findings suggest that the proposed TL-based models, VGG16 and MobileNetV2, exhibited consistent performance on both imbalanced chest X-ray data (Chapter 6) and mixed data used in COVID-19 patient diagnosis, as found in this chapter. However, for imbalanced mixed data, the models ResNet50 and ResNet101 are not ideal choices, although their performance can be improved with additional hyper-tuning, which is beyond the scope of this study.

These results highlight the importance of developing TL-based models that can learn from mixed datasets and improve the interpretability of DL models using LIME in the context of COVID-19 diagnosis. The use of LIME also enabled the explanation of the reasoning behind the proposed TL-based models' predictions. The top features that played a crucial role in the models' final prediction could be matched with the segmented and infected areas, which will be considered in future studies. Future research could also focus on developing other interpretable models with mixed datasets to validate the overall predictions presented in this study and verify the results with healthcare experts.

7.9 Conclusion and Future Works

This study builds upon the research conducted in Chapter 6 and aims to address further the research question of "How does imbalanced data affect the performance of DL-based models?". Specifically, this study focuses on the application of DL-based models for diagnosing COVID-19 patients using both chest X-ray and CT-scan images, with a particular emphasis on imbalanced mixed datasets. The study evaluates six different TL-based models and proposes using LIME to enhance the interpretability of the models. The findings of this study demonstrate the potential of TL-based models, such as VGG16 and MobileNetV2, to perform well on both imbalanced chest X-ray data and mixed data used in COVID-19 patient diagnosis. However, the study also identifies limitations in the performance of some models on imbalanced mixed datasets and highlights the need for future work to address these challenges.

Additionally, future work could focus on developing user-friendly mobile apps and web-based screening systems that employ VGG16 and MobileNetV2 models for COVID-19 screening, incorporating additional data (i.e., age, gender, health conditions), and exploring other image processing techniques, such as fuzzy entropy and divergence, to improve the recognition of edges and contours in X-ray and CT-scan images.

Chapter 8

Defect Analysis of 3D Printed Cylinder Object Using Transfer Learning Approaches

8.1 Introduction

Additive Manufacturing (AM) is a rapidly growing industry that involves printing three-dimensional (3D) objects using Computer-Aided Design (CAD) models (F. Chen, Mac, & Gupta, 2017; Hangge, Pershad, Witting, Albadawi, & Oklu, 2018; Pucci, Christophe, Sisti, & Connolly Jr, 2017). During the printing process, defects can arise due to various reasons, such as deviations from design guidelines or errors in the printing process (J. Liu et al., 2018). Detecting these defects in real time is crucial to ensure the quality of the final product (Trinks, 2021). However, identifying defects in 3D printed products can be challenging due to the complex nature of the images and the imbalanced nature of the data, where the number of defective products is often significantly smaller than the number of non-defective ones (DebRoy, Mukherjee, Wei, Elmer, & Milewski, 2021; S. Guo et al., 2022). Machine Learning (ML) based image classification techniques have shown great potential in addressing such challenges. Therefore, by addressing the primary research question RQ3, which is "How does imbalanced data affect the performance of DL-based models?" in the context of defect analysis in AM, this chapter aims to provide valuable insights into the performance of Transfer Learning (TL) models.

8.2 Motivation

The motivation behind this study is to evaluate the performance of TL approaches in the context of defect analysis in Additive Manufacturing (AM). With the advancements in AM technology, detecting defects during the printing process has become an essential issue to ensure the quality of the printed product. Image classification using TL models has shown promising results in various image-based applications. However, in the context of AM, challenges include imbalanced data, variability in printing conditions, and complexity in the defect patterns (Zeiser, Özcan, Kracke, van Stein, & Bäck, 2023; Westphal & Seitz, 2022). Therefore, evaluating the performance of TL models, such as VGG16, VGG19, ResNet50, ResNet101, and MobileNetV2, is essential to determine their effectiveness in detecting defects in 3D printed products.

Therefore, this study aims to evaluate and compare the performance of various TL models on a dataset of 3D-printed product images to determine the best approach for defect analysis in additive manufacturing. The study aims to provide insights into the effectiveness of TL models and their potential contribution to the field of additive manufacturing. By identifying the best TL model for defect analysis, this research is expected to enhance the quality of 3D-printed products and promote the advancement of the field.

8.3 Chapter Outline

The remainder of the chapter is organized as follows: The methodology of the experiment is described in detail in Section 10.5. The results of the experiment are presented in Section 10.6. A brief discussion of the study's findings is provided in Section 10.7, and the overall conclusions of the research and potential avenues for future research are summarized in Section 10.8.

8.4 Methodology

This section outlines the procedure for acquiring the 3D printing images, the TL model used in this study, the experimental setup, and the evaluation metrics employed to assess the model's performance.

8.4.1 Data Collection

The dataset was generously provided by the Smart Materials and Intelligent Systems (SMIS) Laboratory. It comprises a collection of 3D printed prototype images of cylindrical objects, some of which contain defects while others are non-defective. Figure 8.1 displays a selection of typical images of cylindrical objects from the dataset, including both defective and non-defective examples. Table 8.1 provides an overview of the allocation of data for training and testing of each of the CNN models that were investigated. In both studies, six different DL approaches were investigated: VGG16 (Simonyan & Zisserman, 2014), InceptionResNetV2 (Szegedy et al., 2017), ResNet50 (Akiba et al., 2017), MobileNetV2 (Sandler et al., 2018), ResNet101 (K. He et al., 2016) and VGG19 (Simonyan & Zisserman, 2014).

Table 8.1: Allocation of data for training and testing of TL models used in this study.

Dataset	Label	Train	Test	Total
Study One	Defect	105	31	136
	Non-defect	112	24	136
Study Two	Defect	736	211	947
	Non-defect	1677	393	2070

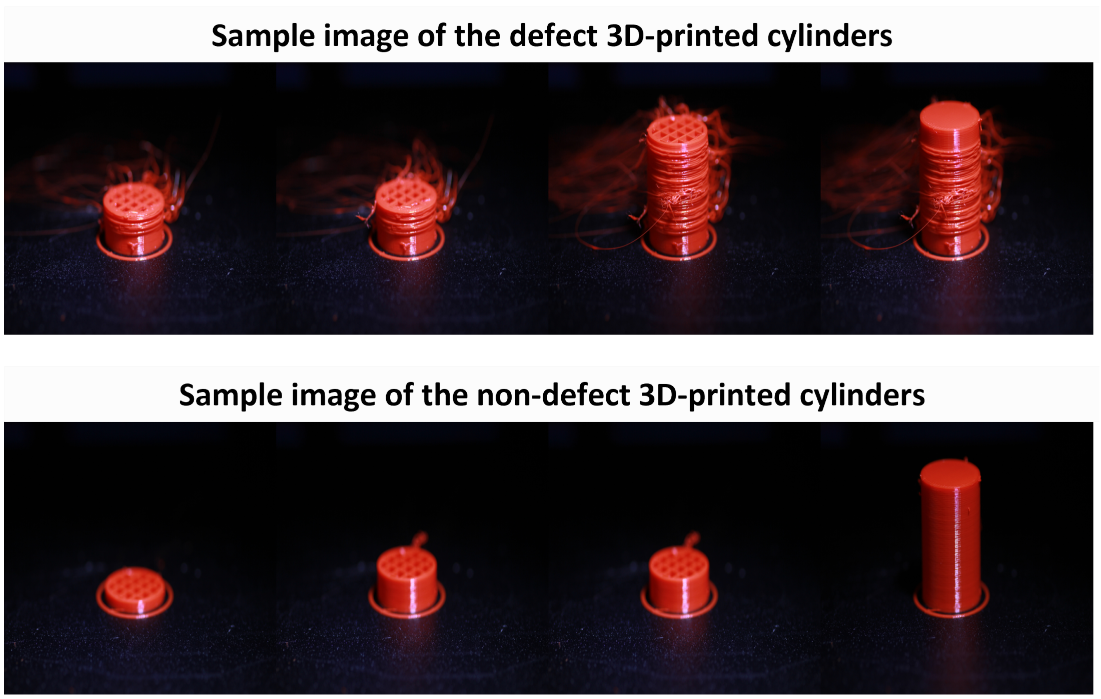

Sample image of the defect 3D-printed cylinders
Sample image of the non-defect 3D-printed cylinders

Figure 8.1: Representative samples of 3D-printed product images obtained from SMIS, which have been utilized in this research.

8.4.2 Experimental Setup

The experiment was carried out with the same computer and software used explicitly for all the experiments mentioned in this dissertation which is already addressed in Chapter 3, Section 3.2. The evaluation metrics employed to measure the performance include accuracy, precision, recall, F1-score, sensitivity, and specificity. The formula of accuracy, precision, recall, and F1-score can be found in Chapter 3, Section 3.2.

8.5 Results

8.5.1 Study One

Table 9.1 presents the performance of six different DL algorithms, namely VGG16, InceptionResNetV2, ResNet50, ResNet101, MobileNetV2, and VGG19, on the train set. The highest performance in terms of all four metrics was achieved by VGG16 and MobileNetV2, both with perfect scores. InceptionResNetV2 also performed well, averaging 99% across all metrics. On the other hand, ResNet50 had the lowest performance among the models, with an average F1-score of 0.34.

Table 8.2: Computational results of the various Transfer Learning models used in Study One on the train set.

Algorithm	Accuracy	Precision	Recall	F1-score
VGG16	1	1	1	1
InceptionResNetV2	.99	.99	.99	.99
ResNet50	0.51	0.25	0.5	0.34
ResNet101	0.86	0.87	0.85	0.85
MobileNetV2	1	1	1	1
VGG19	0.98	0.98	0.98	0.98

The confusion matrix on the train data, as shown in Figure 8.2 for Study One, indicates the true positive, false positive, false negative, and true negative values for each algorithm. The results show that VGG16, InceptionResNetV2, and MobileNetV2 had perfect classification accuracy, with 109, 110, and 110 true positive values, respectively. ResNet50 had the highest number of false negative values, with 106, and a relatively low number of true positives, with only 110. ResNet101 had 27 false negative values but a higher number of true positives than ResNet50, with 106. Finally, VGG19 had the same number of false negatives as ResNet50, with 0, but a higher number of false positives, with 4. Overall, the confusion matrix on the train data shows that VGG16, InceptionResNetV2, and

MobileNetV2 performed the best, while ResNet50 and VGG19 had lower accuracy due to higher false negative and false positive values, respectively.

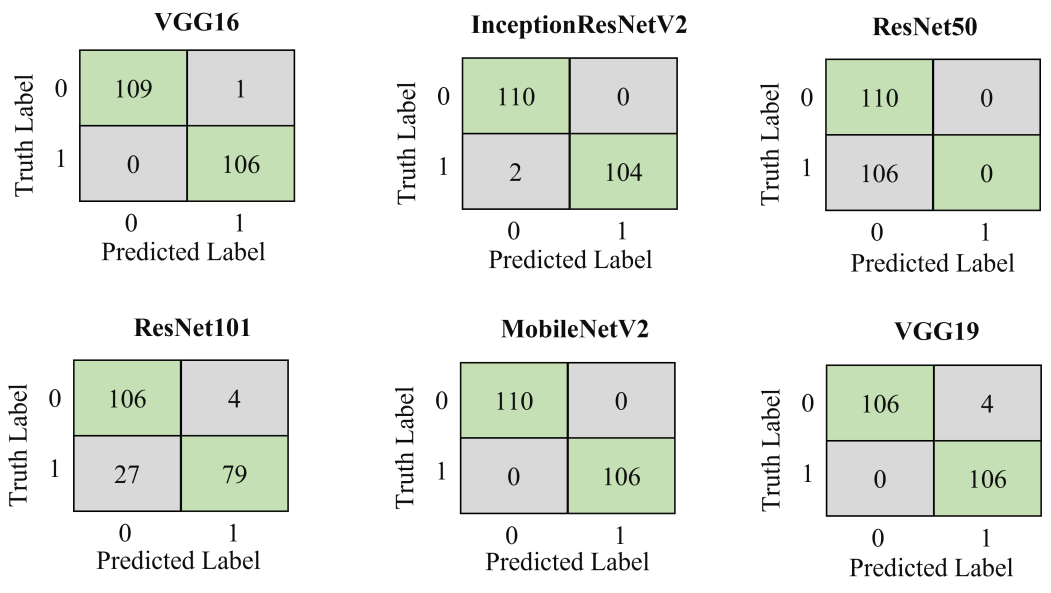


Figure 8.2: Confusion matrices of the different Transfer Learning models used during Study One on the train set.

Table 8.3 presents the results of Study One model performance on the test set. The table shows that VGG16, InceptionResNetV2, and MobileNetV2 achieved perfect scores of 1 for all evaluation metrics, while ResNet50 had the lowest accuracy, precision, and F1-score. ResNet101 achieved high performance with an accuracy of 0.85 and an F1-score of 0.85, while VGG19 had a high accuracy of 0.98 and an F1-score of 0.98. These results demonstrate that different TL algorithms perform differently on the same dataset.

Table 8.3: Computational results of the various Transfer Learning models used in Study One on the test set.

Algorithm	Accuracy	Precision	Recall	F1-score
VGG16	1	1	1	1
InceptionResNetV2	1	1	1	1
ResNet50	0.47	0.24	0.50	0.32
ResNet101	0.85	0.87	0.86	0.85
MobileNetV2	1	1	1	1
VGG19	0.98	0.98	0.98	0.98

Figure 8.7 shows the confusion matrix on the test data for Study One. The algorithms VGG16, InceptionResNetV2, and MobileNetV2 achieved perfect accuracy with true positives (TP) of 26 and no false positives (FP) or false negatives (FN), while ResNet50 had a TP of 26 but a high number of false negatives (FN) at 29. ResNet101 had a slightly lower performance than VGG16, InceptionResNetV2, and MobileNetV2, with a TP of 25 and 1 FP, and 7 FN. VGG19 had a TP of 25 and 1 FP but no FN.

Figure 8.4 demonstrates the top two DL models' AUC scores on the test set.

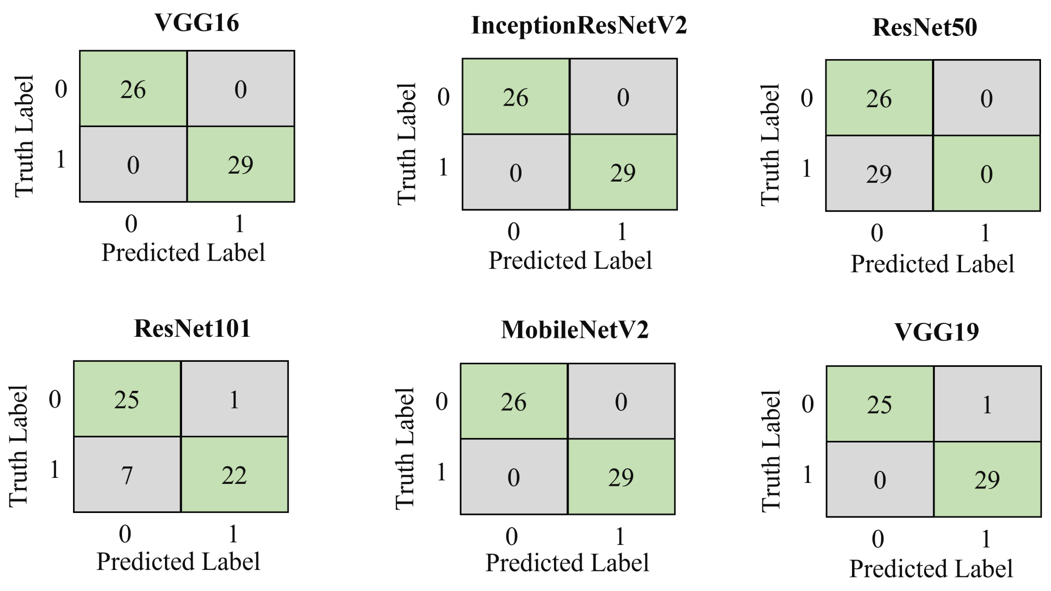


Figure 8.3: Confusion matrices of the different Transfer Learning models used during Study One on the test set.

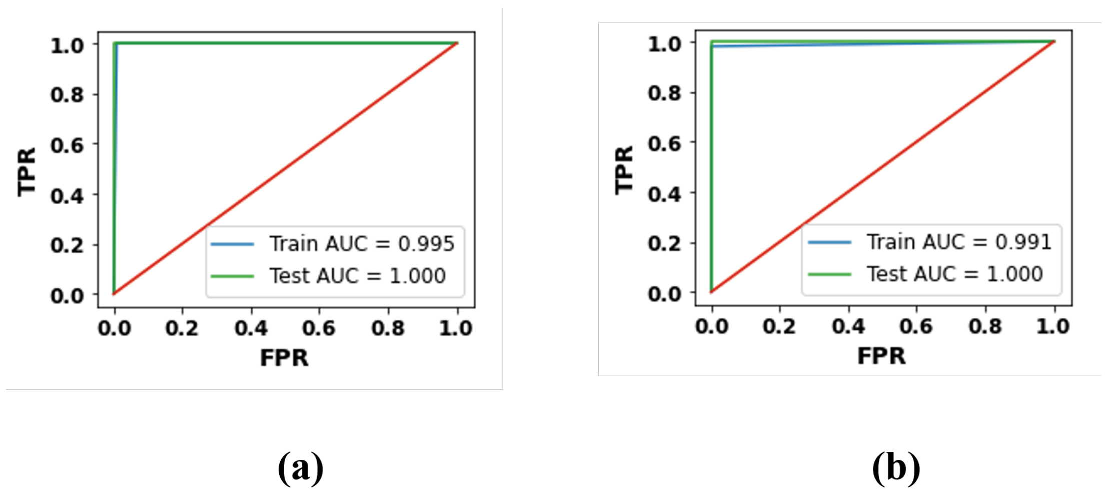
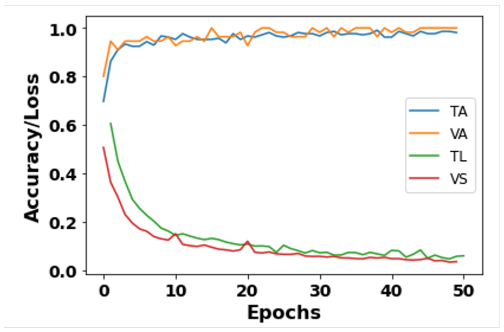
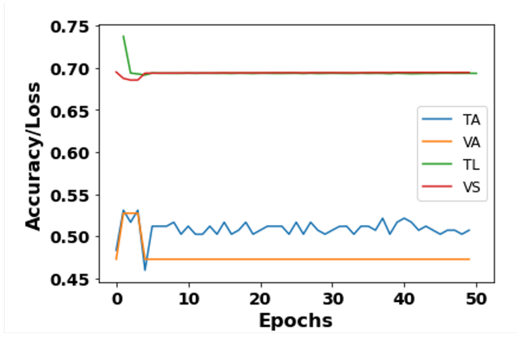


Figure 8.4: AUC score of two well performed TL models (a) VGG16 and (b) InceptionResNetV2.

Figure 8.5 compares the In Figure 8.5, the performance of VGG16 and ResNet50 models during the training phase is compared. The results show that VGG16 continuously improved train accuracy and reduced train loss until epoch 20. At the same time, ResNet50 stopped showing any further improvements in performance after only ten epochs.



(a)



(b)

Figure 8.5: Training and validation performance throughout the training of (a) VGG16 and (b) ResNet50 during Study One. TA – training accuracy; VA – validation accuracy; TL – training loss; VS – validation loss.

8.5.2 Study Two

Table 8.4 presents the results of Study Two model performance on the train set. The best-performing models on the train set in Study Two are VGG16, InceptionResNetV2, MobileNetV2, and VGG19, which achieved perfect scores on all evaluation metrics. The worst-performing model is ResNet50, which achieved the lowest accuracy, precision, recall, and F1-score scores. ResNet101 showed overall good performance, with a score of 0.85 for accuracy and F1-score and precision and recall scores of 0.87 and 0.78, respectively.

Table 8.4: Computational results of the various Transfer Learning models used in Study Two on the train set.

Algorithm	Accuracy	Precision	Recall	F1-score
VGG16	1	1	1	1
InceptionResNetV2	1	1	1	1
ResNet50	.8	.89	.69	.71
ResNet101	0.85	0.87	0.78	0.8
MobileNetV2	1	1	1	1
VGG19	1	1	.99	1

Figure 8.6 shows the confusion matrix on the train data for Study Two. Among the algorithms, InceptionResNetV2 and MobileNetV2 had the highest number of TP, 766 each, while ResNet50 had the lowest, with only 288 TP. InceptionResNetV2 had the fewest FP, with 0, while ResNet50 had the most, with 478. ResNet101 had the highest number of FN, with 45, while VGG16, InceptionResNetV2, MobileNetV2, and VGG19 had none.

Overall, InceptionResNetV2 and MobileNetV2 achieved the best performance, with high numbers of TP and TN and no FN or FP. ResNet50 had the lowest performance, with a low number of TP and a high number of FP.

Table 8.5 presents the results of Study Two model performance on the test set.

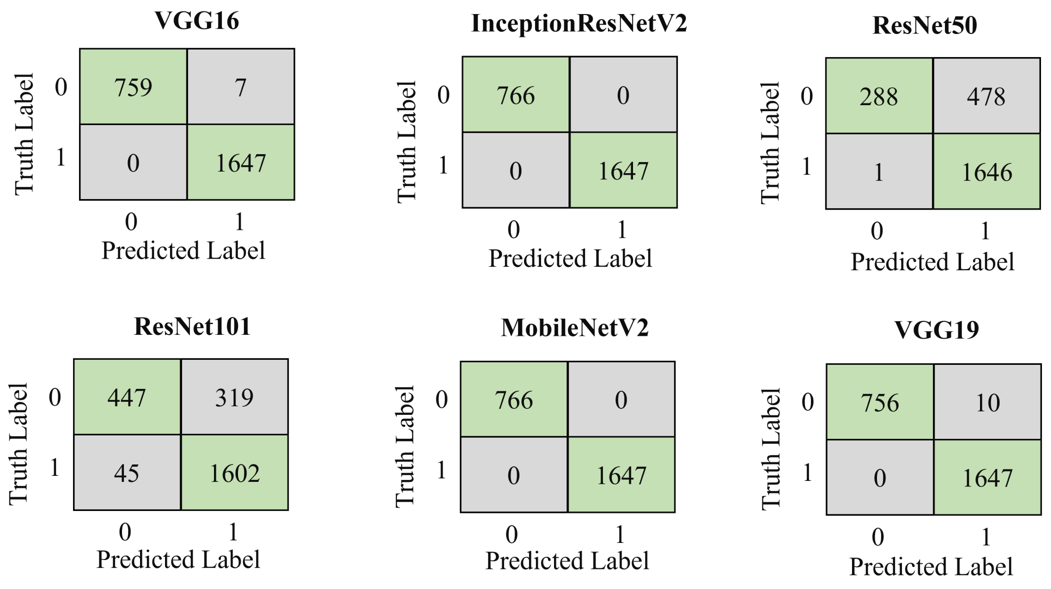


Figure 8.6: Confusion matrices of the different Transfer Learning models used during Study Two on the train set.

The MobileNetV2, VGG16, InceptionResNetV2, and VGG19 models achieved the best results, with perfect accuracy, precision, recall, and F1-score. The ResNet50 and ResNet101 models showed lower performance in terms of accuracy, precision, recall, and F1-score, with the ResNet50 performing the worst among all models.

Table 8.5: Computational results of the various Transfer Learning models used in Study Two on the test set.

Algorithm	Accuracy	Precision	Recall	F1-score
VGG16	1	1	1	1
InceptionResNetV2	1	1	1	1
ResNet50	0.83	0.9	0.72	0.75
ResNet101	0.86	0.88	0.79	0.81
MobileNetV2	1	1	1	1
VGG19	1	1	.99	1

Figure 8.7 shows the confusion matrix on the test data for Study Two. The figure shows that MobileNetV2 had the highest TP and TN, equaling 181, indicating that it correctly classified all positive and negative instances. VGG16 and InceptionResNetV2 also performed well, with TP and TN equaling 180 for each algorithm. ResNet50 had the highest number of FP, with 101, and the lowest number of TP, with only 80. ResNet101 had a lower number of FP (72) but a higher number of FN (12) than other models. VGG19 had the second-highest number of TN, equaling 423, but had 2 FP. Overall, MobileNetV2 showed the best performance, while ResNet50 and ResNet101 had relatively lower performance than other algorithms.

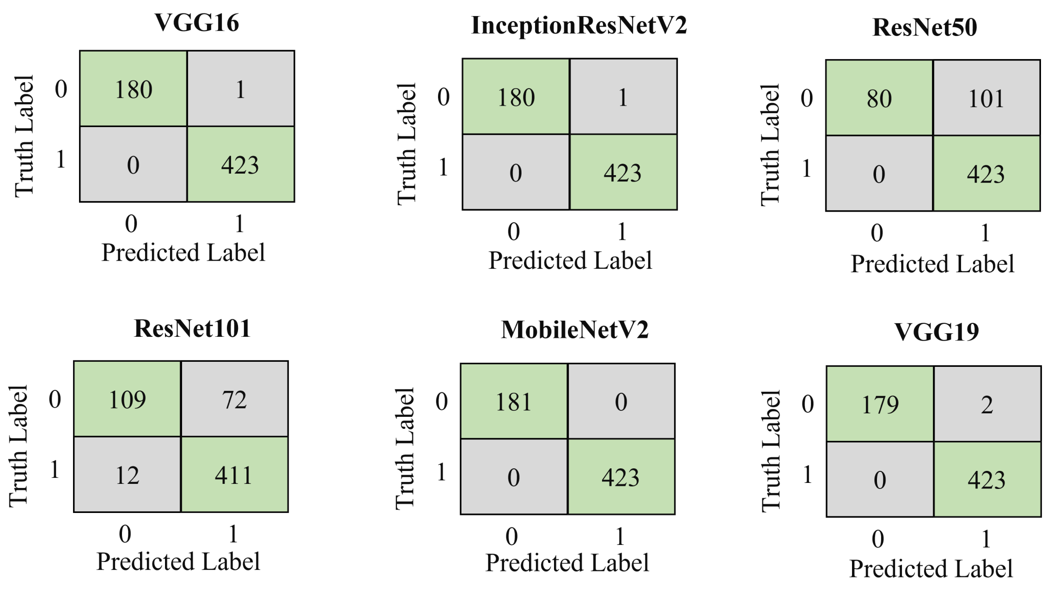


Figure 8.7: Confusion matrices of the different Transfer Learning models used during Study Two on the test set.

Figure 8.8 illustrates the AUC score of VGG16 and ResNet50 models during Study Two. From the figure, it can be observed that the performance of VGG16 is nearly perfect, whereas the performance of ResNet15 demonstrates significantly poor performance.

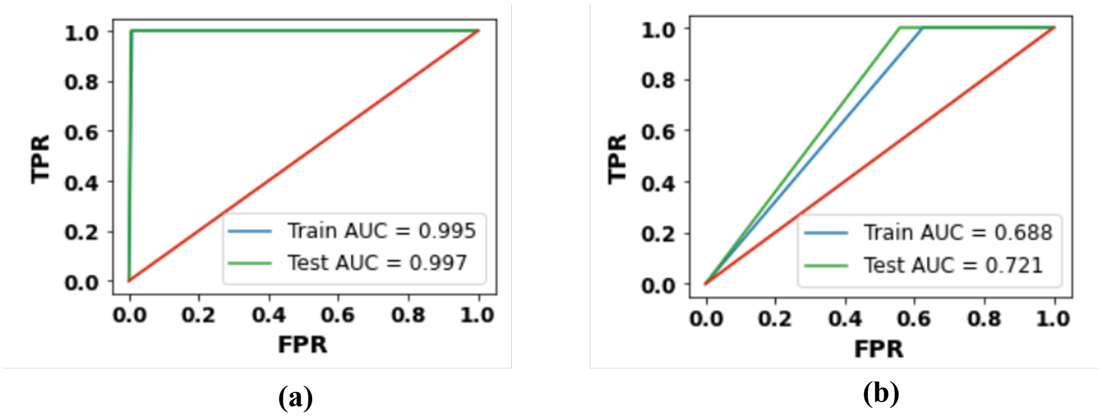
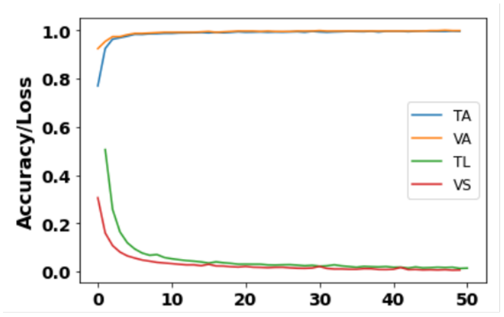
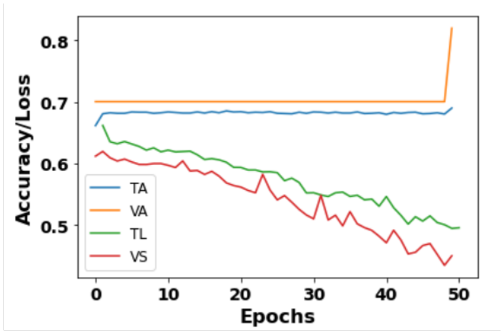


Figure 8.8: AUC score of (a) VGG16 and (b) ResNet50.

Figure 8.9 compares the performance of VGG16 and ResNet50 models during the training phase. The figure indicates that VGG16 constantly improved train accuracy and reduced train loss until epoch 20, whereas ResNet50 ceased showing further performance improvements after only ten epochs.





(a) (b)

Figure 8.9: Training and validation performance throughout the training of (a) VGG16 and (b) ResNet50 during Study Two. TA – training accuracy; VA – validation accuracy; TL – training loss; VS – validation loss.

8.6 Discussions

The current study investigated how deep learning approaches impact imbalanced data analysis. Six deep learning algorithms - VGG16, InceptionResNetV2, ResNet50, ResNet101, MobileNetV2, and VGG19 - were evaluated on an imbalanced data classification task. The results showed that VGG16 and MobileNetV2 achieved perfect scores on all four evaluation metrics, while InceptionResNetV2 performed well, averaging 99% across all metrics. In contrast, ResNet50 had the lowest performance among the models, with an average F1-score of 0.34.

The confusion matrix analysis further supported these findings, showing that VGG16, InceptionResNetV2, and MobileNetV2 had perfect classification accuracy on the train data. In contrast, ResNet50 and VGG19 had lower accuracy due to higher false negative and false positive values. Moreover, on the test data for Study Two, MobileNetV2 demonstrated the best performance, correctly classifying all positive and negative instances. ResNet50 had the highest number of false positives and the lowest number of true positives, while ResNet101 had a higher number of false negatives.

Overall, the findings suggest that deep learning algorithms can perform well on imbalanced data classification tasks, but their performance may vary depending on the specific algorithm and the imbalance level. While some algorithms, such as MobileNetV2, can achieve high classification accuracy, others, such as ResNet50, may struggle with such tasks due to higher false positive and false negative rates. The limitations of this study include the use of a specific dataset and an imbalanced data classification task, which may not generalize to other datasets or tasks. Future research could investigate the impact of various DL algorithms on different imbalanced data classification tasks and explore ways to optimize their performance.

8.6.1 Models Prediction

Figure 8.10 illustrates the model's failure to detect the defect in the image of the 3D-printed cylinder. Based on the results presented in Figure 8.10, it is evident that all three models failed to locate the defect in the 3D-printed cylinder accurately.

Although VGG16 and MobileNetV2 produced bounding boxes that were relatively close to the defect, they were still unable to localize it accurately. These models show little promise and could be used for further investigation. However, similar poor performance was also observed for ResNet101, InceptionResNetV2, and VGG19, indicating that these models may not be well-suited for defect localization in 3D printed objects. These differences in performance may be attributed to variations in the models' architecture and how they handle spatial features. Nonetheless, it is essential to note that despite these differences, all models failed to locate the defect accurately, highlighting the challenges associated with defect localization in 3D printed objects. These findings emphasize the need for further investigation into the factors contributing to model failures and suggest that optimizing model architecture and training strategies may improve localization accuracy and develop more effective solutions for defect detection and classification in additive manufacturing processes.

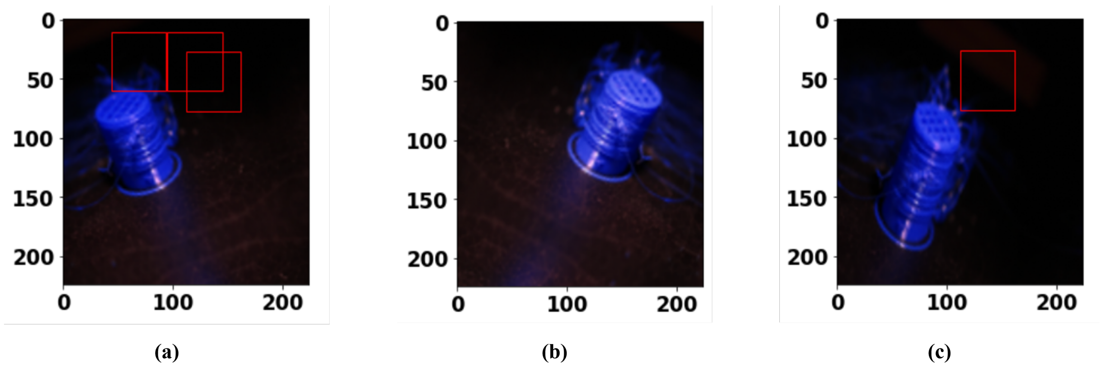


Figure 8.10: Failure to accurately locate the defect in the 3D printed cylinder of (a) VGG16, (b) ResNet50, and (d) MobileNetV2.

8.7 Conclusion

The present study investigated the impact of DL-based approaches on imbalanced data analysis using six popular TL-based DL methods, including VGG16, InceptionResNetV2, ResNet50, MobileNetV2, ResNet101, and VGG19. The experiments were conducted on both small balanced and large imbalanced datasets containing images of the 3D-printed cylinder.

Based on the experiment, it can be answered that even though the performance of TL-based methods on balanced and imbalanced data is very high, the existing popular TL-based approaches failed to locate the potential area, such as defect regions in cylinder images, which indicates that imbalanced data may affect TL-based models' performance. Additionally, VGG16 and MobileNetV2 showed promising results and are a good choice for further investigation.

However, additional experiments with larger and different datasets are necessary to validate the findings. In future work, it is planned to conduct experiments with various datasets and propose advanced TL-based approaches that may help overcome the current challenges observed during this experiment. Ultimately, the study highlights the need for further investigation into the factors contributing to model failures and developing more effective solutions for defect detection and classification in additive manufacturing processes.

Chapter 9

Defect Localization Using Region of Interest and Histogram-Based Enhancement Approaches

9.1 Introduction

Localizing specific regions in computer vision is relatively challenging compared to classifying objects (Bappy, Roy-Chowdhury, Bunk, Nataraj, & Manjunath, 2017; Zou, Chen, Shi, Guo, & Ye, 2023). While Convolutional Neural Network (CNN) based approaches can easily classify between two classes, identifying specific regions can be difficult for the same CNN model (Wahab, Khan, & Lee, 2017; Xue & Li, 2018). In Chapter 8, it was observed that the proposed Transfer Learning (TL)-based approach could accurately classify between defect and non-defect 3D printed cylinder images. However, the model often faced difficulty in localizing the specific region of the defect. This failure could be attributed to the model learning from unnecessary regions. Therefore, pre-processing steps, such as identifying the defect regions from the cylinder image, are crucial to improving the performance of CNN models. The identification of defect regions can significantly reduce the cost and time required for Additive Manufacturing (AM) (Z. Jin, Zhang, Demir, & Gu, 2020; B. Wu et al., 2018).

Finally, interpretation of model predictions is crucial to understand the behavior of the proposed models. To accomplish this, Local Interpretable Model-Agnostic Explanations (LIME) and Gradient-weighted Class Activation Mapping

(Grad-CAM) approaches will be applied. These techniques enable the identification of important features in an input image that contributed to the model's prediction. By generating heatmaps, researchers can gain insight into how the model arrived at its decision, which can provide an understanding of the model's inner workings.

9.2 Motivation

The motivation behind this study is to address the challenges of identifying the defect region in 3D-printed products by incorporating image processing techniques, such as region of interest and histogram equalizer, with Transfer Learning (TL) approaches. While TL-based models can easily classify defect and non-defect cylinders, localizing the specific defect region can be challenging. This limitation can be attributed to the model's learning from unnecessary regions, leading to inefficiencies in the production process.

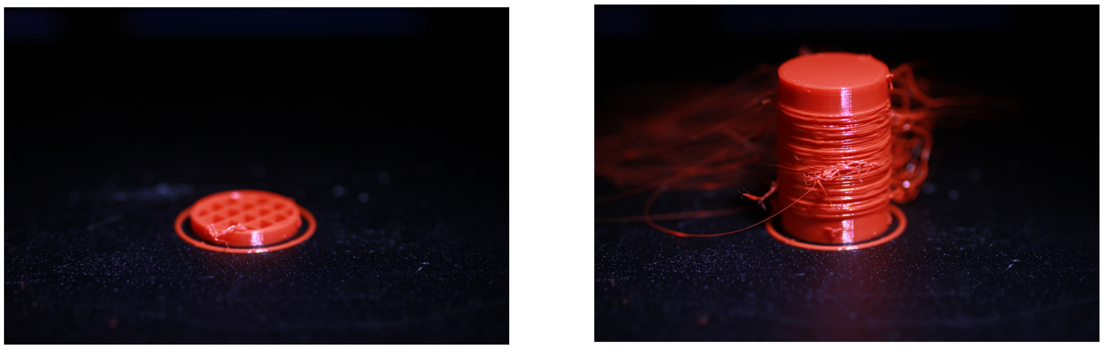
Therefore, this study proposes to update the existing model and evaluate its performance against existing CNN models. Specifically, this study focuses on updating the proposed modified VGG16 model and calculating its efficiency in terms of computational complexity. By localizing the defect region in 3D-printed cylinders, this study is expected to improve the quality of 3D-printed products and reduce production and time costs. The proposed approach will provide an effective solution for industries adopting cost-friendly AI-based AM techniques. This thesis aims to provide valuable insights into the incorporation of image processing techniques and TL-based models for identifying the defect regions in 3D-printed products. The findings of this study will be helpful to researchers, practitioners, and industries seeking to optimize their production processes and increase their competitiveness in the market.

9.3 Chapter Outline

The remainder of the chapter is organized as follows: The methodology of the experiment is described in detail in Section 10.5. The results of the experiment are presented in Section 10.6. A brief discussion of the study's findings is provided in Section 10.7, and the overall conclusions of the research and potential avenues for future research are summarized in Section 10.8. The following chapter is organized as follows: Section 10.4 provides a brief background on the potential reasoning behind the proposed models' inability to identify the defect regions. Section 10.5 outlines the methodology of the experiment in detail. The results of the experiment are presented in Section 10.6, followed by a discussion of the study's findings in Section 10.7. Finally, Section 10.8 summarizes the overall conclusions of the research and suggests potential avenues for future research.

9.4 Background

The traditional TL models have been found to face difficulty in correctly localizing the object in images due to the challenge of identifying specific regions of the image. In Chapter 8, an imbalanced cylinder image dataset containing both defect and non-defect cylinder images was used, and it was found that the proposed modified VGG16 and MobileNetV2 models achieved more than 90% accuracy during classification. However, these models often failed to identify the defect region of the cylinder images. The dataset consisted of images captured at different stages of the 3D printing process, with each image having varying sizes, regions, and pixel densities, as illustrated in Figure 9.1. Figure 9.1(a) shows the cylinder image at the early stage of printing, while Figure 9.1(b) displays the complete cylinder image. It can be observed that for the TL model in Figure 9.1(a), there are many dark and unnecessary regions, while in Figure 9.1(b), the entire cylinder image is present, which allows the model to learn more in detail. As a result, such discrepancies and unnecessary regions may cause the proposed TL-based models to be unable to identify the defect regions. This differs from a steady chest X-ray image used in Chapter 6, where the size and region are constant, making it more challenging to identify the defect region in 3D-printed cylinder images.



(a) **(b)**

Figure 9.1: Sample cylinder image with defect (a) early stage and (b) final stage of 3D printing.

9.5 Research Methodology

The experimental procedure for this study has been kept consistent with the methodology used in Chapter 6 and 8. A comprehensive description of the dataset used in this study can be found in Chapter 8.

9.5.1 Proposed Appraoches

The proposed approach in this study involves three pre-processing steps: Region of Interest (ROI) selection, Histogram equalization (HE), and Details enhancer (DE).

9.5.1.1 Region of Interest (ROI)

ROI selection is used to identify and extract the region of interest from the input image. This technique involves selecting an image's specific area or region to analyze or modify, allowing for more efficient image processing. The ROI is determined by calculating the mean intensity value of the image and selecting a specific threshold value for the selected region. The mathematical formula for ROI selection can be represented as (Erdem, 2020; Girshick, 2015):

$$I_{ROI} = \begin{cases} I(x,y), & \text{if } I(x,y) > T\ 0, \\ \\ \text{otherwise} \end{cases} \tag{9.1}$$

where I_{ROI} is the ROI selected from the input image I, (x, y) represents the pixel coordinates of the image, and T is the threshold value for the selected region.

9.5.1.2 Histogram Equalization (HE)

The second pre-processing step, HE, is a widely-used image processing technique that enhances images' contrast by redistributing pixel values. This technique improves the visual quality of images by normalizing the intensity levels, which improves image detail and clarity. The mathematical formula for HE can be represented as (Joshi, 2022; Dadhich, 2018):

$$g(i,j) = \frac{CDF(f(i,j)) - CDF_{min}}{(MN - CDF_{min})} \times (L-1) \tag{9.2}$$

where $g(i,j)$ is the output image after HE, $f(i,j)$ is the input image, CDF is the cumulative distribution function, CDF_{min} is the minimum cumulative distribution function value, MN is the total number of pixels in the image, and L is the maximum pixel intensity value.

9.5.1.3 Details Enhancer (DE)

The third pre-processing step, DE, is a technique used to enhance the details of an image by increasing its contrast and sharpness. This technique is particularly useful when dealing with low-contrast images, as it can help to bring out more subtle details that might otherwise be difficult to see. The mathematical formula for DE can be represented as (Dadhich, 2018; Bansal, 2022):

$$g(i,j) = (1 + k \cdot (f(i,j) - 128)) \cdot f(i,j) \tag{9.3}$$

where $g(i,j)$ is the output image after DE, $f(i,j)$ is the input image, and k is a constant value that determines the degree of enhancement.

These pre-processing steps are then combined with the proposed modified VGG16 CNN architecture and tested on a designated dataset. These pre-processing techniques are expected to improve the accuracy and efficiency of the proposed model by enhancing the input image data and enabling more accurate defect region identification in 3D-printed cylinder images.

The pre-processing steps are integrated with the proposed CNN models as illustrated in Figure 9.2. In the figure, First formula The first formula calculates the cumulative distribution function of an input image denoted as S_k. It uses a transformation function T to map pixel intensities from the input image to the output image. The formula uses the k-th gray level, r_k, and the normalized histogram of the input image, $P_in(r_j)$, where L represents the total number of possible intensity levels, M represents the number of rows of the image, and N represents the number of columns of the image. It calculates the mapping function for histogram equalization, which redistributes the pixel intensity values in an image to produce a uniform histogram. The second formula calculates the average length of a set of line segments. It uses the length of the i-th line segment, a_i, the number of occurrences of the j-th length in the set of line segments, l_j, and the total number of line segments, A. The formula multiplies each line segment length, a_i, by the number of occurrences, l_j, and adds up these products for all lengths, j. This gives us the sum of the products of the length and frequency of each line segment, which is then divided by the total number of line segments, A, to give the average length of the set of line segments.

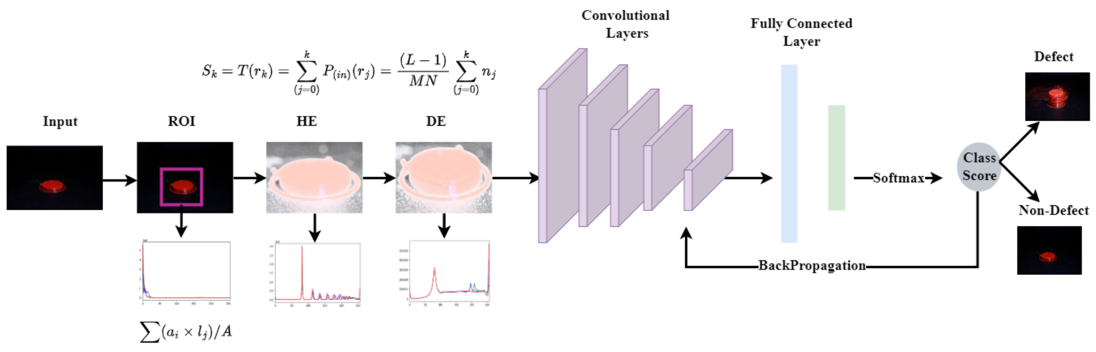


Figure 9.2: Preprocessing steps to identify defect regions before training the proposed CNN model.

The statistical analysis evaluation involved applying metrics such as accuracy, precision, recall, and F1-score, which were adopted from the methodology presented in Chapter Two. Furthermore, the experimental setup was kept consistent with the methodology presented in Chapter Two, except for the changes made to the statistical analysis.

9.5.2 Model Interpretation

In order to comprehend and explicate the proposed models' prediction mechanisms, various model interpretation approaches are being employed. In previous chapters, i.e., chapters 5 and 7, Local Interpretable Model-agnostic Explanations (LIME) has been utilized as an interpretable approach. However, in this study, the Gradient-weighted Class Activation Mapping (Grad-CAM) visualization approach has also been considered in addition to LIME (Selvaraju et al., 2017). One of the main reasons for using both interpretable approaches is to understand how each interpretation technique interprets the proposed model's predictions. In this way, it will also help to evaluate whether the interpretation techniques are trustworthy or not. A detailed explanation of LIME has already been discussed in Chapter 7. In this section, the Grad-CAM visualization approach will be discussed.

Grad-CAM approach is a visualization technique that highlights the regions of an image that contribute most to the prediction made by a CNN model. The method involves generating the class activation map of a model, which is obtained by computing the gradient of the target class output with respect to the feature maps of the last convolutional layer. The importance of the approach lies in its ability to provide interpretability and explainability of the model's prediction, making it possible to understand how the model arrived at its decision. This can be particularly helpful in situations where there is a need to justify the model's output. Additionally, the Grad-CAM approach can be used to identify areas in the image that require further attention or investigation, which can help improve the overall accuracy of the model (Selvaraju et al., 2016, 2017). In summary, the Grad-CAM approach is a powerful tool that provides insight into a CNN model's inner workings and helps improve image classification. The formula of Grad-CAM can be acquired as follows (Selvaraju et al., 2017):

$$L^c_{Grad-CAM} \approx ReLU\left(\sum_k \alpha^c_k A_k\right) \tag{9.4}$$

Here, $L^c_{Grad-CAM}$ is the class activation map for class c, $ReLU$ is the rectified linear unit activation function, α^c_k is the weight of the kth feature map for class c, and A_k is the kth feature map of the last convolutional layer of the CNN. The Grad-CAM equation is used to generate a heatmap that indicates the regions of an input image that were most important for a CNN's prediction of a particular class. By visualizing these heatmaps, researchers can gain insight into how a CNN is making its predictions and identify potential areas for improvement.

9.6 Results

9.6.1 Study One

Table 9.1 shows the performance of different proposed CNN models, including Modified VGG16, ROI Selection (ROIN), ROI Selection with Histogram Equalization (ROIHEN), and ROI Selection with Histogram Equalization and Details Enhancer (ROIHEDEN) on the train and test set of the 3D printed cylinder image small dataset used in this study. The results show that all the proposed approaches achieve promising results, with accuracies ranging from 0.94 to 1 on the test set. Additionally, all the models achieved perfect precision, recall, and F1-score values, indicating no false positives or negatives in identifying the image's defect region. The sensitivity of all the models was also 1, indicating that all models correctly classified the defect region as positive. Furthermore, the specificity of all the models was also high, with values ranging from 0.8750 to 1, indicating a low false-positive rate.

Table 9.1: Proposed different CNN model's performance on the train and test set of the small dataset used during Study One.

Model	Accuracy		Precision		Recall		F1-score		Sensitivity		Specificity	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Modified VGG16	0.94	1	0.94	1	0.94	1	0.94	1	1	1	0.8750	1
ROIN	0.97	1	0.97	1	0.97	1	0.97	1	1	1	0.9464	1
ROIHEN	0.96	1	0.96	1	0.96	1	0.96	1	1	1	0.9196	1
ROIHEDEN	0.96	1	0.96	1	0.96	1	0.96	1	1	1	0.9196	1

Figure 9.3 illustrates the confusion matrices of the proposed ROIHEDEN model on both the train and test sets. The figure shows that the proposed model misclassified only 9 samples from the train set (Figure 9.3(a)) and zero samples from the test set (Figure 9.3(b)). The performance of other models is presented in Appendix .4.1.

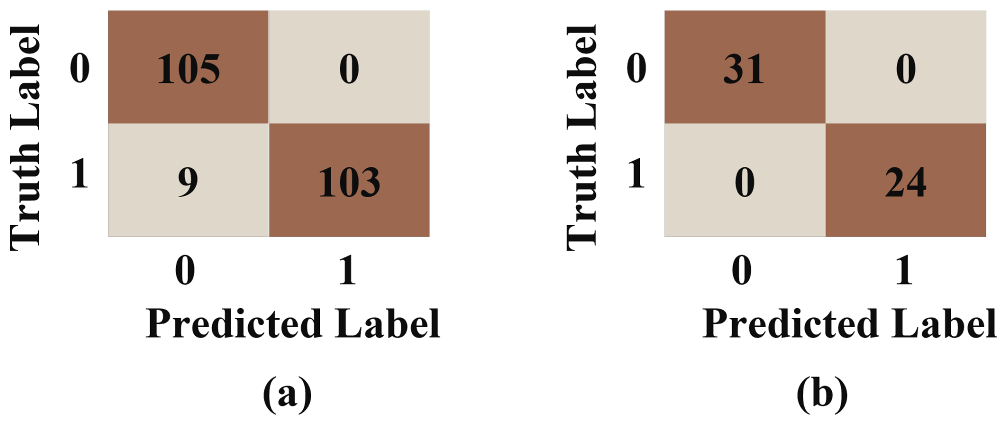


Figure 9.3: Confusion matrices of ROIHEDEN on (a) train and (b) test set.

In Figure 9.4, the performance of modified VGG16, ROIN, ROIHEN, and

ROIHEDEN models during the training phase is compared. The results show that all of the proposed models continuously improved train accuracy and reduced train loss until epoch 30.

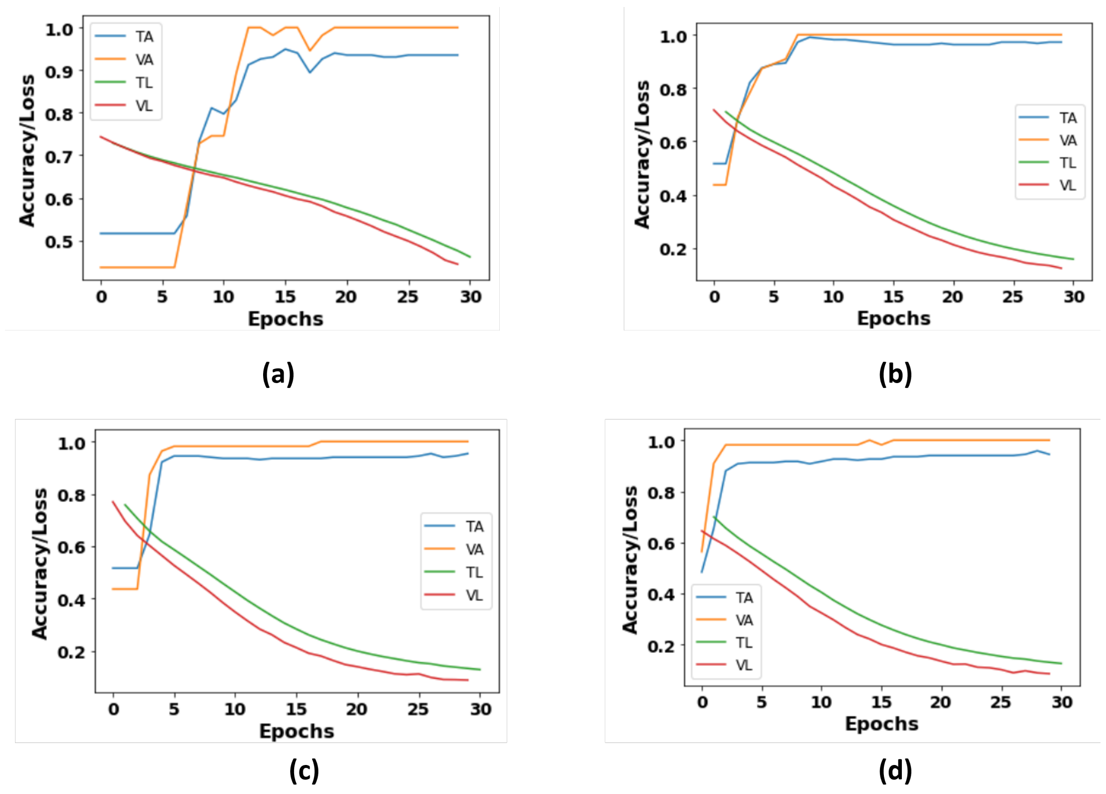


Figure 9.4: Training and validation performance throughout the training of (a) modified VGG16, (b) ROIN, (c) ROIHEN, and (d) ROIHEDEN during Study One. TA – training accuracy; VA – validation accuracy; TL – training loss; VL – validation loss.

9.6.2 Study Two

Table 9.2 presents the performance of the proposed CNN models on the train and test sets of the large dataset used in Study Two. The evaluation metrics used include accuracy, precision, recall, F1-score, sensitivity, and specificity. The models tested are the Modified VGG16, ROIN, ROIHEN, and ROHEDEN. From the table, it can be observed that the Modified VGG16 model achieved the highest accuracy and F1-score on both the train and test sets, while the ROIN model achieved the lowest accuracy and F1-score. However, the ROHEDEN model performed better than the Modified VGG16 model in terms of specificity on the test set. Overall, the results demonstrate that the proposed approaches of ROI selection, histogram equalization, and details enhancer can improve the performance of CNN models in detecting defects in 3D-printed cylinder images.

Table 9.2: Proposed different CNN model's performance on the train and test set of the large dataset used during Study Two.

Model	Accuracy		Precision		Recall		F1-score		Sensitivity		Specificity	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Modified VGG16	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.97	1	1	0.8859	0.9147
ROIN	0.95	0.96	0.96	0.96	0.95	0.96	0.95	0.96	1	1	0.8465	0.8768
ROIHEN	0.94	0.94	0.95	0.95	0.94	0.94	0.94	0.94	1	1	0.8139	0.8389
ROIHEDEN	0.97	0.96	0.97	.96	0.97	0.96	0.96	0.96	1	1	0.8859	0.8957

Figure 9.5 illustrates the confusion matrices of the proposed modified VGG16 model on both the train and test sets during Study Two. The figure shows that the proposed model misclassified 11% samples from the train set (Figure 9.3(a)) and 8.5% of the samples from the test set (Figure 9.3(b)). The performance of other models is presented in Appendix .4.2.

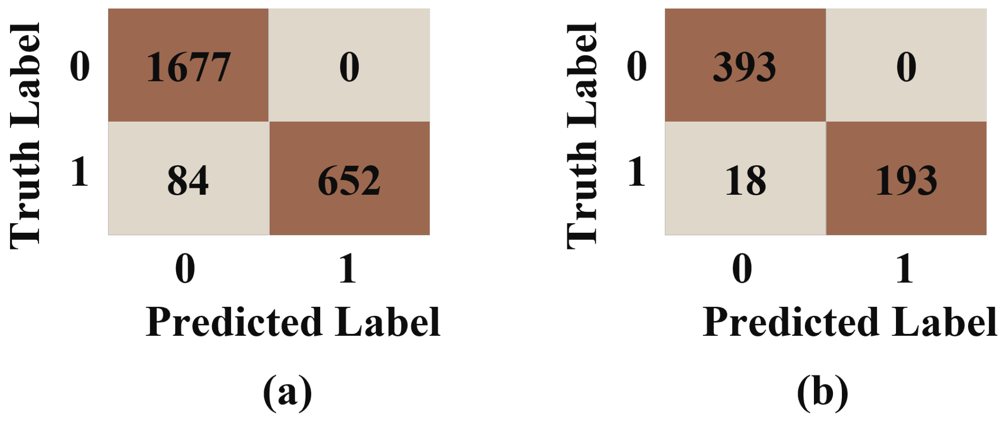


Figure 9.5: Confusion matrices of modified VGG16 on (a) train and (b) test set.

In Figure 9.6, the performance of modified VGG16, ROIN, ROIHEN, and ROIHEDEN models during the training phase is compared for Study Two. The results show that all of the proposed models continuously improved train accuracy and reduced train loss until epoch 30.

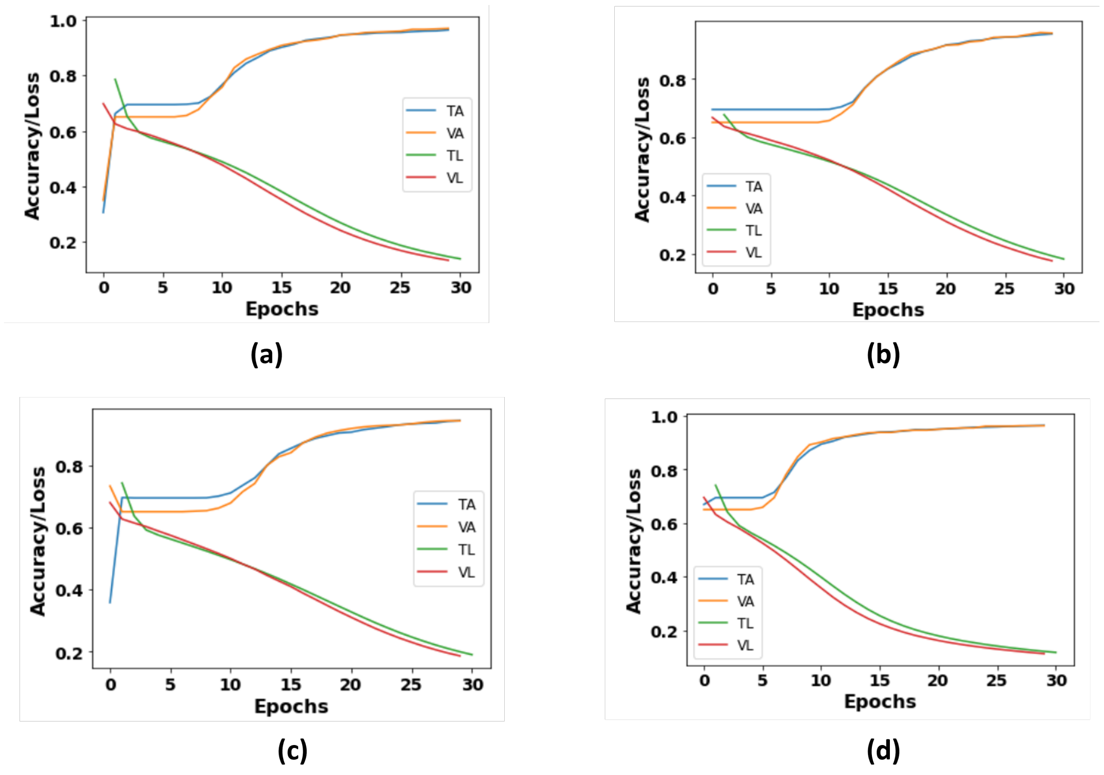


Figure 9.6: Training and validation performance throughout the training of (a) modified VGG16, (b) ROIN, (c) ROIHEN, and (d) ROIHEDEN during Study Two. TA – training accuracy; VA – validation accuracy; TL – training loss; VL – validation loss.

9.6.3 Computational Complexity

The computational complexity of the modified proposed model was evaluated based on two metrics, namely the number of floating-point operations per second (FLOPs) and the number of parameters (NP) in the model's architecture. Table 9.3 presents the comparative computational analysis between the proposed modified VGG16 model and other transfer learning (TL) based models. The results demonstrate that the modified VGG16 model exhibits lower FLOPs and NP compared to other

TL-based models, indicating its superior computational efficiency. Besides the image processing aspect, the TL architecture of the proposed model is similar to modified VGG16 models for other algorithms, such as ROIN, ROIHEN, and ROIHEDEN. Hence, the computational complexity for the proposed algorithms is comparable to that of the modified VGG16 models.

Table 9.3: Comparison of computational complexity between the proposed model and other existing Transfer Learning-based Models.

Algorithm	FLOPS (Millions)	NP
VGG16	30960M	138M
ResNet50	7751M	23.58M
ResNet101	15195M	42.65M
VGG19	39037.83M	20.02M
InceptionResNetV2	26382M	55.87M
Modified VGG16	**30713M**	**15M**

9.6.4 Models Explainability

Figure 9.7 illustrates the result of using a modified ROIN model to detect defects in a 3D-printed object. The proposed ROIN model has successfully identified the defect in the 3D-printed cylinder object. The red square box in the image indicates the location of the defect, which the model has correctly identified. This is a significant achievement as identifying defects in 3D printed objects is challenging, and accurate detection is crucial for quality control.

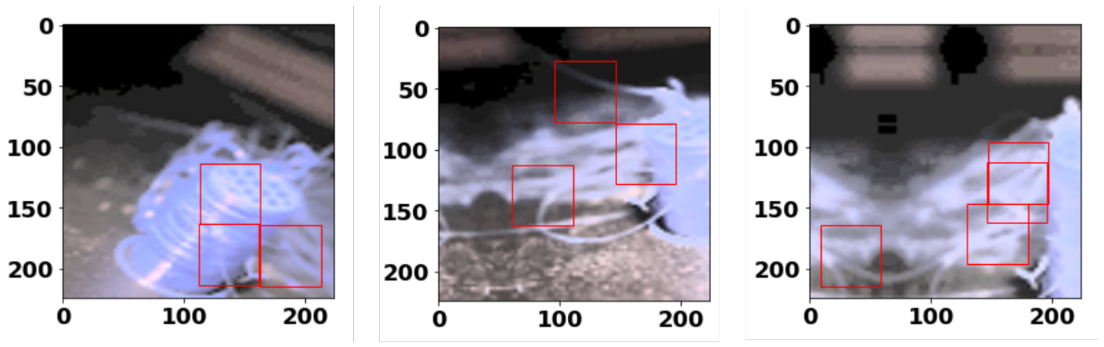


Figure 9.7: Proposed ROIN model's ability to locate the cylinder and its defect regions.

The LIME approach was used to interpret the predictions made by the ROIN model for the given image, as shown in Figure 9.8. From the figure, it can be observed that LIME was able to highlight the regions around the defect as being the most important for the model's prediction, which is consistent with the known characteristics of the defect. The heatmap generated by LIME shows a bright red area in the same region as the defect, indicating that this area was most important in the model's decision.

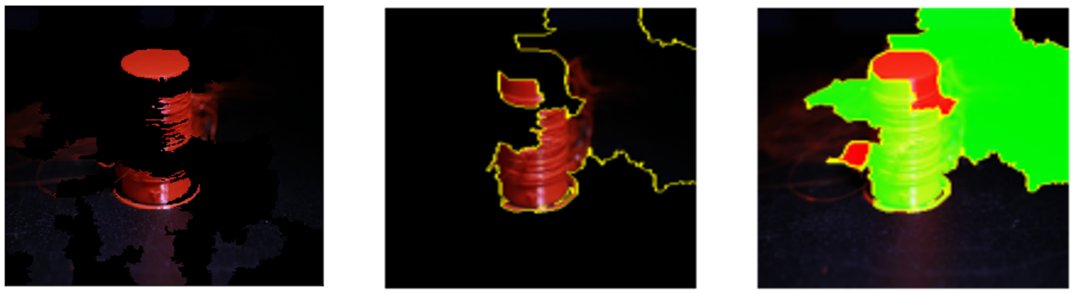


Figure 9.8: Proposed ROIN models prediction interpretation using LIME.

Figure 9.9 illustrates the Grad-CAM approach used to interpret the predictions made by the ROIN model for the given image. The figure shows that Grad-CAM generated the heatmap highlighting the regions of the input image that contributed the most to the model's prediction. In the case of the given image, GradCAM identified the region around the defect as being the most important for the model's prediction, consistent with the findings from LIME.

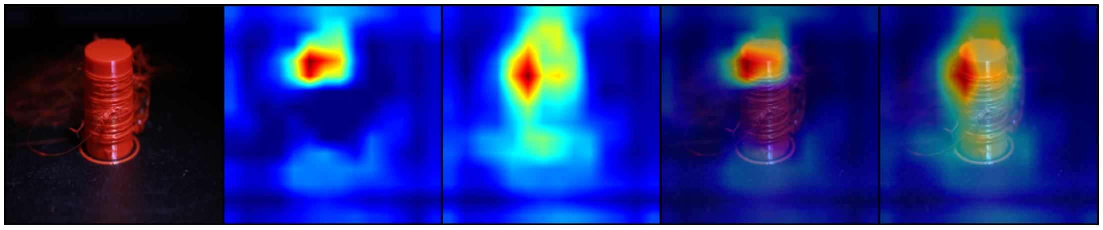


Figure 9.9: Proposed ROIN models prediction interpretation using Grad-CAM.

9.7 Discussion and Overall Findings

This study aimed to develop an approach for detecting defects in 3D-printed cylinder images using TL-based CNN models. While the proposed models achieved high accuracy in identifying defects, they faced challenges localizing the specific defect region. The study found that pre-processing steps, such as identifying the defect regions from the cylinder image, are crucial to improving the performance of CNN models.

The study used the LIME and Grad-CAM approaches to interpret the proposed models' predictions. These techniques enabled identifying essential features in an input image that contributed to the model's prediction. The heatmaps generated from these techniques provided insight into how the model arrived at its decision, allowing for an understanding of its inner workings.

The proposed approaches, including ROIN, ROIHEN, and ROIHEDEN, achieved promising results in detecting defects in 3D-printed cylinder images. The models tested achieved high accuracy, precision, recall, and F1-score values, with all models achieving perfect precision, recall, and F1-score values. The sensitivity of all the models was also 1, indicating that all models correctly classified the defect region as positive. The specificity of all the models was high, with values ranging from 0.8750 to 1.

The results demonstrate that the proposed approaches can improve the performance of CNN models in detecting defects in 3D-printed cylinder images. Furthermore, identifying essential features in the input image using LIME and Grad-CAM approaches can be used to improve the accuracy of the models and reduce the cost and time required for additive manufacturing.

9.8 Conclusion and Future Works

The current study proposed three CNN-based approaches, ROIN, ROIHEN, and ROIHEDEN, integrated with three pre-processing steps, ROI, HE, and DE, for detecting defects in 3D printed cylinder images. The current study addressed the limitation of identifying defect regions, highlighted in Chapter Two, by introducing additional pre-processing steps. The experimental results indicate that the proposed approaches effectively detected the defect regions in the images.

Furthermore, the model's predictions were interpreted using two explainable AI approaches, LIME and Grad-CAM, which generated heatmaps to help understand the models' behavior during predictions. The application of LIME and Grad-CAM approaches provided valuable insights into the inner workings of the models and contributed to the overall interpretability and explainability of the proposed approaches. However, future studies need to validate the generalizability of the proposed approaches on a larger dataset and explore the potential of the approaches in detecting other types of defects.

Chapter 10

Deep MLP-CNN Model Using Mixed-Data

10.1 Introduction

Data balancing approaches are crucial in developing fair Artificial Intelligence (AI)-based diagnostic tools for healthcare. In Chapter 6 and 7, promising results were demonstrated using Convolutional Neural Network (CNN) approaches. However, while CNN approaches can efficiently handle image data, Multi-Layer Perceptron (MLP) models can be used to handle numerical and categorical data (Brouwer, 2002; Bacaksız & Esgin, 2019). In this chapter, a novel approach combining MLP and CNN is proposed to handle mixed data containing numerical, categorical, and image data. The proposed approach is evaluated and tested on both balanced and imbalanced datasets to develop more robust AI-based healthcare diagnostic tools.

10.2 Motivation

The emergence of the Coronavirus Disease 2019 (COVID-19) pandemic has caused a significant global public health crisis, resulting in a large number of cases and fatalities worldwide. Although the Reverse Transcription Polymerase Chain Reaction (RT-PCR) technique is currently considered the gold standard for diagnosis, it has limitations, including a high rate of false-negative results and a shortage of testing kits during the early stages of the outbreak (T. Liang et al., 2020; Pecoraro, Negro, Pirotti, & Trenti, 2022). Therefore, there is an urgent need to develop a cost-effective and reliable early diagnosis screening method, such as AI-based applications. AI has demonstrated potential in medical imaging and diagnosis in recent years, mainly using Deep Learning (DL) techniques (Esteva et al., 2021).

During the pandemic, medical centers and hospitals have explored chest X-ray imaging as a widely available and affordable tool (Elgendi et al., 2020; Calderon-Ramirez et al., 2021). Previous studies have investigated the use of DL techniques on chest X-ray images to detect COVID-19-related pneumonia automatically (N. Kumar, Hashmi, Gupta, & Kundu, 2022; Polat, Özerdem, Ekici, & Akpolat, 2021). Additionally, some studies have employed Machine Learning (ML) algorithms to diagnose COVID-19 patients using numerical and categorical data (Podder, Bharati, Mondal, & Kose, 2021; Huyut, 2023; Dutta, Paul, & Kumar, 2021). However, prior research has yet to integrate chest X-ray images and numerical/categorical data to develop a diagnostic model for COVID-19 patients.

Thus, this study aims to develop a multi-model by taking advantage of MLP and CNN models that leverage image and numerical/categorical data for the early diagnosis of COVID-19 patients. The model is tested on both balanced and imbalanced datasets, and its performance is evaluated. This study offers a reliable alternative for screening COVID-19 patients, contributing to the reduction of mortality rates through early diagnosis, improved treatment, and prevention of disease transmission.

10.3 Chapter Outline

The following chapter is organized as follows: Section 10.4 provides a brief background on COVID-19 and DL-based diagnosis. Section 10.5 outlines the methodology of the experiment in detail. The results of the experiment are presented in Section 10.6, followed by a discussion of the study's findings in Section 10.7. Finally, Section 10.8 summarizes the overall conclusions of the research and suggests potential avenues for future research.

10.4 Background

With the advent of the novel coronavirus (SARS-CoV-2) in December 2019, first detected in the Wuhan Province of China, there was a major outbreak of the associated disease (COVID-19), which causes severe acute respiratory syndrome. More importantly, this virus can be transmitted directly from human to human, making it difficult to be contained. Rapidly, COVID-19 was observed in virtually all countries, triggering a severe public health crisis worldwide (Roosa et al., 2020; Yan et al., 2020). As a consequence, the World Health Organization (WHO) recognized this public health emergency as an ongoing pandemic on March 11, 2020 (Grasselli, Pesenti, & Cecconi, 2020). Coronaviruses (CoV) belong to a large family of viruses that cause diseases related to colds like the Middle East Respiratory Syndrome (MERS-CoV) and the Severe Acute Respiratory Syndrome (SARS-CoV) (Narin et al., 2020).

As of August 30, 2020, the number of Coronavirus cases in the world is approximately hitting the 25.3 million mark, with the total number of deaths surpassing $849,958$ and an associated mortality rate of about 6 percent (Dashbord, June,2020). Statistics show that about 82 percent of the COVID-19 cases have milder symptoms like fever, cough, and dyspnea. However, more serious cases can cause severe acute respiratory syndrome, pneumonia, and multi-organ failure (Narin et al., 2020). With the number of cases increasing daily, most countries find it challenging to keep up with the number of hospitalized patients, more so in Intensive Care Units (ICU). The ICUs are mostly occupied by patients suffering from COVID-19-related pneumonia (Narin et al., 2020). Ultimately, the development of a vaccine is necessary for the prevention and eradication of SARS-CoV-2. However, as the development of such vaccines is still a work in progress, early diagnosis, improved treatment of critical cases, and prevention of the spread through lockdowns are vital to reduce mortality rates (Yuen, Ye, Fung, Chan, & Jin, 2020).

The gold standard for the diagnosis of COVID-19 patients is the RT-PCR technique. However, there has been an inadequate number of testing kits for the SARS-CoV-2 during the disease's early outbreak. The RT-PCR test also produces a high rate of false-negative results, due to sample preparation and quality control in particular (T. Liang et al., 2020). In addition, viruses such as influenza A and influenza B can cause symptoms similar to those of SARS-CoV-2, making it harder to differentiate between COVID-19 and non-COVID-19 cases, more so in the flu season (Zhao, Zhong, Xie, Yu, & Liu, 2020). Uncertainty can lead to a broader spread of the disease if suspected people with the symptoms roam freely without being tested (Zhao et al., 2020). Many overpopulated countries like India and Bangladesh have failed to conduct enough tests due to limited resources for guaranteeing widespread test kit availability (Mohiuddin, 2020; Alam, Alam, Nazir, & Bhuiyan, 2020). Therefore, it is pertinent to develop an early diagnosis screening method considering cost-effectiveness and reliability, such that a larger population is impacted and can benefit from it.

AI is an emerging branch of computer science with demonstrated potential in a wide variety of fields, with applications ranging from decision tools in the energy and financial sectors (Weron, 2014; Ponta, Puliga, Oneto, & Manzini, 2020) to medical imaging and diagnosis. With the unique capabilities of AI, safe, accurate, and efficient imaging solutions can be attained. In fact, AI has recently gained popularity as a useful tool for clinicians (Litjens et al., 2017; Ker, Wang, Rao, & Lim, 2017; D. Shen, Wu, & Suk, 2017; Faust, Hagiwara, Hong, Lih, & Acharya, 2018; Murat et al., 2020; Rizvi et al., 2020). Over the years, similar to many other fields of research, the Deep-learning [1] approach has shown an impressive performance in the field of medical image processing (Narin et al., 2020). By applying Deep-Learning techniques, it is possible to draw meaningful results from medical data (Greenspan, Van Ginneken, & Summers, 2016; D. Shen et al., 2017). By benefiting from Deep-Learning capabilities like image recognition and segmentation, detection and diagnosis of diseases like diabetes mellitus, brain tumors, skin cancer, and breast cancer have been both efficient and useful (Yildirim et al.,

[1]Deep learning is a subset of the machine learning field, inspired by the architecture of the brain (Jakhar & Kaur, 2020).

2019; Saba, Mohamed, El-Affendi, Amin, & Sharif, 2020; Dorj, Lee, Choi, & Lee, 2018; Kassani & Kassani, 2019; Ribli, Horváth, Unger, Pollner, & Csabai, 2018; Celik, Talo, Yildirim, Karabatak, & Acharya, 2020).

Recent studies show that AI-based applications can reduce dependency on the limited RT-PCR test kits (Ozturk et al., 2020). Even if the RT-PCR test shows negative results, symptoms can be identified by examining chest radiological imaging, namely, chest X-ray images (Kanne, Little, Chung, Elicker, & Ketai, 2020; Fang et al., 2020). X-ray machines are popular injury and disease diagnosis tools in most healthcare facilities and have been widely explored by care centers and hospitals during the extent of the current pandemic (Haghanifar, Majdabadi, & Ko, 2020; Ozturk et al., 2020). From a global perspective, X-ray exams are comparatively affordable in developing countries, with exam costs reaching as low as 5 USD (*National Heart Foundation of Bangladesh*, n.d.). In developed countries, as an effect of a more costly healthcare infrastructure, X-ray exams may become more expensive, but are often covered by nearly 100% of public and private health insurance policies in countries like Australia, Canada, Germany, and Japan, and 91% in the USA (*Health System Tracker*, n.d.). Individual charges and copays range from 0 to 50 USD in those countries (*How Much Does an X-ray Cost*, n.d.). Regarding the common concern of exposure to ionizing radiation from X-rays, individual exams are known to be safe and expose the patient to significantly less ionization than, for instance, Computed Tomography (CT) exams (Ozturk et al., 2020).

As a result, chest X-ray imaging recently draws attention to the researcher and practitioner for the early diagnosis of COVID-19 patients with pneumonia symptoms (L. Meng, Hua, & Bian, 2020). For instance, Chen et al. (2020) used a Deep-Learning based model for early detection of COVID-19-related pneumonia using image data from the Renmin Hospital patients at Wuhan University (J. Chen et al., 2020). Narin et al. (2020) describes the use of X-ray images for the coronavirus' automatic detection by implementing a Deep Convolutional Neural Network, achieving an accuracy of around 98% using the ResNet50 model (Narin et al., 2020) . Apart from this, Goshal and Tucker (2020) and Wang and Wong (2020) also developed a Convolutional Neural Network (CNN) to classify COVID-19 and Non-COVID-19 cases using X-ray images, with approximately 92.9% and 83.5% accuracy respectively (Ghoshal & Tucker, 2020; L. Wang & Wong, 2020). Additionally, there are numerous other recent studies carried out with CT images using several Deep Learning models (C. Jin et al., 2020; Song et al., 2020; Butt et al., 2020; F. Shi et al., 2020). Likewise, ML algorithms using numerical/categorical data have also been utilized for the diagnosis of COVID-19. A number of studies (L. Meng et al., 2020; Song et al., 2020; Gong et al., 2020) developed machine learning models based on Lasso regression, and multivariate logistic regression for early identification of COVID-19 patients. Some of the significant factors in these studies were age, temperature, heart rate, blood pressure, fever, sex, uric acid, triglyceride and serum potassium.

Even though the Center for Disease Control and Prevention (CDC) currently doest not recommend, still, many studies in this field of research use Chest radiography or CT scan images to diagnose COVID-19 (*COVID-19 Diagnostic Imaging*

Recommendations, n.d.; *ACR Issues Statement for Use of Chest Radiography, CT for Suspected COVID-19 Infection*, n.d.). For instance, a recent report in the journal of Applied Radiology (2020, March 22) (*ACR Issues Statement for Use of Chest Radiography, CT for Suspected COVID-19 Infection*, n.d.) claimed that using radiological images alone detects patients with ARDS [2] , SARS [3], as COVID-19, which is a drawback since the diseases are misclassified. Articles by Greenfieldboyce and Jewell suggested that a patient's information such as age, gender, temperature, and chronic disease history are significant predictors to identify affected COVID-19 patients (*The New Coronavirus Appears To Take A Greater Toll On Men Than On Women*, n.d.; *Everything You Should Know About the 2019 Coronavirus and COVID-19*, n.d.). Keeping this in mind, some of the studies in this field of research use (numerical or categorical) information such as age, gender, body temperature, and chronic disease history for diagnosis of COVID-19 as well. For instance, Bai et al. (2020), uses CT images (image data) and a combination of demographics, signs, and symptoms (numerical/categorical data) to establish an AI model that predicts patients having mild symptoms with potential malignant progression (Bai et al., 2020). However, none of the previous studies considered numerical, categorical, and chest X-ray images in combination. Thus, developing a model comprising of numerical/categorical data coupled with chest X-ray images may create a new reliable alternative to screen patients with COVID-19 symptoms.

Considering these opportunities, this study focuses on mixed-data analysis us-

[2]ARDS also known as Acute respiratory distress syndrome (Force et al., 2012)
[3]Also known as severe acute respiratory syndrome (Lau et al., 2008)

ing both image and numerical/categorical data to assist in the early diagnosis of COVID-19 patients using a DL approach. A deep Multilayer Perceptron-Convolutional Neural network (Deep MLP-CNN) model is proposed considering the age, gender, temperature, and chest X-ray images of patients. The model was tested under two conditions: a balanced dataset (containing 13 COVID-19 and 13 non-COVID-19 patients), henceforth referred to as Study One, and an imbalanced dataset (containing 112 COVID-19 and 30 non-COVID-19 patients), referred to as Study Two.

10.5 Dataset and Methodology

A COVID-19 dataset containing X-ray images and numerical/categorical data for each patient was collected from the open-source GitHub repository shared by Dr. Joseph Cohen (Cohen et al., 2020). This database is continuously being updated with data shared by several entities around the world and has been used by many studies for detecting COVID-19 patients considering various data mining techniques. At the time of the study, the dataset contains data from 184 different patients with information such as age, gender, temperature, survival, intubation, partial pressure of oxygen dissolved in the blood (PO2), and classification as COVID-19, SARS, Pneumocystis, E. coli, Streptococcus, or "no findings" patients. For simplicity, the dataset has been organized into two groups: COVID-19 patients and all others as non-COVID-19 patients (Figure 10.1).

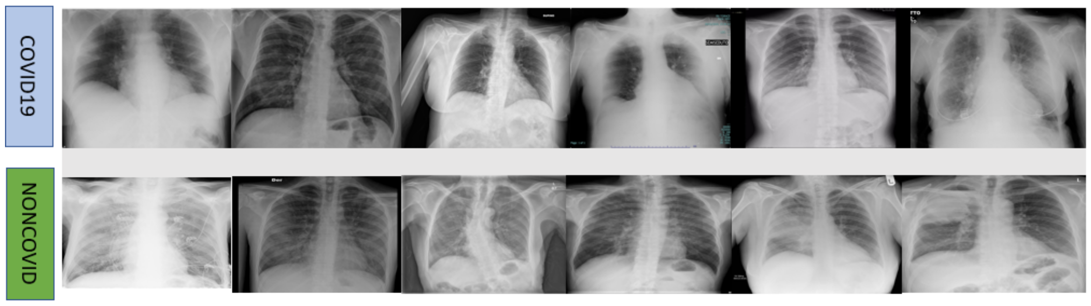


Figure 10.1: Sample set of test images, including the chest X-ray images of COVID-19 and non-COVID-19 patients (Cohen et al., 2020).

One of the challenges associated with this dataset was the missing data for select parameters across patients. In consideration of that limitation, for Study One (balanced dataset), a small dataset was set up with 13 COVID-19 and non-COVID-19 patients considering age, gender, temperature, and chest X-ray images as variables. Since there were numerous missing entries in the temperature column, only rows with complete information of the aforementioned variables were taken into account. No statistically significant difference[4] was found between COVID-19 (6 female, 7 male) and non-COVID-19 (5 female and 8 male) groups in terms of sex distribution ($p^* = 0.69>0.05$), mean of age and temperature ($p^{**} = 0.49>0.05$). Contrarily, the size of the dataset was enlarged by ignoring the "Temperature" column entirely for Study Two. In this case, an imbalanced dataset was constructed with information from 142 patients (112 COVID-19, 30 non-COVID-19) to compare and contrast the model's performance with the imbalanced class. No statistically significant difference was observed between

[4] P value were obtained using t-test (*) and chi-square test (**).

COVID-19 and non-COVID-19 groups in regards to the sex distribution (p* =0.34>0.05) and the mean of age (p** = 0.06>0.05). Table 10.1 summarizes the datasets used for both studies. The implementation of the MLP-CNN models

Table 10.1: Balanced and Imbalanced datasets used in this study.

Dataset	Label	Training Set	Testing Set	Total	$Mean \pm SD$		P-value
					Age (years)	Temperature (celsius)	
Study One	COVID-19	9	4	13	51.29±16.72	38.26 ±.85	0.49
(Balanced dataset)	Non-COVID-19	11	2	13			
Study Two	COVID-19	87	25	112	55.73 ±16.66	—	0.06
(Imbalanced dataset)	Non-COVID-19	26	4	30			

and calculation of computational times took place using the Anaconda modules with Python 3.7, and ran on an office-grade laptop with common specifications (Windows 10, Intel Core I7-7500U, and 16 GB of RAM).

10.5.1 Proposed Model

Neural networks (NN) recently showed promising results than traditional Machine Learning (ML) algorithms like Linear Regression, Logistic Regression, and Random Forest, with high dimensional datasets, primarily when it contains numerical, categorical, and image data combined (Nakada & Imaizumi, 2019). Classical ML approaches may perform better with a small dataset as it is computationally inexpensive and easily interpretable. However, once the size of the data increases (big data), handling such big data becomes challenging for traditional ML approaches. Conversely, deep NN methods guarantee an opportunity to develop a more robust model that perform well on both small and large datasets, mainly due to recent advancements in different NN approaches such as Transfer Learning (TL), Recurrent Neural Network (RNN), and CNN. Additionally, classical ML approaches often require sophisticated feature engineering or dimensionality reduction (Zhou, Pan, Wang, & Vasilakos, 2017). In contrast, deep NN methods: provide better feature engineering methods, can be implemented directly, and achieve good results (Chollet, 2017).

A developed a DL-based model was developed inspired from (Ahmed & Moustafa, 2016). The choice for this architecture was motivated by its predictive performance on visual and textual features, addressed in many recent papers (Law, Paige, & Russell, 2019; F. Wang, Zou, Zhang, & Shi, 2019; Koch et al., 2020; E. S. Kumar, Talasila, Rishe, Kumar, & Iyengar, 2019). The proposed model is a combination of a Multilayer perceptron (MLP) and Convolutional Neural network (CNN). On one hand, MLP was used to handle the numerical/categorical data; on the other, CNN was used to extract features from the X-ray images. Parameter tuning was performed to improve the model's performance, mainly: the number of hidden layers, number of neurons, epochs, and the batch size. At first, hidden layers and the number of neurons were set randomly; however, the optimal parameters were later determined using the grid search method. The optimized parameters using the grid search method are as follows: Learning Rate = 0.001, Batch Size = 5, Epochs = 50. Finally, the proposed MLP model was combined with the CNN architecture, as suggested by (Ahmed & Moustafa, 2016). As shown in Figure 10.2, the highest accuracy (100%) was achieved on 50 epochs,

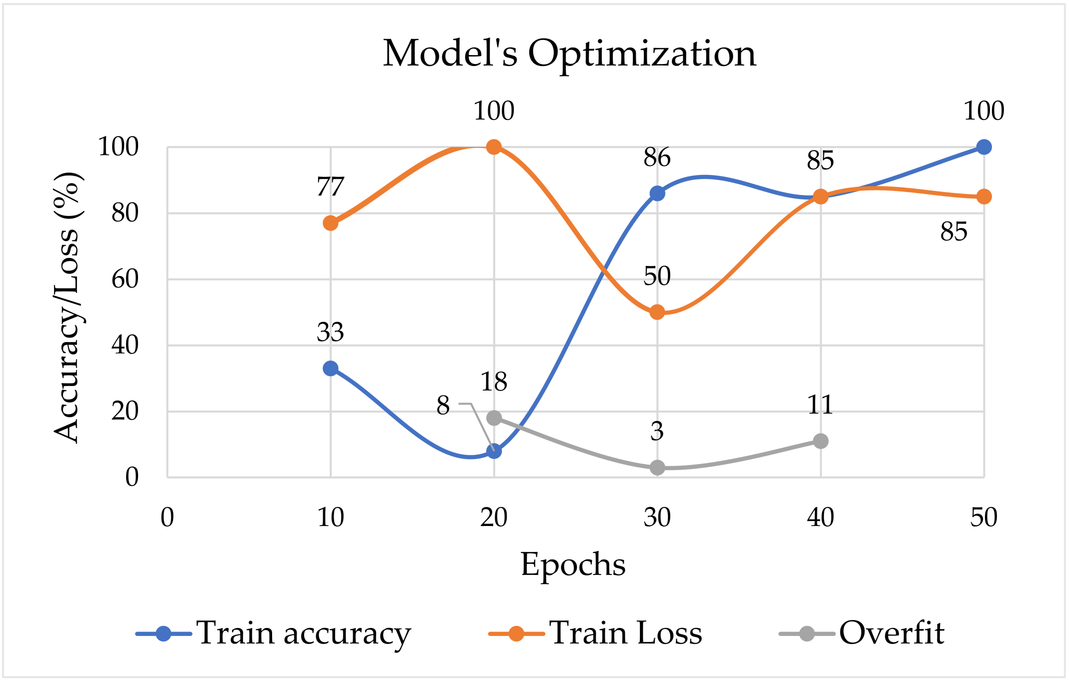


Figure 10.2: Training accuracy and loss with number of epochs.

while the training loss was minimized up to 85%.

Table 10.2 shows how different numbers of neurons and hidden layers affect the MLP models. Based on the experiment, with two hidden layers and four neurons, it is possible to achieve 100% accuracy while reducing the loss up to 100%.

Table 10.2: Model performance with different numbers of neurons and hidden layers.

Number of hidden layers	Neuron	Accuracy (%)	Loss (%)
1	4	42	76
1	8	56	100
2	4	100	100
2	8	35	76
3	4	33	77
3	8	64	100

In order to obtain the best model, optimization algorithms needed to be applied during the training phase (Sutskever, Martens, Dahl, & Hinton, 2013). For that purpose, three popular optimization algorithms were tested: Adaptive learning rate optimization algorithm (Adam) (Kingma & Ba, 2014), Stochastic gradient descent (Sgd) (C. Zhang et al., 2018), and Root mean square propagation (Rmsprop) (Bengio, 2015).

10.5.2 How Proposed MLP-CNN Model Works

Here, the Rectified Linear Unit (ReLU) was applied as the activation of each neuron in the input and hidden layers and the "linear" function was utilized in the final layer (Tang, 2013). The first input layer of the MLP consists of eight neurons and takes the numerical/categorical data as a one-dimensional array. The hidden layer consists of four neurons and the final layer consists of one neuron.

Secondly, the proposed CNN model contains three convolution layers, along with three pooling layers (Max Pooling). The first hidden layer is a convolutional layer with 16 feature maps, each with a kernel size of 64 pixels and a "ReLu" activation function. Then, the pooling layer is then defined, which takes the maximum value and is configured with a pool size of (2,2). The following pooling layer is a dense layer that takes 16 neurons, succeeded by activation function-ReLU. The next layer is another dense layer with four neurons. Two individual outputs emerged from two separate models—one from the MLP model and the other from the CNN model. Both outputs are concatenated and considered as a single input. The newly acquired single input was counted as an initial input followed by additional two dense layers consisting of 4 neurons. The Keras functional API was utilized to concatenate the MLP and the CNN models, as it provides a potential opportunity to develop models that require multiple inputs and outputs. Typically, such models merge the inputs from different layers using an additional layer and combine several tensors, as shown in Figure 10.3, which illustrates the overall diagram of our proposed end-to-end model. In summary, the numerical/categorical and chest X-ray inputs were encoded as vector inputs and then concatenated. Finally, the output layer has one neuron for the two classes and a linear activation function to provide probability-like predictions for each class.

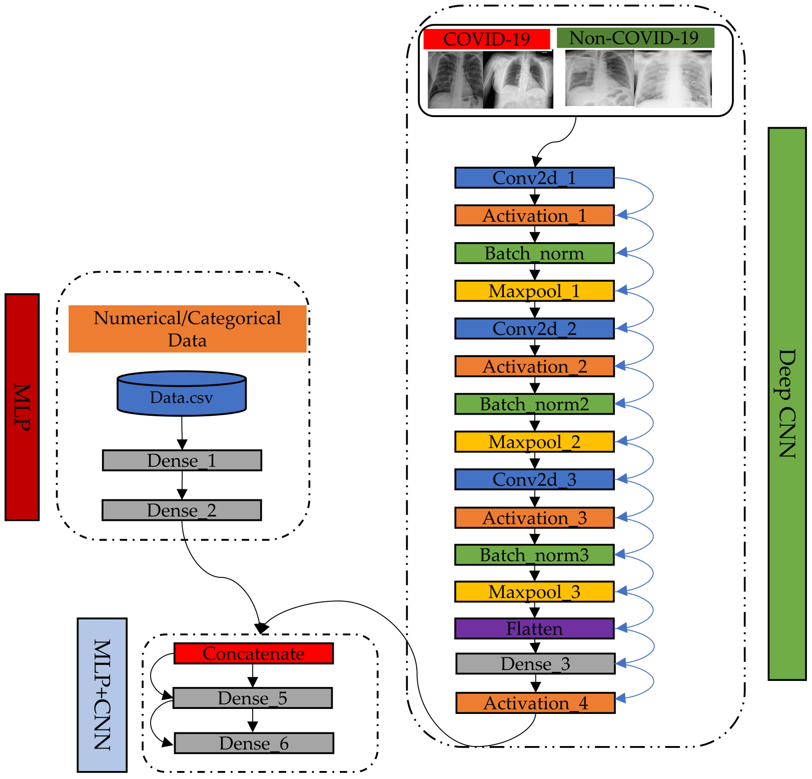


Figure 10.3: Flow diagram of proposed MLP-CNN models. The model contains three components: MLP, CNN, and Merged MLP-CNN.

10.5.3 Experiment Setup

The performance of the model was evaluated using 5-fold cross-validation for both Studies (Study One and Study Two).

The results were presented in terms of Accuracy, Precision, Recall, and F1-score with 95% confidence interval (Ahsan, 2018).

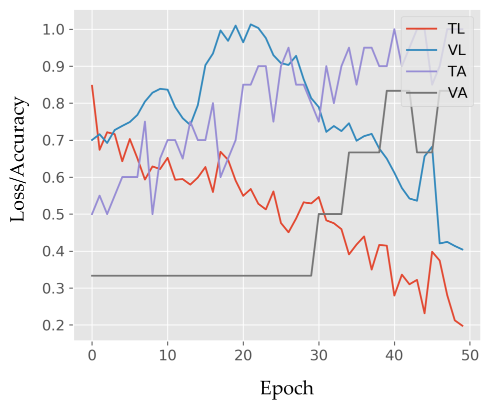
10.6 Computational Results

At first, as means of identifying appropriate training and testing set ratios for validation, we have split our data into the following train set/test set ratios: 75:25, 70:30, 60:40, 85:15, and 80:20. Such split ratios are commonly used in DL techniques for model evaluation and validation (Mohanty et al., 2016; Menzies et al., 2006; Stolfo et al., 2000). The best results in terms of training and testing accuracy were found when the dataset was split randomly into 80% and 20% for training and testing sets, respectively. To exemplify that, Table 10.3 presents the performance of our proposed models with different ratios of randomly split data between training and testing. Since the dataset is comparatively small, reducing training data also reduces the model's ability to achieve better performance in terms of accuracy. In contrast, increasing the training set with a small number of datapoints for testing is not sufficient to confidently measure the model's overall performance.

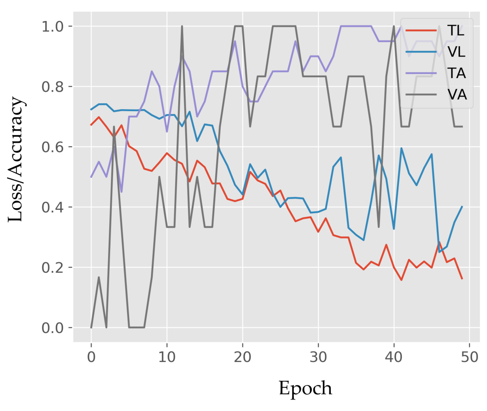
Table 10.3: Proposed MLP-CNN models performance incorporating different amounts of training and testing data in Studies One and Two.

Data ratio (%)	Study One		Study Two	
Training/Testing	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy
75/25	94%	71%	90%	70%
70/30	66%	50%	95%	60%
60/40	73%	45%	70%	71%
80/20	100%	100%	100%	96%

The training stage was carried out up to no more than 50 epochs to avoid overfitting. A graphical illustration of the model's overall performance using Adam and Rmsprop is presented for the 5^{th} fold in Figure 10.4.



(a)



(b)

Figure 10.4: Model accuracy and loss in fold 5. TL – Train loss; VL – Validation loss; TA – Train accuracy; VA – Validation accuracy. In Study One, the model trained with Adam (a) reached 100% accuracy and loss decreased by almost 100% after 50 epochs. Alternatively, for Study Two, the model trained with Rmsprop (b) reached 100% accuracy and loss decreased below 5%.

Each model's average performance on both balanced (Study One) and imbalanced (Study Two) datasets along with 95% confidence intervals are displayed in Table 10.4. For the balanced dataset, Adam has the highest accuracy (96.3%), precision (97.2%), recall (96.3%), and F1-score (96.4%) compared to the other two models-trained with Rmsprop and Sgd. Rmsprop outperformed all other models on the imbalanced dataset. While considering the overall performance on both datasets (average of both studies) the model trained with Adam is the best in terms of accuracy (94.6% ± 3.4%), precision (93.5% ± 3.7%), recall (94.5% ± 3.5%), and F1-score (93.5% ± 3.7%).

Table 10.4: COVID-19 screening performance of our model on Study One and Study Two with 95% Confidence Interval ($\alpha = 0.05$). S_1 – Study One; S_2 – Study Two; CI – Confidence Interval.

Algorithm	Accuracy (%)			Precision (%)			Recall (%)			F1 score (%)		
	S_1	S_2	CI	S_1	S_2	CI	S_1	S_2	CI	S_1	S_2	CI
Adam	96.3	92.9	94.6 ±3.4	97.2	89.9	93.5 ±3.7	96.3	92.8	94.5 ±3.5	96.4	90.7	93.5 ±3.7
Rmsprop	82.5	95.4	88.9 ±4.7	79.4	92.5	85.9±5.3	82.9	95	88.9 ±4.8	79.6	93.6	86.6 ±5.2
Sgd	91.8	85.1	88.4 ±4.9	95.4	82.1	88.7 ±4.8	91.8	86.1	88.5 ±4.8	82.4	83.6	83 ±5.7

Overall execution time for both datasets is shown in Figure 10.5. The lowest registered execution time was 53 seconds for the model trained with Rmsprop and the balanced dataset, whereas the maximum execution time was 79 seconds when the model was trained with Sgd. Conversely, for the imbalanced dataset, Adam showed the lowest execution time of 138 seconds, while Rmsprop displayed the maximum execution time of 163 seconds. In conclusion, when both studies are considered holistically, the average execution time for Adam is lowest in comparison with the other two.

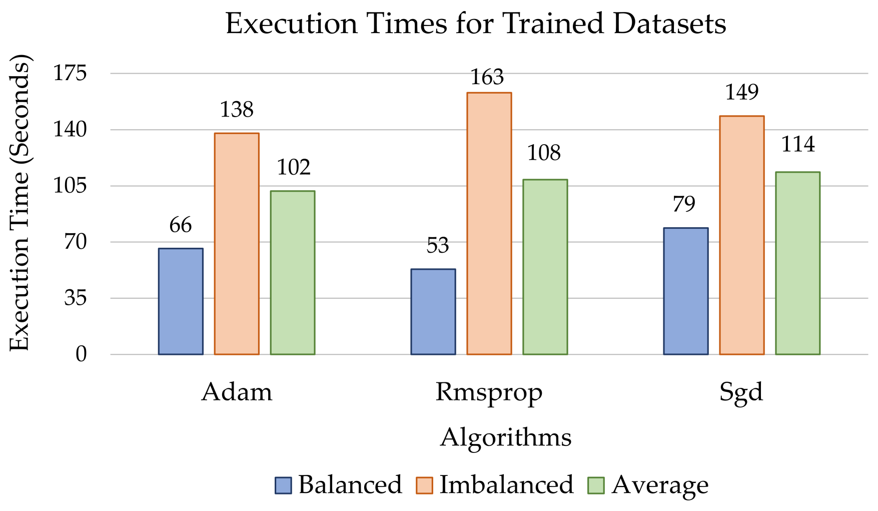


Figure 10.5: Execution time of models trained with Adam, Rmsprop, and Sgd for both balanced and imbalanced datasets.

To evaluate the predictive performance of each model, confusion matrices were generated. Figure 10.6 shows confusion matrices for the models trained with Adam, Rmsprop, and Sgd, respectively, for fold-5. In Study One, the test set contained 6 patients, where 4 were COVID-19, and 2 were non-COVID-19. In this case, both Adam and Sgd correctly classified all the samples, while Rmsprop misclassified 4 out of 6 samples. On the other hand, in Study Two, 29 samples were used for the test set (25 COVID-19 and 4 non-COVID-19). Here, Adam illustrates the best performance by correctly classifying 29 samples, while Rmsprop and Sgd show the worst performance by classifying 27 samples out of 29.

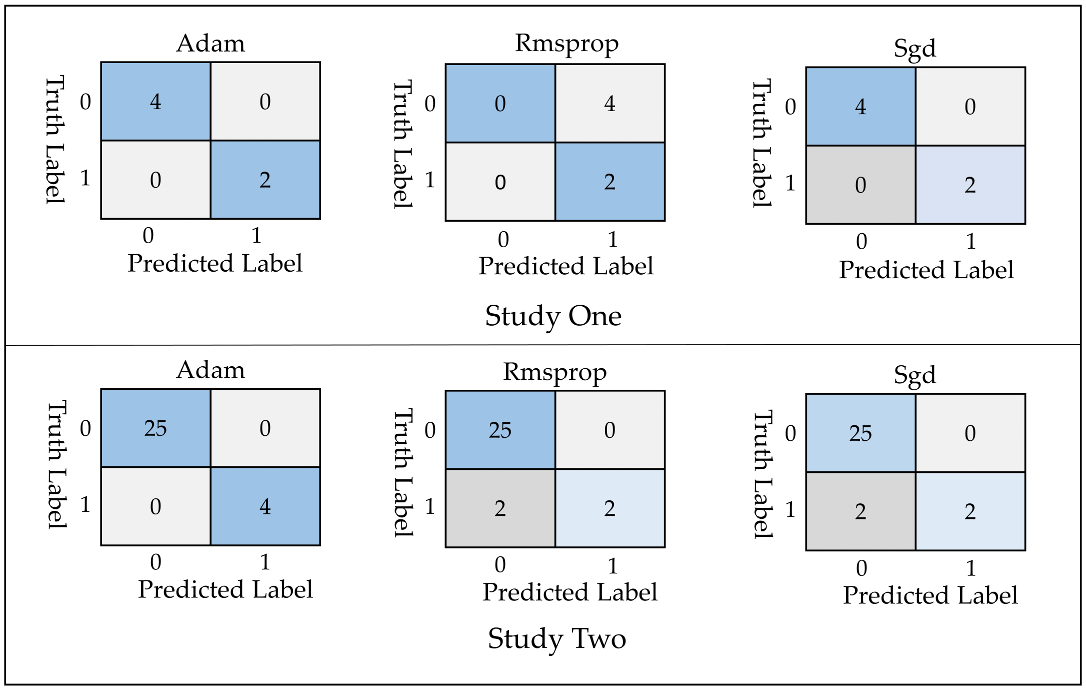


Figure 10.6: Confusion matrix of our models trained with optimization algorithms (Adam, Rmsprop, and Sgd) for both studies at fold 5.

10.7 Discussion

In this study, an MLP-CNN based model was proposed and evaluated for distinguishing between patients with and without COVID-19, demonstrating the superiority of combined MLP-CNN models over traditional CNN or MLP models used exclusively for this purpose. The combined model achieved an accuracy of around 96.3% (using Adam optimization algorithm) in comparison to few published studies that used only CNN (L. Wang & Wong, 2020; Ghoshal & Tucker, 2020; Song et al., 2020) or traditional ML (F. Shi et al., 2020) approaches. On one hand, MLP models are fast and time-efficient when used with numerical and/or categorical data only. On the other hand, CNN models are notably accurate in extracting useful features from chest X-ray images for respiratory disease diagnosis. For instance, Wang and Wong (2020) and Khan et al. (2020) used CNN-based approaches to detect the onset of COVID-19 disease using chest X-ray images and achieved an accuracy of 83.5% and 89.6%, respectively. In comparison, as previously stated, the proposed combined model demonstrated an accuracy of around 95.4% (L. Wang & Wong, 2020; Khan et al., 2020).

In Study One, MLP-CNN model learned from only 26 COVID-19 subjects, which represents 18% of the data used by Zhang et al. (2020) and 2% by Shi et al. (2020) (see Table 10.5 and 10.6) (J. Zhang, Xie, Li, et al., 2020; F. Shi et al., 2020). Therefore, the proposed MLP-CNN model may be used as a useful computer-aided diagnosis tool for low-cost and fast COVID-19 screening considering small datasets.

Table 10.5: Comparative analysis of the proposed MLP-CNN-Based COVID-19 diagnostic method and Other Deep-Learning Approaches utilizing Chest X-ray Images.

Model	Accuracy
(Ghoshal & Tucker, 2020)	92.9%
(J. Zhang, Xie, Li, et al., 2020)	96%
(L. Wang & Wong, 2020)	83.5%
Proposed model	96.3% with Adam

Additionally, the MLP-CNN model performed better with an imbalanced dataset compared to recent studies (C. Jin et al., 2020; Song et al., 2020; Butt et al., 2020; F. Shi et al., 2020; L. Wang & Wong, 2020) that also used imbalanced datasets (Table 10.6). For instance, Jin et al. (2020) used 1882 CT scan images where the data ratio was $1 : 2.78$ (497 COVID-19:1385 other) and achieved 94.1% accuracy (C. Jin et al., 2020). Similarly, the data ratio of a range of other recent studies, particularly Song et al. (2020), Butt et al. (2020), and Shi et al. (2020) were $1 : 2.11$, $1 : 1.82$, and $1 : 1.62$ respectively; while their measured accuracy was 82.9%, 86.7% and 90.7% accordingly (Song et al., 2020; C. Jin et al., 2020; Butt et al., 2020; F. Shi et al., 2020). For the imbalanced dataset, 112 COVID-19 and 30 non-COVID-19 patients' (ratio $1 : 3.73$) chest X-ray images were used. Despite the higher ratio of imbalance in the dataset, the proposed method outperformed several recent studies with similar objectives by achieving a higher accuracy of 95.4% (C. Jin et al., 2020; Song et al., 2020; Butt et al., 2020; F. Shi et al., 2020; L. Wang & Wong, 2020). It should be noted that all studies mentioned in Table 10.6 used only image data in their experiments, while we considered a mixed-data approach, using both numerical/categorical and image data.

In summary, an MLP-CNN based model has been proposed in this study that can determine between COVID-19 and non-COVID-19 patients using information like age, gender, temperature, and chest X-ray images. Both balanced and imbalanced data were considered for the experiments, achieving an average accuracy of around 95% (96.3% from Study One and 95.4% from Study Two).

Table 10.6: Comparison between current literatures with other DL methods developed using imbalanced datasets.

References	Data type	Method	Database size	Accuracy
(C. Jin et al., 2020)	CT	CNN	497 COVID-19, 1385 others	94.1%
(Song et al., 2020)	CT	ResNet50	88 COVID-19, 186 others	82.9%
(Butt et al., 2020)	CT	CNN	219 COVID-19, 399 others	86.7%
(F. Shi et al., 2020)	CT	RF	1658 COVID-19, 1027 others	90.7%
(L. Wang & Wong, 2020)	X-ray	CNN	45 COVID-19, 2794 others	83.5%
(Khan et al., 2020)	X-ray	Xception	284 COVID-19, 967 others	89.6%
Proposed model	X-ray	MLP-CNN + Rmsprop	112 COVID-19, 30 others	95.38%

Finally, the proposed model can be easily adopted by healthcare professionals as it is cost- and time-effective, which accelerates COVID-19 screening procedures and enables patients with the disease to be isolated at earlier stages. Real-time screening of COVID-19 patients using MLP-CNN approaches might be possible with minimal human interaction, provided that chest X-ray images and other relevant information such as age, gender, and temperature of the respective patients are available. Additionally, AI-based screenings can be tailored to a low degree of complexity to the end user, and may not require the training of technicians in the complex computational tools herein described. We identify the following limitations of our study, which present immediate opportunities for future investigations:

1. The size of the dataset adopted is comparatively small, and

2. Only four numerical and categorical parameters were considered.

10.8 Conclusion

In this study, an MLP-CNN based model has been proposed for early diagnosis of patients with COVID-19 symptoms considering mixed-data, in particular, numerical/categorical data (age, gender, and temperature) and image data (chest X-ray images). Results showed using mixed-data, it is possible to develop a more accurate model with a small, and balanced dataset (accuracy 96.30%). Furthermore, the proposed model also performed better (accuracy 95.4%) on the imbalanced dataset compared to other studies, as shown in Table 10.6.

Finally, this experiment could provide valuable insights in developing a screening system to support healthcare providers in distinguishing among COVID-19 and non-COVID-19 patients, such that patients with the disease can be identified and isolated at an earlier stage. Future studies should reapply these methods in larger datasets with more images and complete patient information, work with highly imbalanced data, apply mixed-data analysis using kernel methods, and consider data containing the geographical location of patients.

Chapter 11

Conclusions

This dissertation aims to address class imbalanced problems (CIP) for numerical, categorical, and mixed data by proposing various data balancing approaches to handle CIP issues. The study considers various industrial case scenarios, such as quality, defect, and pattern analysis. The current chapter presents the overall findings and contributions of the dissertation.

Section 11.1 provides closure by revisiting the research questions posed in Chapter 1 and reviewing the answers that have been offered. In Section 11.2, the study outcomes of individual research questions are summarized. Finally, the resulting contributions are then summarized in Section 5.8.

11.1 Overall Summary of All Research Questions

This dissertation investigated various aspects of data balancing approaches in the context of quality, defect, and pattern analysis. In Chapter 3, a comprehensive review of different data balancing approaches was provided, focusing on oversampling and undersampling techniques. The study then delved deeper into one of the most widely used oversampling techniques, SMOTE, and conducted hypothesis testing to identify its impact on ML models' performance with or without parameter tuning. The study aimed to address RQ2: What is the effect of traditional Machine Learning (ML) and Synthetic Minority Over-sampling Technique (SMOTE)-based data-balancing on imbalanced data analysis?

Subsequent chapters proposed three novel data balancing solutions: SSG, GBO (Chapter 4), and BS-GAN (Chapter 5), and compared their performance with existing literature—the proposed solutions aimed to generate more realistic samples to improve the accuracy of ML models. The study also evaluated the effectiveness of data balancing approaches using image data in Chapters 6 and 7 (X-ray and CT-scan images) and Chapters 8 and 9 (images of 3D printed cylinders) using advanced Transfer Learning (TL) approaches. Novel CNN-based approaches such as MVGG16, ROIN, ROIHEN, and ROIHEDEN were introduced to develop robust CNN models to address class imbalance problems. Chapter 10 proposed a novel method, the MLP-CNN model, to handle mixed data (image and tabular data) and addressed RQ3: How does imbalanced data affect the performance of Deep Learning (DL)-based models? Additionally, the study used Local Interpretable Model-Agnostic Explanations (LIME) and Shapley Additive Explanations (SHAP) for the ML and DL-based models' explainability where necessary and applicable to provide better insights into the proposed data balancing solutions.

The dissertation concludes with discussions on opportunities and future work related to all aspects of data balancing approaches, presented at the end of each chapter. The opportunities presented stem from a limitation in the methodology used due to time constraints or represent ideas of interest that can be further investigated in the future.

11.2 Summary of Individual Research Questions

11.2.1 What are the scopes of data-balancing approaches toward the major and minor samples?

The scope of data-balancing approaches towards major and minor samples is significant in improving the performance of Machine Learning (ML) models. The data distribution on imbalanced datasets affects the ML model's performance during the prediction, making it challenging to predict without bias toward the major class. Several studies have suggested data preprocessing steps before applying oversampling methods to address these issues. Data normalization techniques have been utilized to improve data distribution, resulting in high accuracy levels in some studies. However, it is still being determined whether using data normalization techniques significantly affects the overall performance of the ML model. Additionally, using several data normalization techniques before the oversampling approach may change the entire minor class data distributions, which may not be helpful in predicting outlier or minor samples. The performance of ML algorithms on oversampled datasets is inconsistent, and using similar SMOTE approaches may result in varying accuracy levels. The proper parameter tuning of SMOTE and preprocessing steps are necessary to improve ML algorithms' data distribution and performance.

Various undersampling techniques have been explored by researchers to address class imbalance problems (CIP). These techniques include Random Undersampling, Near-Miss (NM) algorithm, Cluster Centroids (CC), and Condensed Nearest Neighbor (CNN). CC has demonstrated promising results in classification accuracy, F1-score, and AUC-ROC compared to other undersampling techniques. However, it may not be appropriate for datasets with high-dimensional features, overlapping classes, or many classes. CNN selects a subset of samples from the majority class by iteratively removing samples that can be classified correctly using the nearest neighbor rule. Recently, a new clustering-based undersampling technique called CUTE has been proposed to improve consistency in various measures. Other undersampling techniques, such as NM and OSDU, have also been proposed to address the class imbalance in datasets.

Undersampling techniques have limitations that can negatively impact their performance. One of the main limitations of undersampling is the loss of information, which can affect the performance of ML algorithms. Additionally, undersampling can increase the bias towards the majority class, resulting in poor classification performance for the minority class. Furthermore, undersampling techniques may not be suitable for datasets with high-dimensional features and many classes or overlapping classes. The characteristics of the datasets may also limit the effectiveness of undersampling techniques. Undersampling techniques may not be effective for highly imbalanced multiclass datasets as they can significantly reduce the size of the majority class. Further research is required to investigate the effectiveness of undersampling techniques on various datasets and to address their potential limitations.

In recent years, oversampling techniques have gained significant attention and have become the most widely used approach compared to undersampling techniques in various applications, including quality control, defect analysis, and pattern recognition. Oversampling techniques are not affected by many of the previously mentioned limitations of undersampling techniques, addressing the class imbalance issue in datasets. Synthetic Minority Over-Sampling Technique (SMOTE), Adaptive Synthetic Sampling (ADASYN), and Borderline-SMOTE are some of the most popular and widely used oversampling techniques used by researchers and practitioners. The increased number of minority class instances through oversampling techniques can enhance the classification performance of ML algorithms, making them more effective in detecting defects or recognizing patterns. However, the effectiveness of oversampling techniques can vary depending on the dataset and specific application. Therefore, careful evaluation and experimentation are necessary to select the appropriate oversampling technique for the given dataset and application. Despite their advantages, oversampling techniques have limitations that can negatively impact their performance, such as overfitting, creating noisy samples, and not working well on highly overlapping classes or noisy features. Additionally, no single oversampling technique can effectively work for all types of datasets. Therefore, additional research is necessary to address these limitations and develop new oversampling techniques that can effectively handle highly imbalanced datasets while addressing these challenges.

SMOTE are among the most popular techniques for handling class-imbalanced problems. One of the significant advantages of using SMOTE is that it creates a more diverse set of synthetic samples than traditional mean or mode-based oversampling techniques. However, a potential drawback of SMOTE is that it can lead to overlapping major and minor classes, resulting in biased statistical outcomes. Several improved versions of SMOTE have been proposed to address this issue, including SVM-SMOTE, Borderline-SMOTE, and adaptive synthetic minority oversampling techniques (ADASYN). Nonetheless, these approaches also have limitations, and many hybrid and improved SMOTE-based oversampling techniques have been proposed to handle class-imbalanced problems more effectively. The performance of different oversampling techniques and ML models varies across datasets, and it is crucial to properly evaluate data distribution after data expansion to ensure accurate model performance.

The combination of over- and under-sampling methods, also known as hybrid methods, has been proposed as a potential solution to the limitations of individual oversampling or undersampling techniques. By combining these methods, it is possible to reduce the impact of the limitations and achieve better performance on imbalanced datasets. Several studies have shown that combining over- and under-sampling methods can outperform individual oversampling or undersampling techniques on imbalanced datasets. However, further research is needed to investigate the optimal combination of techniques and their impact on different classifiers and performance metrics. The SMOTETomek method is a popular hybrid method that combines oversampling with SMOTE and undersampling with Tomek links. While SMOTETomek has been shown to improve classification performance on moderately imbalanced datasets, it may have limited effectiveness on highly imbalanced datasets where the minority class makes up less than 10% of the data. In such cases, more advanced techniques may be necessary. Additionally, SMOTETomek involves a combination of oversampling and undersampling techniques, which can increase the computational time required to train a model. This can be a concern when dealing with large datasets or when the technique is applied repeatedly in an ML pipeline. Finally, the risk of overfitting is also a potential limitation of SMOTETomek, where the oversampling component can introduce bias in the data by creating synthetic samples that closely resemble existing minority class examples. Careful hyperparameter tuning and cross-validation can help mitigate this risk, but it remains an important consideration when using SMOTETomek.

Many references in the literature suggest GAN-based approaches as a potential data-balancing technique in addition to various other types of data-balancing techniques. While SMOTE has been widely used, it may not be effective for highly imbalanced multiclass datasets, which has led to suggestions for GAN-based approaches. GAN has gained popularity in computer vision due to its ability to generate realistic images from random noise, making it useful for nearly any data format. GAN-based approaches have been shown to generate diverse data samples and can solve the CIP ratio problem. However, GAN can be challenging to train, requiring a large amount of data and numerous iterations to achieve optimal results. Moreover, the parameters of GAN are sensitive, making it an unstable approach. Despite these drawbacks, GAN-based approaches have shown promising results in overcoming the multicollinearity and overlapping issues of oversampling approaches. Therefore, if GAN-based algorithms are stable, computationally effective, and require fewer iterations to train, they could be the best alternative to overcome the existing challenges associated with SMOTE and GAN-based approaches.

11.2.2 What is the effect of traditional Machine Learning (ML) and Synthetic Minority Over-sampling Technique (SMOTE)-based data-balancing on imbalanced data analysis?

Based on the findings from RQ1, it was evident that SMOTE-based approaches were commonly used for data balancing. However, to compare their effectiveness with traditional ML algorithms, the study further investigated scenarios with and without parameter tuning. Therefore, the study provided an overview of widely utilized ML algorithms, evaluated their performance on an array of datasets before and after SMOTE-based oversampling through various statistical measures, and executed a hypothesis test to determine the significance of SMOTE-based approaches in data balancing by examining four distinct conditions related to ML model performance. Consequently, the study evaluated the performance of various ML models, including Adaboost, Decision Tree, Gradient Boosting, K-Nearest Neighbors, Logistic Regression, Random Forest, and Support Vector Machine (SVM) models on six different datasets, namely Ionosphere, Pageblocks, Poker, Spambase, Winequality, and Yeast.

The study examined the effect of SMOTE oversampling technique and different feature selection and preprocessing techniques on model performance. The evaluation metrics included accuracy, precision, recall, F1-score, ROC, and G-mean.

The findings revealed that the performance of the models varied depending on the dataset and the chosen conditions. The SMOTE oversampling technique improved the performance of the Adaboost model on Ionosphere, Pageblocks, Poker, and Spambase datasets. The Decision Tree model demonstrated high performance on the Pageblocks dataset but low performance on the Poker dataset. The Gradient Boosting model showed strong performance on most datasets, with conditions two and four showing the highest performance on the Poker dataset. The K-Nearest Neighbors model demonstrated high accuracy and ROC values across all datasets and conditions. The Logistic Regression model's performance varied significantly depending on the dataset and condition used, while the Random Forest and SVM models showed high accuracy and F1-score values on most datasets, with some conditions showing poor performance.

Furthermore, the study provided a detailed breakdown of the performance evaluation metrics for each dataset and condition, highlighting that both Random Forest and SVM models were effective methods for classification tasks. However, their performance could vary depending on the dataset and the condition used. Overall, the study emphasized the significance of selecting the appropriate ML model and tuning the model parameters to achieve optimal performance on a specific dataset.

Later, six hypothesis testing was employed to evaluate the significance of SMOTE-based approaches on data balancing by comparing ML model performance on original and SMOTE-based oversampled datasets, considering the impact of hyperparameter tuning.

Hypothesis One aimed to evaluate the performance differences of ML models with hyperparameter tuning on original and SMOTE-based oversampled datasets. The results indicated that there was no significant difference in performance for most algorithms and benchmark datasets. However, KNN and LR on Ionosphere and Spambase and KNN on Poker showed a significant difference in performance, suggesting that oversampling affected the performance of particular algorithms on specific datasets.

Hypothesis Two focused on determining the significant difference in ML model performance without hyperparameter tuning on original and SMOTE-based oversampled datasets. Overall, the results indicated no significant difference for all algorithms except for LR on Pageblocks, where a significant difference was observed.

Hypothesis Three investigated the performance difference of ML models with and without hyperparameter tuning on the original dataset. The results indicated no significant difference across all algorithms and datasets, suggesting that hyperparameter tuning does not consistently improve performance in the tested scenarios. However, further research may be needed to determine the generalizability of this conclusion.

Hypothesis Four aimed to investigate the performance difference of ML models with and without hyperparameter tuning on the SMOTE-based oversampled dataset. The findings indicated no significant difference across all algorithms and datasets, suggesting that hyperparameter tuning does not consistently impact the performance on the oversampled dataset.

Hypothesis Five was designed to investigate the performance differences of ML models on the original dataset and the original dataset with hyperparameter tuning. The findings indicate that there is no significant difference in performance for most algorithms and datasets. However, a significant difference was observed for DT on Wine quality, LR on Poker, and SVM on wine quality, suggesting that hyperparameter tuning may impact performance in specific cases.

Similarly, Hypothesis Six aimed to evaluate the performance differences of ML models on the SMOTE-based oversampled dataset and the oversampled dataset with hyperparameter tuning. The results reveal that there is no significant difference in performance for most algorithms and datasets. Nevertheless, a significant difference was found for DT and KNN on Wine quality and LR on Poker, indicating that hyperparameter tuning can influence performance in specific instances on oversampled datasets.

In conclusion, this study aimed to examine the impact of the SMOTE and hyperparameter tuning on the performance of various ML algorithms across different datasets. The results suggest that the application of SMOTE and hyperparameter tuning did not consistently lead to significant performance improvements for all algorithms and datasets. This indicates that these techniques may not always be necessary for achieving better model performance. However, it is important to note that specific algorithms and datasets showed significant performance differences when SMOTE or hyperparameter tuning was applied. Therefore, it is recommended that researchers carefully evaluate the need for these techniques based on the specific context and dataset. Overall, this study provides valuable insights into using SMOTE and hyperparameter tuning effectively in ML algorithm selection and model optimization.

11.2.3 How does imbalanced data affect the performance of Deep Learning (DL)-based models?

In RQ3, the study aimed to investigate the effect of imbalanced data on the performance of deep learning-based models. To achieve this goal, the study considered various types of imbalanced image data, including mixed data such as numerical, categorical, and image data. However, due to time constraints, the study focused on two case scenarios: 1) healthcare diagnostics for patients with COVID-19 symptoms and 2) defect analysis from images of 3D-printed cylinders.

As an effect, the study evaluated six transfer learning (TL) algorithms - VGG16, InceptionResNetV2, ResNet50, ResNet101, MobileNetV2, and VGG19 - on imbalanced data analysis in healthcare diagnostics, specifically in the context of COVID-19 diagnosis using chest X-ray CT-scan images. The findings revealed that the proposed TL-based models, VGG16 and MobileNetV2, consistently performed highly on both imbalanced chest X-ray data and mixed data used in COVID-19 patient diagnosis. However, ResNet50 and ResNet101 were not ideal choices for imbalanced mixed data, although their performance could be improved with additional hyper-tuning. To interpret the models' predictions, explainable AI such as LIME was beneficial, providing the reasoning behind the models' predictions and further cross-checking with actual predictions. Moreover, LIME enabled the identification of the regions that the model learned or focused on. Nevertheless, it is essential to acknowledge that other explainable AI-based model interpretability might provide a more reliable understanding of the black box of DL-based models. Therefore, future research could focus on developing other interpretable models with mixed datasets to validate the overall predictions presented in Chapter 2 and verify the results with healthcare experts.

Later on, the study analyzed three separate studies on COVID-19 patients' diagnoses using chest X-ray images to investigate the impact of imbalanced data on DL-based models' performance. The study found that imbalanced data can indeed affect DL-based models' performance in COVID-19 diagnosis, emphasizing the need to address this issue in diagnostic tool development. The best-performing models (VGG16 and MobileNetV2) achieved high accuracy ranges from 97.6% to almost 100% on a highly imbalanced dataset. However, some models extracted irrelevant details from the images, which could affect their performance.

The same approaches were followed to investigate the impact of DL approaches on imbalanced data analysis in additive manufacturing using six TL-based approaches previously used in Chapter 2. Results indicated that VGG16 and MobileNetV2 achieved perfect scores on all four evaluation metrics, while InceptionResNetV2 performed well, averaging 99% across all metrics. However, ResNet50 had the lowest performance among the models, with an average F1-score of 0.34. The confusion matrix analysis further supported these findings, revealing that the model's classification accuracy varied depending on the specific algorithm and the imbalance level. While some algorithms, such as MobileNetV2, achieved high classification accuracy, others, such as ResNet50, struggled with imbalanced data classification tasks due to higher false positive and false negative rates.

Additionally, the models failed to accurately detect the defect in the image of the 3D-printed cylinder, which is a major limitation of the study. Specifically, VGG16, MobileNetV2, and ResNet50 were unable to locate the defect accurately, despite VGG16 and MobileNetV2 producing bounding boxes that were relatively close to the defect. Similar poor performance was observed for ResNet101, InceptionResNetV2, and VGG19, indicating that these models may not be well-suited for defect localization in 3D-printed objects. The differences in performance may be attributed to variations in the models' architecture and how they handle spatial features.

The findings emphasized the need for further investigation into the factors contributing to model failures and suggested that optimizing model architecture and training strategies could improve localization accuracy and develop more effective solutions for defect detection and classification in additive manufacturing processes.

Although TL-based approaches were able to classify defect and non-defect images of 3D-printed cylinders, they were unable to identify the defect regions accurately, which is necessary to develop a cost-efficient and AI-based additive manufacturing system. This challenge arises due to the different sizes, types, and shapes of the cylinders. Therefore, several preprocessing steps based on CNN approaches were proposed and utilized to reduce the impact of imbalanced data on failure to localize defect regions. Four CNN-based solutions were introduced, including Modified VGG16, ROI Selection (ROIN), ROI Selection with Histogram

Equalization (ROIHEN), and ROI Selection with Histogram Equalization and Details Enhancer (ROIHEDEN). The proposed approaches effectively detected the defect regions in the images.

Furthermore, the model's predictions were interpreted using two explainable AI approaches, LIME and Grad-CAM, which generated heatmaps to help understand the models' behavior during predictions. The application of LIME and Grad-CAM approaches provided valuable insights into the inner workings of the models and contributed to the overall interpretability and explainability of the proposed approaches, ultimately providing a better understanding of the effect of imbalanced data on DL-based model predictions. The explainable AI approaches validated how the proposed model could locate the defect regions and the potential regions on which the proposed model was mainly focused.

However, future studies need to validate the generalizability of the proposed approaches on a larger dataset and explore the potential of the approaches in detecting other types of defects.

11.3 Contributions

5.8 The contributions of this dissertation to the existing literature can be summarized as follows:

- Reviewed various approaches to handle class imbalance problems in ML models, identified key factors affecting their performance, presented a fishbone diagram framework of data balancing approaches and provided a comprehensive overview of their potential applications to improve ML models' performance in various real-world domains.

- Experimented considering four different conditions to determine the effect of traditional ML and SMOTE-based data balancing on imbalanced data analysis using six benchmark datasets and evaluated six hypotheses. Found that the SMOTE-based oversampled dataset did not significantly affect ML model performance for most algorithms and benchmarks, but the performance of some algorithms on specific datasets may be impacted by oversampling.

- Proposed two novel oversampling techniques based on generative adversarial networks (GAN) to address the issue of imbalanced datasets in quality (e.g., Winequality) and defect (e.g., Shuttle) analysis. The GAN-based oversampling (GBO) and SVM-SMOTEGAN (SSG) approaches were shown to have improved classification accuracy, precision, recall, and F1-score compared to the original SMOTE-based approach. Additionally, the GBO and SSG approaches outperformed the original SMOTE-based approach regarding the misclassification rate on the nine benchmark datasets used in the study. The results of this study provide valuable insights into the effectiveness of GAN-based oversampling in enhancing the performance of ML models for quality, defect, and pattern analysis tasks.

- A novel BSGAN model was proposed to address class-imbalanced problems and tested on four highly imbalanced datasets. The proposed approach demonstrated superior performance in various statistical measures compared to borderline-SMOTE and GAN-based oversampling techniques. Furthermore, the neural network approaches using the proposed techniques outperformed many existing recent references. The inter-class distance measurement ensured that the data distribution followed a Gaussian distribution after data expansion using BSGAN. To assess the black box behavior of the proposed models, the study used Local Interpretable Model-Agnostic Explanations (LIME), a valuable tool for model interpretability that provides an understanding of the rationales behind the model's predictions through analysis and visualization of individual feature contributions. The SHapley Additive exPlanations (SHAP) framework was also employed to better understand the model's prediction outcomes on the oversampled dataset.

- TL-based approaches were used on mixed chest X-ray and CT-scan images to diagnose COVID-19 patients. The modified VGG16 and MobileNetV2 models achieved an accuracy of almost 99% on both balanced and imbalanced datasets. The model outputs were validated using explainable AI-based diagnostics and cross-checked with healthcare professionals.

- Investigated the impact of DL-based approaches on small balanced and large imbalanced datasets containing images of the 3D-printed cylinder. The study found that existing popular TL-based approaches have limitations in locating potential areas, such as defect regions in cylinder images. VGG16 and

MobileNetV2 were suggested as promising models for further investigation. The need for additional experiments with larger and different datasets to validate findings was highlighted and plans to conduct experiments with various datasets and propose advanced TL-based approaches to overcome current challenges were made.

- Proposed three CNN-based approaches, namely ROIN, ROIHEN, and ROIHEDEN, integrated with three pre-processing steps, namely ROI, HE, and DE, for detecting defects in 3D printed cylinder images. The proposed approaches addressed the limitation of identifying defect regions highlighted in Chapter Two by introducing additional pre-processing steps and demonstrated effective detection of defect regions in the images. LIME and Grad-CAM were used to generate heatmaps for interpreting the models' behavior during predictions to enhance the interpretability and explainability of the proposed approaches. The study highlights the need for future validation of the proposed approaches on larger datasets and exploration of their potential in detecting other types of defects.

- Proposed an MLP-CNN-based model for the early diagnosis of COVID-19 patients using mixed data, including numerical/categorical data and chest X-ray images. The study showed that the proposed model achieved an accuracy of 96.30% on a small, balanced dataset and outperformed other studies on an imbalanced dataset, achieving an accuracy of 95.4%. The study suggested that the proposed model could be used to develop a screening system to assist healthcare providers in identifying and isolating COVID-19 patients at an earlier stage. Moreover, the study recommended future research to validate the proposed method on larger datasets with complete patient information, work with highly imbalanced data, apply mixed-data analysis using kernel methods, and consider patient geographical information.

References

Acr issues statement for use of chest radiography, ct for suspected covid-19 infection. (n.d.). `https://www.appliedradiology.com/communities/CT-Imaging/acr-issues-statement-for-use-of-chest-radiography-ct-for-suspected-covid-19-infection`.

Afshar, P., Heidarian, S., Naderkhani, F., Oikonomou, A., Plataniotis, K. N., & Mohammadi, A. (2020). Covid-caps: A capsule network-based framework for identification of covid-19 cases from x-ray images. *arXiv preprint arXiv:2004.02696*.

Ahmed, E., & Moustafa, M. (2016). House price estimation from visual and textual features. *ArXiv Preprint ArXiv:1609.08399*.

Ahsan, M. M. (2018). *Real time face recognition in unconstrained environment.* Lamar University-Beaumont.

Ahsan, M. M., Abdullah, T. A., Ali, M. S., Jahora, F., Islam, M. K., Alhashim, A. G., & Gupta, K. D. (2022). Transfer learning and local interpretable model agnostic based visual approach in monkeypox disease detection and classification: A deep learning insights. *arXiv preprint arXiv:2211.05633*.

Ahsan, M. M., Ali, M. S., & Siddique, Z. (2022). Imbalanced class data performance evaluation and improvement using novel generative adversarial network-based approach: Ssg and gbo. *arXiv preprint arXiv:2210.12870*.

Ahsan, M. M., E Alam, T., Trafalis, T., & Huebner, P. (2020). Deep mlp-cnn model using mixed-data to distinguish between covid-19 and non-covid-19 patients. *Symmetry*, *12*(9), 1526.

Ahsan, M. M., Gupta, K. D., Islam, M. M., Sen, S., Rahman, M., Shakhawat Hossain, M., et al. (2020). Covid-19 symptoms detection based on nasnetmobile with explainable ai using various imaging modalities. *Machine Learning and Knowledge Extraction*, *2*(4), 490–504.

Ahsan, M. M., Li, Y., Zhang, J., Ahad, M. T., Yazdan, M. M., et al. (2020). Face recognition in an unconstrained and real-time environment using novel bmc-lbph methods incorporates with dji vision sensor. *Journal of Sensor and Actuator Networks*, *9*(4), 54.

Ahsan, M. M., Luna, S. A., & Siddique, Z. (2022). Machine-learning-based disease diagnosis: A comprehensive review. In *Healthcare* (Vol. 10, p. 541).

Ahsan, M. M., & Siddique, Z. (2021). Machine learning-based heart disease diagnosis: A systematic literature review. *arXiv preprint arXiv:2112.06459*.

Ahsan, M. M., Uddin, M. R., Ali, M. S., Islam, M. K., Farjana, M., Sakib, A. N., ... Luna, S. A. (2023). Deep transfer learning approaches for monkeypox disease diagnosis. *Expert Systems with Applications*, 119483.

Ai, T., Yang, Z., Hou, H., Zhan, C., Chen, C., Lv, W., ... Xia, L. (2020). Correlation of chest ct and rt-pcr testing in coronavirus disease 2019 (covid-19) in china: A report of 1014 cases. *Radiology*, 200642.

Akiba, T., Suzuki, S., & Fukuda, K. (2017). Extremely large minibatch sgd: Training resnet-50 on imagenet in 15 minutes. *arXiv preprint arXiv:1711.04325*.

Alam, M. S., Alam, M. Z., Nazir, K. N. H., & Bhuiyan, M. A. B. (2020). The emergence of novel coronavirus disease (covid-19) in bangladesh: Present status, challenges, and future management. *Journal of Advanced Veterinary and Animal Research*, *7*(2), 198–208.

Ali, U., Shamsi, M. H., Bohacek, M., Purcell, K., Hoare, C., Mangina, E., & O'Donnell, J. (2020). A data-driven approach for multi-scale gis-based building energy modeling for analysis, planning and support decision making. *Applied Energy*, *279*, 115834.

Ali-Gombe, A., & Elyan, E. (2019). Mfc-gan: class-imbalanced dataset classification using multiple fake class generative adversarial network. *Neurocomputing*, *361*, 212–221.

Al Majzoub, H., Elgedawy, I., Akaydın, Ö., & Köse Ulukök, M. (2020). Hcab-smote: A hybrid clustered affinitive borderline smote approach for imbalanced data binary classification. *Arabian Journal for Science and Engineering*, *45*(4), 3205–3222.

Almhaithawi, D., Jafar, A., & Aljnidi, M. (2020). Example-dependent cost-sensitive credit cards fraud detection using smote and bayes minimum risk. *SN Applied Sciences*, *2*(9), 1–12.

Anelić, N., Baressi Šegota, S., Lorencin, I., & Glučina, M. (2022). Detection of malicious websites using symbolic classifier. *Future Internet*, *14*(12), 358.

Apostolopoulos, I. D., & Mpesiana, T. A. (2020). Covid-19: Automatic detection from x-ray images utilizing transfer learning with convolutional neural networks. *Physical and Engineering Sciences in Medicine*, 1.

Ardakani, A. A., Kanafi, A. R., Acharya, U. R., Khadem, N., & Mohammadi, A. (2020). Application of deep learning technique to manage covid-19 in routine clinical practice using ct images: Results of 10 convolutional neural networks. *Computers in Biology and Medicine*, 103795.

Ayyadevara, V. K., & Ayyadevara, V. K. (2018). Gradient boosting machine. *Pro machine learning algorithms: A hands-on approach to implementing algorithms in python and R*, 117–134.

Bacaksız, A. H., & Esgin, E. (2019). Extraction of numerical data from categorical data set and artificial neural networks. In *2019 3rd international symposium on multidisciplinary studies and innovative technologies (ismsit)* (pp. 1–4).

Bai, H. X., Hsieh, B., Xiong, Z., Halsey, K., Choi, J. W., Tran, T. M. L., … others (2020). Performance of radiologists in differentiating covid-19 from viral pneumonia on chest ct. *Radiology*, 200823.

Bansal, A. (2022, May). *Image enhancement techniques using opencv and python.* Towards Data Science. Retrieved from `https://towardsdatascience.com/image-enhancement-techniques-using-opencv-and-python-9191d5c30d45`

Bappy, J. H., Roy-Chowdhury, A. K., Bunk, J., Nataraj, L., & Manjunath, B. (2017). Exploiting spatial structure for localizing manipulated image regions. In *Proceedings of the ieee international conference on computer vision* (pp. 4970–4979).

Barstugan, M., Ozkaya, U., & Ozturk, S. (2020). *Coronavirus (covid-19) classification using ct images by machine learning methods.*

Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM*

SIGKDD explorations newsletter, *6*(1), 20–29.

Bell, D., & Mgbemena, C. (2018). Data-driven agent-based exploration of customer behavior. *Simulation*, *94*(3), 195–212.

Bengio, Y. (2015). Rmsprop and equilibrated adaptive learning rates for nonconvex optimization. *Corr abs/1502.04390*.

Bennett, R., Mulla, Z. D., Parikh, P., Hauspurg, A., & Razzaghi, T. (2022). An imbalance-aware deep neural network for early prediction of preeclampsia. *Plos one*, *17*(4), e0266042.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, *13*(Feb), 281–305.

Bernardo, A., & Della Valle, E. (2022). An extensive study of c-smote, a continuous synthetic minority oversampling technique for evolving data streams. *Expert Systems with Applications*, *196*, 116630.

Bernheim, A., Mei, X., Huang, M., Yang, Y., Fayad, Z. A., Zhang, N., ... others (2020). Chest ct findings in coronavirus disease-19 (covid-19): relationship to duration of infection. *Radiology*, 200463.

Bibri, S. E. (2022). The social shaping of the metaverse as an alternative to the imaginaries of data-driven smart cities: A study in science, technology, and society. *Smart Cities*, *5*(3), 832–874.

Blagus, R., & Lusa, L. (2013). Smote for high-dimensional class-imbalanced data. *BMC bioinformatics*, *14*(1), 1–16.

Blake, C. (1998). Uci repository of machine learning databases. *http://www. ics. uci. edu/˜ mlearn/MLRepository. html*.

Brijain, M., Patel, R., Kushik, M., & Rana, K. (2014). A survey on decision tree algorithm for classification.

Brouwer, R. K. (2002). A feed-forward network for input that is both categorical and quantitative. *Neural Networks*, *15*(7), 881–890.

Brownlee, J. (2014). Machine learning mastery. *URL: http://machinelearningmastery. com/discover-feature-engineering-howtoengineer-features-and-how-to-getgood-at-it*.

Brownlee, J. (2016). Machine learning mastery with python. *Machine Learning Mastery Pty Ltd*, *527*, 100–120.

Brownlee, J. (2018). What is the difference between a batch and an epoch in a neural network? *Machine Learning Mastery*.

Brownlee, J. (2020a). *Data preparation for machine learning: data cleaning, feature selection, and data transforms in python*. Machine Learning Mastery.

Brownlee, J. (2020b). *Imbalanced classification with python: Better metrics, balance skewed classes, cost-sensitive learning*. Machine Learning Mastery.

Brownlee, J. (2020c). Random oversampling and undersampling for imbalanced classification. *Machine learning mastery*.

Brownlee, J. (2020d). Undersampling algorithms for imbalanced classification. *Machine Learning Mastrey*, *27*.

Brownlee, J. (2022). *Machine learning mastery*. Machine Learning Mastery.

Brunese, L., Mercaldo, F., Reginelli, A., & Santone, A. (2020a). An ensemble learning approach for brain cancer detection exploiting radiomic features. *Computer methods and programs in biomedicine*, *185*, 105134.

Brunese, L., Mercaldo, F., Reginelli, A., & Santone, A. (2020b). Explainable deep learning for pulmonary disease and coronavirus covid-19 detection from x-rays. *Computer Methods and Programs in Biomedicine*, 105608.

Brunton, S. L., Nathan Kutz, J., Manohar, K., Aravkin, A. Y., Morgansen, K., Klemisch, J., ... others (2021). Data-driven aerospace engineering: reframing the industry with machine learning. *AIAA Journal*, *59*(8), 2820–2847.

Butt, C., Gill, J., Chun, D., & Babu, B. A. (2020). Deep learning system to screen coronavirus disease 2019 pneumonia. *Applied Intelligence*, 1.

Calderon-Ramirez, S., Yang, S., Moemeni, A., Colreavy-Donnelly, S., Elizondo, D. A., Oala, L., ... Molina-Cabello, M. A. (2021). Improving uncertainty estimation with semi-supervised deep learning for covid-19 detection using chest x-ray images. *Ieee Access*, *9*, 85442–85454.

Calo, V. M., Efendiev, Y., Galvis, J., & Li, G. (2016). Randomized oversampling for generalized multiscale finite element methods. *Multiscale Modeling & Simulation*, *14*(1), 482–501.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1721–1730).

Celik, Y., Talo, M., Yildirim, O., Karabatak, M., & Acharya, U. R. (2020). Automated invasive ductal carcinoma detection based using deep transfer learning with whole-slide images. *Pattern Recognition Letters*.

Chandra, B., & Sharma, R. K. (2016). Deep learning with adaptive learning rate using laplacian score. *Expert Systems with Applications*, *63*, 1–7.

Chandra, T. B., Verma, K., Singh, B. K., Jain, D., & Netam, S. S. (2020). Coronavirus disease (covid-19) detection in chest x-ray images using majority voting based classifier ensemble. *Expert Systems with Applications*, *165*, 113909.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Chawla, N. V., Lazarevic, A., Hall, L. O., & Bowyer, K. W. (2003). Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge discovery in databases: Pkdd 2003: 7th european conference on principles and practice of knowledge discovery in databases, cavtat-dubrovnik, croatia, september 22-26, 2003. proceedings 7* (pp. 107–119).

Chen, F., Mac, G., & Gupta, N. (2017). Security features embedded in computer aided design (cad) solid models for additive manufacturing. *Materials & Design*, *128*, 182–194.

Chen, J., Wu, L., Zhang, J., Zhang, L., Gong, D., Zhao, Y., ... others (2020). Deep learning-based model for detecting 2019 novel coronavirus pneumonia on high-resolution computed tomography: A prospective study. *MedRxiv*.

Chen, Z., Duan, J., Kang, L., & Qiu, G. (2021). Class-imbalanced deep learning via a class-balanced ensemble. *IEEE transactions on neural networks and learning systems*, *33*(10), 5626–5640.

Chest x-ray images (pneumonia). (n.d.). `https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia`.

Chollet, F. (2017). *Deep learning with python.* Manning Publications Co.

Chua, F., Armstrong-James, D., Desai, S. R., Barnett, J., Kouranos, V., Kon, O. M., ... others (2020). The role of ct in case ascertainment and management of covid-19 pneumonia in the uk: insights from high-incidence regions. *The Lancet Respiratory Medicine*, *8*(5), 438–440.

Cohen, J. P., Morrison, P., & Dao, L. (2020). Covid-19 image data collection. *arXiv preprint arXiv:2003.11597*.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, *47*(4), 547–553.

Covid-19 chest xray. (June, 2020). `https://www.kaggle.com/search?q=covid-19+datasetFileTypes%3Apng`.

Covid-19 diagnostic imaging recommendations. (n.d.). `https://www.appliedradiology.com/articles/covid-19-diagnostic-imaging-recommendations`.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, *35*(1), 53–65.

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., & Belongie, S. (2019). Class-balanced loss based on effective number of samples. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 9268–9277).

Cui, Y., Ma, H., & Saha, T. (2014). Improvement of power transformer insulation diagnosis using oil characteristics data preprocessed by smoteboost technique. *IEEE Transactions on Dielectrics and Electrical Insulation*, *21*(5), 2363–2373.

Dadhich, A. (2018). *Practical computer vision: Extract insightful information from images using tensorflow, keras, and opencv.* Packt Publishing Ltd.

Dagliati, A., Marini, S., Sacchi, L., Cogni, G., Teliti, M., Tibollo, V., ... Bellazzi, R. (2018). Machine learning methods to predict diabetes complications. *Journal of diabetes science and technology*, *12*(2), 295–302.

Dashbord. (June,2020). *Covid-19 worldmeter.* (`https://www.worldometers.info/coronavirus/`)

Dayton, C. M. (1992). Logistic regression analysis. *Stat*, *474*, 574.

DebRoy, T., Mukherjee, T., Wei, H., Elmer, J., & Milewski, J. (2021). Metallurgy, mechanistic models and machine learning in metal printing. *Nature Reviews Materials*, *6*(1), 48–68.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., & De Freitas, N. (2013). Predicting parameters in deep learning. In *Advances in neural information processing systems* (pp. 2148–2156).

Devi, D., Biswas, S. K., & Purkayastha, B. (2020). A review on solution to class imbalance problem: undersampling approaches. In *2020 international conference on computational performance evaluation (compe)* (pp. 626–631).

Devi, M. S., Aruna, R., Kumar, S. V., Chowdary, G. V., Raju, B. K., & Prasad, M. S. (2022). Statistical oversampling classification based glass type identification through oxide content. In *Innovations in electrical and electronic engineering: Proceedings of iceee 2022, volume 2* (pp. 537–550). Springer.

Ding, H., Chen, L., Dong, L., Fu, Z., & Cui, X. (2022). Imbalanced data classification: A knn and generative adversarial networks-based hybrid approach for intrusion detection. *Future Generation Computer Systems*, *131*, 240–254.

Dorj, U.-O., Lee, K.-K., Choi, J.-Y., & Lee, M. (2018). The skin cancer classification using deep convolutional neural network. *Multimedia Tools and Applications*, *77*(8), 9909–9924.

Douzas, G., & Bacao, F. (2019). Geometric smote a geometrically enhanced drop-in replacement for smote. *Information Sciences*, *501*, 118–135.

Douzas, G., Bacao, F., & Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, *465*, 1–20.

Drucker, H., Wu, D., & Vapnik, V. N. (1999). Support vector machines for spam categorization. *IEEE Transactions on Neural networks*, *10*(5), 1048–1054.

Dutta, P., Paul, S., & Kumar, A. (2021). Comparative analysis of various supervised machine learning techniques for diagnosis of covid-19. In *Electronic devices, circuits, and systems for biomedical applications* (pp. 521–540). Elsevier.

Elgendi, M., Nasir, M. U., Tang, Q., Fletcher, R. R., Howard, N., Menon, C., ... Nicolaou, S. (2020). The performance of deep neural networks in differentiating chest x-rays of covid-19 patients from other bacterial and viral pneumonias. *Frontiers in medicine*, *7*, 550.

El Naqa, I., & Murphy, M. J. (2015). *What is machine learning?* Springer.

Engelmann, J., & Lessmann, S. (2021). Conditional wasserstein gan-based oversampling of tabular data for imbalanced learning. *Expert Systems with Applications*, *174*, 114582.

Erdem, K. (2020, Feb). *Understanding region of interest - part 2 (roi align).* Retrieved from `https://erdem.pl/2020/02/understanding-region-of-interest-part-2-ro-i-align`

Esteva, A., Chou, K., Yeung, S., Naik, N., Madani, A., Mottaghi, A., … Socher, R. (2021). Deep learning-enabled medical computer vision. *NPJ digital medicine*, *4*(1), 5.

Everything you should know about the 2019 coronavirus and covid-19. (n.d.). `https://www.healthline.com/health/coronavirus-covid-19#symptoms`.

Fahimnia, B., Sarkis, J., & Davarzani, H. (2015). Green supply chain management: A review and bibliometric analysis. *International Journal of Production Economics*, *162*, 101–114.

Fang, Y., Zhang, H., Xie, J., Lin, M., Ying, L., Pang, P., & Ji, W. (2020). Sensitivity of chest ct for covid-19: comparison to rt-pcr. *Radiology*, 200432.

Farahany, Z., Wu, J., Islam, K. S., & Madiraju, P. (2022). Oversampling techniques for predicting covid-19 patient length of stay. In *2022 ieee international conference on big data (big data)* (pp. 5253–5262).

Faris, H. (2014). Neighborhood cleaning rules and particle swarm optimization for predicting customer churn behavior in telecom industry. *International Journal of Advanced Science and Technology*, *68*, 11–22.

Faust, O., Hagiwara, Y., Hong, T. J., Lih, O. S., & Acharya, U. R. (2018). Deep learning for healthcare applications based on physiological signals: A review. *Computer Methods and Programs in Biomedicine*, *161*, 1–13.

Fern, S. H., Amir, A., & Azemi, S. N. (2022). Multi-class imbalanced classification problems in network attack detections. In *Proceedings of the 6th international conference on electrical, control and computer engineering* (pp. 1057–1069).

Filipczuk, P., Fevens, T., Krzyżak, A., & Monczak, R. (2013). Computer-aided breast cancer diagnosis based on the analysis of cytological images of fine needle biopsies. *IEEE transactions on medical imaging*, *32*(12), 2169–2178.

Fletcher, R. R., Nakeshimana, A., & Olubeko, O. (2021). Addressing fairness, bias, and appropriate use of artificial intelligence and machine learning in global health. *Frontiers in Artificial Intelligence*, *3*, 116.

Fonseca, J., Douzas, G., & Bacao, F. (2021). Improving imbalanced land cover classification with k-means smote: Detecting and oversampling distinctive minority spectral signatures. *Information*, *12*(7), 266.

Force, A. D. T., Ranieri, V., Rubenfeld, G., Thompson, B., Ferguson, N., Caldwell, E., et al. (2012). Acute respiratory distress syndrome. *JAMA*, *307*(23), 2526–2533.

Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., & Greenspan, H. (2018). Synthetic data augmentation using gan for improved liver lesion classification. In *2018 ieee 15th international symposium on biomedical imaging (isbi 2018)* (pp. 289–293).

Gabriel, R. A., Harjai, B., Simpson, S., Goldhaber, N., Curran, B. P., & Waterman, R. S. (2022). Machine learning-based models predicting outpatient surgery end time and recovery room discharge at an ambulatory surgery center. *Anesthesia and Analgesia*, *135*(1), 159.

Gabrielli, P., Wüthrich, M., Blume, S., & Sansavini, G. (2022). Data-driven modeling for long-term electricity price forecasting. *Energy*, *244*, 123107.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(4), 463–484.

Ganger, H. (2023). *Hargurjeet/ionosphere-data-analysis - jovian.* Retrieved from `https://jovian.com/hargurjeet/ionosphere-data-analysis`

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information sciences*, *180*(10), 2044–2064.

García-Vicente, C., Chushig-Muzo, D., Mora-Jiménez, I., Fabelo, H., Gram, I. T., Løchen, M.-L., … Soguero-Ruiz, C. (2023). Clinical synthetic data generation to predict and identify risk factors for cardiovascular diseases. In *Heterogeneous data management, polystores, and analytics for healthcare: Vldb workshops, poly 2022 and dmah 2022, virtual event, september 9, 2022, revised selected papers* (pp. 75–91).

Gazzah, S., Hechkel, A., & Amara, N. E. B. (2015). A hybrid sampling method for imbalanced data. In *2015 ieee 12th international multi-conference on systems, signals & devices (ssd15)* (pp. 1–6).

Geng, Y., & Luo, X. (2019). Cost-sensitive convolutional neural networks for imbalanced time series classification. *Intelligent Data Analysis*, *23*(2), 357–370.

Ghoshal, B., & Tucker, A. (2020). Estimating uncertainty and interpretability in deep learning for coronavirus (covid-19) detection. *ArXiv Preprint ArXiv:2003.10769*.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the ieee international conference on computer vision* (pp. 1440–1448).

Gnip, P., Vokorokos, L., & Drotár, P. (2021). Selective oversampling approach for strongly imbalanced data. *PeerJ Computer Science*, *7*, e604.

Gök, E. C., & Olgun, M. O. (2021). Smote-nc and gradient boosting imputation based random forest classifier for predicting severity level of covid-19 patients with blood samples. *Neural Computing and Applications*, *33*(22), 15693–15707.

Gong, J., Ou, J., Qiu, X., Jie, Y., Chen, Y., Yuan, L., … others (2020). A tool to early predict severe 2019-novel coronavirus pneumonia (covid-19): A multicenter study using the risk nomogram in wuhan and guangdong, china. *MedRxiv*.

Goodman, J., Sarkani, S., & Mazzuchi, T. (2022). Distance-based probabilistic data augmentation for synthetic minority oversampling. *ACM/IMS*

Transactions on Data Science (TDS), *2*(4), 1–18.

Gosain, A., & Sardana, S. (2017). Handling class imbalance problem using oversampling techniques: A review. In *2017 international conference on advances in computing, communications and informatics (icacci)* (pp. 79–85).

Goyal, S. (2022). Handling class-imbalance with knn (neighbourhood) undersampling for software defect prediction. *Artificial Intelligence Review*, *55*(3), 2023–2064.

Gozes, O., Frid-Adar, M., Greenspan, H., Browning, P. D., Zhang, H., Ji, W., ... Siegel, E. (2020). *Rapid ai development cycle for the coronavirus (covid-19) pandemic: Initial results for automated detection patient monitoring using deep learning ct image analysis.*

Grasselli, G., Pesenti, A., & Cecconi, M. (2020). Critical care utilization for the covid-19 outbreak in lombardy, italy: Early experience and forecast during an emergency response. *JAMA*, *323*(16), 1545–1546.

Greenspan, H., Van Ginneken, B., & Summers, R. M. (2016). Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, *35*(5), 1153–1159.

Gu, X., Angelov, P. P., & Soares, E. A. (2020). A self-adaptive synthetic oversampling technique for imbalanced classification. *International Journal of Intelligent Systems*, *35*(6), 923–943.

Guo, S., Agarwal, M., Cooper, C., Tian, Q., Gao, R. X., Grace, W. G., & Guo, Y. (2022). Machine learning for metal additive manufacturing: Towards a physics-informed data-driven paradigm. *Journal of Manufacturing Systems*, *62*, 145–163.

Guo, Y., Xiong, G., Li, Z., Shi, J., Cui, M., & Gou, G. (2021). Ta-gan: Gan based traffic augmentation for imbalanced network traffic classification. In *2021 international joint conference on neural networks (ijcnn)* (pp. 1–8).

Gupta, K. D., Ahsan, M., Andrei, S., & Alam, K. M. R. (2017). A robust approach of facial orientation recognition from facial features. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience*, *8*(3), 5–12.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, *3*(Mar), 1157–1182.

Haghanifar, A., Majdabadi, M. M., & Ko, S. (2020). Covid-cxnet: Detecting covid-19 in frontal chest x-ray images using deep learning. *ArXiv Preprint ArXiv:2006.13807*.

Hairani, H., Anggrawan, A., & Priyanto, D. (2023). Improvement performance of the random forest method on unbalanced diabetes data classification using smote-tomek link. *JOIV: International Journal on Informatics Visualization*, *7*(1).

Hameed, A. A., Karlik, B., & Salman, M. S. (2016). Back-propagation algorithm with variable adaptive momentum. *Knowledge-Based Systems*, *114*, 79–87.

Han, H., Wang, W.-Y., & Mao, B.-H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing* (pp. 878–887).

Han, L., Yang, G., Yang, X., Song, X., Xu, B., Li, Z., . . . Wu, J. (2022). An explainable xgboost model improved by smote-enn technique for maize lodging detection based on multi-source unmanned aerial vehicle images. *Computers and Electronics in Agriculture*, *194*, 106804.

Hangge, P., Pershad, Y., Witting, A. A., Albadawi, H., & Oklu, R. (2018). Three-dimensional (3d) printing and its applications for aortic diseases. *Cardiovascular diagnosis and therapy*, *8*(Suppl 1), S19.

Hart, P. (1968). The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, *14*(3), 515–516.

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 ieee international joint conference on neural networks (ieee world congress on computational intelligence)* (pp. 1322–1328).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Health system tracker. (n.d.). http://www.nhf.org.bd/hospital_charge.php?id=6.

Hemdan, E. E.-D., Shouman, M. A., & Karar, M. E. (2020). Covidx-net: A framework of deep learning classifiers to diagnose covid-19 in x-ray images. *arXiv preprint arXiv:2003.11055*.

Hira, Z. M., & Gillies, D. F. (2015). A review of feature selection and feature extraction methods applied on microarray data. *Advances in bioinformatics*, *2015*.

Hoens, T. R., & Chawla, N. V. (2013). Imbalanced datasets: from sampling to classifiers. *Imbalanced learning: Foundations, algorithms, and applications*, 43–59.

Hopkins, M., Reeber, E., Forman, G., & Suermondt, J. (1998). *Spambase data set.* Retrieved from `https://archive.ics.uci.edu/ml/datasets/spambase`

How much does an x-ray cost. (n.d.). `https://health.costhelper.com/x-rays.html`.

Hu, Z., Song, C., Xu, C., Jin, G., Chen, Y., Xu, X., ... others (2020). Clinical characteristics of 24 asymptomatic infections with covid-19 screened among close contacts in nanjing, china. *Science China Life Sciences*, 1–6.

Hutter, F., Lücke, J., & Schmidt-Thieme, L. (2015). Beyond manual tuning of hyperparameters. *KI-Künstliche Intelligenz*, *29*(4), 329–337.

Huyut, M. (2023). Automatic detection of severely and mildly infected covid-19 patients with supervised machine learning models. *IRBM*, *44*(1), 100725.

Ikotun, A. M., & Ezugwu, A. E. (2022). Enhanced firefly-k-means clustering with adaptive mutation and central limit theorem for automatic clustering of high-dimensional datasets. *Applied Sciences*, *12*(23), 12275.

Ileberi, E., Sun, Y., & Wang, Z. (2021). Performance evaluation of machine learning methods for credit card fraud detection using smote and adaboost. *IEEE Access*, *9*, 165286–165294.

Ishaq, A., Sadiq, S., Umer, M., Ullah, S., Mirjalili, S., Rupapara, V., & Nappi, M. (2021). Improving the prediction of heart failure patients' survival using smote and effective data mining techniques. *IEEE access*, *9*, 39707–39716.

Jadhav, A. S. (2020). A novel weighted tpr-tnr measure to assess performance of the classifiers. *Expert systems with applications*, *152*, 113391.

Jakhar, D., & Kaur, I. (2020). Artificial intelligence, machine learning and deep learning: Definitions and differences. *Clinical and Experimental Dermatology*, *45*(1), 131–132.

Jalali, A., Heistracher, C., Schindler, A., Haslhofer, B., Nemeth, T., Glawar, R., … De Boer, P. (2019). Predicting time-to-failure of plasma etching equipment using machine learning. In *2019 ieee international conference on prognostics and health management (icphm)* (pp. 1–8).

Jamil, F., Kahng, H. K., Kim, S., & Kim, D.-H. (2021). Towards secure fitness framework based on iot-enabled blockchain network integrated with machine learning algorithms. *Sensors*, *21*(5), 1640.

Jang, B., Kim, M., Harerimana, G., Kang, S.-u., & Kim, J. W. (2020). Bi-lstm model to increase accuracy in text classification: Combining word2vec cnn and attention mechanism. *Applied Sciences*, *10*(17), 5841.

Jia, C., & Zuo, Y. (2017). S-sulfpred: A sensitive predictor to capture s-sulfenylation sites based on a resampling one-sided selection undersampling-synthetic minority oversampling technique. *Journal of theoretical biology*, *422*, 84–89.

Jiang, N., & Li, N. (2021). A wind turbine frequent principal fault detection and localization approach with imbalanced data using an improved synthetic oversampling technique. *International Journal of Electrical Power & Energy Systems*, *126*, 106595.

Jiang, W., Hong, Y., Zhou, B., He, X., & Cheng, C. (2019). A gan-based anomaly detection approach for imbalanced industrial time series. *IEEE Access*, *7*, 143608–143619.

Jin, C., Chen, W., Cao, Y., Xu, Z., Zhang, X., Deng, L., ... Feng, J. (2020). Development and evaluation of an ai system for covid-19 diagnosis. *MedRxiv*.

Jin, Z., Zhang, Z., Demir, K., & Gu, G. X. (2020). Machine learning for advanced additive manufacturing. *Matter*, *3*(5), 1541–1556.

Jo, W., & Kim, D. (2022). Obgan: Minority oversampling near borderline with generative adversarial networks. *Expert Systems with Applications*, *197*, 116694.

Joshi, N. (2022, Mar). *Histogram equalization.* Retrieved from `https://www.analyticsvidhya.com/blog/2022/01/histogram-equalization/`

Junsomboon, N., & Phienthrakul, T. (2017). Combining over-sampling and under-sampling techniques for imbalance dataset. In *Proceedings of the 9th international conference on machine learning and computing* (pp. 243–247).

Kabir, M. F., & Ludwig, S. (2018). Classification of breast cancer risk factors using several resampling approaches. In *2018 17th ieee international conference on machine learning and applications (icmla)* (pp. 1243–1248).

Kanne, J. P., Little, B. P., Chung, J. H., Elicker, B. M., & Ketai, L. H. (2020). *Essentials for radiologists on covid-19: An update—radiology scientific expert panel.* Radiological Society of North America.

Karia, V., Zhang, W., Naeim, A., & Ramezani, R. (2019). Gensample: A genetic algorithm for oversampling in imbalanced datasets. *arXiv preprint* *arXiv:1910.10806*.

Karr, A. F., Sanil, A. P., & Banks, D. L. (2006). Data quality: A statistical perspective. *Statistical Methodology*, *3*(2), 137–173.

Kassani, S. H., & Kassani, P. H. (2019). A comparative study of deep learning architectures on melanoma detection. *Tissue and Cell*, *58*, 76–83.

Kaur, H., Pannu, H. S., & Malhi, A. K. (2019). A systematic review on imbalanced data challenges in machine learning: Applications and solutions. *ACM Computing Surveys (CSUR)*, *52*(4), 1–36.

Ker, J., Wang, L., Rao, J., & Lim, T. (2017). Deep learning applications in medical image analysis. *IEEE Access*, *6*, 9375–9389.

Ketkar, N., & Ketkar, N. (2017). Stochastic gradient descent. *Deep learning with Python: A hands-on introduction*, 113–132.

Khalifa, N. E. M., Taha, M. H. N., Hassanien, A. E., & Elghamrawy, S. (2020). Detection of coronavirus (covid-19) associated pneumonia based on generative adversarial networks and a fine-tuned deep transfer learning model using chest x-ray dataset. *arXiv preprint arXiv:2004.01184*.

khan, A., Gupta, K. D., Kumar, N., & Venugopal, D. (2020). Cidmp: Completely interpretable detection of malaria parasite in red blood cells using lower-dimensional feature space. In *Proceedings of the 2020 international joint conference on neural networks (ijcnn 2020).*

Khan, A. I., Shah, J. L., & Bhat, M. M. (2020). Coronet: A deep neural network for detection and diagnosis of covid-19 from chest x-ray images. *Computer Methods and Programs in Biomedicine*, 105581.

Kim, J., Jeong, K., Choi, H., & Seo, K. (2020). Gan-based anomaly detection in imbalance problems. In *European conference on computer vision* (pp. 128–145).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*.

Koch, D., Despotovic, M., Leiber, S., Sakeena, M., Döller, M., & Zeppelzauer, M. (2020). Real estate image analysis-a literature review (preprint).

Kramer, O., & Kramer, O. (2013). K-nearest neighbors. *Dimensionality reduction with unsupervised nearest neighbors*, 13–23.

Krawczyk, B., Galar, M., Jeleń, Ł., & Herrera, F. (2016). Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy. *Applied Soft Computing*, *38*, 714–726.

Kumar, E. S., Talasila, V., Rishe, N., Kumar, T. S., & Iyengar, S. (2019). Location identification for real estate investment using data analytics. *International Journal of Data Science and Analytics*, *8*(3), 299–323.

Kumar, N., Hashmi, A., Gupta, M., & Kundu, A. (2022). Automatic diagnosis of covid-19 related pneumonia from cxr and ct-scan images. *Engineering, Technology & Applied Science Research*, *12*(1), 7993–7997.

Kumar, V., Lalotra, G. S., Sasikala, P., Rajput, D. S., Kaluri, R., Lakshmanna, K., … Uddin, M. (2022). Addressing binary classification over class imbalanced clinical datasets using computationally intelligent techniques. In *Healthcare* (Vol. 10, p. 1293).

Kwon, C., Park, S., Ko, S., & Ahn, J. (2021). Increasing prediction accuracy of pathogenic staging by sample augmentation with a gan. *Plos one*, *16*(4), e0250458.

Lango, M., & Stefanowski, J. (2022). What makes multi-class imbalanced problems difficult? an experimental study. *Expert Systems with Applications*, *199*, 116962.

Last, F., Douzas, G., & Bacao, F. (n.d.). Oversampling for imbalanced learning based on k-means and smote. arxiv 2017. *arXiv preprint arXiv:1711.00837*.

Lau, A. L., Chi, I., Cummins, R. A., Lee, T. M., Chou, K.-L., & Chung, L. W. (2008). The sars (severe acute respiratory syndrome) pandemic in hong kong: Effects on the subjective wellbeing of elderly and younger people. *Aging and Mental Health*, *12*(6), 746–760.

Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. In *Artificial intelligence in medicine: 8th conference on artificial intelligence in medicine in europe, aime 2001 cascais, portugal, july 1–4, 2001, proceedings 8* (pp. 63–66).

Law, S., Paige, B., & Russell, C. (2019). Take a look around: Using street view and satellite images to estimate house prices. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *10*(5), 1–19.

Le, T., Hoang Son, L., Vo, M. T., Lee, M. Y., & Baik, S. W. (2018). A cluster-based boosting algorithm for bankruptcy prediction in a highly imbalanced dataset. *Symmetry*, *10*(7), 250.

Lee, K.-S., Kim, J. Y., Jeon, E.-t., Choi, W. S., Kim, N. H., & Lee, K. Y. (2020). Evaluation of scalability and degree of fine-tuning of deep convolutional neural networks for covid-19 screening on chest x-ray images using explainable deep-learning algorithm. *Journal of Personalized Medicine*, *10*(4), 213.

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, *18*(1), 559–563.

Lenka, S. R., Bisoy, S. K., Priyadarshini, R., & Nayak, B. (2022). Representative-based cluster undersampling technique for imbalanced credit scoring datasets. In *Innovations in computational intelligence and computer vision: Proceedings of icicv 2021* (pp. 119–129). Springer.

Li, K., Ren, B., Guan, T., Wang, J., Yu, J., Wang, K., & Huang, J. (2022). A hybrid cluster-borderline smote method for imbalanced data of rock groutability classification. *Bulletin of Engineering Geology and the Environment*, *81*(1), 1–15.

Li, L., Qin, L., Xu, Z., Yin, Y., Wang, X., Kong, B., ... others (2020a). Artificial intelligence distinguishes covid-19 from community acquired pneumonia on chest ct. *Radiology*, 200905.

Li, L., Qin, L., Xu, Z., Yin, Y., Wang, X., Kong, B., ... others (2020b). Using artificial intelligence to detect covid-19 and community-acquired pneumonia based on pulmonary ct: evaluation of the diagnostic accuracy. *Radiology*, *296*(2), E65–E71.

Li, Y., & Xia, L. (2020). Coronavirus disease 2019 (covid-19): role of chest ct in diagnosis and management. *American Journal of Roentgenology*, 1–7.

Liang, T., et al. (2020). Handbook of covid-19 prevention and treatment. *The*

First Affiliated Hospital, Zhejiang University School of Medicine. Compiled According to Clinical Experience.

Liang, X., Jiang, A., Li, T., Xue, Y., & Wang, G. (2020). Lr-smote—an improved unbalanced data set oversampling based on k-means and svm. *Knowledge-Based Systems*, *196*, 105845.

Lin, C., Tsai, C.-F., & Lin, W.-C. (2023). Towards hybrid over-and under-sampling combination methods for class imbalanced datasets: An experimental study. *Artificial Intelligence Review*, *56*(2), 845–863.

Lin, W.-C., Tsai, C.-F., Hu, Y.-H., & Jhang, J.-S. (2017). Clustering-based undersampling in class-imbalanced data. *Information Sciences*, *409*, 17–26.

Lin, W.-J., & Chen, J. J. (2013). Class-imbalanced classifiers for high-dimensional data. *Briefings in bioinformatics*, *14*(1), 13–26.

Lin, Z., Khetan, A., Fanti, G., & Oh, S. (2018). Pacgan: The power of two samples in generative adversarial networks. *Advances in neural information processing systems*, *31*.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., … Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, *42*, 60–88.

Liu, C.-L., & Chang, Y.-H. (2022). Learning from imbalanced data with deep density hybrid sampling. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *52*(11), 7065–7077.

Liu, J., Gaynor, A. T., Chen, S., Kang, Z., Suresh, K., Takezawa, A., … others (2018). Current and future trends in topology optimization for additive manufacturing. *Structural and multidisciplinary optimization*, *57*, 2457–2483.

Liu, X.-Y., Wu, J., & Zhou, Z.-H. (2008). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(2), 539–550.

Longadge, R., & Dongre, S. (2013). Class imbalance problem in data mining review. *arXiv preprint arXiv:1305.1707*.

Lu, C., Lin, S., Liu, X., & Shi, H. (2020). Telecom fraud identification based on adasyn and random forest. In *2020 5th international conference on computer and communication systems (icccs)* (pp. 447–452).

Lu, Y., Wu, S., Tai, Y.-W., & Tang, C.-K. (2018). Image generation from sketch constraint using contextual gan. In *Proceedings of the european conference on computer vision (eccv)* (pp. 205–220).

Mahesh, T., Dhilip Kumar, V., Vinoth Kumar, V., Asghar, J., Geman, O., Arulkumaran, G., & Arun, N. (2022). Adaboost ensemble methods using k-fold cross validation for survivability with the early detection of heart disease. *Computational Intelligence and Neuroscience*, *2022*.

Majid, A., Ali, S., Iqbal, M., & Kausar, N. (2014). Prediction of human breast and colon cancers from imbalanced data using nearest neighbor and support vector machines. *Computer Methods and Programs in Biomedicine*, *113*(3), 792–808.

Maldonado, S., Vairetti, C., Fernandez, A., & Herrera, F. (2022). Fw-smote: A feature-weighted oversampling approach for imbalanced classification.

Pattern Recognition, *124*, 108511.

Malerba, D., Esposito, F., & Semeraro, G. (1996). A further comparison of simplification methods for decision-tree induction. *Learning From Data: Artificial Intelligence and Statistics V*, 365–374.

Malviya, R. K., & Kant, R. (2015). Green supply chain management (gscm): a structured literature review and research implications. *Benchmarking: An international journal*.

Manju, B., & Nair, A. R. (2019). Classification of cardiac arrhythmia of 12 lead ecg using combination of smoteenn, xgboost and machine learning algorithms. In *2019 9th international symposium on embedded computing and system design (ised)* (pp. 1–7).

Manzano, C., Meneses, C., Leger, P., & Fukuda, H. (2022). An empirical evaluation of supervised learning methods for network malware identification based on feature selection. *Complexity*, *2022*.

Marina, L. A., Trasnea, B., & Grigorescu, S. M. (2018). A multi-platform framework for artificial intelligence engines in automotive systems. In *2018 22nd international conference on system theory, control and computing (icstcc)* (pp. 559–564).

Mayabadi, S., & Saadatfar, H. (2022). Two density-based sampling approaches for imbalanced and overlapping data. *Knowledge-Based Systems*, *241*, 108217.

Meng, C., Zhou, L., & Liu, B. (2020). A case study in credit fraud detection with smote and xgboost. In *Journal of physics: Conference series* (Vol. 1601, p. 052016).

Meng, L., Hua, F., & Bian, Z. (2020). Coronavirus disease 2019 (covid-19): Emerging and future challenges for dental and oral medicine. *Journal of Dental Research*, *99*(5), 481–487.

Menzies, T., Greenwald, J., & Frank, A. (2006). Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, *33*(1), 2–13.

Minaee, S., Kafieh, R., Sonka, M., Yazdani, S., & Soufi, G. J. (2020). Deep-covid: Predicting covid-19 from chest x-ray images using deep transfer learning. *arXiv preprint arXiv:2004.09363*. Retrieved from `https://arxiv.org/abs/2004.09363`

Mohamad, M., Selamat, A., Subroto, I. M., & Krejcar, O. (2021). Improving the classification performance on imbalanced data sets via new hybrid parameterisation model. *Journal of King Saud University-Computer and Information Sciences*, *33*(7), 787–797.

Mohammed, A. J., Hassan, M. M., & Kadir, D. H. (2020). Improving classification performance for a novel imbalanced medical dataset using smote method. *International Journal*, *9*(3), 3161–3172.

Mohammed, M., Mwambi, H., Mboya, I. B., Elbashir, M. K., & Omolo, B. (2021). A stacking ensemble deep learning approach to cancer type classification based on tcga data. *Scientific reports*, *11*(1), 1–22.

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, *7*, 1419.

Moher, D., Shamseer, L., Clarke, M., Ghersi, D., Liberati, A., Petticrew, M.,

… Stewart, L. A. (2015). Preferred reporting items for systematic review and meta-analysis protocols (prisma-p) 2015 statement. *Systematic reviews*, *4*(1), 1–9.

Mohiuddin, A. K. (2020). Covid-19 situation in bangladesh.

Moreo, A., Esuli, A., & Sebastiani, F. (2016). Distributional random oversampling for imbalanced text classification. In *Proceedings of the 39th international acm sigir conference on research and development in information retrieval* (pp. 805–808).

Mostafa, S. M., Salem, S. A., & Habashy, S. M. (2022). Predictive model for accident severity. *IAENG International Journal of Computer Science*, *49*(1).

Mousavian Anaraki, S. A., Haeri, A., & Moslehi, F. (2022). Generating balanced and strong clusters based on balance-constrained clustering approach (strong balance-constrained clustering) for improving ensemble classifier performance. *Neural Computing and Applications*, *34*(23), 21139–21155.

Mqadi, N., Naicker, N., & Adeliyi, T. (2021). A smote based oversampling data-point approach to solving the credit card data imbalance problem in financial fraud detection. *International Journal of Computing and Digital Systems*, *10*(1), 277–286.

Mqadi, N. M., Naicker, N., & Adeliyi, T. (2021). Solving misclassification of the credit card imbalance problem using near miss. *Mathematical Problems in Engineering*, *2021*, 1–16.

Mukherjee, A., Mukhopadhyay, S., Panigrahi, P. K., & Goswami, S. (2019).

Utilization of oversampling for multiclass sentiment analysis on amazon review dataset. In *2019 ieee 10th international conference on awareness science and technology (icast)* (pp. 1–6).

Mukherjee, M., & Khushi, M. (2021). Smote-enc: A novel smote-based method to generate synthetic data for nominal and continuous features. *Applied System Innovation*, *4*(1), 18.

Murat, F., Yildirim, O., Talo, M., Baloglu, U. B., Demir, Y., & Acharya, U. R. (2020). Application of deep learning techniques for heartbeats detection using ecg signals-analysis and review. *Computers in Biology and Medicine*, 103726.

Mustafa, A. B. (2019). Enhancing learning from imbalanced classes via data preprocessing: A data-driven application in metabolomics data mining. *ISeCure*, *11*(3).

Nagra, A. A., Mubarik, I., Asif, M. M., Masood, K., Ghamdi, M. A. A., & Almotiri, S. H. (2022). Hybrid ga-svm approach for postoperative life expectancy prediction in lung cancer patients. *Applied Sciences*, *12*(21), 10927.

Nakada, R., & Imaizumi, M. (2019). Adaptive approximation and estimation of deep neural network to intrinsic dimensionality. *ArXiv Preprint ArXiv:1907.02177*.

Narin, A., Kaya, C., & Pamuk, Z. (2020). Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. *ArXiv Preprint ArXiv:2003.10849*.

National heart foundation of bangladesh. (n.d.). `http://www.nhf.org.bd/`

hospital_charge.php?id=6.

Ndichu, S., Ban, T., Takahashi, T., & Inoue, D. (2023). Ai-assisted security alert data analysis with imbalanced learning methods. *Applied Sciences*, *13*(3), 1977.

The new coronavirus appears to take a greater toll on men than on women. (n.d.). https://www.npr.org/sections/goatsandsoda/2020/04/10/831883664/the-new-coronavirus-appears-to-take-a-greater-toll-on-men-than-on-women.

Ning, Q., Zhao, X., & Ma, Z. (2021). A novel method for identification of glutarylation sites combining borderline-smote with tomek links technique in imbalanced data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

Nugroho, B., & Yuniarti, A. (2022). Performance of root-mean-square propagation and adaptive gradient optimization algorithms on covid-19 pneumonia classification. In *2022 ieee 8th information technology international seminar (itis)* (pp. 333–338).

Obiedat, R., Qaddoura, R., Ala'M, A.-Z., Al-Qaisi, L., Harfoushi, O., Alrefai, M., & Faris, H. (2022). Sentiment analysis of customers' reviews using a hybrid evolutionary svm-based approach in an imbalanced data distribution. *IEEE Access*, *10*, 22260–22273.

Okoli, C., & Schabram, K. (2010). A guide to conducting a systematic literature review of information systems research.

Ozturk, T., Talo, M., Yildirim, E. A., Baloglu, U. B., Yildirim, O., & Acharya,

U. R. (2020). Automated detection of covid-19 cases using deep neural networks with x-ray images. *Computers in Biology and Medicine*, 103792.

Paleyes, A., Urma, R.-G., & Lawrence, N. D. (2022). Challenges in deploying machine learning: a survey of case studies. *ACM Computing Surveys*, *55*(6), 1–29.

Pan, L., Mu, M., Yang, P., Sun, Y., Wang, R., Yan, J., … others (2020). Clinical characteristics of covid-19 patients with digestive symptoms in hubei, china: a descriptive, cross-sectional, multicenter study. *The American journal of gastroenterology*, *115*.

Pandey, S. K., & Janghel, R. R. (2019). Automatic detection of arrhythmia from imbalanced ecg database using cnn model with smote. *Australasian physical & engineering sciences in medicine*, *42*(4), 1129–1139.

Pandit, M. K., Banday, S. A., Naaz, R., & Chishti, M. A. (2020). Automatic detection of covid-19 from chest radiographs using deep learning. *Radiography*.

Pawlicki, M., Choraś, M., Kozik, R., & Hołubowicz, W. (2020). On the impact of network data balancing in cybersecurity applications. In *Computational science–iccs 2020: 20th international conference, amsterdam, the netherlands, june 3–5, 2020, proceedings, part iv 20* (pp. 196–210).

Pecoraro, V., Negro, A., Pirotti, T., & Trenti, T. (2022). Estimate false-negative rt-pcr rates for sars-cov-2. a systematic review and meta-analysis. *European journal of clinical investigation*, *52*(2), e13706.

Pereira, R. M., Costa, Y. M., & Silla Jr, C. N. (2020). Mltl: A multi-label approach for the tomek link undersampling algorithm. *Neurocomputing*, *383*, 95–105.

Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

Podder, P., Bharati, S., Mondal, M. R. H., & Kose, U. (2021). Application of machine learning for the diagnosis of covid-19. In *Data science for covid-19* (pp. 175–194). Elsevier.

Polat, H., Özerdem, M. S., Ekici, F., & Akpolat, V. (2021). Automatic detection and localization of covid-19 pneumonia using axial computed tomography images and deep convolutional neural networks. *International Journal of Imaging Systems and Technology*, *31*(2), 509–524.

Ponta, L., Puliga, G., Oneto, L., & Manzini, R. (2020). Identifying the determinants of innovation capability with machine learning and patents. *IEEE Transactions on Engineering Management*.

Prudêncio, R. B. (2020). Cost sensitive evaluation of instance hardness in machine learning. In *Machine learning and knowledge discovery in databases: European conference, ecml pkdd 2019, würzburg, germany, september 16–20, 2019, proceedings, part ii* (pp. 86–102).

Pucci, J. U., Christophe, B. R., Sisti, J. A., & Connolly Jr, E. S. (2017). Three-dimensional printing: technologies, applications, and limitations in neurosurgery. *Biotechnology advances*, *35*(5), 521–529.

Puri, A., & Kumar Gupta, M. (2022). Improved hybrid bag-boost ensemble with k-means-smote–enn technique for handling noisy class imbalanced data. *The*

Computer Journal, *65*(1), 124–138.

Qaddoura, R., Al-Zoubi, A., Almomani, I., & Faris, H. (2021). A multi-stage classification approach for iot intrusion detection based on clustering with oversampling. *Applied Sciences*, *11*(7), 3022.

Qadrini, L. (2022). Handling unbalanced data with smote adaboost. *Jurnal Mantik*, *6*(2), 2332–2336.

Qian, W., & Li, S. (2020). A novel class imbalance-robust network for bearing fault diagnosis utilizing raw vibration signals. *Measurement*, *156*, 107567.

Qing, Z., Zeng, Q., Wang, H., Liu, Y., Xiong, T., & Zhang, S. (2022). Adasyn-lof algorithm for imbalanced tornado samples. *Atmosphere*, *13*(4), 544.

Qolomany, B., Maabreh, M., Al-Fuqaha, A., Gupta, A., & Benhaddou, D. (2017). Parameters optimization of deep learning models using particle swarm optimization. In *2017 13th international wireless communications and mobile computing conference (iwcmc)* (pp. 1285–1290).

Rachna, C. (2020). Difference between x-ray and ct scan. *Rahman, T., Chowdhury, M., & Khandakar, A.(2020). COVID-19 Radiography Database. Kaggle. Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., & Shpanskaya, K.(2017). Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. ArXiv Preprint ArXiv*, *1711*, 2352–2449.

Radziukas, R., Maskeliūnas, R., & Damaševičius, R. (2019). Prediction of poker moves using sequential model and tensorflow. In *Information and software technologies: 25th international conference, icist 2019, vilnius, lithuania,* *october 10–12, 2019, proceedings 25* (pp. 516–525).

Rahman, M. J., & Zhu, H. (2023). Predicting accounting fraud using imbalanced ensemble learning classifiers–evidence from china. *Accounting & Finance*.

Rai, H. M., Chatterjee, K., Dubey, A., & Srivastava, P. (2021). Myocardial infarction detection using deep learning and ensemble technique from ecg signals. In *Proceedings of second international conference on computing, communications, and cyber-security: Ic4s 2020* (pp. 717–730).

Rajabi, A., & Garibay, O. O. (2022). Tabfairgan: Fair tabular data generation with generative adversarial networks. *Machine Learning and Knowledge Extraction*, *4*(2), 488–501.

Rajesh, K. N., & Dhuli, R. (2018). Classification of imbalanced ecg beats using re-sampling techniques and adaboost ensemble classifier. *Biomedical Signal Processing and Control*, *41*, 242–254.

Rajkumar, K. V., et al. (2022). Development of hybrid methods for rainfall classification using ant colony optimization and neural networks.

Rana, C., Chitre, N., Poyekar, B., & Bide, P. (2021). Stroke prediction using smote-tomek and neural network. In *2021 12th international conference on computing communication and networking technologies (icccnt)* (pp. 1–5).

Rath, A., Mishra, D., Panda, G., & Satapathy, S. C. (2021). Heart disease detection using deep learning methods from imbalanced ecg samples. *Biomedical Signal Processing and Control*, *68*, 102820.

Reddy, S., Seshadri, S. B., Bothra, G. S., Suhas, T., & Thundiyil, S. C. (2020). Detection of arrhythmia in real-time using ecg signal analysis and convolutional neural networks. In *2020 ieee 21st international conference on computational problems of electrical engineering (cpee)* (pp. 1–4).

Ribli, D., Horváth, A., Unger, Z., Pollner, P., & Csabai, I. (2018). Detecting and classifying lesions in mammograms with deep learning. *Scientific Reports*, *8*(1), 1–7.

Rivera, W. A., & Xanthopoulos, P. (2016). A priori synthetic over-sampling methods for increasing classification sensitivity in imbalanced data sets. *Expert Systems with Applications*, *66*, 124–135.

Rizvi, A. S., Murtaza, G., Yan, D., Irfan, M., Xue, M., Meng, Z. H., & Qu, F. (2020). Development of molecularly imprinted 2d photonic crystal hydrogel sensor for detection of l-kynurenine in human serum. *Talanta*, *208*, 120403.

Roosa, K., Lee, Y., Luo, R., Kirpich, A., Rothenberg, R., Hyman, J., ... Chowell, G. (2020). Real-time forecasts of the covid-19 epidemic in china from february 5th to february 24th, 2020. *Infectious Disease Modelling*, *5*, 256–263.

Rosolia, A., & Osterrieder, J. (2021). Analyzing deep generated financial time series for various asset classes. *Available at SSRN 3898792*.

Rothan, H. A., & Byrareddy, S. N. (2020). The epidemiology and pathogenesis of coronavirus disease (covid-19) outbreak. *Journal of autoimmunity*, 102433.

Roy, K., Ahmad, M., Waqar, K., Priyaah, K., Nebhen, J., Alshamrani, S. S., ... Ali, I. (2021). An enhanced machine learning framework for type 2 diabetes classification using imbalanced data with missing values. *Complexity*, *2021*, 1–21.

Rubio-Solis, A., Panoutsos, G., Beltran-Perez, C., & Martinez-Hernandez, U.

(2020). A multilayer interval type-2 fuzzy extreme learning machine for the recognition of walking activities and gait events using wearable sensors. *Neurocomputing*, *389*, 42–55.

Saba, T., Mohamed, A. S., El-Affendi, M., Amin, J., & Sharif, M. (2020). Brain tumor detection using fusion of hand crafted and deep learning features. *Cognitive Systems Research*, *59*, 221–230.

Sadgali, I., Nawal, S., & Benabbou, F. (2019). Fraud detection in credit card transaction using machine learning techniques. In *2019 1st international conference on smart systems and data science (icssd)* (pp. 1–4).

Sahu, A., Harshvardhan, G., & Gourisaria, M. K. (2020). A dual approach for credit card fraud detection using neural network and data mining techniques. In *2020 ieee 17th india council international conference (indicon)* (pp. 1–7).

Saini, A. (2022, Aug). *An introduction to random forest algorithm for beginners.* Retrieved from `https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/`

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4510–4520).

Santosh, K. (2020). Ai-driven tools for coronavirus outbreak: need of active learning and cross-population train/test models on multitudinal/multimodal data. *Journal of medical systems*, *44*, 1–5.

Sawangarreerak, S., & Thanathamathee, P. (2020). Random forest with sampling techniques for handling imbalanced prediction of university student depression. *Information*, *11*(11), 519.

Schapire, R. E. (2013). Explaining adaboost. In *Empirical inference* (pp. 37–52). Springer.

Schmidt, C. O., Struckmann, S., Enzenbach, C., Reineke, A., Stausberg, J., Damerow, S., … Richter, A. (2021). Facilitating harmonized data quality assessments. a data quality framework for observational health research data collections with software implementations in r. *BMC Medical Research Methodology*, *21*(1), 1–15.

Sekeroglu, B., & Ozsahin, I. (2020). ¡? covid19?¿ detection of covid-19 from chest x-ray images using convolutional neural networks. *SLAS TECHNOLOGY: Translating Life Sciences Innovation*, 2472630320958376.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the ieee international conference on computer vision* (pp. 618–626).

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., & Batra, D. (2016). Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*.

Sethy, P. K., & Behera, S. K. (2020). Detection of coronavirus disease (covid-19) based on deep features. *Preprints*, *2020030300*, 2020.

Sevastyanov, L. A., & Shchetinin, E. Y. (2020). On methods for improving the accuracy of multi-class classification on imbalanced data. In *Ittmm* (pp. 70–82).

Shafi, O., Rai, C., Sen, R., & Ananthanarayanan, G. (2021). Demystifying tensorrt: Characterizing neural network inference engine on nvidia edge devices. In *2021 ieee international symposium on workload characterization (iiswc)* (pp. 226–237).

Shahriare Satu, M., Atik, S. T., & Moni, M. A. (2020). A novel hybrid machine learning model to predict diabetes mellitus. In *Proceedings of international joint conference on computational intelligence: Ijcci 2019* (pp. 453–465).

Shamsudin, H., Yusof, U. K., Jayalakshmi, A., & Khalid, M. N. A. (2020). Combining oversampling and undersampling techniques for imbalanced classification: A comparative study using credit card fraudulent transaction dataset. In *2020 ieee 16th international conference on control & automation (icca)* (pp. 803–808).

Shankar, S., Garcia, R., Hellerstein, J. M., & Parameswaran, A. G. (2022). Operationalizing machine learning: An interview study. *arXiv preprint arXiv:2209.09125*.

Sharma, A., Singh, P. K., & Chandra, R. (2022). Smotified-gan for class imbalanced pattern classification problems. *IEEE Access*.

Sharma, H., & Gosain, A. (2023). Oversampling methods to handle the class imbalance problem: A review. In *Soft computing and its engineering applications: 4th international conference, icsoftcomp 2022, changa, anand, india, december 9–10, 2022, proceedings* (pp. 96–110).

Shelke, A., Inamdar, M., Shah, V., Tiwari, A., Hussain, A., Chafekar, T., & Mehendale, N. (2020). Chest x-ray classification using deep learning for automated covid-19 screening. *medRxiv*. Retrieved from `https://`

www.medrxiv.org/content/10.1101/2020.06.21.20136598v1

Shen, D., Wu, G., & Suk, H.-I. (2017). Deep learning in medical image analysis. *Annual Review of Biomedical Engineering*, *19*, 221–248.

Shen, F., Zhao, X., Kou, G., & Alsaadi, F. E. (2021). A new deep learning ensemble credit risk evaluation model with an improved synthetic minority oversampling technique. *Applied Soft Computing*, *98*, 106852.

Shen, J., Wu, J., Xu, M., Gan, D., An, B., & Liu, F. (2021). A hybrid method to predict postoperative survival of lung cancer using improved smote and adaptive svm. *Computational and mathematical methods in medicine*, *2021*.

Shen, M., Zhou, Y., Ye, J., Al-Maskri, A. A. A., Kang, Y., Zeng, S., & Cai, S. (2020). Recent advances and perspectives of nucleic acid detection for coronavirus. *Journal of pharmaceutical analysis*, *10*(2), 97–101.

Shi, F., Xia, L., Shan, F., Wu, D., Wei, Y., Yuan, H., ... Shen, D. (2020). Large-scale screening of covid-19 from community acquired pneumonia using infection size-aware classification. *ArXiv Preprint ArXiv:2003.09860*.

Shi, H., Zhang, Y., Chen, Y., Ji, S., & Dong, Y. (2022). Resampling algorithms based on sample concatenation for imbalance learning. *Knowledge-Based Systems*, 108592.

Shim, W., Luo, Y., Seo, J.-S., & Yu, S. (2020). Investigation of read disturb and bipolar read scheme on multilevel rram-based deep learning inference engine. *IEEE Transactions on Electron Devices*, *67*(6), 2318–2323.

Shon, H. S., Batbaatar, E., Kim, K. O., Cha, E. J., & Kim, K.-A. (2020). Classification of kidney cancer data using cost-sensitive hybrid deep learning approach. *Symmetry*, *12*(1), 154.

Shrinidhi, M., Kaushik Jegannathan, T., & Jeya, R. (2023). Classification of imbalanced datasets using various techniques along with variants of smote oversampling and ann. *Advances in Science and Technology*, *124*, 504–511.

Siddappa, N. G., & Kampalappa, T. (2019). Adaptive condensed nearest neighbor for imbalance data classification. *International Journal of Intelligent Engineering and Systems*, *12*(2), 104–113.

Sigillito, V. G., Wing, S. P., Hutton, L. V., & Baker, K. B. (1989). Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, *10*(3), 262–266.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sisodia, D. S., Reddy, N. K., & Bhandari, S. (2017). Performance evaluation of class balancing techniques for credit card fraud detection. In *2017 ieee international conference on power, control, signals and instrumentation engineering (icpcsi)* (pp. 2747–2752).

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*.

Smith, S. L., Kindermans, P.-J., Ying, C., & Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

Smiti, S., & Soui, M. (2020). Bankruptcy prediction using deep learning approach based on borderline smote. *Information Systems Frontiers*, *22*, 1067–1083.

Song, Y., & Peng, Y. (2019). A mcdm-based evaluation approach for imbalanced classification methods in financial risk prediction. *IEEE Access*, *7*, 84897–84906.

Song, Y., Zheng, S., Li, L., Zhang, X., Zhang, X., Huang, Z., ... others (2020). Deep learning enables accurate diagnosis of novel coronavirus (covid-19) with ct images. *MedRxiv*.

Srivastava, S., Divekar, A. V., Anilkumar, C., Naik, I., Kulkarni, V., & Pattabiraman, V. (2021). Comparative analysis of deep learning image detection algorithms. *Journal of Big Data*, *8*(1), 1–27.

Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A., & Chan, P. K. (2000). Cost-based modeling for fraud and intrusion detection: Results from the jam project. In *Proceedings darpa information survivability conference and exposition. discex'00* (Vol. 2, pp. 130–144).

Sun, J., Li, H., Fujita, H., Fu, B., & Ai, W. (2020). Class-imbalanced dynamic financial distress prediction based on adaboost-svm ensemble combined with smote and time weighting. *Information Fusion*, *54*, 128–144.

Sun, Y., Que, H., Cai, Q., Zhao, J., Li, J., Kong, Z., & Wang, S. (2022). Borderline smote algorithm and feature selection-based network anomalies detection strategy. *Energies*, *15*(13), 4751.

Sun, Z., Zhang, J., Sun, H., & Zhu, X. (2020). Collaborative filtering-based recommendation of sampling methods for software defect prediction. *Applied Soft Computing*, *90*, 106163.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning* (pp. 1139–1147).

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first aaai conference on artificial intelligence.*

Tahamtan, A., & Ardebili, A. (2020). *Real-time rt-pcr in covid-19 detection: issues affecting the results.* Taylor & Francis.

Taneja, S., Suri, B., & Kothari, C. (2019). Application of balancing techniques with ensemble approach for credit card fraud detection. In *2019 international conference on computing, power and communication technologies (gucon)* (pp. 753–758).

Tang, Y. (2013). Deep learning using linear support vector machines. *ArXiv Preprint ArXiv:1306.0239*.

Tao, X., Li, Q., Guo, W., Ren, C., He, Q., Liu, R., & Zou, J. (2020). Adaptive weighted over-sampling for imbalanced datasets based on density peaks clustering with heuristic filtering. *Information Sciences*, *519*, 43–73.

Tao, X., Li, Q., Guo, W., Ren, C., Li, C., Liu, R., & Zou, J. (2019). Self-adaptive cost weights-based support vector machine cost-sensitive ensemble for imbalanced data classification. *Information Sciences*, *487*, 31–56.

Tesfahun, A., & Bhaskari, D. L. (2013). Intrusion detection using random forests classifier with smote and feature reduction. In *2013 international conference on cloud & ubiquitous computing & emerging technologies* (pp. 127–132).

Thabtah, F., Hammoud, S., Kamalov, F., & Gonsalves, A. (2020). Data imbalance in classification: Experimental evaluation. *Information Sciences*, *513*, 429–441.

Tricco, A. C., Lillie, E., Zarin, W., O'Brien, K. K., Colquhoun, H., Levac, D., ... others (2018). Prisma extension for scoping reviews (prisma-scr): checklist and explanation. *Annals of internal medicine*, *169*(7), 467–473.

Triguero, I., García-Gil, D., Maillo, J., Luengo, J., García, S., & Herrera, F. (2019). Transforming big data into smart data: An insight on the use of the k-nearest neighbors algorithm to obtain quality data. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *9*(2), e1289.

Trinks, S. (2021). Real time quality assurance and defect detection in industry 4.0. In *Lwda* (pp. 258–267).

Uyun, S., & Sulistyowati, E. (2020). Feature selection for multiple water quality status: integrated bootstrapping and smote approach in imbalance classes. *International Journal of Electrical and Computer Engineering*, *10*(4), 4331.

Velmurugan, T., & Santhanam, T. (2010). Computational complexity between k-means and k-medoids clustering algorithms for normal and uniform distributions of data points. *Journal of computer science*, *6*(3), 363.

Wahab, N., Khan, A., & Lee, Y. S. (2017). Two-phase deep convolutional neural network for reducing class skewness in histopathological images based breast cancer detection. *Computers in biology and medicine*, *85*, 86–97.

Wang, F., Zou, Y., Zhang, H., & Shi, H. (2019). House price prediction approach based on deep learning and arima model. In *2019 ieee 7th international conference on computer science and network technology (iccsnt)* (pp. 303–

307).

Wang, L. (2005). *Support vector machines: theory and applications* (Vol. 177). Springer Science & Business Media.

Wang, L., Wang, H., & Fu, G. (2020). Multiple kernel learning with minority oversampling for classifying imbalanced data. *IEEE Access*, *9*, 565–580.

Wang, L., & Wong, A. (2020). Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *ArXiv Preprint ArXiv:2003.09871*.

Wang, S., Dai, Y., Shen, J., & Xuan, J. (2021). Research on expansion and classification of imbalanced data based on smote algorithm. *Scientific reports*, *11*(1), 1–11.

Wang, S., Kang, B., Ma, J., Zeng, X., Xiao, M., Guo, J., … others (2021). A deep learning algorithm using ct images to screen for corona virus disease (covid-19). *European radiology*, *31*, 6096–6104.

Wang, S., & Yao, X. (2013). Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, *62*(2), 434–443.

Wang, X., Deng, X., Fu, Q., Zhou, Q., Feng, J., Ma, H., … Zheng, C. (2020). A weakly-supervised framework for covid-19 classification and lesion localization from chest ct. *IEEE transactions on medical imaging*, *39*(8), 2615–2625.

Wang, X., Liu, J., Qiu, T., Mu, C., Chen, C., & Zhou, P. (2020). A real-time collision prediction mechanism with deep learning for intelligent transportation system. *IEEE transactions on vehicular technology*, *69*(9), 9497–9508.

Wang, Y., Kung, L., & Byrd, T. A. (2018). Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations. *Technological forecasting and social change*, *126*, 3–13.

Watson, H. J. (2019). Update tutorial: Big data analytics: Concepts, technology, and applications. *Communications of the Association for Information Systems*, *44*(1), 21.

Weron, R. (2014). Electricity price forecasting: A review of the state-of-the-art with a look into the future. *International Journal of Forecasting*, *30*(4), 1030–1081.

Westphal, E., & Seitz, H. (2022). Machine learning for the intelligent analysis of 3d printing conditions using environmental sensor data to support quality assurance. *Additive Manufacturing*, *50*, 102535.

Wibowo, P., & Fatichah, C. (2021). An in-depth performance analysis of the oversampling techniques for high-class imbalanced dataset. *Register: Jurnal Ilmiah Teknologi Sistem Informasi*, *7*(1), 63–71.

Wu, B., Pan, Z., Ding, D., Cuiuri, D., Li, H., Xu, J., & Norrish, J. (2018). A review of the wire arc additive manufacturing of metals: properties, defects and quality improvement. *Journal of Manufacturing Processes*, *35*, 127–139.

Wu, J., Zhao, Z., Sun, C., Yan, R., & Chen, X. (2021). Learning from class-imbalanced data with a model-agnostic framework for machine intelligent diagnosis. *Reliability Engineering & System Safety*, *216*, 107934.

Wu, Y., & Cao, D. (2021). A diagnostic algorithm diagnosing the failure of railway signal equipment. *Diagnostyka*, *22*(4), 33–38.

Wulf, J., & Blohm, I. (2020). Fostering value creation with digital platforms: A unified theory of the application programming interface design. *Journal of Management Information Systems*, *37*(1), 251–281.

Xiao, Y., Wu, J., & Lin, Z. (2021). Cancer diagnosis using generative adversarial networks based on deep learning from imbalanced data. *Computers in Biology and Medicine*, *135*, 104540.

Xiaolong, X., Wen, C., & Yanfei, S. (2019). Over-sampling algorithm for imbalanced data classification. *Journal of Systems Engineering and Electronics*, *30*(6), 1182–1191.

Xin, L. K., et al. (2021). Prediction of depression among women using random oversampling and random forest. In *2021 international conference of women in data science at taif university (widstaif)* (pp. 1–5).

Xu, X., Jiang, X., Ma, C., Du, P., Li, X., Lv, S., … others (2020). A deep learning system to screen novel coronavirus disease 2019 pneumonia. *Engineering*.

Xu, Z., Shi, L., Wang, Y., Zhang, J., Huang, L., Zhang, C., … others (2020). Pathological findings of covid-19 associated with acute respiratory distress syndrome. *The Lancet Respiratory Medicine*, *8*(4), 420–422.

Xue, Y., & Li, Y. (2018). A fast detection method via region-based fully convolutional neural networks for shield tunnel lining defects. *Computer-Aided Civil and Infrastructure Engineering*, *33*(8), 638–654.

Yan, L., Zhang, H.-T., Xiao, Y., Wang, M., Sun, C., Liang, J., … others (2020). Prediction of criticality in patients with severe covid-19 infection using three clinical features: A machine learning-based prognostic model with clinical data in wuhan. *MedRxiv*.

Yang, F., Wang, K., Sun, L., Zhai, M., Song, J., & Wang, H. (2022). A hybrid sampling algorithm combining synthetic minority over-sampling technique and edited nearest neighbor for missed abortion diagnosis. *BMC Medical Informatics and Decision Making*, *22*(1), 344.

Yang, H., & Li, M. (2022). Software defect prediction based on smote-tomek and xgboost. In *Bio-inspired computing: Theories and applications: 16th international conference, bic-ta 2021, taiyuan, china, december 17–19, 2021, revised selected papers, part ii* (pp. 12–31).

Yang, Y., & Xu, Z. (2020). Rethinking the value of labels for improving class-imbalanced learning. *Advances in Neural Information Processing Systems*, *33*, 19290–19301.

Yildirim, O., Talo, M., Ay, B., Baloglu, U. B., Aydin, G., & Acharya, U. R. (2019). Automated detection of diabetic subject using pre-trained 2d-cnn models with frequency spectrum images extracted from heart rate signals. *Computers in Biology and Medicine*, *113*, 103387.

Yinka-Banjo, C., & Ugot, O.-A. (2020). A review of generative adversarial networks and its application in cybersecurity. *Artificial Intelligence Review*, *53*, 1721–1736.

Yuen, K.-S., Ye, Z.-W., Fung, S.-Y., Chan, C.-P., & Jin, D.-Y. (2020). Sars-cov-2 and covid-19: The most important research questions. *Cell & Bioscience*, *10*(1), 1–5.

Yurochkin, M., Bower, A., & Sun, Y. (2019). Training individually fair ml models with sensitive subspace robustness. *arXiv preprint arXiv:1907.00020*.

Zeiser, A., Özcan, B., Kracke, C., van Stein, B., & Bäck, T. (2023). A data-centric approach to anomaly detection in layer-based additive manufacturing. *at-Automatisierungstechnik*, *71*(1), 81–89.

Zeng, M., Zou, B., Wei, F., Liu, X., & Wang, L. (2016). Effective prediction of three common diseases by combining smote with tomek links technique for imbalanced medical data. In *2016 ieee international conference of online analysis and computing science (icoacs)* (pp. 225–228).

Zhang, A., Yu, H., Huan, Z., Yang, X., Zheng, S., & Gao, S. (2022). Smote-rknn: A hybrid re-sampling method based on smote and reverse k-nearest neighbors. *Information Sciences*, *595*, 70–88.

Zhang, C., Liao, Q., Rakhlin, A., Miranda, B., Golowich, N., & Poggio, T. (2018). Theory of deep learning iib: Optimization properties of sgd. *ArXiv Preprint ArXiv:1801.02254*.

Zhang, J., Xie, Y., Li, Y., Shen, C., & Xia, Y. (2020). Covid-19 screening on chest x-ray images using deep learning based anomaly detection. *ArXiv Preprint ArXiv:2003.12338*.

Zhang, J., Xie, Y., Pang, G., Liao, Z., Verjans, J., Li, W., ... others (2020). Viral pneumonia screening on chest x-rays using confidence-aware anomaly detection. *IEEE transactions on medical imaging*, *40*(3), 879–890.

Zhang, J., Zhang, Y., & Li, K. (2020). A network intrusion detection model based on the combination of relieff and borderline-smote. In *Proceedings of the 2020 4th high performance computing and cluster technologies conference &*

2020 3rd international conference on big data and artificial intelligence (pp. 199–203).

Zhang, W., Xia, E., Dai, R., Tang, W., Bin, Y., & Xia, J. (2022). Predapp: predicting anti-parasitic peptides with undersampling and ensemble approaches. *Interdisciplinary Sciences: Computational Life Sciences*, 1–11.

Zhang, X., Ran, J., & Mi, J. (2019). An intrusion detection system based on convolutional neural network for imbalanced network traffic. In *2019 ieee 7th international conference on computer science and network technology (iccsnt)* (pp. 456–460).

Zhang, Z., Krawczyk, B., Garcia, S., Rosales-Pérez, A., & Herrera, F. (2016). Empowering one-vs-one decomposition with ensemble learning for multi-class imbalanced data. *Knowledge-Based Systems*, *106*, 251–263.

Zhao, W., Zhong, Z., Xie, X., Yu, Q., & Liu, J. (2020). Relation between chest ct findings and clinical conditions of coronavirus disease (covid-19) pneumonia: A multicenter study. *American Journal of Roentgenology*, *214*(5), 1072–1077.

Zheng, X. (2020). *Smote variants for imbalanced binary classification: heart disease prediction*. University of California, Los Angeles.

Zhou, L., Pan, S., Wang, J., & Vasilakos, A. V. (2017). Machine learning on big data: Opportunities and challenges. *Neurocomputing*, *237*, 350–361.

Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object detection in 20 years: A survey. *Proceedings of the IEEE*.

Zuech, R., Hancock, J., & Khoshgoftaar, T. M. (2021). Detecting web attacks using random undersampling and ensemble learners. *Journal of Big Data*, *8*(1), 1–20.

Appendices

.1 Appendix A

.1.1 Performance of Adaboost

Table A1: Adaboost ML model performance evaluation on Pageblocks dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.979	1	1	1	0.968	0.9894
	Precision	1	1	1	1	0.667	0.9334
	Recall	0.667	0.6	1	1	1	0.8534
	F1-score	0.8	0.571	1	1	0.8	0.8342
	ROC	1	1	1	1	0.996	0.9992
	G-mean	0.816	1	1	1	0.983	0.9598
Two	Accuracy	1	1	1	1	0.983	0.9966
	Precision	1	1	1	1	0.967	0.9934
	Recall	1	1	1	1	1	1
	F1-score	1	1	1	1	0.983	0.9966
	ROC	1	1	1	1	0.994	0.9988
	G-mean	1	1	1	1	0.983	0.9966

Table A1: *Cont.*

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Three	Accuracy	0.979	0.979	1	1	0.968	0.9852
	Precision	1	1	1	1	0.667	0.9334
	Recall	0.667	0.6	1	1	1	0.8534
	F1-score	0.8	0.75	1	1	0.8	0.87
	ROC	1	1	1	1	0.996	0.9992
	G-mean	0.816	1	1	1	0.983	0.9598
Four	Accuracy	1	1	1	1	0.983	0.9966
	Precision	1	1	1	1	0.967	0.9934
	Recall	1	1	1	1	1	1
	F1-score	1	1	1	1	0.983	0.9966
	ROC	1	1	1	1	0.994	0.9988
	G-mean	1	1	1	1	0.983	0.9966

Table A2: Adaboost ML model performance evaluation on Poker dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.986	0.99	0.99	0.99	0.983	0.9878
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.667	0.45	0.63	0.658	0.446	0.5702
	G-mean	0	0	0	0	0	0
Two	Accuracy	0.964	0.976	0.978	0.971	0.964	0.9706
	Precision	0.944	0.963	0.963	0.951	0.939	0.952
	Recall	0.986	0.99	0.993	0.993	0.993	0.991
	F1-score	0.965	0.976	0.978	0.971	0.965	0.971
	ROC	0.991	0.992	0.997	0.994	0.993	0.9934
	G-mean	0.964	0.976	0.978	0.971	0.963	0.9704
Three	Accuracy	0.986	0.99	0.99	0.99	0.986	0.9884
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.551	0.247	0.433	0.501	0.366	0.4196

Table A2: *Cont.*

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0	0	0	0	0	0
Four	Accuracy	0.748	0.733	0.772	0.768	0.744	0.753
	Precision	0.68	0.663	0.7	0.7	0.678	0.6842
	Recall	0.938	0.945	0.952	0.938	0.932	0.941
	F1-score	0.788	0.78	0.807	0.802	0.785	0.7924
	ROC	0.759	0.742	0.78	0.779	0.754	0.7628
	G-mean	0.724	0.701	0.751	0.75	0.72	0.7292

Table A3: Adaboost ML model performance evaluation on Spambase dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.93	0.941	0.946	0.948	0.929	0.9388
	Precision	0.919	0.923	0.943	0.934	0.923	0.9284
	Recall	0.904	0.928	0.917	0.934	0.895	0.9156
	F1-score	0.911	0.926	0.93	0.934	0.909	0.922
	ROC	0.972	0.983	0.981	0.985	0.969	0.978
	G-mean	0.925	0.939	0.94	0.945	0.923	0.9344

Table A3 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Two	Accuracy	0.939	0.943	0.938	0.941	0.933	0.9388
	Precision	0.934	0.959	0.944	0.93	0.93	0.9394
	Recall	0.944	0.925	0.932	0.953	0.935	0.9378
	F1-score	0.939	0.942	0.938	0.941	0.933	0.9386
	ROC	0.981	0.986	0.984	0.981	0.982	0.9828
	G-mean	0.939	0.942	0.938	0.941	0.933	0.9386
Three	Accuracy	0.938	0.951	0.947	0.953	0.934	0.9446
	Precision	0.93	0.93	0.943	0.939	0.929	0.9342
	Recall	0.912	0.948	0.92	0.942	0.901	0.9246
	F1-score	0.921	0.939	0.931	0.941	0.914	0.9292
	ROC	0.975	0.984	0.982	0.986	0.97	0.9794
	G-mean	0.933	0.95	0.942	0.951	0.927	0.9406
Four	Accuracy	0.945	0.949	0.941	0.947	0.936	0.9436
	Precision	0.934	0.956	0.936	0.935	0.932	0.9386
	Recall	0.959	0.941	0.946	0.961	0.941	0.9496
	F1-score	0.946	0.949	0.941	0.948	0.937	0.9442
	ROC	0.983	0.987	0.982	0.983	0.984	0.9838
	G-mean	0.945	0.949	0.941	0.947	0.936	0.9436

Table A4: Adaboost ML model performance evaluation on Winequality dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.962	0.962	0.947	0.969	0.977	0.9634
	Precision	0	0	0.2	0.5	1	0.34
	Recall	0	0	0.25	0.5	0.25	0.2
	F1-score	0	0	0.222	0.5	0.4	0.2244
	ROC	0.906	0.805	0.766	0.984	0.707	0.8336
	G-mean	0	0	0.492	0.702	0.5	0.3388
Two	Accuracy	0.969	0.953	0.965	0.976	0.984	0.9694
	Precision	0.954	0.939	0.941	0.969	0.977	0.956
	Recall	0.984	0.969	0.992	0.984	0.992	0.9842
	F1-score	0.969	0.953	0.966	0.977	0.984	0.9698
	ROC	0.985	0.971	0.981	0.986	0.999	0.9844
	G-mean	0.969	0.953	0.964	0.976	0.984	0.9692
Three	Accuracy	0.977	0.977	0.969	0.969	0.969	0.9722
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.625	0.84	0.522	0.925	0.646	0.7116

Table A4 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0	0	0	0	0	0
Four	Accuracy	0.847	0.875	0.875	0.839	0.87	0.8612
	Precision	0.814	0.823	0.838	0.813	0.836	0.8248
	Recall	0.898	0.953	0.93	0.883	0.921	0.917
	F1-score	0.854	0.883	0.881	0.846	0.876	0.868
	ROC	0.873	0.894	0.894	0.86	0.89	0.8822
	G-mean	0.846	0.871	0.873	0.838	0.869	0.8594

Table A5: Adaboost ML model performance evaluation on Yeast dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.961	0.932	0.981	0.922	0.951	0.9494
	Precision	0.75	0.667	0.909	0.667	0.778	0.7542
	Recall	0.9	0.6	0.909	0.4	0.7	0.7018
	F1-score	0.818	0.632	0.909	0.5	0.737	0.7192
	ROC	0.982	0.886	0.994	0.796	0.837	0.899
	G-mean	0.933	0.762	0.948	0.626	0.828	0.8194

Table A5 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Two	Accuracy	0.973	0.968	0.978	0.984	0.967	0.974
	Precision	0.978	0.939	0.989	0.989	0.957	0.9704
	Recall	0.967	1	0.968	0.978	0.978	0.9782
	F1-score	0.973	0.968	0.978	0.984	0.968	0.9742
	ROC	0.97	0.986	0.996	0.999	0.996	0.9894
	G-mean	0.973	0.967	0.978	0.984	0.967	0.9738
Three	Accuracy	0.971	0.932	0.981	0.971	0.951	0.9612
	Precision	0.818	0.667	0.909	1	0.778	0.8344
	Recall	0.9	0.6	0.909	0.7	0.7	0.7618
	F1-score	0.857	0.632	0.909	0.824	0.737	0.7918
	ROC	0.998	0.965	0.994	0.982	0.953	0.9784
	G-mean	0.938	0.762	0.948	0.837	0.828	0.8626
Four	Accuracy	0.957	0.957	0.968	0.968	0.957	0.9614
	Precision	0.967	0.929	0.978	0.958	0.929	0.9522
	Recall	0.946	0.989	0.957	0.978	0.989	0.9718
	F1-score	0.956	0.958	0.967	0.968	0.958	0.9614
	ROC	0.991	0.987	0.994	0.998	0.992	0.9924
	G-mean	0.957	0.956	0.968	0.967	0.956	0.9608

Performance of Decision Tree

Table A6: Decision Tree ML model performance evaluation on Ionosphere dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.845	0.871	0.9	0.914	0.886	0.8832
	Precision	0.881	0.891	0.846	0.977	0.953	0.9096
	Recall	0.867	0.889	0.978	0.911	0.911	0.9112
	F1-score	0.899	0.879	0.928	0.93	0.933	0.9138
	ROC	0.845	0.847	0.849	0.936	0.927	0.8808
	G-mean	0.826	0.833	0.884	0.947	0.906	0.8792
Two	Accuracy	0.867	0.867	0.956	0.889	0.878	0.8914
	Precision	0.878	0.909	0.846	0.833	0.927	0.8786
	Recall	0.8	0.889	0.978	0.911	0.822	0.88
	F1-score	0.864	0.889	0.945	0.882	0.897	0.8954
	ROC	0.856	0.9	0.933	0.878	0.889	0.8912
	G-mean	0.833	0.9	0.955	0.866	0.911	0.893
Three	Accuracy	0.831	0.929	0.857	0.814	0.914	0.869
	Precision	0.848	0.891	0.857	0.913	0.957	0.8932
	Recall	0.844	0.889	0.956	0.889	1	0.9156
	F1-score	0.897	0.913	0.863	0.925	0.978	0.9152

Table A6 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.803	0.882	0.847	0.929	0.925	0.8772
	G-mean	0.847	0.894	0.885	0.822	0.935	0.8766
Four	Accuracy	0.756	0.911	0.933	0.933	0.867	0.88
	Precision	0.87	0.809	0.863	0.8	0.917	0.8518
	Recall	0.8	0.867	0.889	0.889	0.911	0.8712
	F1-score	0.882	0.899	0.891	0.879	0.909	0.892
	ROC	0.754	0.975	0.974	0.942	0.954	0.9198
	G-mean	0.842	0.922	0.889	0.886	0.877	0.8832

Table A7: Decision Tree ML model performance evaluation on Poker dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.983	0.98	0.986	0.986	0.983	0.9836
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.5	0.493	0.497	0.497	0.497	0.4968
	G-mean	0	0	0	0	0	0

Table A7 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Two	Accuracy	0.985	0.993	0.995	0.991	0.985	0.9898
	Precision	0.99	1	0.993	0.986	0.99	0.9918
	Recall	0.979	0.993	0.997	0.993	0.986	0.9896
	F1-score	0.988	0.995	0.995	0.991	0.986	0.991
	ROC	0.981	0.997	0.995	0.993	0.986	0.9904
	G-mean	0.983	0.997	0.998	0.991	0.988	0.9914
Three	Accuracy	0.98	0.986	0.99	0.983	0.983	0.9844
	Precision	0	0	0	0	0.571	0.1142
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0.5	1	0.3
	ROC	0.495	0.493	0.497	0.49	0.5	0.495
	G-mean	0	0	0	0	0	0
Four	Accuracy	0.943	0.959	0.962	0.962	0.942	0.9536
	Precision	0.831	0.976	0.93	0.948	0.915	0.92
	Recall	0.979	0.993	0.993	0.99	0.993	0.9896
	F1-score	0.96	0.96	0.959	0.968	0.953	0.96
	ROC	0.967	0.971	0.986	0.979	0.964	0.9734
	G-mean	0.949	0.972	0.955	0.988	0.942	0.9612

Table A8: Decision Tree ML model performance evaluation on Spambase dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.892	0.904	0.923	0.908	0.905	0.9064
	Precision	0.879	0.867	0.905	0.849	0.893	0.8786
	Recall	0.857	0.89	0.895	0.928	0.856	0.8852
	F1-score	0.864	0.887	0.902	0.888	0.887	0.8856
	ROC	0.89	0.9	0.921	0.917	0.889	0.9034
	G-mean	0.89	0.904	0.916	0.903	0.899	0.9024
Two	Accuracy	0.937	0.935	0.932	0.942	0.925	0.9342
	Precision	0.934	0.94	0.928	0.932	0.902	0.9272
	Recall	0.952	0.927	0.941	0.961	0.937	0.9436
	F1-score	0.938	0.929	0.936	0.944	0.921	0.9336
	ROC	0.93	0.935	0.929	0.939	0.92	0.9306
	G-mean	0.937	0.933	0.935	0.943	0.917	0.933
Three	Accuracy	0.923	0.912	0.913	0.922	0.907	0.9154
	Precision	0.863	0.889	0.882	0.892	0.892	0.8836
	Recall	0.873	0.882	0.878	0.881	0.865	0.8758
	F1-score	0.862	0.88	0.894	0.889	0.875	0.88
	ROC	0.904	0.887	0.924	0.905	0.908	0.9056

Table A8 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.906	0.895	0.899	0.908	0.896	0.9008
Four	Accuracy	0.925	0.931	0.925	0.923	0.906	0.922
	Precision	0.921	0.927	0.932	0.922	0.929	0.9262
	Recall	0.921	0.901	0.909	0.93	0.937	0.9196
	F1-score	0.925	0.933	0.925	0.921	0.936	0.928
	ROC	0.927	0.932	0.92	0.932	0.938	0.9298
	G-mean	0.927	0.931	0.913	0.926	0.93	0.9254

Table A9: Decision Tree ML model performance evaluation on Winequality dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.962	0.939	0.954	0.977	0.939	0.9542
	Precision	0.333	0	0.333	0	0.167	0.1666
	Recall	0	0	0.25	0	0.25	0.1
	F1-score	0	0	0.444	0.4	0.222	0.2132
	ROC	0.484	0.48	0.605	0.625	0.726	0.584
	G-mean	0.571	0	0.693	0	0.69	0.3908

Table A9 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Two	Accuracy	0.984	0.949	0.976	0.98	0.98	0.9738
	Precision	0.992	0.945	0.955	0.969	0.977	0.9676
	Recall	0.992	0.953	0.992	0.984	1	0.9842
	F1-score	0.992	0.949	0.973	0.973	0.981	0.9736
	ROC	0.992	0.953	0.972	0.973	0.984	0.9748
	G-mean	0.992	0.953	0.968	0.972	0.976	0.9722
Three	Accuracy	0.969	0.947	0.969	0.962	0.969	0.9632
	Precision	0	0.2	0	0	0	0.04
	Recall	0	0	0.25	0.5	0.5	0.25
	F1-score	0	0.4	0.25	0.571	0	0.2442
	ROC	0.484	0.484	0.612	0.616	0.606	0.5604
	G-mean	0	0	0.496	0	0.702	0.2396
Four	Accuracy	0.988	0.949	0.969	0.969	0.965	0.968
	Precision	0.962	0.969	0.962	0.977	0.977	0.9694
	Recall	0.984	0.953	0.992	0.969	0.969	0.9734
	F1-score	0.973	0.957	0.973	0.977	0.965	0.969
	ROC	0.988	0.953	0.992	0.988	0.961	0.9764
	G-mean	0.976	0.965	0.968	0.957	0.976	0.9684

Table A10: Decision Tree ML model performance evaluation on Yeast dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.961	0.932	0.971	0.931	0.941	0.9472
	Precision	0.75	0.636	0.833	0.75	0.667	0.7272
	Recall	0.9	0.7	0.818	0.6	0.8	0.7636
	F1-score	0.818	0.667	0.87	0.667	0.667	0.7378
	ROC	0.934	0.828	0.944	0.789	0.878	0.8746
	G-mean	0.928	0.818	0.895	0.762	0.875	0.8556
Two	Accuracy	0.951	0.978	0.973	0.968	0.94	0.962
	Precision	0.937	0.968	0.978	0.968	0.956	0.9614
	Recall	0.967	0.989	0.989	0.978	0.957	0.976
	F1-score	0.952	0.978	0.978	0.973	0.95	0.9662
	ROC	0.951	0.978	0.978	0.973	0.951	0.9662
	G-mean	0.951	0.978	0.989	0.973	0.962	0.9706
Three	Accuracy	0.961	0.932	0.961	0.941	0.941	0.9472
	Precision	0.818	0.625	0.889	0.667	0.583	0.7164
	Recall	0.9	0.7	0.909	0.5	0.6	0.7218
	F1-score	0.818	0.6	0.783	0.471	0.667	0.6678
	ROC	0.939	0.782	0.853	0.728	0.889	0.8382

Table A10 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.978	0.758	0.938	0.766	0.77	0.842
Four	Accuracy	0.951	0.978	0.978	0.973	0.946	0.9652
	Precision	0.938	0.938	0.978	0.957	0.978	0.9578
	Recall	0.957	0.989	0.957	0.957	0.989	0.9698
	F1-score	0.957	0.979	0.961	0.974	0.94	0.9622
	ROC	0.978	0.978	0.978	0.984	0.967	0.977
	G-mean	0.957	0.978	0.968	0.978	0.956	0.9674

Performance of Gradient Boosting

Table A11: Gradient Boosting ML model performance evaluation on Ionosphere dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.901	0.943	0.9	0.971	0.957	0.9344
	Precision	0.932	0.918	0.88	0.957	1	0.9374
	Recall	0.911	1	0.978	1	0.978	0.9734
	F1-score	0.921	0.957	0.926	0.978	0.989	0.9542
	ROC	0.973	0.95	0.947	0.978	0.998	0.9692

Table A11 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.898	0.917	0.862	0.959	0.989	0.925
Two	Accuracy	0.878	0.956	0.944	0.978	0.956	0.9424
	Precision	0.854	0.918	0.917	0.957	0.935	0.9162
	Recall	0.911	1	0.978	1	0.956	0.969
	F1-score	0.882	0.957	0.946	0.978	0.956	0.9438
	ROC	0.952	0.996	0.989	0.987	0.989	0.9826
	G-mean	0.877	0.955	0.944	0.978	0.944	0.9396
Three	Accuracy	0.901	0.929	0.9	0.971	0.971	0.9344
	Precision	0.896	0.9	0.88	0.957	0.977	0.922
	Recall	0.956	0.978	1	1	1	0.9868
	F1-score	0.935	0.946	0.928	0.978	0.978	0.953
	ROC	0.972	0.966	0.966	0.966	0.996	0.9732
	G-mean	0.899	0.917	0.862	0.959	0.969	0.9212
Four	Accuracy	0.889	0.967	0.956	0.944	0.956	0.9424
	Precision	0.889	0.917	0.957	0.956	0.936	0.931
	Recall	0.889	0.978	0.978	0.956	0.978	0.9558
	F1-score	0.889	0.968	0.957	0.956	0.957	0.9454
	ROC	0.955	0.994	0.993	0.985	0.989	0.9832
	G-mean	0.889	0.944	0.955	0.956	0.955	0.9398

Table A12: Gradient Boosting ML model performance evaluation on Pageblocks dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.979	1	1	1	0.979	0.9916
	Precision	1	1	1	1	0.75	0.95
	Recall	0.667	1	1	1	1	0.9334
	F1-score	0.8	1	1	1	0.857	0.9314
	ROC	0.838	1	1	1	0.989	0.9654
	G-mean	0.816	1	1	1	0.989	0.961
Two	Accuracy	1	1	1	1	0.989	0.9978
	Precision	1	1	1	1	0.978	0.9956
	Recall	1	1	1	1	1	1
	F1-score	1	1	1	1	0.989	0.9978
	ROC	1	1	1	1	0.989	0.9978
	G-mean	1	1	1	1	0.989	0.9978
Three	Accuracy	0.979	1	1	1	0.979	0.9916
	Precision	1	1	1	1	0.75	0.95
	Recall	0.667	1	1	1	1	0.9334
	F1-score	0.8	1	1	1	0.857	0.9314

Table A12 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.838	1	1	1	0.989	0.9654
	G-mean	0.816	1	1	1	0.989	0.961
Four	Accuracy	1	1	1	1	0.989	0.9978
	Precision	1	1	1	1	0.978	0.9956
	Recall	1	1	1	1	1	1
	F1-score	1	1	1	1	0.989	0.9978
	ROC	1	1	1	1	0.989	0.9978
	G-mean	1	1	1	1	0.989	0.9978

Table A13: Gradient Boosting ML model performance evaluation on Spambase dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.948	0.949	0.94	0.96	0.935	0.9464
	Precision	0.949	0.936	0.945	0.95	0.936	0.9432
	Recall	0.917	0.931	0.901	0.948	0.895	0.9184
	F1-score	0.931	0.932	0.922	0.948	0.915	0.9296
	ROC	0.987	0.987	0.987	0.99	0.974	0.985

Table A13 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.939	0.945	0.933	0.958	0.927	0.9404
Two	Accuracy	0.949	0.956	0.949	0.948	0.946	0.9496
	Precision	0.95	0.967	0.953	0.953	0.953	0.9552
	Recall	0.948	0.944	0.944	0.946	0.941	0.9446
	F1-score	0.949	0.956	0.948	0.95	0.946	0.9498
	ROC	0.987	0.989	0.988	0.988	0.986	0.9876
	G-mean	0.949	0.956	0.949	0.948	0.946	0.9496
Three	Accuracy	0.946	0.947	0.958	0.97	0.941	0.9524
	Precision	0.938	0.931	0.96	0.966	0.938	0.9466
	Recall	0.931	0.928	0.928	0.956	0.906	0.9298
	F1-score	0.936	0.939	0.941	0.964	0.921	0.9402
	ROC	0.99	0.988	0.99	0.993	0.98	0.9882
	G-mean	0.943	0.943	0.95	0.964	0.941	0.9482
Four	Accuracy	0.955	0.967	0.957	0.969	0.957	0.961
	Precision	0.955	0.973	0.956	0.959	0.957	0.96
	Recall	0.964	0.953	0.961	0.971	0.961	0.962
	F1-score	0.955	0.966	0.957	0.967	0.959	0.9608
	ROC	0.992	0.993	0.99	0.992	0.992	0.9918
	G-mean	0.954	0.966	0.956	0.967	0.96	0.9606

Table A14: Gradient Boosting ML model performance evaluation on Winequality dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.969	0.954	0.954	0.977	0.977	0.9662
	Precision	0	0	0.25	1	1	0.45
	Recall	0	0	0.25	0.25	0.25	0.15
	F1-score	0	0	0.25	0.4	0.4	0.21
	ROC	0.852	0.878	0.872	0.992	0.929	0.9046
	G-mean	0	0	0.494	0.5	0.5	0.2988
Two	Accuracy	0.988	0.973	0.98	0.984	0.992	0.9834
	Precision	0.984	0.955	0.962	0.984	0.992	0.9754
	Recall	0.992	0.992	1	0.984	0.992	0.992
	F1-score	0.988	0.973	0.981	0.984	0.992	0.9836
	ROC	1	0.998	1	0.996	1	0.9988
	G-mean	0.988	0.972	0.98	0.984	0.992	0.9832
Three	Accuracy	0.977	0.977	0.969	0.969	0.969	0.9722
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0

Table A14 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.635	0.363	0.684	0.951	0.873	0.7012
	G-mean	0	0	0	0	0	0
Four	Accuracy	0.961	0.91	0.949	0.945	0.933	0.9396
	Precision	0.933	0.861	0.901	0.914	0.882	0.8982
	Recall	0.992	0.976	1	0.992	1	0.992
	F1-score	0.962	0.915	0.948	0.948	0.937	0.942
	ROC	0.982	0.949	0.962	0.98	0.971	0.9688
	G-mean	0.96	0.908	0.943	0.948	0.931	0.938

Table A15: Gradient Boosting ML model performance evaluation on Yeast dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.971	0.913	0.971	0.912	0.961	0.9456
	Precision	0.818	0.556	0.818	0.571	0.875	0.7276
	Recall	0.9	0.5	0.909	0.4	0.7	0.6818
	F1-score	0.857	0.526	0.818	0.471	0.778	0.69
	ROC	0.992	0.955	0.991	0.972	0.968	0.9756

Table A15 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.938	0.692	0.895	0.622	0.832	0.7958
Two	Accuracy	0.968	0.978	0.984	0.978	0.984	0.9784
	Precision	0.967	0.958	0.979	0.958	0.989	0.9702
	Recall	0.967	1	0.989	0.989	0.978	0.9846
	F1-score	0.967	0.979	0.984	0.979	0.984	0.9786
	ROC	0.997	0.997	0.998	0.999	0.996	0.9974
	G-mean	0.968	0.978	0.984	0.978	0.989	0.9794
Three	Accuracy	0.971	0.932	0.971	0.941	0.971	0.9572
	Precision	0.818	0.667	0.818	0.75	0.8	0.7706
	Recall	0.9	0.6	0.818	0.6	0.8	0.7436
	F1-score	0.857	0.632	0.818	0.667	0.842	0.7632
	ROC	0.991	0.956	0.99	0.967	0.9	0.9608
	G-mean	0.938	0.762	0.89	0.766	0.885	0.8482
Four	Accuracy	0.968	0.973	0.984	0.968	0.967	0.972
	Precision	0.946	0.948	0.979	0.948	0.968	0.9578
	Recall	0.957	1	0.989	0.989	0.978	0.9826
	F1-score	0.951	0.974	0.984	0.963	0.973	0.969
	ROC	0.996	0.996	0.999	0.998	0.997	0.9972
	G-mean	0.951	0.978	0.984	0.967	0.973	0.9706

Performance of K-Nearest Neighbors

Table A16: K-Nearest Neighbors ML model performance evaluation on Ionosphere dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.817	0.843	0.8	0.871	0.9	0.8462
	Precision	0.808	0.804	0.772	0.833	0.88	0.8194
	Recall	0.933	1	0.978	1	0.978	0.9778
	F1-score	0.866	0.891	0.863	0.909	0.926	0.891
	ROC	0.875	0.952	0.888	0.897	0.929	0.9082
	G-mean	0.758	0.748	0.685	0.8	0.862	0.7706
Two	Accuracy	0.867	0.9	0.933	0.956	0.922	0.9156
	Precision	0.837	0.833	0.898	0.918	0.896	0.8764
	Recall	0.911	1	0.978	1	0.956	0.969
	F1-score	0.872	0.909	0.936	0.957	0.925	0.9198
	ROC	0.923	0.961	0.972	0.989	0.99	0.967
	G-mean	0.866	0.894	0.932	0.955	0.922	0.9138
Three	Accuracy	0.803	0.871	0.829	0.871	0.914	0.8576
	Precision	0.804	0.833	0.8	0.833	0.882	0.8304
	Recall	0.911	1	0.978	1	1	0.9778

Table A16 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	F1-score	0.854	0.909	0.88	0.909	0.938	0.898
	ROC	0.836	0.92	0.855	0.88	0.935	0.8852
	G-mean	0.749	0.8	0.74	0.8	0.872	0.7922
Four	Accuracy	0.867	0.922	0.956	0.933	0.933	0.9222
	Precision	0.867	0.865	0.936	0.882	0.915	0.893
	Recall	0.867	1	0.978	1	0.956	0.9602
	F1-score	0.867	0.928	0.957	0.938	0.935	0.925
	ROC	0.913	0.954	0.965	0.989	0.961	0.9564
	G-mean	0.867	0.919	0.955	0.931	0.933	0.921

Table A17: K-Nearest Neighbors ML model performance evaluation on Pageblocks dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.968	1	0.968	0.989	0.979	0.9808
	Precision	1	1	0.667	1	1	0.9334
	Recall	0.5	1	0.8	0.833	0.667	0.76
	F1-score	0.667	1	0.727	0.909	0.8	0.8206

Table A17 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.997	1	0.996	0.999	1	0.9984
	G-mean	0.707	1	0.884	0.913	0.816	0.864
Two	Accuracy	0.989	0.994	0.989	1	0.994	0.9932
	Precision	0.978	0.989	0.978	1	0.989	0.9868
	Recall	1	1	1	1	1	1
	F1-score	0.989	0.994	0.989	1	0.994	0.9932
	ROC	1	1	1	1	1	1
	G-mean	0.989	0.994	0.989	1	0.994	0.9932
Three	Accuracy	0.989	0.989	1	0.989	0.989	0.9912
	Precision	1	0.833	1	1	1	0.9666
	Recall	0.833	1	1	0.833	0.833	0.8998
	F1-score	0.909	0.909	1	0.909	0.909	0.9272
	ROC	0.915	1	1	1	1	0.983
	G-mean	0.913	0.994	1	0.913	0.913	0.9466
Four	Accuracy	1	0.994	0.994	1	1	0.9976
	Precision	1	0.989	0.989	1	1	0.9956
	Recall	1	1	1	1	1	1
	F1-score	1	0.994	0.994	1	1	0.9976
	ROC	1	1	1	1	1	1

Table A17 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	1	0.994	0.994	1	1	0.9976

Table A18: K-Nearest Neighbors ML model performance evaluation on Poker dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.99	0.993	0.993	0.99	0.986	0.9904
	Precision	1	1	1	0	0	0.6
	Recall	0.25	0.333	0.333	0	0	0.1832
	F1-score	0.4	0.5	0.5	0	0	0.28
	ROC	0.87	0.828	0.828	0.83	0.991	0.8694
	G-mean	0.5	0.577	0.577	0	0	0.3308
Two	Accuracy	0.997	0.991	0.991	0.993	0.991	0.9926
	Precision	0.993	0.983	0.983	0.986	0.983	0.9856
	Recall	1	1	1	1	1	1
	F1-score	0.997	0.992	0.992	0.993	0.992	0.9932
	ROC	1	0.998	0.998	0.998	1	0.9988
	G-mean	0.997	0.991	0.991	0.993	0.991	0.9926

Table A18 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Three	Accuracy	0.99	0.993	0.993	0.993	0.99	0.9918
	Precision	1	1	1	1	1	1
	Recall	0.25	0.333	0.333	0.333	0.25	0.2998
	F1-score	0.4	0.5	0.5	0.5	0.4	0.46
	ROC	0.874	0.83	0.83	0.831	0.996	0.8722
	G-mean	0.5	0.577	0.577	0.577	0.5	0.5462
Four	Accuracy	1	0.998	0.995	0.998	0.995	0.9972
	Precision	1	0.997	0.99	0.997	0.99	0.9948
	Recall	1	1	1	1	1	1
	F1-score	1	0.998	0.995	0.998	0.995	0.9972
	ROC	1	0.998	0.998	0.998	0.998	0.9984
	G-mean	1	0.998	0.995	0.998	0.995	0.9972

Table A19: K-Nearest Neighbors ML model performance evaluation on Winequality dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	Accuracy	0.977	0.977	0.969	0.969	0.969	0.9722

One

Table A19 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.604	0.616	0.798	0.582	0.714	0.6628
	G-mean	0	0	0	0	0	0
Two	Accuracy	0.933	0.933	0.937	0.949	0.965	0.9434
	Precision	0.882	0.882	0.889	0.908	0.934	0.899
	Recall	1	1	1	1	1	1
	F1-score	0.937	0.937	0.941	0.952	0.966	0.9466
	ROC	0.984	0.984	0.969	0.988	1	0.985
	G-mean	0.931	0.931	0.935	0.947	0.964	0.9416
Three	Accuracy	0.977	0.977	0.977	0.969	0.977	0.9754
	Precision	0	0	1	0	1	0.4
	Recall	0	0	0.25	0	0.25	0.1
	F1-score	0	0	0.4	0	0.4	0.16
	ROC	0.625	0.633	0.825	0.596	0.73	0.6818
	G-mean	0	0	0.5	0	0.5	0.2
Four	Accuracy	0.949	0.945	0.941	0.961	0.969	0.953
	Precision	0.907	0.901	0.895	0.928	0.941	0.9144

Table A19 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	Recall	1	1	1	1	1	1
	F1-score	0.951	0.948	0.945	0.962	0.969	0.955
	ROC	0.984	0.984	0.969	0.988	1	0.985
	G-mean	0.948	0.944	0.939	0.96	0.968	0.9518

Table A20: K-Nearest Neighbors ML model performance evaluation on Yeast dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.971	0.942	0.981	0.941	0.951	0.9572
	Precision	0.889	1	1	1	0.857	0.9492
	Recall	0.8	0.4	0.818	0.4	0.6	0.6036
	F1-score	0.842	0.571	0.9	0.571	0.706	0.718
	ROC	0.996	0.832	0.953	0.872	0.877	0.906
	G-mean	0.89	0.632	0.905	0.632	0.77	0.7658
Two	Accuracy	0.957	0.968	1	0.946	0.962	0.9666
	Precision	0.929	0.939	1	0.903	0.929	0.94
	Recall	0.989	1	1	1	1	0.9978

Table A20 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	F1-score	0.958	0.968	1	0.949	0.963	0.9676
	ROC	0.987	0.978	1	0.995	0.982	0.9884
	G-mean	0.956	0.967	1	0.944	0.961	0.9656
Three	Accuracy	0.981	0.942	0.981	0.951	0.951	0.9612
	Precision	1	1	1	1	0.857	0.9714
	Recall	0.8	0.4	0.818	0.5	0.6	0.6236
	F1-score	0.889	0.571	0.9	0.667	0.706	0.7466
	ROC	0.997	0.832	0.953	0.931	0.922	0.927
	G-mean	0.894	0.632	0.905	0.707	0.77	0.7816
Four	Accuracy	0.962	0.962	0.995	0.957	0.957	0.9666
	Precision	0.929	0.929	1	0.921	0.92	0.9398
	Recall	1	1	0.989	1	1	0.9978
	F1-score	0.963	0.963	0.995	0.959	0.958	0.9676
	ROC	0.987	0.978	1	0.994	0.987	0.9892
	G-mean	0.962	0.962	0.995	0.956	0.956	0.9662

Performance of Logistic Regression

Table A21: Logistic Regression ML model performance evaluation on Ionosphere dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.887	0.9	0.843	0.857	0.914	0.8802
	Precision	0.911	0.88	0.827	0.857	0.898	0.8746
	Recall	0.911	0.978	0.956	0.933	0.978	0.9512
	F1-score	0.911	0.926	0.887	0.894	0.936	0.9108
	ROC	0.894	0.925	0.877	0.892	0.937	0.905
	G-mean	0.878	0.862	0.782	0.82	0.884	0.8452
Two	Accuracy	0.856	0.889	0.9	0.9	0.9	0.889
	Precision	0.848	0.857	0.891	0.86	0.86	0.8632
	Recall	0.867	0.933	0.911	0.956	0.956	0.9246
	F1-score	0.857	0.894	0.901	0.905	0.905	0.8924
	ROC	0.915	0.97	0.94	0.955	0.954	0.9468
	G-mean	0.855	0.888	0.9	0.898	0.898	0.8878
Three	Accuracy	0.845	0.914	0.843	0.886	0.929	0.8834
	Precision	0.905	0.898	0.827	0.863	0.917	0.882
	Recall	0.844	0.978	0.956	0.978	0.978	0.9468
	F1-score	0.874	0.936	0.887	0.917	0.946	0.912

Table A21 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.881	0.9	0.849	0.892	0.9	0.8844
	G-mean	0.845	0.884	0.782	0.839	0.906	0.8512
Four	Accuracy	0.811	0.9	0.889	0.9	0.922	0.8844
	Precision	0.804	0.86	0.889	0.86	0.896	0.8618
	Recall	0.822	0.956	0.889	0.956	0.956	0.9158
	F1-score	0.813	0.905	0.889	0.905	0.925	0.8874
	ROC	0.904	0.961	0.935	0.956	0.945	0.9402
	G-mean	0.811	0.898	0.889	0.898	0.922	0.8836

Table A22: Logistic Regression ML model performance evaluation on Pageblocks dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.968	0.979	0.968	0.947	0.957	0.9638
	Precision	1	0.8	1	0.667	1	0.8934
	Recall	0.5	0.8	0.4	0.333	0.333	0.4732
	F1-score	0.667	0.8	0.571	0.444	0.5	0.5964
	ROC	0.985	0.989	0.973	0.97	0.972	0.9778

Table A22 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.707	0.889	0.632	0.574	0.577	0.6758
Two	Accuracy	0.983	0.972	0.972	0.983	0.96	0.974
	Precision	0.967	0.946	0.946	0.967	0.927	0.9506
	Recall	1	1	1	1	1	1
	F1-score	0.983	0.972	0.972	0.983	0.962	0.9744
	ROC	0.986	0.982	0.993	0.987	0.975	0.9846
	G-mean	0.983	0.972	0.972	0.983	0.959	0.9738
Three	Accuracy	0.958	0.979	0.968	0.947	0.979	0.9662
	Precision	1	0.8	1	0.667	1	0.8934
	Recall	0.333	0.8	0.4	0.333	0.667	0.5066
	F1-score	0.5	0.8	0.571	0.444	0.8	0.623
	ROC	1	0.993	0.98	0.981	0.979	0.9866
	G-mean	0.577	0.889	0.632	0.574	0.816	0.6976
Four	Accuracy	1	0.972	0.972	0.977	0.96	0.9762
	Precision	1	0.946	0.946	0.957	0.927	0.9552
	Recall	1	1	1	1	1	1
	F1-score	1	0.972	0.972	0.978	0.962	0.9768
	ROC	1	0.984	0.994	0.988	0.982	0.9896
	G-mean	1	0.972	0.972	0.977	0.959	0.976

Table A23: Logistic Regression ML model performance evaluation on Poker dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.986	0.99	0.99	0.99	0.986	0.9884
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.306	0.24	0.46	0.283	0.216	0.301
	G-mean	0	0	0	0	0	0
Two	Accuracy	0.598	0.604	0.639	0.569	0.561	0.5942
	Precision	0.601	0.594	0.628	0.567	0.563	0.5906
	Recall	0.579	0.658	0.682	0.584	0.548	0.6102
	F1-score	0.59	0.624	0.654	0.575	0.556	0.5998
	ROC	0.653	0.603	0.667	0.612	0.615	0.63
	G-mean	0.597	0.602	0.637	0.569	0.561	0.5932
Three	Accuracy	0.986	0.99	0.99	0.99	0.986	0.9884
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0

Table A23 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.343	0.202	0.441	0.211	0.217	0.2828
	G-mean	0	0	0	0	0	0
Four	Accuracy	0.57	0.586	0.618	0.568	0.537	0.5758
	Precision	0.575	0.578	0.609	0.566	0.541	0.5738
	Recall	0.538	0.634	0.661	0.577	0.5	0.582
	F1-score	0.556	0.605	0.634	0.571	0.52	0.5772
	ROC	0.634	0.593	0.661	0.598	0.598	0.6168
	G-mean	0.569	0.584	0.617	0.568	0.536	0.5748

Table A24: Logistic Regression ML model performance evaluation on Spambase dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.911	0.939	0.918	0.937	0.913	0.9236
	Precision	0.914	0.923	0.936	0.925	0.915	0.9226
	Recall	0.854	0.923	0.851	0.914	0.859	0.8802
	F1-score	0.883	0.923	0.891	0.919	0.886	0.9004
	ROC	0.971	0.979	0.967	0.978	0.965	0.972

Table A24 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.9	0.936	0.905	0.933	0.902	0.9152
Two	Accuracy	0.925	0.926	0.929	0.923	0.926	0.9258
	Precision	0.929	0.936	0.93	0.927	0.941	0.9326
	Recall	0.919	0.914	0.928	0.917	0.91	0.9176
	F1-score	0.924	0.925	0.929	0.922	0.925	0.925
	ROC	0.968	0.974	0.976	0.967	0.975	0.972
	G-mean	0.925	0.925	0.929	0.923	0.926	0.9256
Three	Accuracy	0.92	0.942	0.917	0.935	0.916	0.926
	Precision	0.919	0.921	0.931	0.917	0.918	0.9212
	Recall	0.873	0.934	0.854	0.917	0.865	0.8886
	F1-score	0.895	0.927	0.89	0.917	0.89	0.9038
	ROC	0.972	0.979	0.966	0.977	0.966	0.972
	G-mean	0.911	0.941	0.905	0.932	0.906	0.919
Four	Accuracy	0.922	0.923	0.931	0.924	0.926	0.9252
	Precision	0.926	0.932	0.932	0.924	0.939	0.9306
	Recall	0.918	0.912	0.93	0.923	0.91	0.9186
	F1-score	0.922	0.922	0.931	0.924	0.924	0.9246
	ROC	0.968	0.974	0.978	0.968	0.975	0.9726
	G-mean	0.922	0.923	0.931	0.924	0.925	0.925

Table A25: Logistic Regression ML model performance evaluation on Yeast dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.981	0.932	0.961	0.951	0.941	0.9532
	Precision	0.9	0.714	1	1	0.75	0.8728
	Recall	0.9	0.5	0.636	0.5	0.6	0.6272
	F1-score	0.9	0.588	0.778	0.667	0.667	0.72
	ROC	0.997	0.9	0.964	0.93	0.915	0.9412
	G-mean	0.944	0.699	0.798	0.707	0.766	0.7828
Two	Accuracy	0.935	0.924	0.946	0.935	0.908	0.9296
	Precision	0.955	0.906	0.956	0.918	0.879	0.9228
	Recall	0.913	0.946	0.935	0.957	0.946	0.9394
	F1-score	0.933	0.926	0.946	0.937	0.911	0.9306
	ROC	0.964	0.975	0.972	0.981	0.961	0.9706
	G-mean	0.935	0.924	0.946	0.935	0.907	0.9294
Three	Accuracy	0.981	0.951	0.951	0.941	0.951	0.955
	Precision	0.9	1	1	1	0.857	0.9514
	Recall	0.9	0.5	0.545	0.4	0.6	0.589
	F1-score	0.9	0.667	0.706	0.571	0.706	0.71

Table A25 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.998	0.887	0.967	0.935	0.911	0.9396
	G-mean	0.944	0.707	0.739	0.632	0.77	0.7584
Four	Accuracy	0.941	0.924	0.946	0.957	0.913	0.9362
	Precision	0.955	0.906	0.956	0.947	0.888	0.9304
	Recall	0.924	0.946	0.935	0.968	0.946	0.9438
	F1-score	0.939	0.926	0.946	0.957	0.916	0.9368
	ROC	0.963	0.974	0.97	0.982	0.962	0.9702
	G-mean	0.94	0.924	0.946	0.957	0.912	0.9358

Performance of Random Forest

Table A26: Random Forest ML model performance evaluation on Ionosphere dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.915	0.943	0.9	0.971	0.986	0.943
	Precision	0.915	0.918	0.896	0.957	0.978	0.9328
	Recall	0.911	1	0.933	0.978	1	0.9644
	F1-score	0.945	0.957	0.925	0.967	0.978	0.9544
	ROC	0.975	0.99	0.974	0.961	0.997	0.9794

Table A26 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.898	0.917	0.874	0.948	0.969	0.9212
Two	Accuracy	0.867	0.989	0.922	0.956	0.967	0.9402
	Precision	0.848	0.957	0.915	0.978	0.938	0.9272
	Recall	0.911	1	0.933	0.956	1	0.96
	F1-score	0.879	0.978	0.923	0.978	0.968	0.9452
	ROC	0.965	1	0.991	0.987	0.993	0.9872
	G-mean	0.855	0.978	0.933	0.978	0.978	0.9444
Three	Accuracy	0.93	0.943	0.9	0.957	0.971	0.9402
	Precision	0.909	0.938	0.896	0.957	0.978	0.9356
	Recall	0.956	1	0.978	0.978	1	0.9824
	F1-score	0.923	0.968	0.913	0.967	0.978	0.9498
	ROC	0.974	0.98	0.973	0.963	0.999	0.9778
	G-mean	0.919	0.938	0.864	0.948	0.98	0.9298
Four	Accuracy	0.867	0.978	0.933	0.967	0.956	0.9402
	Precision	0.851	0.938	0.915	0.978	0.936	0.9236
	Recall	0.911	0.978	0.956	1	0.978	0.9646
	F1-score	0.894	0.978	0.923	0.978	0.968	0.9482
	ROC	0.961	1	0.99	0.988	0.997	0.9872
	G-mean	0.878	0.967	0.911	0.978	0.966	0.94

Table A27: Random Forest ML model performance evaluation on Pageblocks dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.968	1	0.989	0.989	0.989	0.987
	Precision	1	1	0.833	1	1	0.9666
	Recall	0.667	1	1	0.833	1	0.9
	F1-score	0.667	1	0.909	0.909	0.909	0.8788
	ROC	0.998	1	1	1	1	0.9996
	G-mean	0.707	1	0.994	0.913	0.913	0.9054
Two	Accuracy	1	1	1	1	1	1
	Precision	1	1	1	1	1	1
	Recall	1	1	1	1	1	1
	F1-score	1	1	1	1	1	1
	ROC	1	1	1	1	1	1
	G-mean	1	1	1	1	1	1
Three	Accuracy	0.979	1	0.989	0.989	1	0.9914
	Precision	1	1	1	1	1	1
	Recall	0.5	1	1	1	1	0.9
	F1-score	0.667	1	0.909	1	1	0.9152
	ROC	0.99	1	1	1	1	0.998

Table A27 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.707	1	0.994	1	1	0.9402
Four	Accuracy	1	1	1	1	0.989	0.9978
	Precision	1	1	1	1	0.978	0.9956
	Recall	1	1	1	1	1	1
	F1-score	1	1	1	1	0.989	0.9978
	ROC	1	1	1	1	1	1
	G-mean	1	1	1	1	0.989	0.9978

Table A28: Random Forest ML model performance evaluation on Poker dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.986	0.99	0.99	0.99	0.986	0.9884
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.98	0.859	0.939	0.9	0.996	0.9348
	G-mean	0	0	0	0	0	0
Two	Accuracy	0.993	0.995	0.998	0.998	0.998	0.9964

Table A28 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	Precision	1	1	1	1	1	1
	Recall	0.986	0.99	0.997	0.993	0.997	0.9926
	F1-score	0.993	0.995	0.998	0.997	0.998	0.9962
	ROC	0.999	1	1	1	1	0.9998
	G-mean	0.993	0.995	0.998	0.997	0.998	0.9962
Three	Accuracy	0.986	0.99	0.99	0.99	0.986	0.9884
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.984	0.91	0.974	0.86	0.956	0.9368
	G-mean	0	0	0	0	0	0
Four	Accuracy	0.993	0.995	0.998	1	0.998	0.9968
	Precision	1	1	1	1	1	1
	Recall	0.986	0.99	0.997	1	0.997	0.994
	F1-score	0.993	0.995	0.998	0.998	0.998	0.9964
	ROC	0.997	1	1	1	1	0.9994
	G-mean	0.993	0.995	0.998	1	0.998	0.9968

Table A29: Random Forest ML model performance evaluation on Spambase dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.946	0.95	0.957	0.962	0.952	0.9534
	Precision	0.948	0.944	0.966	0.967	0.962	0.9574
	Recall	0.923	0.931	0.928	0.95	0.906	0.9276
	F1-score	0.93	0.936	0.946	0.957	0.936	0.941
	ROC	0.988	0.988	0.987	0.994	0.978	0.987
	G-mean	0.946	0.949	0.95	0.96	0.946	0.9502
Two	Accuracy	0.961	0.97	0.956	0.966	0.962	0.963
	Precision	0.958	0.976	0.961	0.961	0.959	0.963
	Recall	0.971	0.966	0.953	0.968	0.961	0.9638
	F1-score	0.962	0.968	0.962	0.964	0.962	0.9636
	ROC	0.992	0.992	0.99	0.992	0.993	0.9918
	G-mean	0.964	0.967	0.956	0.968	0.961	0.9632
Three	Accuracy	0.949	0.951	0.954	0.96	0.946	0.952
	Precision	0.954	0.944	0.965	0.961	0.964	0.9576
	Recall	0.915	0.923	0.92	0.953	0.898	0.9218
	F1-score	0.938	0.931	0.943	0.954	0.928	0.9388
	ROC	0.987	0.988	0.985	0.992	0.98	0.9864

Table A29 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.943	0.941	0.947	0.959	0.943	0.9466
Four	Accuracy	0.959	0.967	0.957	0.964	0.958	0.961
	Precision	0.966	0.978	0.96	0.955	0.961	0.964
	Recall	0.962	0.952	0.953	0.962	0.959	0.9576
	F1-score	0.956	0.966	0.956	0.959	0.957	0.9588
	ROC	0.99	0.99	0.99	0.993	0.992	0.991
	G-mean	0.956	0.968	0.956	0.965	0.959	0.9608

Table A30: Random Forest ML model performance evaluation on Winequality dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.977	0.977	0.977	0.969	0.977	0.9754
	Precision	0	0	1	0	1	0.4
	Recall	0	0	0.25	0	0.25	0.1
	F1-score	0	0	0.4	0	0.4	0.16
	ROC	0.885	0.852	0.829	0.969	0.942	0.8954
	G-mean	0	0	0.5	0	0.5	0.2

Table A30 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Two	Accuracy	0.996	0.976	0.984	0.984	0.996	0.9872
	Precision	1	0.976	0.97	0.992	1	0.9876
	Recall	1	0.992	1	0.984	1	0.9952
	F1-score	0.996	0.972	0.981	0.984	1	0.9866
	ROC	1	0.999	1	0.999	1	0.9996
	G-mean	1	0.976	0.988	0.984	0.992	0.988
Three	Accuracy	0.977	0.977	0.977	0.969	0.977	0.9754
	Precision	0	0	1	0	1	0.4
	Recall	0	0	0.25	0	0.25	0.1
	F1-score	0	0	0.4	0	0.4	0.16
	ROC	0.786	0.811	0.88	0.965	0.951	0.8786
	G-mean	0	0	0.5	0	0	0.1
Four	Accuracy	0.992	0.973	0.976	0.988	1	0.9858
	Precision	0.992	0.976	0.955	0.984	1	0.9814
	Recall	1	0.984	1	0.992	1	0.9952
	F1-score	0.996	0.972	0.977	0.984	0.996	0.985
	ROC	1	0.999	1	1	1	0.9998
	G-mean	0.996	0.973	0.984	0.988	1	0.9882

Performance of Support Vector Machine

Table A31: Support Vector ML model performance evaluation on Pageblocks dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.968	0.989	0.968	0.947	0.957	0.9658
	Precision	1	1	1	1	1	1
	Recall	0.5	0.8	0.4	0.167	0.333	0.44
	F1-score	0.667	0.889	0.571	0.286	0.5	0.5826
	ROC	1	0.996	0.987	0.996	0.996	0.995
	G-mean	0.707	0.894	0.632	0.408	0.577	0.6436
Two	Accuracy	0.994	0.989	0.983	0.983	0.977	0.9852
	Precision	0.989	0.978	0.967	0.967	0.957	0.9716
	Recall	1	1	1	1	1	1
	F1-score	0.994	0.989	0.983	0.983	0.978	0.9854
	ROC	1	0.994	0.989	0.991	0.98	0.9908
	G-mean	0.994	0.989	0.983	0.983	0.977	0.9852
Three	Accuracy	0.968	0.989	0.989	1	1	0.9892
	Precision	1	1	1	1	1	1
	Recall	0.5	0.8	0.8	1	1	0.82
	F1-score	0.667	0.889	0.889	1	1	0.889

Table A31 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	ROC	0.998	0.998	1	1	1	0.9992
	G-mean	0.707	0.894	0.894	1	1	0.899
Four	Accuracy	1	1	0.994	0.989	0.983	0.9932
	Precision	1	1	0.989	0.978	0.967	0.9868
	Recall	1	1	1	1	1	1
	F1-score	1	1	0.994	0.989	0.983	0.9932
	ROC	1	1	0.996	1	0.998	0.9988
	G-mean	1	1	0.994	0.989	0.983	0.9932

Table A32: Support Vector ML model performance evaluation on Poker dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.986	0.99	0.99	0.99	0.986	0.9884
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.98	0.944	0.985	0.939	0.995	0.9686
	G-mean	0	0	0	0	0	0

Table A32 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Two	Accuracy	1	1	1	1	1	1
	Precision	1	1	1	1	1	1
	Recall	1	1	1	1	1	1
	F1-score	1	1	1	1	1	1
	ROC	1	1	1	1	1	1
	G-mean	1	1	1	1	1	1
Three	Accuracy	0.993	0.997	0.997	0.997	0.993	0.9954
	Precision	1	1	1	1	1	1
	Recall	0.5	0.667	0.667	0.667	0.5	0.6002
	F1-score	0.667	0.8	0.8	0.8	0.667	0.7468
	ROC	1	0.999	1	1	0.973	0.9944
	G-mean	0.707	0.816	0.816	0.816	0.707	0.7724
Four	Accuracy	1	1	1	1	1	1
	Precision	1	1	1	1	1	1
	Recall	1	1	1	1	1	1
	F1-score	1	1	1	1	1	1
	ROC	1	1	1	1	1	1
	G-mean	1	1	1	1	1	1

Table A33: Support Vector ML model performance evaluation on Spambase dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.926	0.941	0.924	0.947	0.924	0.9324
	Precision	0.93	0.926	0.948	0.938	0.932	0.9348
	Recall	0.879	0.926	0.854	0.925	0.87	0.8908
	F1-score	0.904	0.926	0.898	0.932	0.9	0.912
	ROC	0.978	0.981	0.97	0.981	0.963	0.9746
	G-mean	0.917	0.938	0.91	0.943	0.913	0.9242
Two	Accuracy	0.927	0.939	0.935	0.928	0.945	0.9348
	Precision	0.933	0.961	0.947	0.933	0.964	0.9476
	Recall	0.921	0.916	0.923	0.923	0.925	0.9216
	F1-score	0.927	0.938	0.935	0.928	0.944	0.9344
	ROC	0.973	0.981	0.976	0.973	0.982	0.977
	G-mean	0.927	0.939	0.935	0.928	0.945	0.9348
Three	Accuracy	0.93	0.937	0.933	0.953	0.929	0.9364
	Precision	0.931	0.92	0.952	0.939	0.93	0.9344
	Recall	0.89	0.92	0.873	0.942	0.887	0.9024
	F1-score	0.91	0.92	0.911	0.941	0.908	0.918
	ROC	0.982	0.985	0.979	0.987	0.968	0.9802

Table A33 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.923	0.934	0.921	0.951	0.921	0.93
Four	Accuracy	0.941	0.944	0.946	0.943	0.948	0.9444
	Precision	0.941	0.959	0.948	0.938	0.956	0.9484
	Recall	0.941	0.928	0.944	0.95	0.939	0.9404
	F1-score	0.941	0.944	0.946	0.944	0.947	0.9444
	ROC	0.981	0.983	0.982	0.981	0.985	0.9824
	G-mean	0.941	0.944	0.946	0.943	0.948	0.9444

Table A34: Support Vector ML model performance evaluation on Winequality dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.977	0.977	0.969	0.969	0.969	0.9722
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.81	0.74	0.76	0.703	0.758	0.7542
	G-mean	0	0	0	0	0	0

Table A34 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Two	Accuracy	0.965	0.957	0.957	0.961	0.976	0.9632
	Precision	0.934	0.92	0.921	0.928	0.955	0.9316
	Recall	1	1	1	1	1	1
	F1-score	0.966	0.958	0.959	0.962	0.977	0.9644
	ROC	1	1	0.971	0.988	0.999	0.9916
	G-mean	0.964	0.956	0.956	0.96	0.976	0.9624
Three	Accuracy	0.977	0.977	0.969	0.969	0.969	0.9722
	Precision	0	0	0	0	0	0
	Recall	0	0	0	0	0	0
	F1-score	0	0	0	0	0	0
	ROC	0.828	0.568	0.829	0.927	0.553	0.741
	G-mean	0	0	0	0	0	0
Four	Accuracy	0.914	0.929	0.902	0.906	0.945	0.9192
	Precision	0.852	0.876	0.841	0.842	0.901	0.8624
	Recall	1	1	0.992	1	1	0.9984
	F1-score	0.92	0.934	0.91	0.914	0.948	0.9252
	ROC	0.944	0.949	0.901	0.929	0.96	0.9366
	G-mean	0.91	0.927	0.897	0.901	0.943	0.9156

Table A35: Support Vector ML model performance evaluation on Yeast dataset under four different conditions.

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
One	Accuracy	0.981	0.942	0.99	0.951	0.951	0.963
	Precision	0.9	0.833	1	1	0.857	0.918
	Recall	0.9	0.5	0.909	0.5	0.6	0.6818
	F1-score	0.9	0.625	0.952	0.667	0.706	0.77
	ROC	0.999	0.937	0.994	0.968	0.957	0.971
	G-mean	0.944	0.703	0.953	0.707	0.77	0.8154
Two	Accuracy	0.968	0.962	0.989	0.957	0.967	0.9686
	Precision	0.957	0.929	0.989	0.929	0.939	0.9486
	Recall	0.978	1	0.989	0.989	1	0.9912
	F1-score	0.968	0.963	0.989	0.958	0.968	0.9692
	ROC	0.994	0.994	0.999	0.997	0.99	0.9948
	G-mean	0.968	0.962	0.989	0.956	0.967	0.9684
Three	Accuracy	0.981	0.942	0.99	0.951	0.951	0.963
	Precision	0.9	0.833	1	1	0.857	0.918
	Recall	0.9	0.5	0.909	0.5	0.6	0.6818
	F1-score	0.9	0.625	0.952	0.667	0.706	0.77
	ROC	0.999	0.937	0.994	0.968	0.957	0.971

Table A35 continued from previous page

Conditions	Evaluation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
	G-mean	0.944	0.703	0.953	0.707	0.77	0.8154
Four	Accuracy	0.968	0.962	0.989	0.957	0.967	0.9686
	Precision	0.957	0.929	0.989	0.929	0.939	0.9486
	Recall	0.978	1	0.989	0.989	1	0.9912
	F1-score	0.968	0.963	0.989	0.958	0.968	0.9692
	ROC	0.994	0.994	0.999	0.997	0.99	0.9948
	G-mean	0.968	0.962	0.989	0.956	0.967	0.9684

.2 Appendix B

.2.1 Optimal Parameters

Table B36: Optimal AB ML hyper-parameters used to experiment with conditions two and four.

Dataset	Optimal parameters
Ionosphere	learning_rate: 1, n_estimators: 50
Pageblocks	learning_rate: 1, n_estimators: 50
Poker	learning_rate: 0.01, n_estimators: 50
Spambase	learning_rate: 1, n_estimators: 100
Winequality	learning_rate: 0.01, n_estimators: 50
Yeast	learning_rate: 0.01, n_estimators: 100

Table B37: Optimal DT ML hyper-parameters used to experiment with conditions two and four.

Dataset	Optimal parameters
Ionosphere	criterion: gini, max_depth: 10 max_features: 5 min_samples_split: 4
Pageblocks	criterion: gini, max_depth: 10 max_features: 5 min_samples_split: 4
Poker	criterion: gini, max_depth: 10 max_features: 6 min_samples_split: 3
Spambase	criterion: gini, max_depth: 50 max_features: 8 min_samples_split: 4
Winequality	criterion: gini, max_depth: 50 max_features: 2 min_samples_split: 4
Yeast	criterion: gini, max_depth: 50 max_features: 6 min_samples_split: 3

Table B38: Optimal GB ML hyper-parameters used to experiment with conditions two and four.

Dataset	Optimal parameters
Ionosphere	learning_rate: 1, max_depth: 3, n_estimators: 100
Pageblocks	learning_rate: 0.1, max_depth: 3, n_estimators: 100
Poker	learning_rate: 0.1, max_depth: 5, n_estimators: 150
Spambase	learning_rate: 0.1, max_depth: 7, n_estimators: 150
Winequality	learning_rate: 0.01, max_depth: 5, n_estimators: 50
Yeast	learning_rate: 0.1, max_depth: 5, n_estimators: 150

Table B39: Optimal KNN ML hyper-parameters used to experiment with conditions two and four.

Dataset	Optimal parameters
Ionosphere	n_neighbors: 3, weights: uniform
Pageblocks	n_neighbors: 3, weights: uniform
Poker	n_neighbors: 3, weights: uniform
Spambase	n_neighbors: 7, weights: distance
Winequality	n_neighbors: 5, weights: distance
Yeast	n_neighbors: 7, weights: uniform

Table B40: Optimal LR ML hyper-parameters used to experiment with conditions two and four.

Dataset	Optimal parameters
Ionosphere	C: 10, penalty: l2
Pageblocks	C: 10, penalty: l2
Poker	C: 0.01, penalty: l2
Spambase	C: 10, penalty: l2
Winequality	C: 0.01, penalty: l2
Yeast	C: 0.1, penalty: l2

Table B41: Optimal RF ML hyper-parameters used to experiment with conditions two and four.

Dataset	Optimal parameters
Ionosphere	max_depth: 20, max_features: 5, min_samples_split: 4, n_estimators: 50
Pageblocks	max_depth: 20, max_features: 8, min_samples_split: 4, n_estimators: 50
Poker	max_depth: 10, max_features: 2, min_samples_split: 3, n_estimators: 50
Spambase	max_depth: 20, max_features: 6, min_samples_split: 3, n_estimators: 100
Winequality	max_depth: 10, max_features: 2, min_samples_split: 3, n_estimators: 100
Yeast	max_depth: 20, max_features: 2, min_samples_split: 5, n_estimators: 100

Table B42: Optimal SVM ML hyper-parameters used to experiment with conditions two and four.

Dataset	Optimal parameters
Ionosphere	C: 1, gamma: scale, kernel: rbf
Pageblocks	C: 10, gamma: scale, kernel:rbf
Poker	C: 10, gamma: scale, kernel:rbf
Spambase	C: 10, gamma: scale, kernel:rbf
Winequality	C: 0.1, gamma: scale, kernel:linear
Yeast	C: 1, gamma: scale, kernel:rbf

.3 Appendix C

.3.1 Hypothesis Testing

.3.1.1 Adaboost

Table C43: Results of hypothesis testing for Adaboost algorithms on Pageblocks dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9992	0.0016	ML	ML + SMOTE	0.0004	-0.0008	1.224745	0.27522	Fail to reject H0
ML + SMOTE	0.9988	0.0024	ML	ML(HP)	0	0	–	–	Fail to reject H0
ML(HP)	0.9992	0.0016	ML	ML(HP) + SMOTE	0.0004	-0.0008	1.224745	0.27522	Fail to reject H0
ML(HP) + SMOTE	0.9988	0.0024	ML + SMOTE	ML(HP)	-0.0004	0.0008	-1.22474	0.27522	Fail to Reject H0
			ML + SMOTE	ML(HP) + SMOTE	0	0	–	–	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	0.0004	-0.0008	1.224745	0.27522	Fail to reject H0

Table C44: Results of hypothesis testing for Adaboost algorithms on Poker dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.5702	0.100527	ML	ML + SMOTE	-0.4232	0.098468	-10.3618	0.000144	Reject H0
ML + SMOTE	0.9934	0.002059	ML	ML(HP)	0.1506	-0.006	7.821855	0.000548	Reject H0
ML(HP)	0.4196	0.106522	ML	ML(HP) + SMOTE	-0.1926	0.085811	-5.25756	0.003306	Reject H0
ML(HP) + SMOTE	0.7628	0.014716	ML + SMOTE	ML(HP)	0.5738	-0.10446	13.20252	4.45E-05	Reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.2306	-0.01266	42.64525	1.34E-07	Reject H0
			ML(HP)	ML(HP) + SMOTE	-0.3432	0.091807	-8.61645	0.000347	Reject H0

Table C45: Results of hypothesis testing for Adaboost algorithms on Spambase dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.978	0.006325	ML	ML + SMOTE	-0.0048	0.004385	-2.02356	0.09892	Fail to reject H0
ML + SMOTE	0.9828	0.001939	ML	ML(HP)	-0.0014	0.000338	-4.28661	0.007814	Fail to reject H0
ML(HP)	0.9794	0.005987	ML	ML(HP) + SMOTE	-0.0058	0.004604	-2.2531	0.073989	Fail to reject H0
ML(HP) + SMOTE	0.9838	0.00172	ML + SMOTE	ML(HP)	0.0034	-0.00405	1.494836	0.195194	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.001	0.000219	-1.58114	0.174688	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0044	0.004266	-1.79033	0.133408	Fail to reject H0

Table C46: Results of hypothesis testing for Adaboost algorithms on Winequality dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.8336	0.099275	ML	ML + SMOTE	-0.1508	0.09025	-3.63658	0.014957	Fail to reject H0
ML + SMOTE	0.9844	0.009024	ML	ML(HP)	0.122	-0.0489	2.481644	0.055729	Fail to reject H0
ML(HP)	0.7116	0.148174	ML	ML(HP) + SMOTE	-0.0486	0.085734	-1.06521	0.335495	Fail to reject H0
ML(HP) + SMOTE	0.8822	0.013541	ML + SMOTE	ML(HP)	0.2728	-0.13915	4.431765	0.006817	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.1022	-0.00452	14.10584	3.22E-05	Reject H0
			ML(HP)	ML(HP) + SMOTE	-0.1706	0.134633	-2.68577	0.043518	Fail to reject H0

Table C47: Results of hypothesis testing for Adaboost algorithms on Yeast dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.899	0.078149	ML	ML + SMOTE	-0.0904	0.067498	-2.61774	0.047227	Fail to reject H0
ML + SMOTE	0.9894	0.010651	ML	ML(HP)	-0.0794	0.06103	-2.8659	0.035163	Fail to reject H0
ML(HP)	0.9784	0.017118	ML	ML(HP) + SMOTE	-0.0934	0.074537	-2.8827	0.034479	Fail to reject H0
ML(HP) + SMOTE	0.9924	0.003611	ML + SMOTE	ML(HP)	0.011	-0.00647	1.146414	0.303508	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.003	0.00704	-0.8037	0.45806	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.014	0.013507	-2.10263	0.089454	Fail to reject H0

.3.1.2 Decision Tree

Table C48: Results of hypothesis testing for DT algorithms on Ionosphere dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.8808	0.041513	ML	ML + SMOTE	-0.0104	0.016047	-0.47713	0.653394	Fail to reject H0
ML + SMOTE	0.8912	0.025467	ML	ML(HP)	0.0036	-0.00625	0.361133	0.732757	Fail to reject H0
ML(HP)	0.8772	0.047768	ML	ML(HP) + SMOTE	-0.039	-0.04232	-1.16796	0.295482	Fail to reject H0
ML(HP) + SMOTE	0.9198	0.083829	ML + SMOTE	ML(HP)	0.014	-0.0223	0.66129	0.537677	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.0286	-0.05836	-1.05754	0.338659	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0426	-0.03606	-1.68764	0.152283	Fail to reject H0

Table C49: Results of hypothesis testing for DT algorithms on Poker dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.4968	0.002227	ML	ML + SMOTE	-0.4936	-0.00376	-152.765	2.28E-10	Reject H0
ML + SMOTE	0.9904	0.005987	ML	ML(HP)	0.0018	-0.00118	1.206271	0.281671	Fail to reject H0
ML(HP)	0.495	0.003406	ML	ML(HP) + SMOTE	-0.4766	-0.00584	-135.858	4.1E-10	Reject H0
ML(HP) + SMOTE	0.9734	0.008065	ML + SMOTE	ML(HP)	0.4954	0.002581	152.8352	2.28E-10	Reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.017	-0.00208	6.790951	0.001054	Reject H0
			ML(HP)	ML(HP) + SMOTE	-0.4784	-0.00466	-120.456	7.48E-10	Reject H0

Table C50: Results of hypothesis testing for DT algorithms on Spambase dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9034	0.013366	ML	ML + SMOTE	-0.0272	0.006959	-5.91304	0.001971	Reject H0
ML + SMOTE	0.9306	0.006406	ML	ML(HP)	-0.0022	0.001591	-0.41215	0.697307	Fail to reject H0
ML(HP)	0.9056	0.011775	ML	ML(HP) + SMOTE	-0.0264	0.007352	-3.69047	0.014139	Fail to reject H0
ML(HP) + SMOTE	0.9298	0.006013	ML + SMOTE	ML(HP)	0.025	-0.00537	3.986205	0.010466	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.0008	0.000393	0.202375	0.8476	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0242	0.005761	-3.71824	0.013737	Fail to reject H0

Table C51: Results of hypothesis testing for DT algorithms on Winequality dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.584	0.092846	ML	ML + SMOTE	-0.3908	0.079679	-10.6327	0.000127	Reject H0
ML + SMOTE	0.9748	0.013167	ML	ML(HP)	0.0236	0.030385	1.191919	0.286775	Fail to reject H0
ML(HP)	0.5604	0.062462	ML	ML(HP) + SMOTE	-0.3924	0.076739	-10.1749	0.000157	Reject H0
ML(HP) + SMOTE	0.9764	0.016107	ML + SMOTE	ML(HP)	0.4144	-0.04929	16.32914	1.57E-05	Reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.0016	-0.00294	-0.25762	0.806968	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.416	0.046354	-17.2211	1.21E-05	Reject H0

Table C52: Results of hypothesis testing for DT algorithms on Yeast dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.8746	0.059758	ML	ML + SMOTE	-0.0916	0.047214	-3.44764	0.018284	Fail to reject H0
ML + SMOTE	0.9662	0.012544	ML	ML(HP)	0.0364	-0.01543	2.281067	0.071443	Fail to reject H0
ML(HP)	0.8382	0.075184	ML	ML(HP) + SMOTE	-0.1024	0.054245	-4.05732	0.009755	Fail to reject H0
ML(HP) + SMOTE	0.977	0.005514	ML + SMOTE	ML(HP)	0.128	-0.06264	3.688184	0.014172	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.0108	0.007031	-2.58712	0.049007	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.1388	0.06967	-4.32801	0.007513	Fail to reject H0

.3.1.3 Gradient Boosting

Table C53: Results of hypothesis testing for GB algorithms on Ionosphere dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9692	0.018883	ML	ML + SMOTE	-0.0134	0.00328	-1.22577	0.274866	Fail to reject H0
ML + SMOTE	0.9826	0.015603	ML	ML(HP)	-0.004	0.007248	-0.83649	0.441014	Fail to reject H0
ML(HP)	0.9732	0.011634	ML	ML(HP) + SMOTE	-0.014	0.004427	-1.29136	0.253053	Fail to reject H0
ML(HP) + SMOTE	0.9832	0.014455	ML + SMOTE	ML(HP)	0.0094	0.003968	1.188319	0.288069	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.0006	0.001147	-0.58835	0.581869	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.01	-0.00282	-1.32376	0.242874	Fail to reject H0

Table C54: Results of hypothesis testing for GB algorithms on Page-blocks dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9654	0.063842	ML	ML + SMOTE	-0.0324	0.059442	-1.22474	0.27522	Fail to reject H0
ML + SMOTE	0.9978	0.0044	ML	ML(HP)	0	0			Fail to reject H0
ML(HP)	0.9654	0.063842	ML	ML(HP) + SMOTE	-0.0324	0.059442	-1.22474	0.27522	Fail to reject H0
ML(HP) + SMOTE	0.9978	0.0044	ML + SMOTE	ML(HP)	0.0324	-0.05944	1.224745	0.27522	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0	0			Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0324	0.059442	-1.22474	0.27522	Fail to reject H0

Table C55: Results of hypothesis testing for GB algorithms on Spambase dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.985	0.005621	ML	ML + SMOTE	-0.0026	0.004602	-1.30436	0.248924	Fail to reject H0
ML + SMOTE	0.9876	0.00102	ML	ML(HP)	-0.0032	0.001221	-4.89898	0.004478	Reject H0
ML(HP)	0.9882	0.0044	ML	ML(HP) + SMOTE	-0.0068	0.004642	-2.88384	0.034433	Fail to reject H0
ML(HP) + SMOTE	0.9918	0.00098	ML + SMOTE	ML(HP)	-0.0006	-0.00338	-0.38411	0.716689	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.0042	4E-05	-7.75476	0.00057	Reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0036	0.00342	-1.88691	0.117826	Fail to reject H0

Table C56: Results of hypothesis testing for GB algorithms on Winequality dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9046	0.050539	ML	ML + SMOTE	-0.0942	0.048939	-4.46103	0.006634	Fail to reject H0
ML + SMOTE	0.9988	0.0016	ML	ML(HP)	0.2034	-0.15487	2.919478	0.033034	Fail to reject H0
ML(HP)	0.7012	0.205413	ML	ML(HP) + SMOTE	-0.0642	0.038351	-3.30198	0.021429	Fail to reject H0
ML(HP) + SMOTE	0.9688	0.012189	ML + SMOTE	ML(HP)	0.2976	-0.20381	3.543266	0.016505	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.03	-0.01059	5.937006	0.001935	Reject H0
			ML(HP)	ML(HP) + SMOTE	-0.2676	0.193225	-3.33214	0.02073	Fail to reject H0

Table C57: Results of hypothesis testing for GB algorithms on Yeast dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9756	0.014151	ML	ML + SMOTE	-0.0218	0.013131	-3.82437	0.012318	Fail to reject H0
ML + SMOTE	0.9974	0.00102	ML	ML(HP)	0.0148	-0.01908	1.359229	0.232166	Fail to reject H0
ML(HP)	0.9608	0.033235	ML	ML(HP) + SMOTE	-0.0216	0.012984	-3.84815	0.012024	Fail to reject H0
ML(HP) + SMOTE	0.9972	0.001166	ML + SMOTE	ML(HP)	0.0366	-0.03222	2.747611	0.040425	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.0002	-0.00015	0.5	0.638299	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0364	0.032069	-2.70032	0.042767	Fail to reject H0

.3.1.4 K-Nearest Neighbors

Table C58: Results of hypothesis testing for KNN algorithms on Yeast dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9082	0.028238	ML	ML + SMOTE	-0.0588	0.003702	-4.88937	0.004516	Reject H0
ML + SMOTE	0.967	0.024536	ML	ML(HP)	0.023	-0.00931	3.475288	0.017748	Fail to reject H0
ML(HP)	0.8852	0.037552	ML	ML(HP) + SMOTE	-0.0482	0.003551	-3.64399	0.014841	Fail to reject H0
ML(HP) + SMOTE	0.9564	0.024687	ML + SMOTE	ML(HP)	0.0818	-0.01302	6.763026	0.001074	Reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.0106	-0.00015	2.657763	0.045005	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0712	0.012865	-4.87671	0.004566	Reject H0

Table C59: Results of hypothesis testing for KNN algorithms on Pageblocks dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9984	0.001625	ML	ML + SMOTE	-0.0016	0.001625	-2.41209	0.060707	Fail to reject H0
ML + SMOTE	1	0	ML	ML(HP)	0.0154	-0.03238	1.1317	0.309099	Fail to reject H0
ML(HP)	0.983	0.034	ML	ML(HP) + SMOTE	-0.0016	0.001625	-2.41209	0.060707	Fail to reject H0
ML(HP) + SMOTE	1	0	ML + SMOTE	ML(HP)	0.017	-0.034	1.224745	0.27522	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0	0			Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.017	0.034	-1.22474	0.27522	Fail to reject H0

Table C60: Results of hypothesis testing for KNN algorithms on Poker dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.8694	0.062876	ML	ML + SMOTE	-0.1294	0.061897	-5.10394	0.003758	Reject H0
ML + SMOTE	0.9988	0.00098	ML	ML(HP)	-0.0028	-0.00129	-4.66667	0.005499	Fail to reject H0
ML(HP)	0.8722	0.06417	ML	ML(HP) + SMOTE	-0.129	0.062076	-5.02538	0.004017	Reject H0
ML(HP) + SMOTE	0.9984	0.0008	ML + SMOTE	ML(HP)	0.1266	-0.06319	4.892056	0.004505	Reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.0004	0.00018	1.224745	0.27522	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.1262	0.06337	-4.81778	0.004808	Reject H0

Table C61: Results of hypothesis testing for KNN algorithms on Winequality dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.6628	0.081335	ML	ML + SMOTE	-0.3222	0.071415	-9.22043	0.000252	Reject H0
ML + SMOTE	0.985	0.00992	ML	ML(HP)	-0.019	-0.00328	-10.1079	0.000162	Reject H0
ML(HP)	0.6818	0.084615	ML	ML(HP) + SMOTE	-0.3222	0.071415	-9.22043	0.000252	Reject H0
ML(HP) + SMOTE	0.985	0.00992	ML + SMOTE	ML(HP)	0.3032	-0.0747	8.331557	0.000407	Reject H0
			ML + SMOTE	ML(HP) + SMOTE	0	0			Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.3032	0.074696	-8.33156	0.000407	Reject H0

Table C62: Results of hypothesis testing for KNN algorithms on Yeast dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.906	0.059636	ML	ML + SMOTE	-0.0824	0.051521	-3.58926	0.01572	Fail to reject H0
ML + SMOTE	0.9884	0.008114	ML	ML(HP)	-0.021	0.005521	-2.00166	0.101723	Fail to reject H0
ML(HP)	0.927	0.054115	ML	ML(HP) + SMOTE	-0.0832	0.052222	-3.60505	0.015461	Fail to reject H0
ML(HP) + SMOTE	0.9892	0.007414	ML + SMOTE	ML(HP)	0.0614	-0.046	3.011445	0.029706	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.0008	0.000701	-0.91766	0.400894	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0622	0.046701	-3.05024	0.028416	Fail to reject H0

.3.1.5 Logistic Regression

Table C63: Results of hypothesis testing for LR algorithms on Ionosphere dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.905	0.022352	ML	ML + SMOTE	-0.0418	0.003833	-5.17564	0.003538	Reject H0
ML + SMOTE	0.9468	0.018519	ML	ML(HP)	0.0206	0.00333	3.92779	0.011095	Fail to reject H0
ML(HP)	0.8844	0.019022	ML	ML(HP) + SMOTE	-0.0352	0.002139	-3.69348	0.014095	Fail to reject H0
ML(HP) + SMOTE	0.9402	0.020213	ML + SMOTE	ML(HP)	0.0624	-0.0005	8.162618	0.000448	Reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.0066	-0.00169	3.78536	0.012819	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0558	-0.00119	-6.51781	0.001271	Reject H0

Table C64: Results of hypothesis testing for LR algorithms on Page-blocks dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9778	0.007679	ML	ML + SMOTE	-0.0068	0.001725	-1.63836	0.162273	Fail to reject H0
ML + SMOTE	0.9846	0.005953	ML	ML(HP)	-0.0088	-0.00073	-5.64908	0.002413	Reject H0
ML(HP)	0.9866	0.008405	ML	ML(HP) + SMOTE	-0.0118	0.001057	-3.1582	0.025146	Fail to reject H0
ML(HP) + SMOTE	0.9896	0.006621	ML + SMOTE	ML(HP)	-0.002	-0.00245	-0.48131	0.650621	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.005	-0.00067	-2.43975	0.058672	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.003	0.001784	-0.9649	0.37891	Fail to reject H0

Table C65: Results of hypothesis testing for LR algorithms on Poker dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.301	0.085529	ML	ML + SMOTE	-0.329	0.060326	-12.3576	6.15E-05	Reject H0
ML + SMOTE	0.63	0.025203	ML	ML(HP)	0.0182	-0.00899	1.217598	0.277699	Fail to reject H0
ML(HP)	0.2828	0.094514	ML	ML(HP) + SMOTE	-0.3158	0.058985	-12.5174	5.77E-05	Reject H0
ML(HP) + SMOTE	0.6168	0.026544	ML + SMOTE	ML(HP)	0.3472	-0.06931	12.16368	6.64E-05	Reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.0132	-0.00134	6.868544	0.001	Reject H0
			ML(HP)	ML(HP) + SMOTE	-0.334	0.067971	-12.0334	6.99E-05	Reject H0

Table C66: Results of hypothesis testing for LR algorithms on Spambase dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.972	0.005657	ML	ML + SMOTE	0	0.001915	-1.1E-14	1	Fail to reject H0
ML + SMOTE	0.972	0.003742	ML	ML(HP)	0	0.000253	0	1	Fail to reject H0
ML(HP)	0.972	0.005404	ML	ML(HP) + SMOTE	-0.0006	0.001677	-0.17486	0.868047	Fail to reject H0
ML(HP) + SMOTE	0.9726	0.00398	ML + SMOTE	ML(HP)	0	-0.00166	1.13E-14	1	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.0006	-0.00024	-1.83712	0.125612	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0006	0.001424	-0.17688	0.866545	Fail to reject H0

Table C67: Results of hypothesis testing for LR algorithms on Yeast dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9412	0.035051	ML	ML + SMOTE	-0.0294	0.027768	-1.9013	0.11567	Fail to reject H0
ML + SMOTE	0.9706	0.007283	ML	ML(HP)	0.0016	-0.00436	0.608816	0.569244	Fail to reject H0
ML(HP)	0.9396	0.039414	ML	ML(HP) + SMOTE	-0.029	0.027664	-1.84947	0.123632	Fail to reject H0
ML(HP) + SMOTE	0.9702	0.007386	ML + SMOTE	ML(HP)	0.031	-0.03213	1.816628	0.128968	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.0004	-0.0001	0.816497	0.451349	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0306	0.032027	-1.77348	0.136334	Fail to reject H0

.3.1.6 Random Forest

Table C68: Results of hypothesis testing for RF algorithms on Ionosphere dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9794	0.012722	ML	ML + SMOTE	-0.0078	0.000849	-1.4428	0.208665	Fail to reject H0
ML + SMOTE	0.9872	0.011873	ML	ML(HP)	0.0016	0.000799	0.888889	0.414776	Fail to reject H0
ML(HP)	0.9778	0.011923	ML	ML(HP) + SMOTE	-0.0078	-0.0011	-1.36695	0.229894	Fail to reject H0
ML(HP) + SMOTE	0.9872	0.013819	ML + SMOTE	ML(HP)	0.0094	-5E-05	1.6487	0.160124	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0	-0.00195	0	1	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0094	-0.0019	-1.59253	0.172143	Fail to reject H0

Table C69: Results of hypothesis testing for RF algorithms on Page-blocks dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9996	0.0008	ML	ML + SMOTE	-0.0004	0.0008	-1.22474	0.27522	Fail to reject H0
ML + SMOTE	1	0	ML	ML(HP)	0.0016	-0.0032	1.224745	0.27522	Fail to reject H0
ML(HP)	0.998	0.004	ML	ML(HP) + SMOTE	-0.0004	0.0008	-1.22474	0.27522	Fail to reject H0
ML(HP) + SMOTE	1	0	ML + SMOTE	ML(HP)	0.002	-0.004	1.224745	0.27522	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0	0			Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.002	0.004	-1.22474	0.27522	Fail to reject H0

Table C70: Results of hypothesis testing for RF algorithms on Poker dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9348	0.050523	ML	ML + SMOTE	-0.065	0.050123	-3.14018	0.025661	Fail to reject H0
ML + SMOTE	0.9998	0.0004	ML	ML(HP)	-0.002	0.004486	-0.13073	0.901089	Fail to reject H0
ML(HP)	0.9368	0.046037	ML	ML(HP) + SMOTE	-0.0646	0.049323	-3.09838	0.026903	Fail to reject H0
ML(HP) + SMOTE	0.9994	0.0012	ML + SMOTE	ML(HP)	0.063	-0.04564	3.337119	0.020617	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.0004	-0.0008	1.224745	0.27522	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0626	0.044837	-3.28607	0.021808	Fail to reject H0

Table C71: Results of hypothesis testing for RF algorithms on Spambase dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.987	0.005138	ML	ML + SMOTE	-0.0048	0.004158	-2.11308	0.088276	Fail to reject H0
ML + SMOTE	0.9918	0.00098	ML	ML(HP)	0.0006	0.001209	0.981981	0.371206	Fail to reject H0
ML(HP)	0.9864	0.003929	ML	ML(HP) + SMOTE	-0.004	0.003873	-1.89264	0.116963	Fail to reject H0
ML(HP) + SMOTE	0.991	0.001265	ML + SMOTE	ML(HP)	0.0054	-0.00295	3.131641	0.025909	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.0008	-0.00029	1.680336	0.153725	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0046	0.002664	-2.86754	0.035095	Fail to reject H0

Table C72: Results of hypothesis testing for RF algorithms on Winequality dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.8954	0.052895	ML	ML + SMOTE	-0.1042	0.052405	-4.8148	0.00482	Reject H0
ML + SMOTE	0.9996	0.00049	ML	ML(HP)	0.0168	-0.01901	0.81432	0.452486	Fail to reject H0
ML(HP)	0.8786	0.071907	ML	ML(HP) + SMOTE	-0.1044	0.052495	-4.84958	0.004676	Reject H0
ML(HP) + SMOTE	0.9998	0.0004	ML + SMOTE	ML(HP)	0.121	-0.07142	4.118729	0.009185	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.0002	8.99E-05	-1.22474	0.27522	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.1212	0.071507	-4.1394	0.009002	Fail to reject H0

.3.1.7 Support Vector Machine

Table C73: Results of hypothesis testing for SVM algorithms on Pageblocks dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.995	0.00429	ML	ML + SMOTE	0.0042	-0.00226	1.623409	0.165429	Fail to reject H0
ML + SMOTE	0.9908	0.006554	ML	ML(HP)	-0.0042	0.00331	-2.09303	0.09055	Fail to reject H0
ML(HP)	0.9992	0.00098	ML	ML(HP) + SMOTE	-0.0038	0.00269	-3.1096	0.026563	Fail to reject H0
ML(HP) + SMOTE	0.9988	0.0016	ML + SMOTE	ML(HP)	-0.0084	0.005575	-2.80416	0.037808	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.008	0.004954	-3.36067	0.020093	Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	0.0004	-0.00062	0.420084	0.691866	Fail to reject H0

Table C74: Results of hypothesis testing for SVM algorithms on Poker dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9686	0.022703	ML	ML + SMOTE	-0.0314	0.022703	-3.38779	0.019508	Fail to reject H0
ML + SMOTE	1	0	ML	ML(HP)	-0.0258	0.011996	-2.10032	0.089717	Fail to reject H0
ML(HP)	0.9944	0.010707	ML	ML(HP) + SMOTE	-0.0314	0.022703	-3.38779	0.019508	Fail to reject H0
ML(HP) + SMOTE	1	0	ML + SMOTE	ML(HP)	0.0056	-0.01071	1.281137	0.256345	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0	0			Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0056	0.010707	-1.28114	0.256345	Fail to reject H0

Table C75: Results of hypothesis testing for SVM algorithms on Spambase dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.9746	0.00706	ML	ML + SMOTE	-0.0024	0.003213	-0.61478	0.565598	Fail to reject H0
ML + SMOTE	0.977	0.003847	ML	ML(HP)	-0.0056	0.000384	-7.39579	0.000711	Reject H0
ML(HP)	0.9802	0.006675	ML	ML(HP) + SMOTE	-0.0078	0.005563	-2.32793	0.067385	Fail to reject H0
ML(HP) + SMOTE	0.9824	0.001497	ML + SMOTE	ML(HP)	-0.0032	-0.00283	-0.82919	0.444765	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	-0.0054	0.00235	-5.29514	0.003205	Reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0022	0.005179	-0.67915	0.527202	Fail to reject H0

Table C76: Results of hypothesis testing for SVM algorithms on Winequality dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.7542	0.034597	ML	ML + SMOTE	-0.2374	0.023344	-17.1729	1.23E-05	Reject H0
ML + SMOTE	0.9916	0.011253	ML	ML(HP)	0.0132	-0.11718	0.203681	0.846633	Fail to reject H0
ML(HP)	0.741	0.151777	ML	ML(HP) + SMOTE	-0.1824	0.014195	-11.8991	7.39E-05	Reject H0
ML(HP) + SMOTE	0.9366	0.020402	ML + SMOTE	ML(HP)	0.2506	-0.14052	3.882167	0.011616	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0.055	-0.00915	13.28745	4.32E-05	Reject H0
			ML(HP)	ML(HP) + SMOTE	-0.1956	0.131376	-2.88232	0.034494	Fail to reject H0

Table C77: Results of hypothesis testing for SVM algorithms on Yeast dataset. ML– Machine learning; HP– Hyper-parameter; Std– Standard deviation.

Algorithm	Mean	Std	Conditions		Mean difference	Std difference	t-value	p-value	Hypothesis
ML	0.971	0.023126	ML	ML + SMOTE	-0.0238	0.020066	-2.6627	0.044739	Fail to reject H0
ML + SMOTE	0.9948	0.003059	ML	ML(HP)	0	0			Fail to reject H0
ML(HP)	0.971	0.023126	ML	ML(HP) + SMOTE	-0.0238	0.020066	-2.6627	0.044739	Fail to reject H0
ML(HP) + SMOTE	0.9948	0.003059	ML + SMOTE	ML(HP)	0.0238	-0.02007	2.662697	0.044739	Fail to reject H0
			ML + SMOTE	ML(HP) + SMOTE	0	0			Fail to reject H0
			ML(HP)	ML(HP) + SMOTE	-0.0238	0.020066	-2.6627	0.044739	Fail to reject H0

.4 Appendix D

.4.1 Study One

.4.1.1 Confusion Matrix

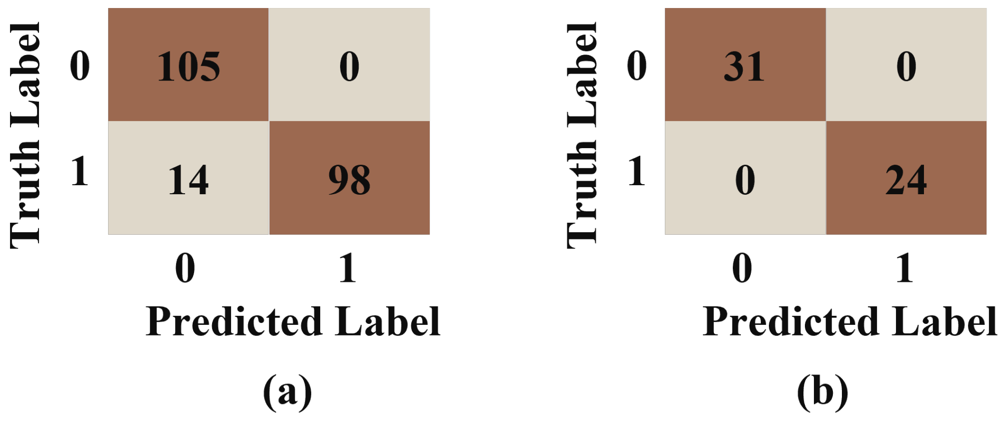


Figure 1: Confusion matrices of modified VGG16 on (a) train and (b) test set.

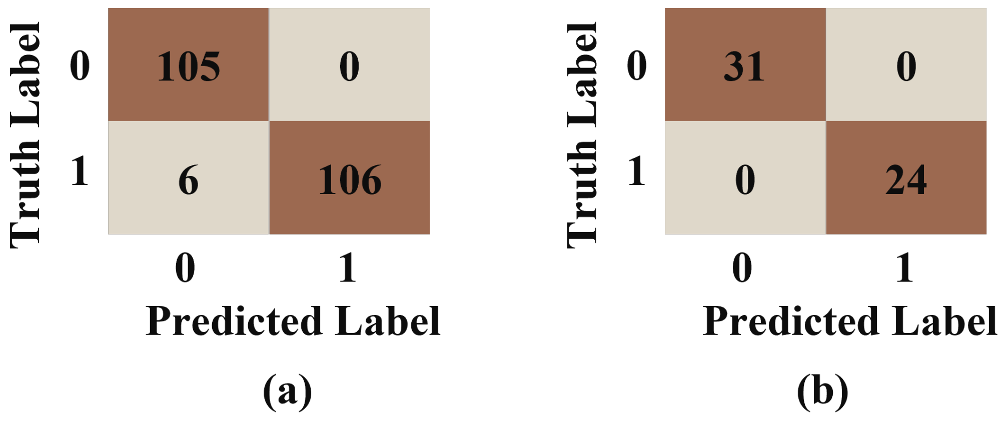


Figure 2: Confusion matrices of ROIN on (a) train and (b) test set.

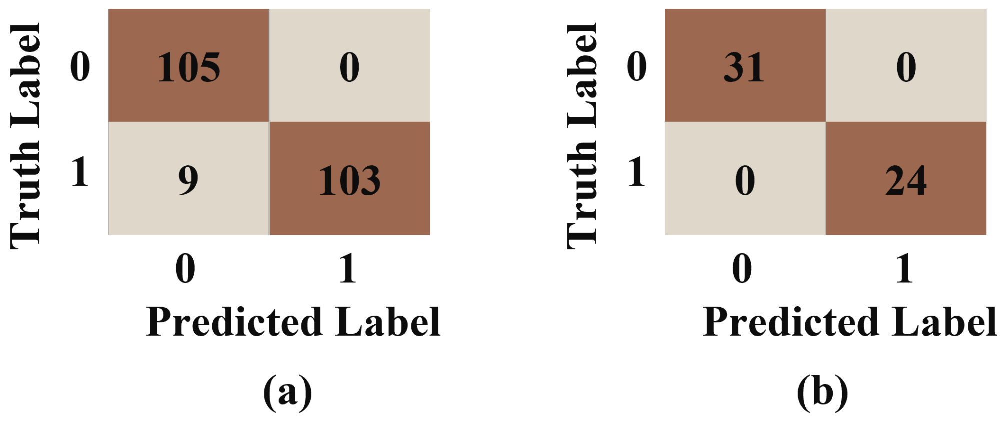


Figure 3: Confusion matrices of ROIHEN on (a) train and (b) test set.

.4.2 Study Two

.4.2.1 Confusion Matrix

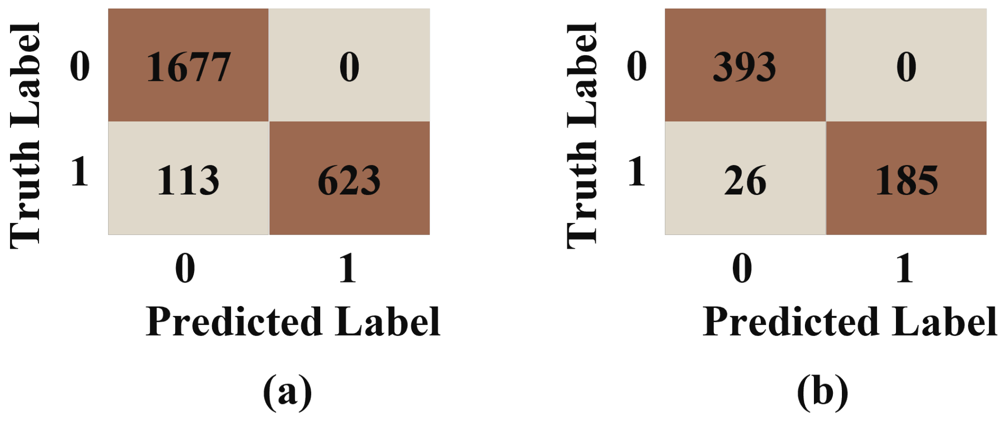


Figure 4: Confusion matrices of ROIN on (a) train and (b) test set.

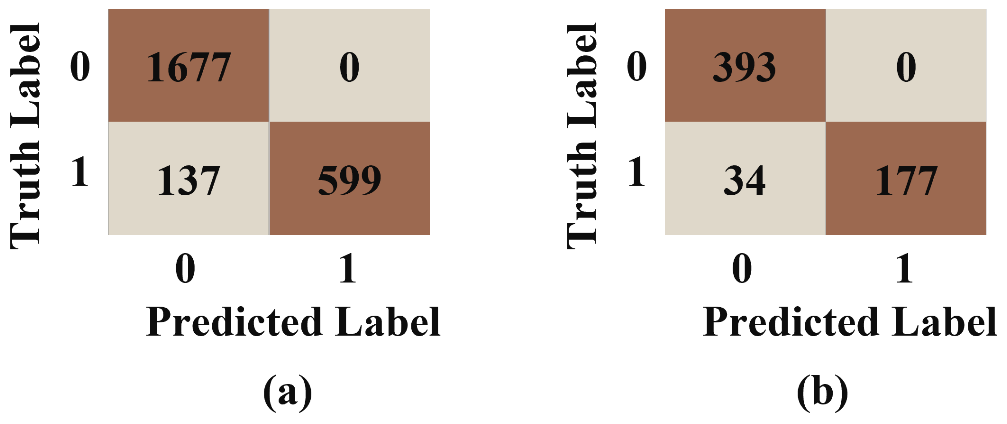


Figure 5: Confusion matrices of ROIHEN on (a) train and (b) test set.

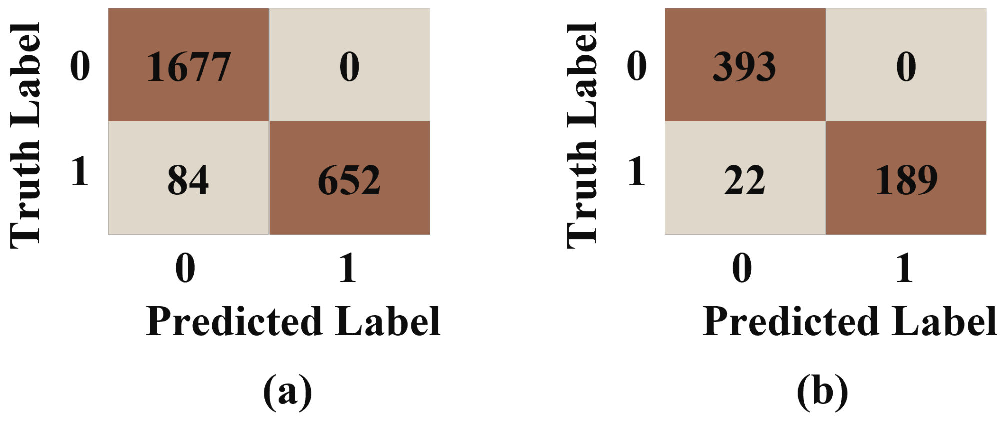


Figure 6: Confusion matrices of ROIHEDEN on (a) train and (b) test set.