UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

IMBALANCED LEARNING WITH PARAMETRIC LINEAR PROGRAMMING SUPPORT VECTOR MACHINE FOR WEATHER DATA APPLICATION

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

ELAHEH JAFARIGOL
Norman, Oklahoma
2019

Downloaded from SHAREOK. Persistent Handle URL: https://hdl.handle.net/11244/319660

IMBALANCED LEARNING WITH PARAMETRIC LINEAR PROGRAMMING

SUPPORT VECTOR MACHINE FOR WEATHER DATA APPLICATION

A THESIS APPROVED FOR THE
SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

BY

Dr. Theodore Trafalis, Chair

Dr. Shima Mohebbi

Dr. Michael Richman

© Copyright by ELAHEH JAFARIGOL 2019
All Rights Reserved.

To my best friend and true love, Hadi and

to all the little girls in corners of the world who pursue their dreams.

Acknowledgment

First, I would like to express my sincere gratitude to my thesis advisor Dr. Theodore Trafalis for his unceasing support, motivation and immense knowledge. Dr. Trafalis was always available whenever I had questions or ran into problems for my thesis. I could not have imagined a better advisor and mentor for my master's thesis.

I would like to acknowledge my committee members Dr. Michael Richman from School of Meteorology and Dr. Shima Mohebbi from the School of Industrial and Systems Engineering. I am grateful for their insightful comments and input towards the completion of this work.

A very special gratitude goes out to Dr. Guifu Zhang from School of Meteorology as this project would not have started without his support.

Finally, I would like to thank my beloved husband, Hadi Saeidi Manesh for his endless love, continuous encouragement and support throughout this venture. This accomplishment would not have been possible without him.

Table of Contents

Acknowledgment iv

Table of Contents vi

List of Figures vii

List of Tables viii

Abstract ix

1 Introduction 1

1.1 Imbalanced learning 2
1.2 Related work and existing models 4
1.3 Literature review 8
2 Methodology 12

2.1 Missing value imputation 12
2.2 Feature selection 13
2.3 Feature extraction 20
2.4 Resampling Techniques for imbalanced learning 22
2.5 Classification – Support Vector Machines 24
2.5.1 Linearly Separable Data – Hard margin Support Vector Machines 25
2.5.2 Linearly non-separable data – Soft margin Support Vector Machines 27
2.6 Defining norms 29
2.7 Parametric modeling 36
2.8 Evaluation metrics 42
3 Experiment and results 47

3.1 Data preprocessing 47
3.1.1 Missing value imputation 47
3.1.2 Feature selection 52
3.1.3 Feature Extraction 57
3.2 Model selection 59
3.3 Classification and parameter tuning 60
4 Conclusion and future work 66

References 68

List of Figures

Figure 2-1: Violin plot of scaled data 16

Figure 2-2: Correlation Matrix of the scaled data 18

Figure 2-3: Linearly separable SVM 26

Figure 2-4: Non-linearly separable SVM 27

Figure 2-5: Kernel SVM 35

Figure 2-6: Receiver Operating Characteristic and Area Under the Curve 45

Figure 2-7: K-fold Cross Validation 46

Figure 3-1: Distribution of data in the minority and majority class in the selected locations 50

Figure 3-2: Missing values in the data set before and after dropping the features with 30% or higher missing values 52

Figure 3-3: Joint plot of highly correlated features 53

Figure 3-4: Confusion matrix – RF classification 53

Figure 3-5: Confusion matrix – RFE-RF vs. RFE-SVM classification 54

Figure 3-6: RFE-RF with 5-fold cross-validation 55

Figure 3-7: Feature importance plot 56

Figure 3-8: Optimal number of principal components and the correlation matrix 58

Figure 3-9: Confusion matrix - RF classification on the new feature set after PCA 58

Figure 3-10: Model comparison 60

Figure 3-11: Confusion matrices of modified SVM classification 63

Figure 3-12: ROC curves 63

List of Tables

Table 1-1: General approaches to imbalanced learning 3

Table 2-1: Kernel functions 34

Table 2-2: Confusion matrix 42

Table 3-1: Numerical and categorical features of the data set 47

Table 3-2: Observation locations and sampled data sets 48

Table 3-3: Sampled data sets 49

Table 3-4: Features of the data set and the percentage of missing values 50

Table 3-5: Results – RF classification 53

Table 3-6: Results – RFE-RF vs. RFE-SVM classification 54

Table 3-7: Results – Comparison of RF classification of the feature set 1 and 2 57

Table 3-8: Results - RF classification on the new feature set after PCA 58

Table 3-9: Model comparison 59

Table 3-10: Parametric modeling values 61

Table 3-11: Results – Standard Kernelized SVM 61

Table 3-12: Results - Kernelized SVM with modified objective function classification 61

Table 3-13: Cross-Validation 64

Table 3-14: Cross-Validation with randomness 65

Abstract

Learning from imbalanced data sets is one of the aspects of predictive modeling and machine learning that has taken a lot of attention in the last decade. Multiple research projects have been carried out to adjust the existing algorithms for accurate predictions of both classes. The model proposed in this thesis is a linear Support Vector Machine model with L_1-norm objective function with applications on weather data collected from the Bureau of Meteorology system in Australia. Apart from model selection and modifications we have also introduced a parametric modeling algorithm based on a novel parametric simplex approach for parameter tuning of Support Vector Machine. The combination of the two proposed approaches has yielded a significant improvement in predicting the minority class and decrease the model's bias towards the majority class as is seen in most machine learning algorithms.

1 Introduction

In recent years, the use of machine learning, and data mining techniques have gained a lot of attention from the data analytics society. The generality and functionality of these techniques have made them applicable in vast areas of science and technology. Meteorology is one of the fields that has greatly benefitted from these techniques. Machine learning algorithms such as Support Vector Machines and Artificial Neural Networks have been used to classify large amounts of data and extract knowledge from historical data to be used in predicting future data.

This work is organized into four chapters. In the first chapter, we will go over the problem of imbalanced learning, popular techniques in supervised learning and a literature review of the related work.

In chapter 2, the methodology of this work is discussed. The first part of chapter 2 focuses on preprocessing techniques, feature selection, and feature extraction. After that, we will go over the theory of Support Vector Machines and parametric modeling as a replacement for a grid search for finding the best regularization parameter. And a new L_1-norm SVM with the kernel is introduced to be used for classifying imbalanced data sets. Finally, the evaluation metrics used for imbalanced learning are discussed.

In chapter 3, the results obtained from implementing the algorithms discussed in the chapter is presented. Finally, chapter 4 includes the conclusion and some of our ideas for future work.

1.1 Imbalanced learning

Imbalanced learning has taken a lot of attention in recent years from industry to academia. Imbalanced learning is the art of retaining knowledge from a data set where the number of instances in one class called the majority class is significantly larger than the number of instances in the other class known as the minority class. The minority class consists of rare cases that are more important from the learning perspective. The main concern in imbalanced learning is that most classifiers are biased towards the majority class, and despite having high accuracy score, they fail to classify the instances in the minority class correctly, and the misclassification error of the minority class is often ignored [1]. From the ratio of the minority to the majority class, imbalanced learning problems are categorized into three sections:

- Marginally imbalanced: the imbalance ratio is 2:1
- Modestly imbalanced: the imbalance ratio is 10:1
- Extremely imbalanced: the imbalance ratio is 1000:1

Generally, class imbalance refers to the relative proportion of instances belonging to each class; however, the absolute number of examples available for learning is significant. Based on the type of data we are dealing with the issue of imbalanced learning can rise at different levels.

- Problem definition issues: This problem is caused by lack of enough information to accurately define the learning problem.
- Data issues: The problem of an absolute rarity occurs where there are not enough examples associated with one or more classes to effectively learn the class.

- Algorithm issues: This problem is the result of inadequacies in a learning algorithm that may cause poor performance.

Table 1-1 briefly describes the general approaches towards the different types of imbalanced learning problems mentioned before.

Table 1-1: General approaches to imbalanced learning

Problem	Approach
Problem definition issues	Using suitable evaluation metrics
	Redefining the problem
Data issues	Dynamic learning
	Exploring different sampling methods
	Using various information procurement methods
Algorithm issues	Avoiding greed and recursive partitioning in search methods
	Using metrics designed for imbalanced learning in search methods
	Developing biases that are more suitable for imbalanced data
	Modifying algorithms that implicitly or explicitly contribute to learning rare cases
	Developing models that only focus on the minority class

The problem definition issue is mostly concerned with the evaluation metrics used to assess the classifier. Accuracy and error rate which are two commonly used evaluation metrics, usually fail to represent misclassification in the minority class. These metrics perform reasonably well only when the respective error of both classes have uniform cost. However, with datasets that suffer from the imbalanced issue, the costs of errors are not symmetric.

A lack of adequate training data causes data issues, that can result in imbalanced data. When there are fewer instances in the minority class, they are more likely to be selected in the sample, and the class will be misrepresented in the model, which leads to higher misclassification rate for the minority class.

Another problem with data is when noise exists in the data. Generally, having a robust model that can handle noisy data is challenging. This problem is magnified with

imbalanced data as noise can affect the rare instances more severely and makes it more difficult to identify and correctly label those instances [2][3].

Algorithm issues focus on the model's failure in reaching optimization while learning the criteria necessary for classification. This issue is concerned with ample focus on accuracy as an evaluation metric, while metrics that evaluate the performance of the classifier for each class can provide more insight and avoid misclassification in each class regardless of the size of the class [4].

The problem with the rare cases is that their true nature is not known; hence they are difficult to identify. Training classifiers for imbalanced data is a critical issue in machine learning. Throughout the years, researches have done extensive work to resolve this matter, and have proposed different solutions. Multiple machine learning algorithms have been used to classify the data concerning both cases accurately, but none of these approaches has reached optimization yet, and the need for more powerful algorithms still exists.

1.2 Related work and existing models

All research projects start with learning about the nature of the problem and characteristics which make it unique. Learning about the problem and the literature leads us toward the problem statement and helps developing ideas to address the issue.

As science and technology evolve, we are exposed to larger and more complex datasets. It is not always easy to interpret the patterns and extract knowledge from the data, so the demand for high performance and more accurate algorithms are not lessening in the foreseeable future. Machine learning is the art of gaining knowledge and discovering patterns from data, and there are many studies on ways to improve the existing algorithms.

In machine learning, there is no best model and model selection is based on the type of data we are using. To start the learning process, we must split the data set into a training, and testing dataset. The goal is to train the model based on the training set and predict or classify the output variable in the test data. Based on the knowledge we have about the output variable in the historical data, we have three approaches towards learning.

- Supervised learning: In supervised learning we know the class each data point belongs to before running the algorithm, known as the label, and we use that to evaluate the performance of the model.
- Unsupervised learning: Unlike supervised learning, the output variable is not known in advance, and we try to find the patterns in the data.
- Semi-supervised learning: this is the most common type of data in the real world, which means some of the observations are labeled, and some of them are not.

The models discussed in this work are commonly used for supervised learning. The goal is to predict new data based on historical input data which is known as predictive modeling. In predicting modeling, we try to minimize the error by making the most accurate predictions, while being able to explain the behavior of the model. These predictive models are used for classification, meaning that we are trying to develop a concise model of the class label distribution using the features in the data, then use that model to predict the output value also known as a label for the instances in the test data. In other words, a classification model uses the known data to train the model and predict the output value for the unknown instances in the test data. An overview of the state-of-the-art algorithms is presented in this chapter as follows [5].

1. Logistic Regression (LR): Linear/Logistic Regression is probably the most well-known and widely used machine learning algorithm. Linear regression is used for predicting values that are in a range, but logistic regression is appropriate when we are trying to predict categorical output values such as binary classification. Logistic regression is presented by a non-linear function, and the data is classified based on the features correlated with the output variable [6].

2. Support Vector Machine (SVM): SVMs are commonly used for classifying large data sets. The data is classified based on its location on either side of a hyperplane, which splits the input variable space. The separating hyperplane is not unique; however, the best hyperplane is the one that maximizes the margin of separation while minimizing the misclassification error. SVM is the main algorithm used in this work for classification, and we will discuss the relevant concepts in more detail in section 2.5.

3. K-Nearest Neighbors (KNN): KNN is a simple yet efficient algorithm which uses the whole dataset for classification. To classify a new data point, KNN uses the data points closer to the designated point based on their Euclidean distance. Then it summarizes their output values and assigns the result as the label of the new data point. In KNN, training, and testing is combined in one step which increases the effectiveness of the model.

4. Naïve Bayes: The first assumption in this method is that all the data points are independent of one another, which is unrealistic, but this technique uses this assumption to predict new data effectively. In this method, the training data is used to calculate the probability of each class and the conditional probability of each class for a given data point. These two pieces of information are used to predict the class of new data points.

5. Gaussian Naïve Bayes: It is a modification of Naïve Bayes method, except that for the input data in $\mathbb{R}$ (Real values), a Gaussian distribution is assumed, which makes calculating the probabilities easier.

6. Perceptron: Perceptron algorithm is one of the oldest machine learning algorithms. In this method, we need to associate a weight to the data points and define a threshold, known as bias. The weights and the threshold are extracted from the data. The weighted sum of the input data is calculated for predicting the output value. The label is one if the sum is greater than the designated threshold, and zero otherwise. In the Perceptron algorithm, the goal is to find the set of weights that best classifies the data.

7. Stochastic Gradient Descent: Stochastic Gradient Descent method is based on minimizing the misclassification error by predicting each point in the training set and calculating the error. Then the model is optimized, so the error is minimized for the next prediction.

8. Decision Tree: Decision Tree is a classification algorithm that splits the data set into smaller subsets to predict the output value of the test data. The conditions by which the data is split are called leaves, and the decision is known as a branch. The data is split until we have reached the depth of the tree and no further split is possible. Decision Tree is a fast and simple algorithm in which the process of classification and inquiries made are clear.

9. Random Forest: Random Forest is a powerful ensemble method, which is an aggregation of less accurate predictive models to create a better model. This model is used for regression or classification. In random forest classification, decision trees are used to

introduce randomness when selecting the suboptimal splits, and the goal is to aggregate as many uncorrelated trees as possible and improve the accuracy at each step.

10. Linear Discriminant Analysis: This method is used for both binary and multi-class classification, and it uses statistical parameters such as the mean value of each class and the overall variance of the classes to calculate a discriminant value for each class and predict the output value of the class. The data is assumed to follow a Gaussian distribution and removing the outliers is an important step since it can affect the variance significantly [7].

None of these methods is superior over the other ones. So, for a given dataset, multiple machine learning algorithms must be used, and the most suitable model is the one with the most accurate predictions based on its application on real-world data. Among all areas that have benefited from advances in machine learning, weather applications in machine learning have been one of the most popular fields, that has been growing extensively in the past few decades.

1.3 Literature review

The use of machine learning algorithms in weather applications was initiated to improve the lead times to important events such as thunderstorms or tornadoes. In the research carried out by Marzban and Stumpf [8] the MDA- Mesocyclone Detection Algorithm attributes based on Doppler radar observations were used to train artificial neural network algorithms and predict tornadic events. Later Lakshmanan combined MDA data with near storm environment (NSE) data which improved the results marginally [9]. In recent years, other data collection methods based on observations made by polarimetric radar have become available. The dual polarized polarimetric radar observations provide a

2-dimensional snapshot of the data that can provide more insight into weather phenomena and new advances specially in phased array radar is required to significantly imporve storm and rain detection algorithms [10] [11] [12] [13][14].

Since weather events such as rain are considered less common in some locations, the data could be imbalanced, and most machine learning algorithms are biased towards the majority class. So, the interest in machine learning algorithms that can be used with imbalanced data has grown significantly. Different algorithms such as ANNs and SVM were used to improve the accuracy of predictions in the minority class [15]. The problem of imbalanced data has attracted a lot of attention from the data mining community. In a paper published by He and Garcia in 2009, they have explored different sampling methods to address the data-related issues in imbalanced learning. Cost-sensitive methods have also been studied to address the issues regarding the problem definition. Kernel-based learning methods such as Support Vector Machine have also been investigated as an algorithmic approach [16]. An important issue that arises with imbalanced learning is that the standard evaluation metrics do not provide an appropriate representative of both classes and they fail to assess the performance of the classifier in each class. Shaza et al. [17] have provided a broad review of different evaluation metrics developed for imbalanced learning such as F-measure and Geometric mean which are thoroughly explained in section 2.8. In addition, convenient visualization tools such as Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) as a means of performance assessment are discussed.

To explore the distinctive features of imbalanced data sets, Lopez et al. [18] did extensive work on using the intrinsic characteristics for classification and the issues related to these characteristics. In this paper different experiments were carried out to investigate

the approaches suitable for imbalanced data in the presence of small disjuncts, the absence of density and enough information in the training data, the presence of noisy data, etc.

SVM is a popular approach to imbalanced learning. Trafalis et al. [19] explored the use of SVMs for classifying imbalanced data, and they found that SVM can out-perform ANNs for imbalanced learning. They have also investigated modified SVMs such as Bayesian SVM and SVM-RFE with threshold adjustment on SVM. These algorithms proved to be more accurate and performed better in comparison with previous work. Throughout the years, various changes have been made to SVM algorithm to make it more efficient with imbalanced data. A modification to the objective function and using different norms is a simplified approach as it is discussed by Zhu et al. [20]. In this paper, L_1-norm SVM is introduced as a replacement for quadratic SVM in case of redundant and noisy data. This approach provides a linear programming problem which can be solved using existing software. Zhou et al. also investigated L1-norm SVMs [21]. In this work linear and kernelized Support Vector Machines with L1-norm objective function is introduced and simulated to be compared with quadratic SVMs and the results of the experiments are presented [22].

In another paper by Askan and Sayin [23], a multi-objective approach is introduced which incorporates an individual objective function for the positive and negative error sums of minority and majority class. This three-objective optimization model with the L1-norm SVM approach presents a significant improvement in the performance of the classifier. Another work that implements the principles of multi-objective optimization in classification is carried out by Paolo Soda [24]. He proposed using an algorithm that can compare and choose between an algorithm trained on the skewed data set and one that is

expected to be suitable for imbalanced learning. This paper shows that accuracy of the classifier and the geometric mean are improved with their proposed approach. In another paper published by Suttorp and Igel [25], a multi-objective SVM approach is represented which investigates the trade-offs between the set of Pareto-optimal points and finds the optimal solution from that set [26]. This approach is relatively new as many multi-objective problems are solved through scalarization. The model is implemented using the pedestrian detection problem, which is a real-life problem. This approach has had promising results that were evaluated and visualized with a ROC curve.

2 Methodology

Data preprocessing is the initial step in machine learning. Some of the issues that need to be addressed before any further analysis is making sure that the data is clean, without noise, missing values and it is scaled. Although data analysts are continually trying to improve the robustness of machine learning algorithms to be capable of high performance in the presence of missing values or noise, the quality of the results is still affected by the input data.

2.1 Missing value imputation

Preliminary analysis of the data reveals that there are missing values in the data set and for the first step we need to address this issue and impute the missing values.

Having missing values is common in weather data, and there are a variety of methods we can use to handle this issue, some of which are more complicated than the others. To handle missing values, knowing the reason behind having missing values in the dataset is helpful and based on the randomness of missing values we have several categories:

- Missingness completely at random
- Missingness at random
- Missingness that depends on unobserved predictors
- Missingness that depends on the missing value itself

Among the four types, the last two are more difficult to handle because the replaced value might not represent the actual observation. The easiest way of handling missing value is discarding features with the missing value. Removing incomplete features is not the best

approach because the number of complete cases might be too few. It is also possible that the discarded values include valuable information and removing them might bias the results.

Instead of deleting features with a missing value, we can impute the missing values, which is replacing the missing data with a new value. Various imputation methods can retain the dataset and use the complete data set [27].

Although removing the features with missing values is the obvious solution, it can result in losing valuable information. So, we can use imputation methods to replace the missing values with new data systematically. Imputing missing values allows us to consider more features rather than removing all the observations with missing values. Imputing missing values keeps the full data set and avoids biased results. However, it can alter the results to some extent too. After dropping the features with the highest percentage of missing values, we decided to impute the remaining missing values using mean imputation technique. In this method, which is one of the most accessible and commonly used methods, the missing value in each feature is filled in with their respective mean of the observed values. This method is implemented in Python. It is worth mentioning that imputing the missing values is an important step in data preprocessing as it is the input for other steps. There are more advanced imputation methods from linear correlation to Support Vector Regression.

2.2 Feature selection

The next step in preprocessing the data is feature selection. In recent years, development of technology has resulted in an exponential growth in the amount of data stored, in terms of size and dimensionality. Hence, storing, managing and analyzing large

scale problems is an increasing challenge. In the last decade, many machine learning algorithms have been developed to help remove irrelevant and redundant data and extract features that can contribute most to understanding the patterns and knowledge in the data. Feature selection methods provide a subset of the full-size data, in which only the relevant features are selected [28] [29].

Dimensionality reduction approaches are among the most frequently used techniques in machine learning. These techniques can be divided into two categories, feature selection, and feature extraction. Learning from a smaller subset not only increases the learning speed, and makes the process less computationally expensive, it can lead to better performance, with improved learning accuracy and the model is more interpretable.

Apart from feature selection, feature extraction techniques such as Principal Component Analysis try to reduce the dimensionality by creating a new set of features that can capture the variations in the data and reduce the dimensionality without compromising the performance of the classification algorithms.

Both feature selection and feature extraction techniques are essential steps in preparing the data for classification. They can enhance the performance of the classifiers and decrease the computational complexity which reduces the time and storage required to build and run the model.

In general, feature selection methods are divided into three categories:

- Filter method: This approach is a preprocessing method that ranks the variables based on their correlation coefficient. According to the filter method, a useful feature is one that is highly correlated with the class, but uncorrelated with other features. Such methods perform independently of machine learning algorithms.

- Wrapper methods: This approach uses a machine learning algorithm to rank the features based on their importance to the model. Then the classifier is validated using Cross-validation, and the subset that has the highest accuracy is selected.
- Embedded methods: Embedded methods perform feature selection as part of the learning process, without splitting the data into train/test set.

Before proceeding with future selection and feature extraction techniques, the data must be scaled. Scaling is necessary for preparing the data because we can avoid having features in different numeric ranges, which can affect the classifier if some of the features are in a significantly higher numerical range and cause problems in the classification process. Scikit-learn preprocessing package in python is used to scale the data to the range of [-1, 1] before further analysis. The same method of scaling is used for both training and test set. After scaling the data, having a visual understanding of the data and how it is dispersed can be useful in making decisions in the next steps. Here, we have used violin plots to show how the data is distributed in each class. It is essential to make sure that the data is scaled or normalized before plotting.

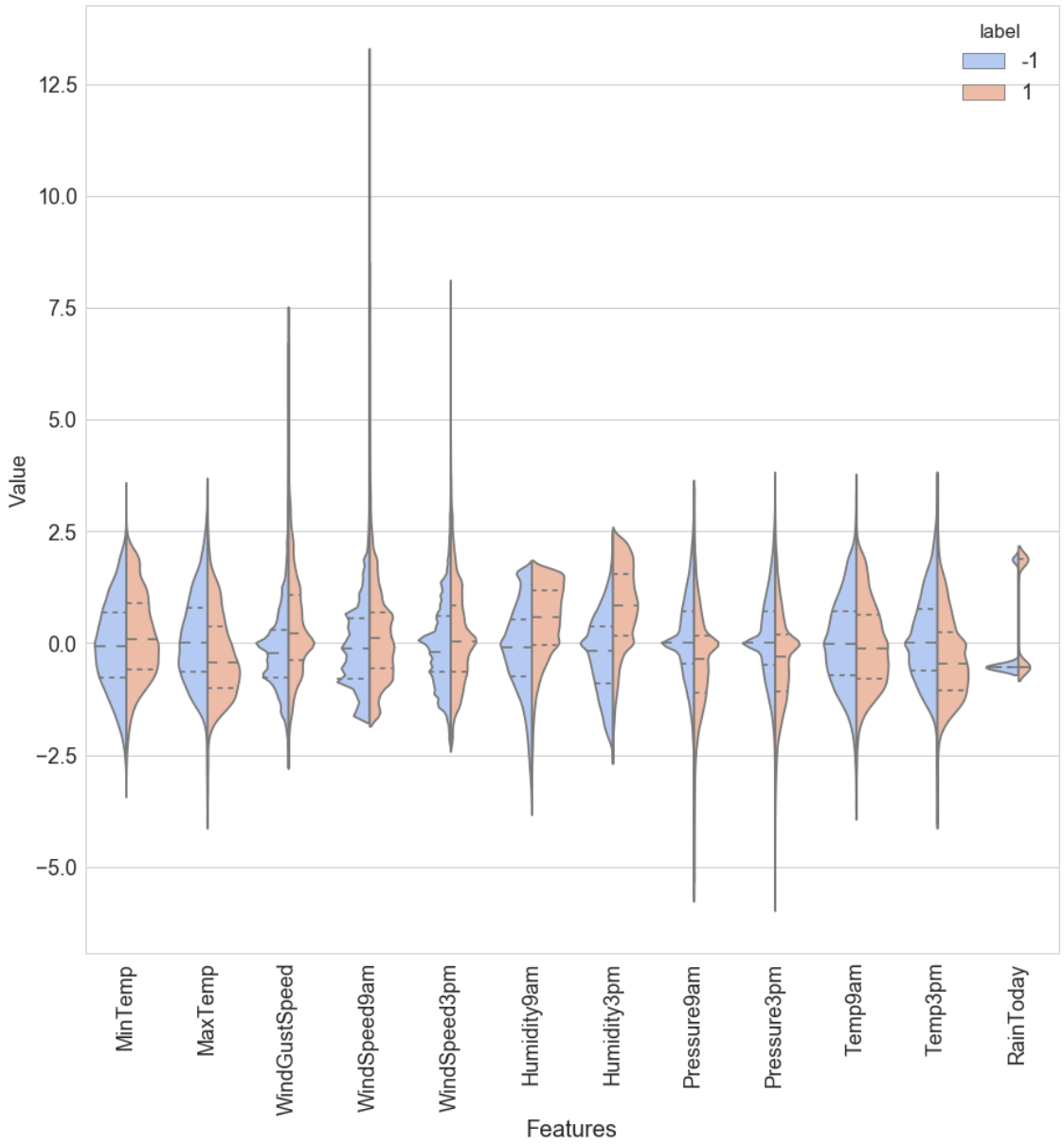


Figure 2-1: Violin plot of scaled data

As it is shown in Figure 2-1, the features that have a significant difference in the median of the two classes potentially provide more information for classification. However, more accurate methods are needed to find the best feature set for classification.

Some of the popular feature selection methods in the literature are described below:

1. Feature selection with correlation

The goal of feature selection is to use an algorithm that can identify and select the features that are not correlated with each other yet are predictive of the class so if a variable is independent of the class, it is considered irrelevant and it can be discarded. This task is accomplished by evaluating the inner feature correlation in the data set and their contribution to the classification model and determine its worth.

Pearson correlation coefficient is one of the most straightforward criteria for evaluating the interdependencies in the data. If x_i is the i^{th} observation in the data set and y is the class label, the Pearson correlation coefficient is defined as:

$$R_{(i)} = \frac{cov(x_i, y)}{\sqrt{var(x_i)var(y)}} \qquad 2.1$$

In this equation, *cov* designates the covariance and *var* is the variance. The complete mathematical formulation is given by:

$$R_{(i)} = \frac{\sum_{k=1}^{m}(x_{k,i} - \bar{x}_i)(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{m}(x_{k,i} - \bar{x}_i)^2 \sum_{k=1}^{m}(y_k - \bar{y})^2}} \qquad 2.2$$

For implementation, a correlation map is a visual tool for demonstration of the correlation between the columns. As it is shown in Figure 2-2 the highest score on the map is 1.

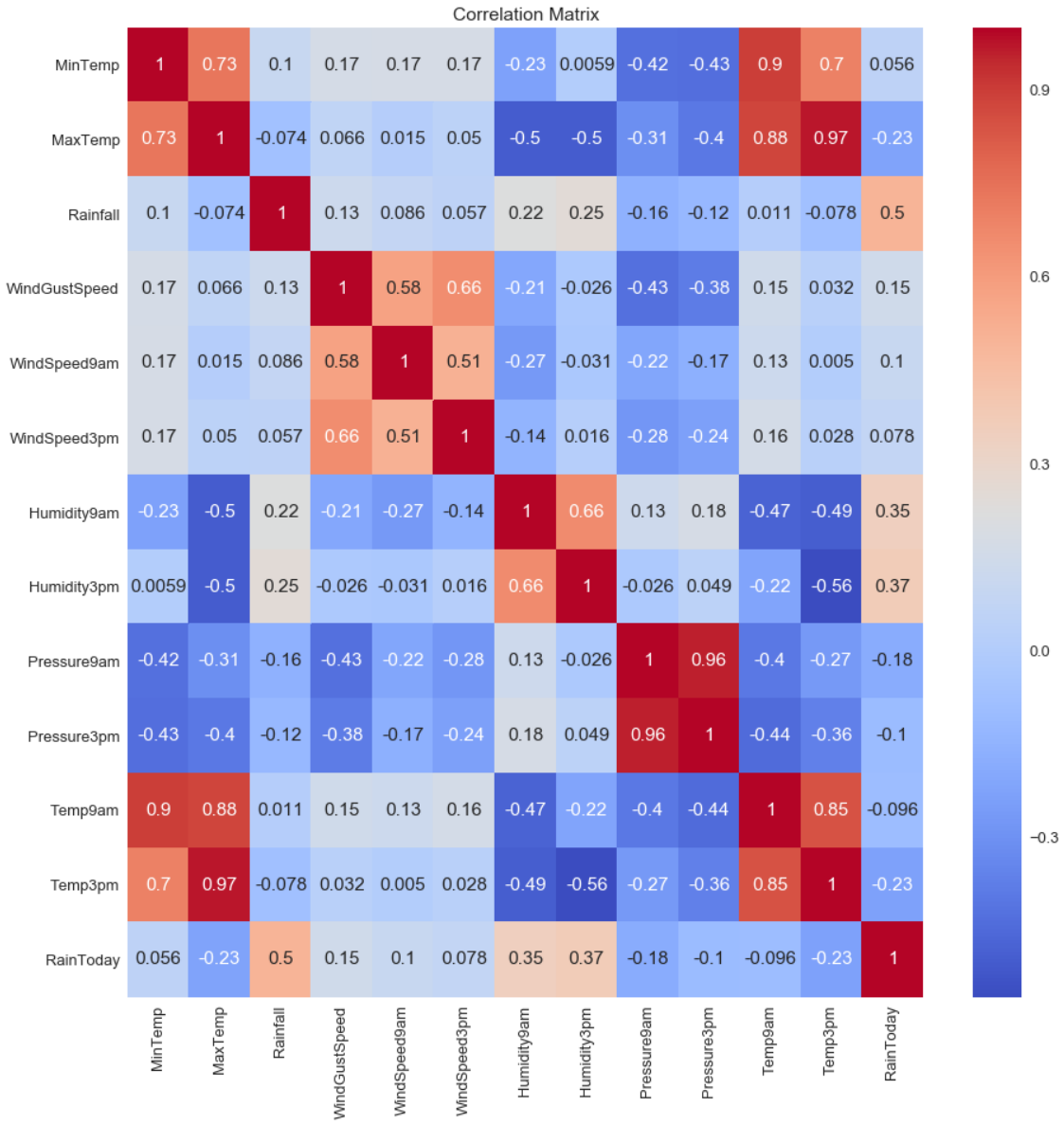


Figure 2-2: Correlation Matrix of the scaled data

In this algorithm, highly correlated features are selected in each pair and one of them is dropped. Then we will use a classification algorithm to see how the performance of the classifier changes with the new subset. At this step, looking closely at the correlation between the highly correlated set of features is useful [30]. Although feature selection with

correlation coefficient can provide valuable insight into the data set, it relies mostly on tried and error and the results are not optimized.

2. Recursive Feature Elimination

A more recently developed method of feature selection is Recursive Feature Elimination (RFE). RFE is a wrapper method that starts with all the features in the data set. RFE is designed to find the best features by fitting the model and removing the least discriminant feature until the model reaches optimization or the desired number of features remains.

To implement RFE, we need to split the data into train/test sets. A machine learning algorithm such as SVM or Random Forest is required to calculate the feature relevance weights. The absolute value of the relevance weights is calculated to rank the features according to their importance; the lowest weights are the least important and the highest ones are the most important features. After dropping the least important feature, the model is trained on the new subset using the selected machine learning algorithm. The accuracy of the classifier is used to evaluate the model's performance. This process is repeated until all the desired number of features is obtained.

In the fitting part, a classification model is required to analyze how the accuracy measure improves after a new subset is created. The most common versions of RFE are recursive feature elimination with a linear SVM at its core or random forest. Both classifiers provide good learning performance, high accuracy and in many cases an acceptable computational time.

3. Recursive Feature Elimination with Cross-Validation

In most cases the preferred number of features is not determined in advance so by combining RFE with Cross Validation, a new subset is selected at each iteration. After fitting the model, we can compare the cross-validated scores and decide on the best number of features that yields higher classification accuracy. Implementing RFE as a preprocessing technique before classification is a useful tool to recognize the inter-correlated features and the patterns in the data. RFE can efficiently remove non-relevant and redundant data. However, there are a few drawbacks to this method. Computational intensity is an important issue to consider. RFE requires the user to select the number of features before training the data. To address this issue, other feature selection methods have been used to reach the optimal data set.

4. Tree-based feature selection (Random Forest Feature importance)

Random forest is a fast-growing algorithm used for large-scale data analysis. Random forest applies to high dimensional data with nonlinear interactions among the features. It is also capable of ranking the variables to assess their importance in predicting the desired value, by finding the features that contribute most to the predictor variable.

2.3 Feature extraction

Principal Component Analysis is one of the most popular feature extraction techniques designed to find a lower dimensional basis from the original data set that can capture most of the variance in the data. The lower dimensional basis is constructed from the linear transformation of the correlated features into fewer uncorrelated ones, known as principal components.

What we try to do in PCA is finding the lower dimensional surfaces, onto which to project the data to minimize the projection error. The lower dimensional surfaces are a set

of vectors. In general, the best surface is the one that can satisfy two conditions. First, maximizing the projected variance, and the second, minimizing the mean squared error. These conditions must be true in all dimensionalities. The vectors in PCA are called principal components, and they are the dimensions or directions that capture the most variance. For example, the first principal component is the direction that has the largest projected variance. The second principal component is the vector perpendicular to the first component and captures the second largest projected variance. All the principal components are the directions that satisfy both conditions mentioned above and yield an optimal subspace. Principal components are always sorted from the largest captured variance, so we can determine the number of components that capture a specific desired threshold named α, that is usually 99% or 95% of the variance.

From the mathematical point of view, PCA is a linear transformation that rotates the points to a new coordinate system, making the correlation between them vanish. PCA does the transformation through projecting the data into a lower dimensional space while minimizing the mean squared projected error which is the difference between the original data and the projected data. The mean squared projection error is formulated as:

$$\text{Mean square projection error} = \frac{1}{m}\sum_{i=1}^{m}\left\|x^i - x^i_{approx}\right\|^2 \qquad 2.3$$

To determine the variance retained by the components, we need the ratio of the mean squared projection error over the total variance to be less than (1-α) % or 1% for 99% threshold, which is formulated as:

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\left\|x^i - x^i_{approx}\right\|^2}{\frac{1}{m}\sum_{i=1}^{m}\left\|x^i\right\|^2} \leq 0.01 \qquad 2.4$$

The steps of the PCA algorithm for a given threshold of α is:

1. Compute the mean
2. Center the data
3. Compute the covariance matrix
4. Compute the eigenvalues
5. Compute the eigenvectors
6. Compute the fraction of total variance using equation 2.4
7. Select the dimensionality that is the smallest number of components, so the fraction of total variance is less than α.

Now we can create a reduced basis and have the data with lower dimensionality [31]. Since many features in real life data sets are correlated or redundant, reducing the dimensionality using PCA can still retain the variance, and the accuracy of the classifier will be intact, yet it helps the learning algorithms run more efficiently and decrease the computation time.

2.4 Resampling Techniques for imbalanced learning

When dealing with imbalanced data, resampling methods can enhance the performance of the classifier. In this work, we have combined resampling techniques with an algorithmic approach through modifications in the classifier to address the issue of imbalanced learning in the data set. Resampling techniques improve the performance of classifier by trying to create a balance between the classes. Generally, resampling methods follow two strategies: one is removing instances from the majority class known as under-sampling and second is adding new instances to the minority class, which is known as over-sampling. The sampling tool used in this work is Imbalanced-learn API package in Python,

which provides fast and accurate sampling strategies for imbalanced data. The available under-sampling methods include:

1. Under-sampling based nearest neighbor method with multiple variations such as condensed nearest neighbor edited the nearest neighbor and repeated edited nearest neighbor.

2. Under-sampling based on the instance hardness threshold.

3. Under-sampling based on random under-sampling and also using Tomek's links. This package also provides multiple over-sampling techniques such as:

1. Over-sampling based on Adaptive Synthetic sampling method (ADASYN)

2.Random Over-sampling

3. Over-sampling based on Synthetic Minority Over-sampling Technique (SMOTE)

These methods can be applied alone or in combination with each other to imbalanced data sets to improve classification results. The explanation and the details regarding these methods are beyond the scope of this work [32].

The sampling method used in the preprocessing part of this work is Over-sampling with Synthetic Minority over-sampling Technique (SMOTE). Using SMOTE, we are trying to increase the number of instances in the minority class by syntactically creating new instances instead of merely replicating the existing instances. Therefore the classifier tends to be biased towards the minority class [33]. SMOTE generates data in feature space, and it depends on introducing new instances based on the nearest neighbors. In this method, the new examples are added near the line segment that joins the nearest neighbors of the minority class. The nearest neighbors are selected to create the instances required for over-

sampling. To implement this method, we have used the Python Imbalanced-learn API package for resampling the imbalanced data.

2.5 Classification – Support Vector Machines

SVM is a popular classification algorithm with applications in fraud detection, identifying cancer cells from healthy ones, face recognition, weather predictions, etc. SVM is developed to find a binary classifier using the training data, for which the data is already labeled by the supervisor, hence it is called a supervised learning classifier. There are multiple variations of this problem in the literature, but binary SVM classification is the most popular one [34][35].

First, we will discuss the theory behind classic SVM and some general properties of this algorithm.

Classification algorithms are used to maximize performance while maintaining the generalization for unknown data. In other words, there is a trade-off between fitting the data and the model's generalization ability. SVM algorithm classifies the data by finding a hyperplane that can separate the two classes. This approach is represented as 2.5 in the compact form:

$$\hat{y} = sign\,(H(x)), \tag{2.5}$$

where $H(x)$ is the decision function in this formulation. The separating hyperplane is the set of all points that can satisfy the following condition 2.6:

$$\begin{aligned} H(x) &= w^T x + b = 0, \\ x, w &\in \mathbb{R}^n, \text{ and } b \in \mathbb{R}, \end{aligned} \tag{2.6}$$

Where *x* is the vector of features, *w* is the weight vector, and *b* is the offset. In the linear equation above, the weight vector determines the orientation of the hyperplane in space. The hyperplane is perpendicular to the weight vector, and *b* is the offset or the distance of the hyperplane from the origin. The hyperplane divides the input space into two half-spaces. The important property of this hyperplane is that $H(x) > 0$ in one of the half spaces and $H(x) < 0$ in the other one and $H(x) = 0$ for all the data points on the hyperplane. This hyperplane is used for classifying the test data into two classes, where $H(x) > 0$ corresponds to $+1$ label and $H(x) < 0$ corresponds to -1 label.

$$\hat{y} = \begin{cases} 1 & w^T x + b > 0 \\ -1 & w^T x + b < 0 \end{cases} \qquad 2.7$$

The distance from the nearest data point in the training set to the separating hyperplane is called the margin of separation. Although we can find multiple hyperplanes that satisfy condition 2.7, the hyperplane that has the maximum margin of separation between the two classes is unique and is found through optimization. The maximum margin of separation is necessary because it increases the model's generalization or the ability to better handle the noise in the test data and the data points that lie on the margin are classified based on their location on the band.

Another critical term that this method is named after are Support Vectors. Support Vectors are the data points that their distance from the separating hyperplane is equal to one after normalization.

2.5.1 Linearly Separable Data – Hard margin Support Vector Machines

As it is mentioned before, the goal of SVM is to find a hyperplane that separates the data points in two classes. As it is shown in Figure 2-3 achieving this goal is easier if the data

is linearly separable, meaning there exists a hyperplane that can perfectly separate the data points into two classes without having any points labeled *+1* that lies on the subspace $H(x) < 0$ and points labeled *-1* that rests on the subspace $H(x) > 0$. In the case of separable data, multiple hyperplanes exist that can classify the data, so an optimization problem is defined to find the hyperplane with a maximum margin of separation.

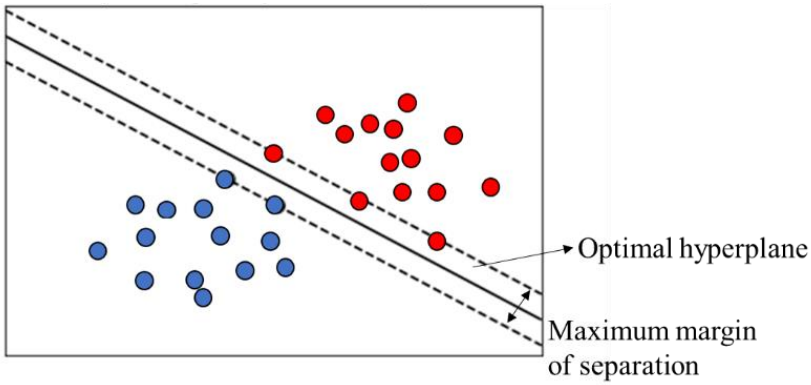


Figure 2-3: Linearly separable SVM

An SVM problem is designed to find a hyperplane with the maximum margin of separation that can classify the data points into two classes.

As discussed earlier, the separating hyperplane is defined as $H(x) = w^T x + b = 0$, where $x, w \in \mathbb{R}^n$ and $b \in \mathbb{R}$. In this equation, w is the N-dimensional weight vector, and b is the offset. The margin of separation denoted as $\frac{1}{\|w\|}$ is defined as the distance between the support vectors and the hyperplane. To find the maximum margin of separation, instead of maximizing $\frac{1}{\|w\|}$ we can minimize $\|w\|$ subject to the constraints. Referring to the definition of support vectors, and the separable data, for a given data set $D = \{x_i, y_i\}_{i=1}^n$ that $x_i \in \mathbb{R}^n$ and $y_i \in \{-1, +1\}$, the solution of the optimization problem is optimal if it satisfies the following constraint:

$$y_i (w^T x_i + b) \geq 1, \ \forall x_i \in D \qquad 2.8$$

So, the problem is formulated as:

$$\underset{w,b}{Min}\ \{\frac{\|w\|^2}{2}\} \qquad 2.9$$

$$Subject\ to:\ y_i(w^T x_i + b) \geq 1,\ \forall x_i \in D$$

This problem is a convex quadratic minimization problem that can be solved using quadratic programming algorithms in existing software such as MATLAB and Python.

2.5.2 *Linearly non-separable data – Soft margin Support Vector Machines*

One drawback of this approach is that most real-life, large data sets are not linearly separable and perfect separation of the two classes is not possible; that is why Soft Margin SVM is introduced.

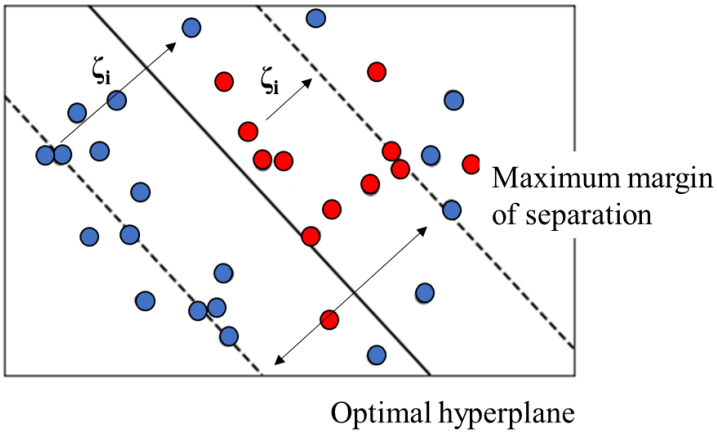


Figure 2-4: Non-linearly separable SVM

As it is shown in Figure 2-4, in soft margin SVM, some of the points might be on the separating margin or the wrong side of the hyperplane; therefore, the violation from the separating condition, previously defined as

$H(x) = w^T x + b = 0$, where $x, w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ must be measured. Thus, the slack variables ζ_i are introduced to handle non-separable cases. The value of ζ_i can be within

three ranges. If $\zeta_i = 0$, then there is no misclassification cost for x_i and the point is either a support vector or on the right side of the margin. Next, if $0 < \zeta_i < 1$, that the point is located on the margin and its distance from the separating hyperplane is less than $\frac{1}{\|w\|}$. Finally, $\zeta_i \geq 1$ indicates that the point is on the wrong side of the margin and it is misclassified.

Now, the objective is to maximize the margin of separation, while minimizing the slack variables with the correct regularization parameter. Regularization parameter denoted as C is defined in the objective function to control the trade-off between the two terms, maximizing the margin of separation and minimizing the slack variables that represents the cost of misclassification.

Mathematical formulation of Soft Margin SVM is:

$$\underset{w,b,\zeta i}{Min}\ \{\frac{\|w\|^2}{2} + C\sum_{i=1}^{n}\ (\zeta_i)^k\}$$

$$\text{Subject to:}\quad y_i\,(w^T x_i + b) \geq 1 - \zeta_i,\ \forall x_i \in D \qquad 2.10$$

$$\zeta_i \geq 0,\ \forall x_i \in D$$

In this problem, the term $\sum_{i=1}^{n}\ (\zeta_i)^k$ represents the misclassification cost and it is known as the loss, that is deviation from the separable case. Usually, k is set to be equal to 1 or 2. K=1 is called hinge loss and k=2 is known as the quadratic loss. The goal is to minimize the sum of slack variables or the squared slack variables, respectively. In this work, since we are trying to solve a linear programming problem, we have assumed k = 1.

Once the problem is solved, the values of w_i determine the support vectors. If $w_i = 0$, the corresponding x_i is not a support vector, and if $w_i > 0$, the corresponding data points

satisfy the separating condition and $y_i (w^T x_i + b) = 1 - \zeta_i$. In this case, support vectors are all the points on the margin with $\zeta_i = 0$ or $\zeta_i > 0$.

SVM is a systematic approach based on minimizing a convex cost function. Due to the properties of convex functions, local minima do not exist, which makes the optimization problem less complicated. Also, SVM performs better with kernels. Therefore, they are beneficial for the problems with non-linearly separable data. To summarize, SVM is based in the theory of convex optimization. However, for training, we need to solve a quadratic optimization problem which is computationally complex, and it requires powerful computer systems for large scale problems. Based on the theory of quadratic SVM, linear SVM is proposed, which is less complicated and computationally expensive in comparison with the classic SVM.

2.6 Defining norms

In linear algebra, the norm of a vector $\vec{v}$ is a useful way of measuring the size or length of the vector. A vector norm is a function defined from $\mathbb{R}^n \rightarrow \mathbb{R}$ and based on its definition it has different properties. In general, Lp-norm is defined as:

$$\|\vec{v}\|_p = \sqrt[p]{\sum_{i=1}^{m} |v_i|^p} \qquad 2.11$$

Lemma 1*: For any given vector in N-dimension linear space V^n, let $\|\vec{x}\|_\alpha$ and $\|\vec{x}\|_\beta$ be any vector norms in V^n, there exist two positive constants $0 < c_1 < +\infty$ and $0 < c_2 < +\infty$ such that condition:*

$$C_1\|x\|_\beta \leq \|x\|_\alpha \leq C_2\|x\|_\beta, \quad \forall x \in V^n \qquad 2.12$$

holds.

This lemma proves that based on the definition of convergence if a set of vectors converges to a vector C with respect to a specific norm, it converges to the same vector when using other norms. So, we can say that for two constants C_1 and C_2, if condition *2.12* stands, we say that the two vector norms are equivalent.

Theorem 1: *Given the examples set* $\{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_l, y_l)\}$, $(x, y) \in (R^n, R)$ *and the vectors x belong to a sphere of radius R. If the set of* $m_\Delta -$ *margin separating hyperplanes of* $H(x,w,b) = w^T x + b$ *classifies vectors x as follows:*

$$\hat{y} = \begin{cases} 1 & w^T x + b > \Delta, \\ -1 & w^T x + b < \Delta, \end{cases} \quad \Delta \geq 0 \qquad 2.13$$

Then there exists a constant $0 < C < +\infty$ *without depending on examples (x, y) and the parameters w and b such that the bound inequality holds:*

$$h \leq min\left(\left[\frac{c^2 . R^2 . \|w\|_\beta^2}{\Delta^2}\right], n\right) + 1 \qquad 2.14$$

where $||w||_\beta$ *is any vector norm.*

The lemma and theorem can be found in [15].

Based on the above lemma and theorem, SVM formulation can be modified using different norms other than just L_2-norm.

The most commonly used norms are L_1-norm, L_2-norm, and $L\infty$-norm denoted as $\|\vec{v}\|_p$ where p is 1, 2, or ∞, respectively. In this section, we will discuss how each norm is defined and we'll touch upon their properties.

One Norm (L_1-norm or mean norm) is defined as the sum of the absolute values of the vector components:

$$\|\vec{v}\|_1 = \sum_{i=1}^{m} |v_i| \qquad 2.15$$

Two Norm (L_2-norm or least squares norm) is defined as the square root of the sum of the squares of the absolute values of the vector components:

$$\|\vec{v}\|_2 = \sqrt{\sum_{i=1}^{m} |v_i|^2} \qquad 2.16$$

Infinity-Norm (L_∞-norm, max norm or uniform norm) is defined as the maximum of the absolute values of the vector components:

$$\|\vec{v}\|_\infty = max\ \{|v_i|\ for\ i = 1, 2..., m\} \qquad 2.17$$

We can use these variations of norms to define the distance between the points in feature space. These terms are used to define a modified SVM problem where the quadratic objective function is replaced with a linear function using L_∞-norm function. For the weight vector $w = (w_1, w_2, ..., w_n)$ where $n \in \mathbb{N}$ is the size of the features set. If $\tau = Max\ \{|w_1|, |w_2|, \ldots, |w_n|\}$, SVM is formulated as:

$$\underset{w,b,\zeta i}{Min}\{\ \lambda_1 \tau + \lambda_2 \sum_{i=1}^{n} \ \zeta_i\} \qquad 2.18$$

$$Subject\ to:\quad y_i\,(w^T x_i + b) \geq 1 - \zeta_i,\ \forall x_i \in D$$

$$\zeta_i \geq 0,\ \forall x_i \in D$$

$$|w_i| \ \leq \ \tau$$

$$\lambda_1 + \ \lambda_2 = 1$$

$$\lambda_1, \lambda_2 \geq 0$$

Another vector norm that could replace the L_2-norm SVM is L_1-norm that yields a linear programming problem, which is of more interest to us. As it is shown before, classic SVM is defined as a quadratic problem, and it is popular due to its acceptable performance. Although L_2-norm SVM has a reasonable performance in binary classification problems, L_1-norm SVM is more suitable for datasets with redundant noisy features. In the case of imbalanced data, the size of the error vectors is crucial for us because we can compare the accuracy of the classifier in each class. So, using L_1-norm is preferable, and after minimization, in most cases, the slack variable will have a value of 0 or close, and a few variables will have a high error. The drawback of this approach is the existence of outliers, which we have dealt with this issue in the preprocessing section.

For the weight vector $w = (w_1, w_2, \ldots, w_n)$ where $n \in \mathbb{N}$ is the size of the feature set and $l \in \mathbb{N}$ is the number of observations, the mathematical formulation of an L_1-norm SVM is presented below:

$$Min_{w,b,\zeta}\{ \lambda_1 \sum_{i=1}^{l} |w_i| + \lambda_2 \sum_{i=1}^{n} \zeta_i\} \qquad 2.19$$

$$Subject\ to:\ y_i (w^T x_i + b) \geq 1 - \zeta_i$$

$$\zeta_i \geq 0\ ,\ \forall x_i \in D$$

$$\lambda_1 + \lambda_2 = 1$$

$$\lambda_1, \lambda_2 \geq 0$$

To classify the data, we need to train the classifier based on the model, which is attained after optimizing the LP problem.

So far, the standard SVM with modification to the objective function is presented. However, SVM classifier as discussed in the previous section does not yield the most accurate results, so Kernel SVM has risen to improve the performance of the classifier.

Kernel trick is a method of data presentation based on similarity in the data which facilitates the analysis and classification of data. The data is generally presented in the input space, and kernel trick is used to map the data to a higher dimensional space called feature space. To do so, for a given data point x, we need a map denoted as $\varphi(x)$, which is a vector representation of the data point in feature space and φ is the function that transforms the data from input space to the feature space. This transformation makes a complex analysis of the data using numerical probabilistic and algebraic methods attainable. The drawback to this transformation is that feature space is a high dimensional space, and consequently, the curse of dimensionality is unavoidable. Another challenge while mapping the data is finding the kernel map. However, this issue is solved by transforming the data using the similarity values between the points or the kernel matrix denoted as $K(x_i, x_j)$. K is the kernel function and it is defined for any two points in the input pace via the dot product between the images of the two points in the feature space. The kernel function K must satisfy condition 2.20:

$$K(x_i, x_j) = \varphi(x_i)^T . \varphi(x_j) \qquad 2.20$$

This condition proves that we can map the data from the input space into the feature space without explicitly defining the kernel function and avoid the computational complexity; this is known as the kernel trick. Kernelization is widely used in various machine learning algorithms, and it can considerably enhance their performance. One of the algorithms that

have greatly benefited from kernelization is SVM. Comparing to non-kernelized SVM, kernel-SVM is more flexible, and it can predict non-linearly separable data better.

One of the most important characteristics of the kernel matrix is that the matrix is symmetric and positive semidefinite, meaning that for a function *K* that maps the data from input space to the feature space using their respective dot product, $K(x_i, x_j) = \varphi(x_i)^T . \varphi(x_j)$. Function *K* is positive semidefinite if and only if $K(x_i, x_j) = K(x_j, x_i)$ which shows that *K* is symmetric.

So, the kernel matrix K is also symmetric and positive semidefinite if for any given vector

$$q \in \mathbb{R}^n: \qquad (2.21)$$

$$Q^T K Q \geq 0$$

This equation can be rewritten as:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} q_i q_j K(xi, xj) \geq 0 \qquad (2.22)$$

Based on these conditions, it could be said that we can create many positive semidefinite kernels that correspond to the dot product of the data points in a feature space. According to this, different kernel functions have been developed and used over the years. Some of the most popular ones are presented in Table 2-1.

Table 2-1: Kernel functions

Kernel	Formulation
Linear	$K(x, y) = x^T y$
Gaussian	$K(x, y) = \exp\{\frac{-\|\|x-y\|\|^2}{2\delta^2}\}$
Polynomial	$K(x, y) = (c + x^T y)^q$
Sigmoid	$K(x, y) = \tanh(x^T y + c)$

Choosing the suitable kernel and assigning the best parameter value is very important as it can significantly affect the performance of the classifier. To better understand the role of the kernel in SVM classification, four SVMs with and without kernels are defined to classify a randomly generated dataset, and the results are presented in Figure 2-5.

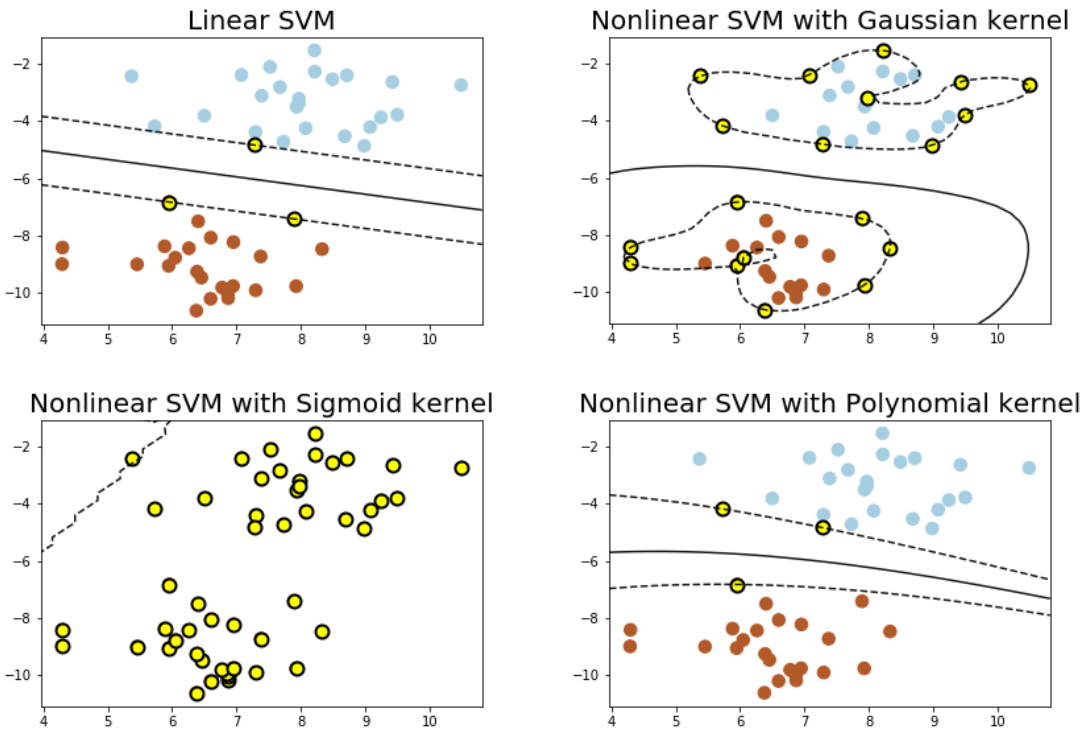


Figure 2-5: Kernel SVM

For implementing the kernel matrix into classification problems using SVM, the decision function and the constraint must be reformulated to reflect the use of kernel and transform the data from input space into feature space. The mathematical formulation of SVM for the weight vector w_i where $l \in \mathbb{N}$ is the number of observations in the data set, with a given kernel $K(x_i, x_j)$ is presented as follows:

$$Min_{\,w,b,\zeta}\{\,\lambda_1 \sum_{i=1}^{l} |w_i| + \lambda_2 \sum_{i=1}^{l} \zeta_i\,\} \qquad 2.23$$

$$Subject\ to:\ y_i\left[\sum_{i=1}^{l}\ \sum_{j=1}^{l}\ w_i.\ k(x_i, x_j) + b\right] \geq 1 - \zeta_i$$

$$\lambda_1 + \lambda_2 = 1$$

$$\zeta_i \geq 0,\ w_i \geq 0$$

$$\lambda_1, \lambda_2 \geq 0$$

2.7 Parametric modeling

SVM is a powerful algorithm for predictive analysis. However, it can be computationally expensive. To achieve the best results using SVM we need to tune the model parameters through a regularization parameter which is usually denoted by *C*. The regularization parameter controls the trade-off between the two objectives, which are minimizing the error term and minimizing the norm of the weight vector and achieving the maximum margin of separation. Tuning this parameter is quite essential as it can affect the performance of SVM classifier. There are not many robust approaches for parameter tuning in SVM, and grid search is often used for this purpose. To perform a grid search for parameter selection, the model is trained over a set of hyperparameters to find the optimal parameters for the given model. A grid search is computationally expensive, and it requires powerful computing systems if the data set is large, because it builds multiple combinations of the model with the given parameter values and selects the best one after comparing the results for all combinations.

In this work, we have proposed using an algorithmic approach based on the parametric simplex method to exhaustively search the solution path and find the optimal value of the regularization parameter.

Since the proposed SVM algorithm proposed in this work is a linear programming problem, by defining this issue as parametric linear programming, we can use parametric simplex method to attain all the values for the parameter and choose the optimal value among a small set of potential values. Various versions of the parametric simplex method exist which are different in the variables we choose to update at each iteration [36]. In a paper published by Nguyen Dinh Dan and Le Dung Muu [37], the parametric simplex algorithm for multi-objective optimization problems is introduced. In this paper, the proposed algorithm finds the global solution of the linear programming problem.

The method used in this work was initially introduced in a book by Ehrgott et al. [38]. Using the parametric simplex algorithm for two-objective linear optimization problems, we can find the optimal value of the regularization parameter of an optimization model in SVM. In theory, the general form of this algorithm applies to multi-objective linear programming problems too.

Before further discussing the algorithm, a few terms must be defined. As mentioned before, the algorithm starts with solving the augmented form of the original problem to check the feasibility of the problem. The solution obtained from solving this problem is the augmented solution which is the optimal values of the decision variables in the augmented LP. The corner-point solution of the augmented LP is a basic solution. The basic solution can be either feasible or not, and it is called a basic feasible solution (BFS) if it is feasible. The corner point solution and a basic solution are only different in including the value of slack variables. The corner-point solution consists of basic variables and non-basic variables. Non-basic variables are the values set equal to zero in the basic solution, and the remaining variables are the basic variables. The set of all basic variables is known as the

basis. Another term that needs to be defined is the reduced cost vector, which is the amount the coefficients of the objective function have to decrease when minimizing the objective function, to achieve the optimal value. Reduced cost vector is denoted as $\bar{c}$. In a two-objective LP, $\bar{c}_i^1$ represents the components of the reduced cost vector corresponding to the first objective and $\bar{c}_i^2$ represents the components corresponding to the second objective. Based on the inseparable data, kernelized Support Vector Machine with L_1-norm is introduced for classification and the problem is formulated as:

$$Min_{w,b,\zeta}\{ \lambda_1 \sum_{i=1}^{l} |w_i| + \lambda_2 \sum_{i=1}^{l} \zeta_i \} \quad (2.24)$$

$$Subject\ to: y_i [\sum_{i=1}^{l} \sum_{j=1}^{l} w_i . k(x_i, x_j) + b] \geq 1 - \zeta_i$$

$$\lambda_1 + \lambda_2 = 1$$

$$\zeta_i \geq 0,\ w_i \geq 0$$

$$\lambda_1, \lambda_2 \geq 0$$

There are multiple methods for solving a multi-objective problem [39]. The objective function in this problem is a weighted sum of two separate objective functions that we are trying to minimize simultaneously. The goal is to find the optimal value of λ_1 and λ_2. To make the problem compatible to the form mentioned in [32] let's set $\lambda_1 = \lambda$ and $\lambda_2 = 1 - \lambda$.

Theorem 2: *Let $\hat{x} \in X$ be an optimal solution of the weighted sum LP*

1. If $\lambda \geqslant 0$ then $\hat{x}$ is weakly efficient.

2. If $\lambda > 0$ then $\hat{x}$ is efficient. [32]

Based on Theorem 2, the problem is solved when the optimal value of λ is identified. If $\lambda \geq 0$, then the optimal solution of the LP is weakly efficient and if $\lambda > 0$ then the solution is efficient. This algorithm starts with the feasibility evaluation of the LP problem, and the augmented form of the problem is solved to find the optimal basis and optimal basis feasible solution to the problem. The augmented form of the problem is given as:

$$Min_{w,b,\zeta}\{ \lambda \sum_{i=1}^{l} |w_i| + (1-\lambda) \sum_{i=1}^{l} \zeta_i \} \quad 2.25$$

$$Subject\ to: y_i [\sum_{i=1}^{l} \sum_{j=1}^{l} w_i . k(x_i, x_j) + b] - Z_i = 1 - \zeta_i$$

$$\zeta_i \geq 0,\ w_i \geq 0$$

$$\lambda \geq 0$$

where Z_i is the added i^{th} slack variable. The problem is implemented and solved using the simplex algorithm in Gurobi optimization tool in Python.

Proposition 1: *The LP is feasible, i.e. $X \neq \emptyset$, if and only if the auxiliary LP has an optimal solution $(\hat{x}, \hat{z})$ with $\hat{z} = 0$.*

If $\hat{x}$ in an optimal solution of the auxiliary LP, is not a BFS of the original LP, it can always be easily converted into one [32].

Based on Proposition 1, if the added slack variables are equal to zero and the problem has an optimal solution, then the problem would be feasible.

Theorem 3: *1. A basic feasible solution $(x_B, 0)$ of a linear programming problem is an extreme point of the feasible set X. However, several feasible bases may define the same basic feasible solution and therefore the same extreme point (in case of degeneracy).*

2. If $X \neq \varnothing$ and the LP is bounded, the set of all optimal solutions of the LP is either X itself or a face of X.

3. For each extreme point $\hat{x}$ of X there exists a cost vector $c \in R^n$ such that $\hat{x}$ is an optimal solution of $min\{c^T x : x \in X\}$ [32].

Based on Theorem 3, the basic feasible solution of LP in the augmented format is an extreme point belonging to the feasible set and a cost vector $c \in \mathbb{R}^n$ exists for each extreme point of the feasibility set so the objective function can be written as:

$$Min\ \{(c^1)^T \sum_{i=1}^{l} \alpha_i\ ,\ (c^2)^T \sum_{i=1}^{l} \zeta_i\ \} \qquad 2.26$$

Subject to the constraints and finding efficient solutions to this problem would yield the solution to the two-objective LP. The parametric simplex algorithm is implemented in 3 phases.

Phase 1: As it is discussed in the previous section, phase one is solving the augmented LP problem and finding the optimal basis and the optimal basic feasible solution. The problem is infeasible if the set of optimal basic feasible solution is empty. Therefore, the algorithm stops.

Phase 2: In the next step we will solve the following LP for $\lambda = 1$ starting from the basis obtained from phase one. When implementing the problem in Python, there are various optimization tools to be selected, since this approach is based on parametric simplex for two-objective optimization, the simplex algorithm must be selected.

The output of this phase is the optimal basis. The optimal values of the decision variables are updated as well.

Phase 3: At this stage, the set of non-basic variables is denoted by N, and the reduced cost vector for each objective is also calculated. The new value for λ is calculated using the following equation:

$$\lambda = max \frac{-\bar{c}_i^2}{\bar{c}_i^1 - \bar{c}_i^2} \qquad 2.27$$

for $i \epsilon I$, where $I = \{ i \epsilon N, \bar{c}_i^2 < 0 , \bar{c}_i^1 > 0 \}$

Using equation 2.27 the value of λ and the decision variables is updated after each simplex iteration until $I = \emptyset$ and the algorithm ends while providing a finite number of λs to be tested as an optimal parameter for the LP. Using this method, we can significantly reduce the computational complexity and obtain more accurate results for tuning the parameters in comparison with methods such as grid search. This method is implemented using Gurobi in Python, and the algorithm is presented as follows:

1. Set the model parameter equal to 1 to use Simplex as the optimization solver.

model.params.method = 1

2. Set the objective function equal to zero.

model.setObjective (0.0)

3. Solve the auxiliary LP and obtain the basis and basic feasible solution.

Basis = model.VBasis

4. If $Z_i = 0$, the problem is feasible; otherwise the LP is unbounded.

5. Set $\lambda = 1$

6. Solve the auxiliary LP, starting from the basis found in step 3.

PStart = Basis

7. Obtain a new basis and the reduced cost of the components of the objective functions.

For j = number of components of the objective function:

Reduced cost of vector W: $\bar{c}_i^1 = W[j].RC$

Reduced cost of vector S: $\bar{c}_i^2 = S[j].RC$

8. Calculate λ as*:* $\lambda = max \frac{-\bar{c}_i^2}{\bar{c}_i^1 - \bar{c}_i^2}$ *for* $i \epsilon I$, where $I = \{ i \epsilon N, \bar{c}_i^2 < 0 , \bar{c}_i^1 > 0 \}$

9. While $I \neq \emptyset$, update λ and repeat steps 7 - 9.

10. The output is a finite number of λ values to be used as a regularization parameter.

2.8 Evaluation metrics

Assessment is the next important step after classification. Some of the most popular evaluation metrics are accuracy and error rate.

Before defining the metrics, four key terms need to be defined. For a binary classification problem, if {P, N} is the predicted labels for the data points in the test set, and the majority class is donated by N, and the minority class is denoted by P, we can use Confusion Matrix as presented in Table 2-2 as a visual presentation of classifiers' performance:

Table 2-2: Confusion matrix

		Predicted values
Actual values	True Positive (TP)	False Negative (FN)
	False Positive (FP)	True Negative (TN)

Based on this notation, accuracy and error rate are defined as:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Error rate = 1 - accuracy

Although these metrics are used in almost all machine learning problems, they are not appropriate for imbalanced data because their results are biased towards the majority class. These evaluation metrics often fail to present the poor performance of the classifier on the minority class, which in most cases is more important for us. So, for binary classification problems of imbalanced data, other metrics are defined and used.

1. Precision = $\frac{TP}{TP+FP}$

2. Recall = $\frac{TP}{TP+FN}$

3. F-measure = $\frac{(1+\beta)^2.\ \text{Recall.}\ \text{Precision}}{\beta^2.\ \text{Recall}+\text{Precision}}$, where β is the relative importance of precision versus recall, and it is usually set equal to one.

4. Geometric mean/ G-mean = $\sqrt{\frac{TP}{TP+FN} . \frac{TN}{TN+FP}}$

In imbalanced learning, precision measures the proportion of instances that were labeled correctly among those with the positive label in the test data. In other words, it measures how exact the model is, concerning the minority class. While, recall measures the portion of positive instances in the test data that were labeled correctly, and it measures the completeness of the model. Precision and recall have an inverse relationship. Precision is dependent on the distribution in the data, but the recall is more robust. Precision and recall, when used together can provide valid insight into the performance of the classifier with regards to the minority class. That is why F-measure is a valuable evaluation metric in imbalanced learning, which can assess the trade-off between precision and recall.

Finally, G-mean is the metric that is used explicitly for imbalanced learning and tries to maximize the accuracy of the model over each class by considering both classes for evaluation. In the evaluation section of this work, all the metrics mentioned above are used to compare the models and select the best one.

To visualize and summarize the performance of the classifier, we have used Receiver Operating Characteristic curve also known as ROC curve [40]. ROC curve is a popular tool, which presents the trade-off between true positive rate (TP_{Rate}) and false positive rate (FP_{Rate}) defined as:

$$TP_{Rate} = \frac{TP}{TP+FN} \qquad 2.28$$

$$FP_{Rate} = \frac{TN}{TN+FP}$$

The area under the ROC curve denoted as AUC is defined as:

$$AUC = \frac{TP_{Rate} + TN_{Rate}}{2} \qquad 2.29$$

AUC represents the probability of correctly classifying positive instances while minimizing the number of false positives. It provides a great measure to compare different models as it is independent of the classification model. Figure 2-6 shows all the key points in the ROC curve and AUC. If used appropriately, they can be a valuable tool in model assessment for imbalanced learning.

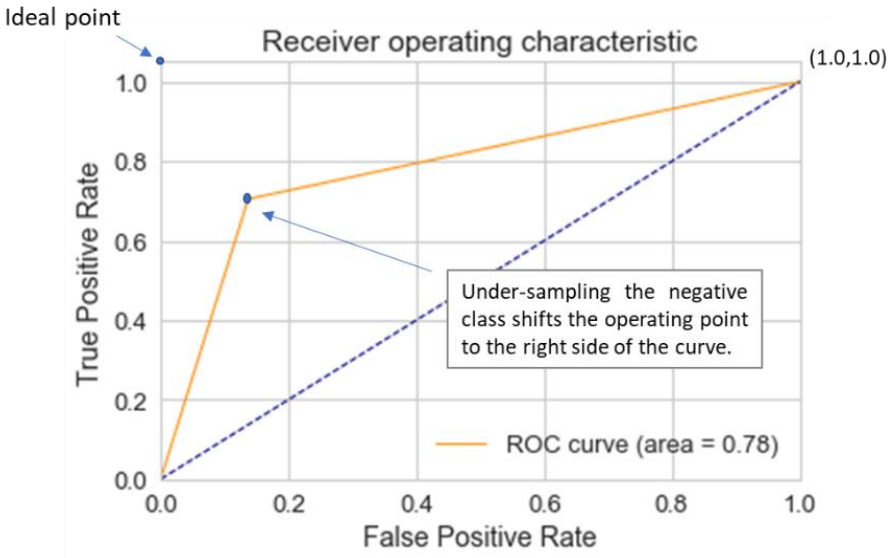


Figure 2-6: Receiver Operating Characteristic and Area Under the Curve

Finally, K-fold cross validation is used to evaluate the classifiers over the data set. For most steps of this work, 20 percent of the data is randomly selected for testing and the remaining data set is used for training. For evaluation, 5-fold cross validation is proposed as a standard evaluation technique. In 5-fold cross validation, the data set is divided into five subsets, and in each iteration, one subset is used for testing and the remaining is used for training.

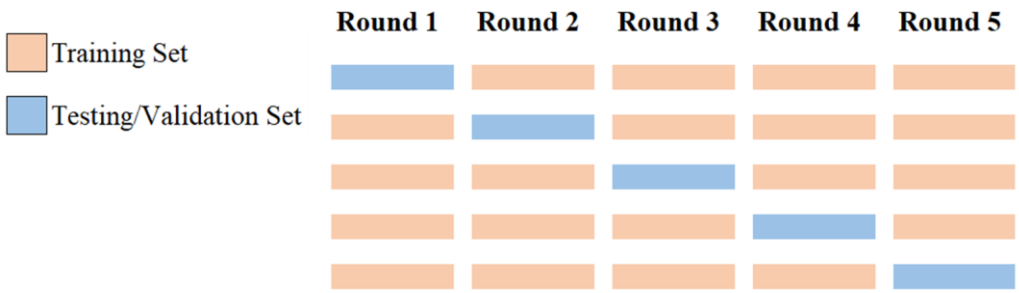


Figure 2-7: K-fold Cross Validation

The final results are the average of the results found at each round.

3 Experiment and results

3.1 Data preprocessing

3.1.1 Missing value imputation

The data used in this work is collected by the Bureau of meteorology's system in Australia. Bureau of Meteorology's system is responsible for collecting the data regarding weather phenomena in Australia. Bureau weather stations collect data from real-time observations including temperature, rainfall, wind speed and direction, sunshine, etc. The most common ones are the stations that measure rainfall. The historical data is collected from the early 1830s, at stations in Adelaide West Terrace, Parramatta, Port Arthur, and Colombo Creek. The number of stations has grown significantly since then, and currently, there are more than seven thousand active stations in Australia. Bureau weather stations are Meteorology offices in Australia that record various weather-related observations. Most Bureau weather stations observe and record a type of weather event based on the specific equipment available in the station. The observations are recorded in different time intervals such as daily, or hourly, and it varies based on the location of the station and the weather phenomena observed. The Bureau stations have a unique name, and ID number and all the observations are gathered and stored in the Bureau's climate database, also known as the Australian Data Archive for Meteorology.

The historical data includes various numerical and categorical features, which are defined in Table 3-1. (http://www.bom.gov.au/climate/data-services/data-requests.shtml):

Table 3-1: Numerical and categorical features of the data set

Measurements		Meaning	Unit
Date		Day of the month	
Day		Day of the week	First two letters
Temperature	Min	Minimum temerature in the 24 hrs to 9 AM	Degrees Celsius
	Max	Minimum temerature in the 24 hrs to 9 AM	Degrees Celsius
Rain		Precipitation (rainfall) in the 24 hrs to 9 AM	Millimeters
Evaporation		"Class A" pan evaporation in the 24 hrs to 9 AM	Millimeters
Sunshine		Bright sunshine in the 24 hrs to midnight	Hours
Max Wind Gust	Direction	Direction of the strongest wind gust in the 24 hrs to midnight	16 compass points
	Speed	Spped of the strongest wind gust in the 24 hrs to midnight	Kilometers per hour
	Time	Time of the strongest wind gust	Lcoal time hh:mm
9:00 AM	Temperature	Temperature at 9 AM	Degrees Celsius
	Relative Humidity	Relative humidity at 9 AM	Percent
	Cloud	Fraction of sky obsured by cloud at 9 AM	Eights/Oktas
	Direction	Wind direction averaged over 10 times prior to 9 AM	Compass points
	Speed	Wind speed averaged over 10 times prior to 9 AM	Kilometers per hour
	Mean Sea Level Pressure	Atmospheric pressure reduced to mean sea level at 9 AM	Hectopascals
3:00 PM	Temperature	Temperature at 9 AM	Degrees Celsius
	Relative Humidity	Relative humidity at 9 AM	Percent
	Cloud	Fraction of sky obsured by cloud at 9 AM	Eights/Oktas
	Direction	Wind direction averaged over 10 times prior to 9 AM	Compass points
	Speed	Wind speed averaged over 10 times prior to 9 AM	Kilometers per hour
	Mean Sea Level Pressure	Atmospheric pressure reduced to mean sea level at 9 AM	Hectopascals

The data used in this work is collected from multiple stations in Australia. Due to limitations of CPU and Memory, the 25 stations are divided into 5 sample groups. Each group is treated as one data set and is used for analysis. Table 3-2 shows the number and percentage of observations in each class for the chosen stations.

Table 3-2: Observation locations and sampled data sets

Location		Rain	Clear	Total
Group 1	Portland	1095	1901	2996
	Walpole	949	1870	2819
	Cairns	950	2038	2988
	Dartmoor	922	2021	2943
	NorfolkIsland	919	2045	2964
Group 2	MountGambier	920	2110	3030
	Albany	902	2114	3016
	Witchcliffe	879	2073	2952
	CoffsHarbour	869	2084	2953
	MountGinini	819	2088	2907
Group 3	NorahHead	808	2121	2929
	Williamtown	700	1853	2553
	Darwin	852	2340	3192
	GoldCoast	775	2205	2980
	Ballarat	781	2247	3028
Group 4	SydneyAirport	774	2231	3005
	Newcastle	731	2224	2955
	Watsonia	738	2261	2999
	Wollongong	713	2270	2983
	Hobart	761	2427	3188
Group 5	Launceston	699	2329	3028
	Brisbane	709	2452	3161
	Adelaide	688	2402	3090
	MelbourneAirport	653	2356	3009
	Sale	643	2357	3000

The groups are selected with a slightly different ratio of class imbalance. Table 3-3 and Figure 3-1 presents an overview of the percentage of observations in each class.

Table 3-3: Sampled data sets

Sample	Rain	Clear	Total	Minority (%)	Majority(%)
Group 1	2318	12691	15009	15.44	84.56
Group 2	2669	14123	16792	15.89	84.11
Group 3	2355	12638	14993	15.71	94.29
Group 4	2516	14208	16724	15.04	84.96
Group 5	2413	13617	16030	15.05	84.95

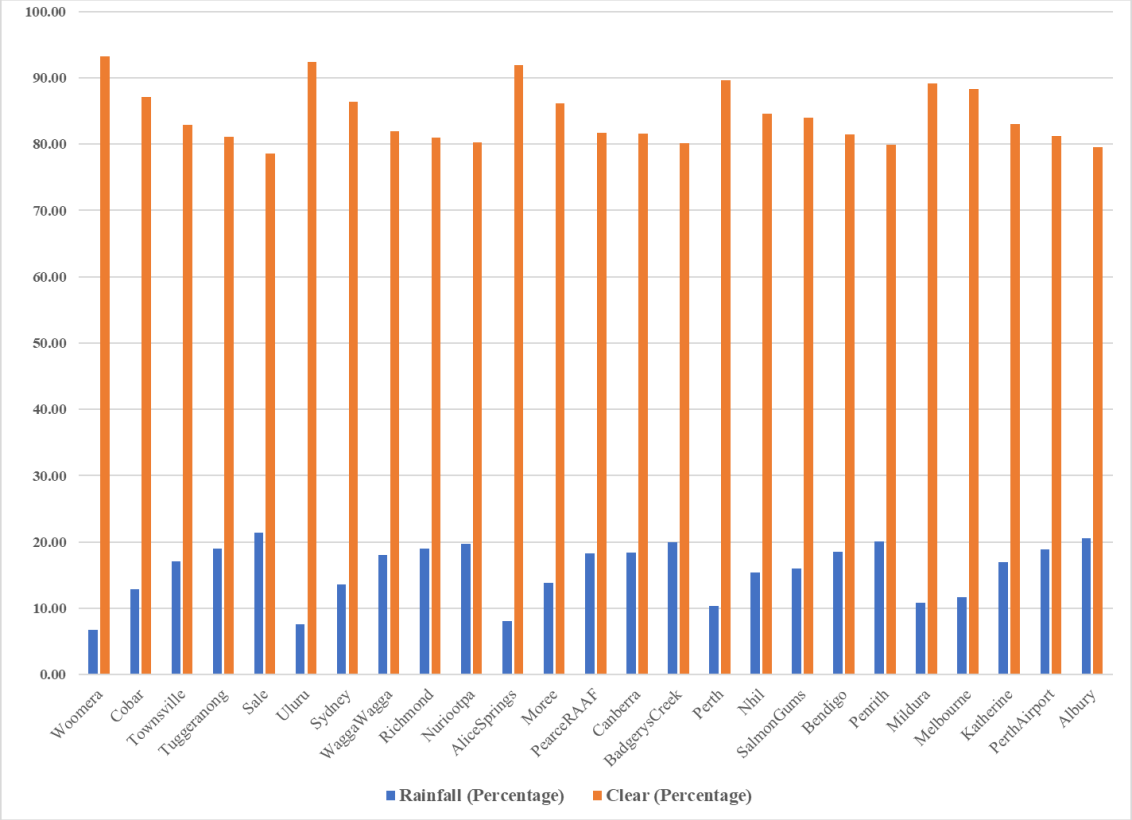


Figure 3-1: Distribution of data in the minority and majority class in the selected locations

Before proceeding with the analysis, it is essential to check the data for missing values and outliers. We need to know which features have missing values and what percentage of the data is missing. The data initially had 142193 observations and 24 attributes. For simplification, most of the attributes are written in abbreviations. The complete list of attributes is with the respective missing value percentage is provided in Table 3-4.

Table 3-4: Features of the data set and the percentage of missing values

Feature Name	Meaning	Missing values (%)
Date	Date	0.00
Location	Location	0.00
RainToday	Rain Today	0.00
RISK_MM	Chance of rain (mm)	0.00
RainTomorrow	Rain Tomorrow	0.00
MaxTemp	Maximum Temperature	0.23
MinTemp	Minimum Temperature	0.45
Temp9am	Temperature at 9 am	0.64
WindSpeed9am	Wind Speed at 9 am	0.95
Rainfall	Rainfall	0.99
Humidity9am	Humidity at 9 am	1.25
WindSpeed3pm	Wind Speed at 3 pm	1.85
Temp3pm	Temperature at 3 pm	1.92
Humidity3pm	Humidity at 3 pm	2.54
WindDir3pm	Wind Direction at 3 pm	2.66
WindGustSpeed	WindnGust Speed	6.52
WindGustDir	Wind Gust Direction	6.56
WindDir9am	Wind Direction at 9 am	7.04
Pressure3pm	Pressure at 3 pm	9.83
Pressure9am	Pressure at 9 am	9.86
Cloud9am	Cloud at 9 am	37.73
Cloud3pm	Cloud at 3 pm	40.15
Evaporation	Evaporation	42.79
Sunshine	Sunshine	47.69

As it is shown, the features “Evaporation,” “Sunshine,” “Cloud9am” and “Cloud3pm” have more than 30% missing values. In this step, the columns with categorical values such as “WindDir9am” and “WindDir3pm” are also removed. The “RISK_MM” column is removed because it includes measurements of rainfall in the day that we are trying to predict, and it leaks information to the classifier. Information leakage is misleading, and it can result in false high accuracy. Figure 3-2 compares the missing values in the data set before and after dropping the features.

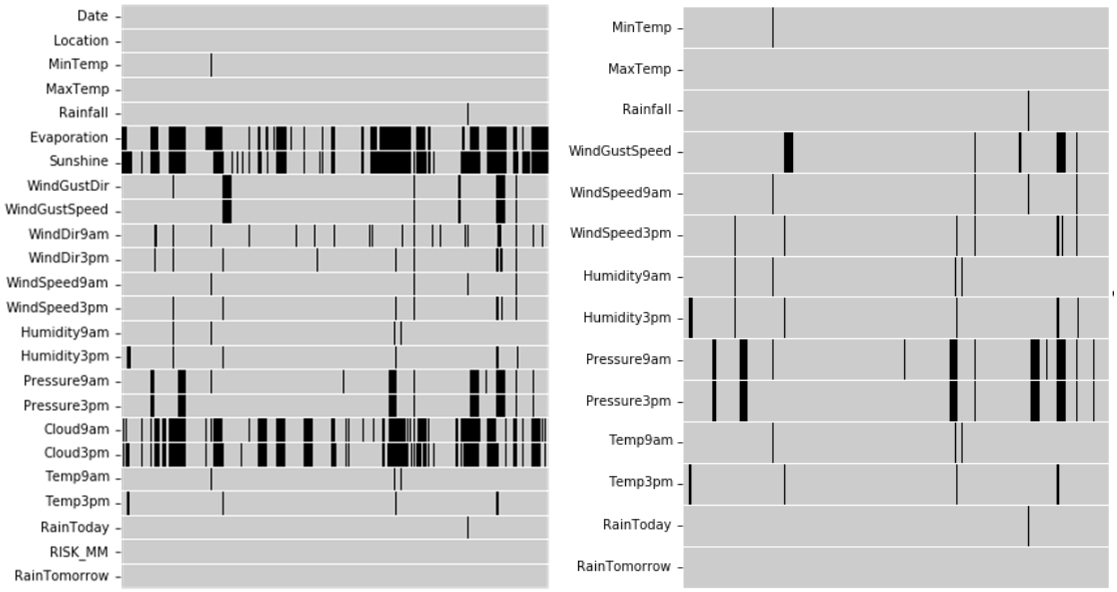


Figure 3-2: Missing values in the data set before and after dropping the features with 30% or higher missing values

The missing values in the remaining features are filled with the mean of the observed instances in each feature. The number of features in the data set is reduced to 14, which includes redundant and noisy data so feature selection methods are used to reduce the sample size and find the optimal number of features.

3.1.2 Feature selection

As it is discussed in chapter 2, several feature selection methods have been used, and the results are represented as follows.

1. Feature selection with correlation

Based on the correlation heat map shown in Figure 2-2, three pairs of features with the highest correlation coefficient are selected and as is shown in Figure 3-3, the Pearson correlation coefficients for Pressure3pm and Pressure9am, MaxTemp and Temp3pm and MinTemp and Temp9am is 0.96, 0.97 and 0.9 respectively which indicates a high correlation between the features and one of the features in each set must be dropped.

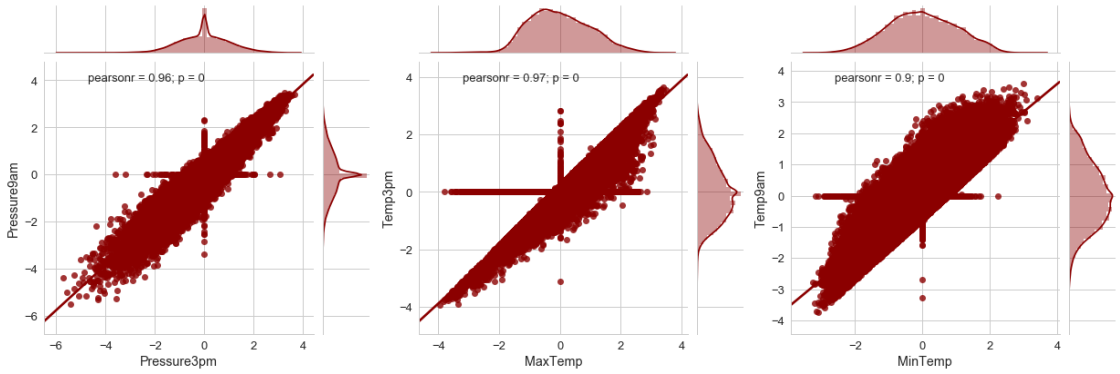


Figure 3-3: Joint plot of highly correlated features

To improve the performance of the classifier based on feature selection using correlation coefficient, the features 'Humidity9am', 'Pressure9am', 'WindSpeed9am' are dropped. The model is trained using RF classifier, and the results are shown in Table 3-5 and Figure 3-4.

Table 3-5: Results – RF classification

Metrics	RF classification
Accuracy	0.84
Precision	0.71
Recall	0.46
F-Measure	0.56
G-Mean	0.66

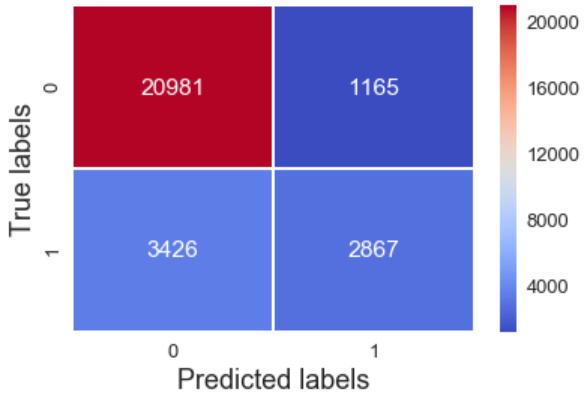


Figure 3-4: Confusion matrix – RF classification

2. Recursive Feature Elimination

As it is discussed in chapter 2, RFE can be implemented with different classification algorithms. The results obtained from RFE with Random Forest and Support Vector Machine as core classifier are presented in Table 3-6 and Figure 3-5 for comparison.

Table 3-6: Results – RFE-RF vs. RFE-SVM classification

Metrics	RFE-RF	RFE-SVM
Accuracy	0.80	0.84
Precision	0.57	0.72
Recall	0.41	0.43
F-Measure	0.48	0.54
G-Mean	0.61	0.64

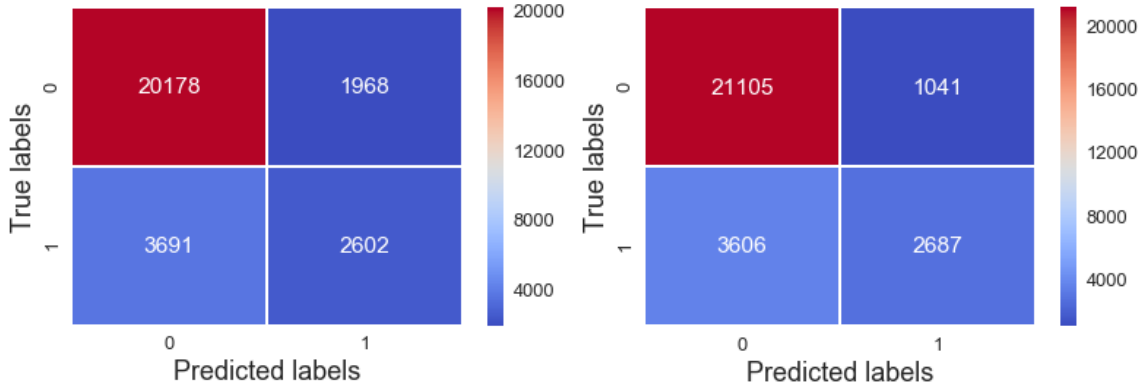


Figure 3-5: Confusion matrix – RFE-RF vs. RFE-SVM classification

Based on the results from the confusion matrices in Table 3-6 and Figure 3-5 the two classifiers have relatively close performance based on the measurement metrics used for evaluation. However, RFE with Random forest at its core is considerably faster. Therefore, it is selected for training in the remaining feature selection and feature extraction methods.

3. Recursive Feature Elimination with Cross-Validation

Recursive feature elimination with 5-fold cross validation using Random Forest is used, to find the optimal number of features for classification, and the results are presented in Figure 3-6.

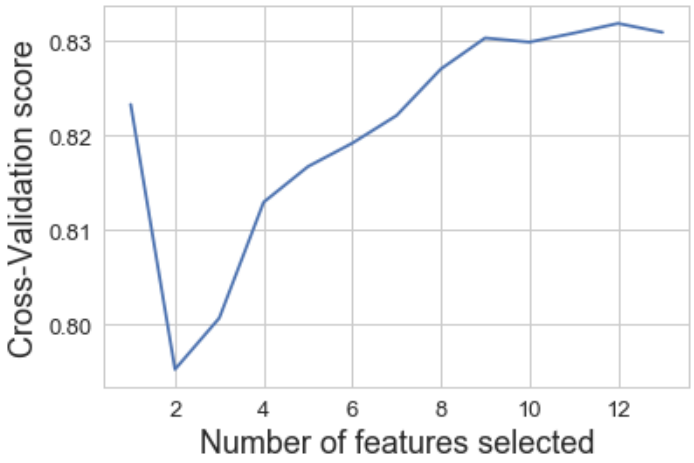


Figure 3-6: RFE-RF with 5-fold cross-validation

4. Tree-based feature selection (Random Forest Feature importance)

Random forest feature importance is a practical tool for most data sets. However, it performs poorly if the dataset includes different types of data or the data varies in their scale, so we have applied the algorithm on scaled data which includes only numeric values.

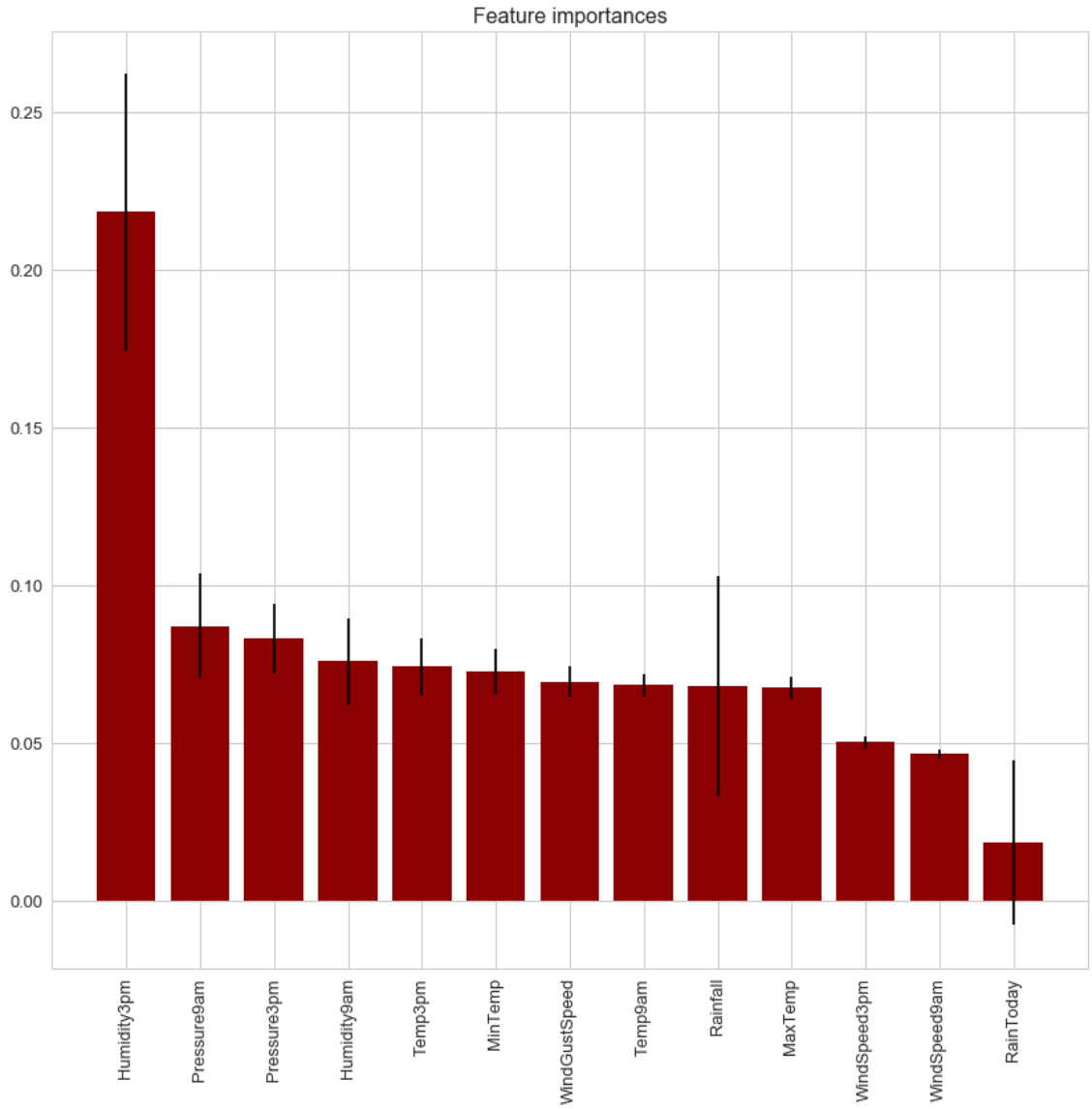


Figure 3-7: Feature importance plot

As it is shown in Figure 3-7, “RainToday” has the lowest importance so it can be discarded. Based on previous analysis, Rainfall has the highest number of outliers, and it is not vital for classification. Thus, they are both dropped from the data set. For comparison, the data is trained before and after dropping the two features, and the results are presented in Table 3-7. The feature first set includes all the features and the second feature set includes all the features except “RainToday” and “Rainfall.”

Table 3-7: Results – Comparison of RF classification of the feature set 1 and 2

Metrics	Feature set 1	Feature set 2
Accuracy	0.84	0.83
Precision	0.70	0.70
Recall	0.45	0.45
F-Measure	0.55	0.55
G-Mean	0.66	0.65

Based on the results presented in Table 3-7, the accuracy of the classifier does not drastically change after the two features were dropped. However, the new subset decreases the computational time so, training/testing feature set two is more efficient, and it is selected for further analysis.

3.1.3 Feature Extraction

From the machine learning aspect, the number of principal components is determined based on different variables such as storage, the capacity of the system, training time, performance of the machine learning algorithm, etc. A sensible way of choosing the best number of principal components is plotting the variance against principal components, as it is shown in Figure 3-8, which shows the minimum number of principal components required to retain %99 variance. And the correlation coefficient map is an indication of low correlation among the new features.

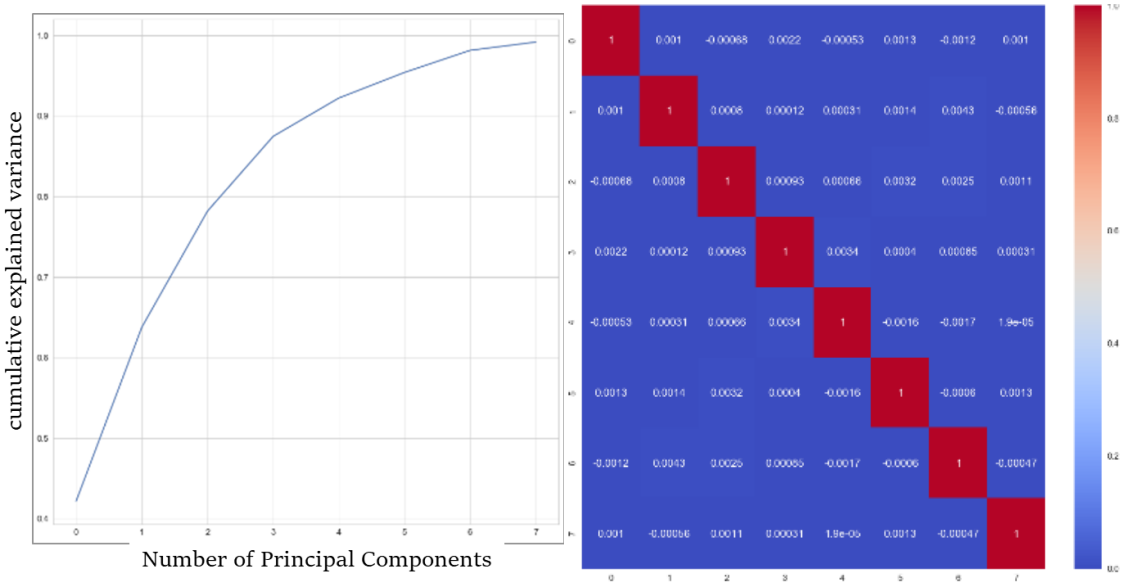


Figure 3-8: Optimal number of principal components and the correlation matrix

To assess the performance of the classifier the constructed data set is trained and tested. The results are provided in Table 3-8 and Figure 3-9.

Table 3-8: Results - RF classification on the new feature set after PCA

Metrics	Results
Accuracy	0.83
Precision	0.71
Recall	0.45
F-Measure	0.55
G-Mean	0.65

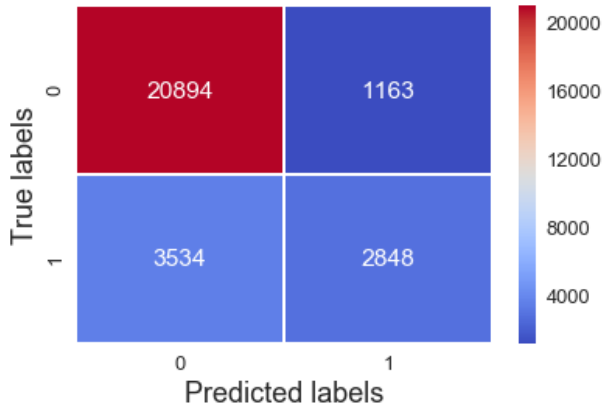


Figure 3-9: Confusion matrix - RF classification on the new feature set after PCA

3.2 Model selection

As it is discussed in chapter 2, no single classifier performs flawlessly for all data sets, so choosing the right classifier can affect the accuracy of results. To this end, ten algorithms that are suitable for classifying large data sets are used to train and test the data, and the results are presented in Table 3-9.

Table 3-9: Model comparison

Model	Accuracy	Precision	Recall	F-Measure	G-Mean
Support Vector Machines_RBF	84.92	77.82	44.53	56.65	65.52
Random Forest	84.49	72.05	48.90	58.26	68.02
Logistic Regression	84.11	71.89	46.31	56.33	66.28
Linear Discriminat Analysis	83.95	70.46	47.33	56.62	66.83
Support Vector Machines-Linear	83.93	73.33	43.07	54.27	64.15
Stochastic Gradient Decent	83.47	76.84	36.26	49.27	59.27
Gaussian Naive Bayes	82.57	66.74	42.34	51.81	63.09
Naive Bayes	82.57	66.74	42.34	51.81	63.09
KNN	82.23	62.11	50.48	55.69	67.87
Support Vector Machines_Poly	80.38	80.38	14.16	24.21	37.48
Perceptron	79.21	54.42	44.25	79.21	58.29
Decision Tree	77.09	48.32	50.75	49.50	65.51

For comparison among the models, the overall performance of the classifier determines if it is the accurate classifier for the data. As is shown in Table 3-9 and Figure 3-10, Support Vector Machine with RBF kernel has the highest accuracy, so SVM is the most suitable algorithm for classifying this data set.

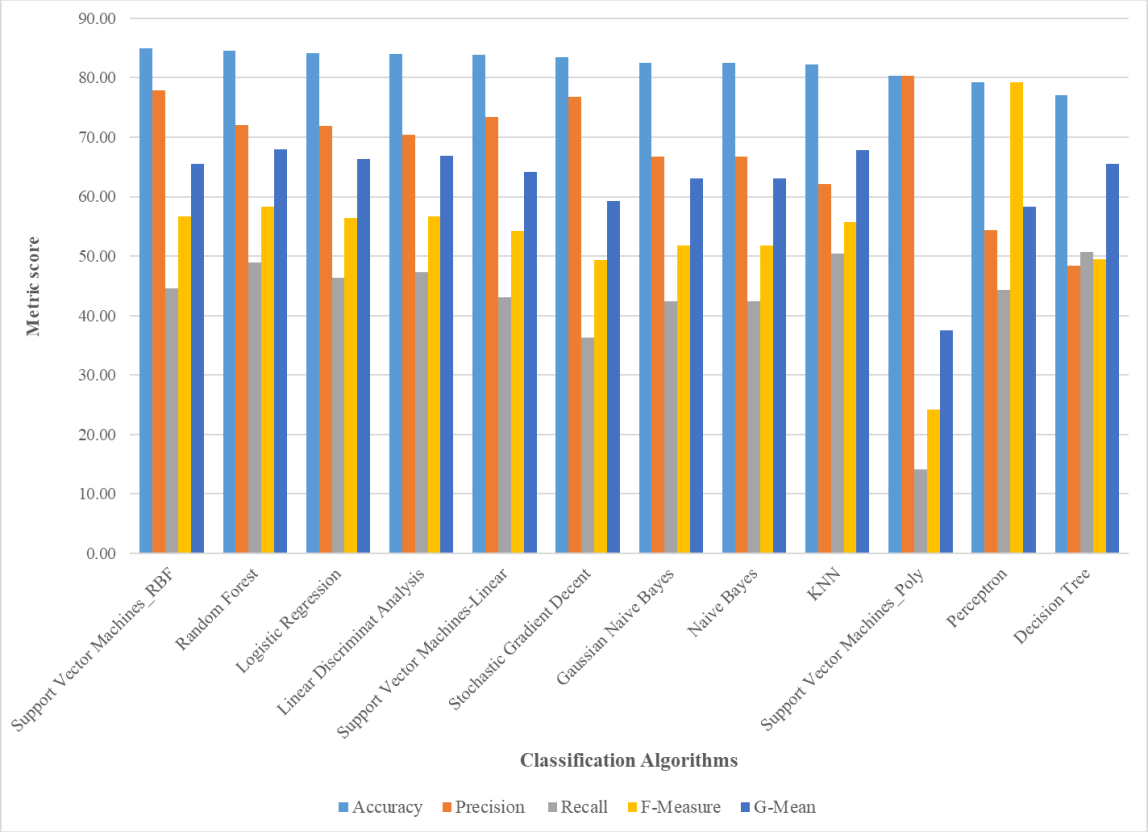


Figure 3-10: Model comparison

3.3 Classification and parameter tuning

After preprocessing the data and selecting the Support Vector Machine as a classifier, we have trained the data using the proposed linear SVM. The model is tuned based on the parameter found using the parametric modeling algorithm proposed in chapter 2. According to the results obtained from the proposed algorithm, the optimal value of $\lambda = 0.9999$, so the objective function of the optimization problem is updated to:

$$Min_{\,w,b,\zeta}\{\, 0.9999 \sum_{i=1}^{l} |w_i| + 0.0001 \sum_{i=1}^{n} \zeta_i \,\}$$

Since the value of λ is obtained from the first iteration of the algorithm, we have trained the first sample data using other values of λ, and the results of the optimization model are presented in Table 3-10.

Table 3-10: Parametric modeling values

Group 1	
λ	Optimal objective
0.9	0.101
0.7	1.495
0.5	2.492
0.3	3.488
0.1	4.484

Based on the results presented in Table 3-10, $\lambda = 0.9999$ has the lowest value of the objective function, which is the optimal value of λ. We have also compared the results from the result obtained from a grid search on the set of C = { 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000} and the best value of C = 10 was found which used to train and test the data. The results are shown in Table 3-12.

Table 3-11: Results – Standard Kernelized SVM

Metrics	Linear	RBF	Polynomial
Accuracy	0.83	0.84	0.81
Precision	0.73	0.77	0.83
Recall	0.43	0.44	0.14
F-Measure	0.54	0.56	0.24
G-Mean	0.64	0.65	0.37

Table 3-12: Results - Kernelized SVM with modified objective function classification

Kernel	Sample	Sample size	Accuracy	Precision	Recall	F-Measure	G-Mean
Linear	Group 1	135081	0.72	0.66	0.86	0.75	0.71
	Group 2	124083	0.71	0.66	0.83	0.74	0.70
	Group 3	134937	0.77	0.73	0.84	0.78	0.77
	Group 4	123435	0.70	0.66	0.82	0.73	0.68
	Group 5	117189	0.70	0.66	0.82	0.73	0.69
RBF	Group 1	135081	0.49	0.49	1.00	0.66	0.00
	Group 2	124083	0.49	0.49	1.00	0.66	0.00
	Group 3	134937	0.48	0.48	1.00	0.65	0.00
	Group 4	123435	0.50	0.50	1.00	0.66	0.00
	Group 5	117189	0.50	0.50	1.00	0.67	0.00
Polynomial	Group 1	135081	0.49	0.49	1.00	0.66	0.00
	Group 2	124083	0.49	0.49	1.00	0.66	0.00
	Group 3	134937	0.48	0.48	1.00	0.65	0.00
	Group 4	123435	0.50	0.50	1.00	0.66	0.00
	Group 5	117189	0.50	0.50	1.00	0.67	0.00

Based on the results presented in Table 3-12, the proposed kernel SVM with adjustments to the objective function of the training algorithm performs well on classifying the minority class. However, the results obtained from running the algorithm on the data reveals that SVM with RBF and second-degree polynomial kernel tend to overfit the data, which doesn't affect the performance of the classifier on the minority class, but the majority class is misclassified. As is shown in the confusion matrices and ROC curves in Figure 3-11and Figure 3-12, this problem presents itself in the G-mean that is equal to zero and the ROC curve where the area under the curve is equal to zero.

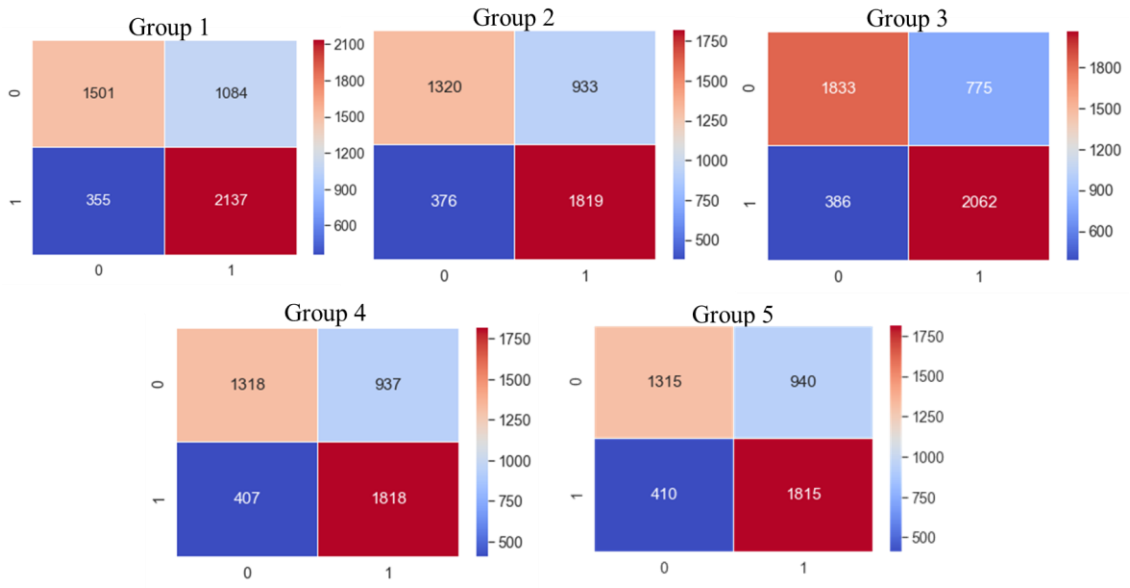


Figure 3-11: Confusion matrices of modified SVM classification

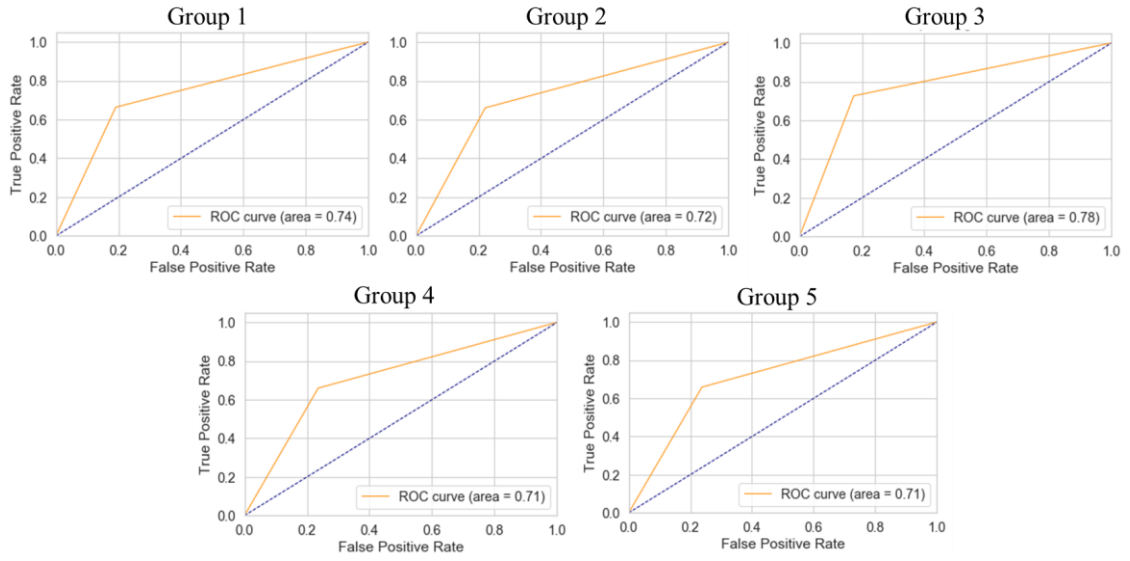


Figure 3-12: ROC curves

To avoid this issue, the use of a linear kernel is proposed and implemented, and the results obtained from training and testing the data proves that the proposed SVM with linear kernel provides enough generalization to perform equally well on both classes. The AUC score derived from the ROC curve presented in Figure 3-12 also shows the

improvement in classifying the test data in both classes in comparison with standard Support Vector Machine classifier, with a quadratic objective function.

As it is mentioned before, the meaningful range of AUC score is between 0.5 and 1 and the closer we get to 1, the better the classifier is. This score is important because it also determines the probability of ranking an unknown instance positive which in the case of our work, is the probability of having rain. So, the probability of predicting a randomly selected instance as positive is 0.7 which is a reasonable result for an imbalanced learning problem.

To evaluate the results 5-fold cross validation is also used. Due to time limitations, we have applied 5-fold cross-validation on the first group of data, and the results are presented in Table 3-13, which shows a lower average G-mean score. The lower score is due to the fact that rainy observations occur at different time intervals. Randomly selecting the train/test data sets can provide less correlation between the observations and overall higher scores comparing to cross-validation in which the data is divided into five sections, and at each iteration, one is selected as testing data, and the remaining is the training data.

Table 3-13: Cross-Validation

	Group 1				
k-Fold	**Accuracy**	**Precision**	**Recall**	**F-Measure**	**G-Mean**
Round 1	0.51	0.15	0.90	0.25	0.65
Round 2	0.43	0.23	0.94	0.36	0.54
Round 3	0.38	0.26	0.91	0.41	0.45
Round 4	0.84	1.00	0.84	0.92	0.00
Round 5	0.84	1.00	0.84	0.91	0.00
Average	0.60	0.53	0.89	0.57	0.33

To improve the results of cross-validation we have randomized the data before 5-fold cross-validation and as it is shown in the scores are closer to what we have with randomly splitting the data.

Table 3-14: Cross-Validation with randomness

	Group 1				
k-Fold	**Accuracy**	**Precision**	**Recall**	**F-Measure**	**G-Mean**
Round 1	0.72	0.67	0.87	0.76	0.71
Round 2	0.73	0.68	0.87	0.77	0.72
Round 3	0.74	0.70	0.87	0.78	0.73
Round 4	0.71	0.66	0.84	0.74	0.70
Round 5	0.73	0.68	0.87	0.77	0.72
Average	0.73	0.68	0.87	0.76	0.71

As for other metrics, having the relatively close performance of the classifier for both classes is an indication that the minority class is not ignored, and without loss of generalization, this approach could be used for imbalanced data sets.

4 Conclusion and future work

Two main contributions towards improving the classification of imbalanced data are:

1. Introducing an algorithmic approach for finding the optimal regularization parameter in Classification algorithms such as Support Vector Machines. This approach is based on parametric simplex optimization that searches the solution path and provides a finite number of values that could be used as a possible regularization parameter. The most important advantage of this method is that it is built upon a solid mathematical background. It is also considerably faster than grid search.

2. Developing a linear Support Vector Machine with L1-norm objective function based on principles of multi-criteria optimization that can effectively predict the minority class as well as the majority class. This modified SVM can decrease the computational complexity associated with solving a quadratic optimization problem, and it can be solved using the existing software suitable for convex LP problems.

There are also other things that we can consider to further improve the results. For example, training the data with smaller number of features before PCA could result in less overfitting. It is also a good idea to study the rotation of Principal Components [41], which could result in better interpretation of PCs and data reduction.

In Evaluation, due to the application of LP SVM in weather data. comparison of the results with evaluation metrics used in meteorology could also provide a better understanding of the results.

As for future work, despite the advances made in data analytics, there are still various opportunities for research projects to be carried out to provide insight into the machine learning algorithms. Large scale problems are becoming more popular, and we will continue to face the challenges associated with them.

For the parametric simplex method introduced for parametric modeling, we propose implementing this method for identifying the hyperparameters in different algorithms other than SVM, as it can reduce the computational complexity and effectively improve the parameter tuning in statistical learning.

As for imbalanced learning, we tend to combine the proposed method with weighted SVM and explore how the algorithmic approach can work in combination with data-related methods to improve classification performance.

References

[1] B. Krawczyk, “Learning from imbalanced data : open challenges and future directions,” *Prog. Artif. Intell.*, vol. 5, no. 4, pp. 221–232, 2016.

[2] G. M. Weiss, *Foundations of Imbalanced Learning*. Wiley Online Library, 2012.

[3] K. Veropoulos, C. Campbell, and N. Cristianini, “Controlling the Sensitivity of Support Vector Machines,” *Proc. Int. Jt. Conf. Artif. Intell. Stock. Sweden*, pp. 55–60, 1999.

[4] I. Journal and C. Science, “Class Imbalance Problem in Data Mining : Review,” vol. 2, no. 1, 2013.

[5] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, “Machine learning: a review of classification and combining techniques,” *Artif. Intell. Rev.*, vol. 26, no. 3, pp. 159–190, Nov. 2006.

[6] M. Maalouf and T. B. Trafalis, “Robust weighted kernel logistic regression in imbalanced and rare events data,” *Comput. Stat. Data Anal.*, vol. 55, no. 1, pp. 168–183, Jan. 2011.

[7] D. P. Agrawal, “Welcome message from the editor-in-chief,” *Open Computer Science*, vol. 1, no. 1, Walter de Gruyter GmbH, p. 1, 01-Mar-2011.

[8] C. Marzban and G. J. Stumpf, “A Neural Network for Tornado Prediction Based on Doppler Radar-Derived Attributes,” *J. Appl. Meteorol.*, vol. 35, no. 5, pp. 617–626, May 2002.

[9] G. J. Stumpf and A. Witt, “A Neural Network for Detecting and Diagnosing Tornadic Circulations using the Mesocyclone Detection and Near Storm Environment Algorithms,” no. Bishop 1995, pp. 1–17, 2004.

[10] H. Saeidi-Manesh and G. Zhang, “Cross-polarization and sidelobe suppression in dual linear polarization antenna arrays,” *Electron. Lett.*, vol. 53, no. 9, pp. 577–578, 2017.

[11] H. Saeidi-Manesh, M. Mirmozafari, and G. Zhang, “Low cross-polarisation high-isolation frequency scanning aperture coupled microstrip patch antenna array with matched dual-polarisation radiation patterns,” *Electron. Lett.*, vol. 53, no. 14, pp. 901–902, 2017.

[12] H. Saeidi-Manesh and G. Zhang, “Low Cross-Polarization, High-Isolation Microstrip Patch Antenna Array for Multi-Mission Applications,” *IEEE Access*, vol. 7, pp. 5026–5033, 2019.

[13] H. Saeidi-Manesh, S. Karimkashi, G. Zhang, and R. J. Doviak, “High-Isolation Low Cross-Polarization Phased-Array Antenna for MPAR Application,” *Radio Sci.*, vol. 52, no. 12, pp. 1544–1557, 2017.

[14] H. Saeidi-Manesh and G. Zhang, “High-Isolation, Low Cross-Polarization, Dual- Polarization, Hybrid Feed Microstrip Patch Array Antenna for MPAR Application,” *IEEE Trans. Antennas Propag.*, vol. 66, no. 5, pp. 2326–2332, 2018.

[15] N. V. Chawla, “Data Mining for Imbalanced Datasets: An Overview,” *Data Min. Knowl. Discov. Handb.*, pp. 875–886, 2010.

[16] H. Haibo and E. A. Garcia, “Learning from Imbalanced Data (重重点),” *Knowl. Data Eng. IEEE Trans.*, vol. 21, no. 9, pp. 1263–1284, 2009.

[17] S. M. Abd Elrahman and A. Abraham, “A Review of Class Imbalance Problem,” *J. Netw. Innov. Comput.*, vol. 1, pp. 332–340, 2013.

[18] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, “An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics,” *Inf. Sci. (Ny).*, vol. 250, pp. 113–141, Nov. 2013.

[19] T. B. Trafalis, I. Adrianto, M. B. Richman, and S. Lakshmivarahan, “Machine-learning classifiers for imbalanced tornado data,” *Comput. Manag. Sci.*, vol. 11, no. 4, pp. 403–418, Oct. 2014.

[20] H. Zou and M. Yuan, “the F ∞ -Norm Support Vector Machine,” *Stat. Sin.*, vol. 18, pp. 379–398, 2008.

[21] W. Zhou, L. Zhang, and L. Jiao, “Linear programming support vector

machines," *Pattern Recognit.*, vol. 35, pp. 2927–2936, 2002.

[22] K. P. Bennett and C. Campbell, "Support vector machines: Hype or Hallelujah?," *J. Chem. Inf. Model.*, vol. Volume 2, no. Issue 2, pp. 1–13, 2000.

[23] A. Aşkan and S. Sayın, "SVM classification for imbalanced data sets using a multiobjective optimization framework," *Ann. Oper. Res.*, vol. 216, no. 1, pp. 191–203, May 2014.

[24] P. Soda, "A multi-objective optimisation approach for class imbalance learning," *Pattern Recognit.*, vol. 44, no. 8, pp. 1801–1810, 2011.

[25] T. Suttorp and C. Igel, "Multi-Objective Optimization of Support Vector Machines LOKI Learning to organize complex systems View project Music Classification on Personal Mobile Devices View project Multi-objective optimization of support vector machines," vol. 16, pp. 199–220, 2006.

[26] S. Datta and S. Das, "Multiobjective Support Vector Machines: Handling Class Imbalance With Pareto Optimality," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 5, pp. 1602–1608, 2018.

[27] M. B. Richman, T. B. Trafalis, and I. Adrianto, "Missing Data Imputation Through Machine Learning Algorithms," in *Artificial Intelligence Methods in the Environmental Sciences*, Dordrecht:

Springer Netherlands, 2009, pp. 153–169.

[28] J. Miao and L. Niu, “A Survey on Feature Selection,” *Procedia - Procedia Comput. Sci.*, vol. 91, no. Itqm, pp. 919–926, 2016.

[29] I. Guyon, “An Introduction to Variable and Feature Selection 1 Introduction,” vol. 3, pp. 1157–1182, 2003.

[30] I. Guyon and A. Elisseeff, “An Introduction to Variable and Feature Selection,” *J. Mach. Learn. Res.*, vol. 3, no. March, pp. 1157–1182, 2011.

[31] M. J. Zaki and W. J. Meira, *Data Mining and Analysis: fundamental concepts and algorithms.* Cambridge University Press, 2014.

[32] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Journal of machine learning research : JMLR.,” *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.

[33] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *J. Artif. Intell. Res.*, 2002.

[34] G. Kim, B. Kevin, and C. David, “A support vector machine (SVM) approach to imbalanced datasets of customer responses : comparison with other customer response models,” pp. 167–182, 2013.

[35] B. Scho, “Support Vector Machines and Kernels for Computational

Biology," vol. 4, no. 10, 2008.

[36] H. Pang, Z. Tuo, V. Robert, and L. Han, "A Parametric Simplex Approach to Statistical Learning Problems," *Unpubl. manuscript. http//www. princeton. edu/*, 2015.

[37] N. D. Dan and L. E. D. Muu, "A parametric simplex method for optimizing a linear function over the efficient set of a bicriteria linear problem," vol. 21, no. 1, pp. 59–67, 1996.

[38] T. M. Rassias and A. Bacopoulos, *Multicriteria Optimization*. Springer-Verlag Berlin Heidelberg, 2012.

[39] G. Mavrotas, "Generation of efficient solutions in Multiobjective Mathematical Programming problems using GAMS . Effective implementation of the ε -constraint method 1 . Multiobjective Mathematical Programming and efficient solutions 2 . Classification of the MMP metho," no. x.

[40] J. S. Baras, "B-ROC Curves for the Assessment of Classifiers over Imbalanced Data Sets B-ROC Curves," pp. 1581–1584, 2006.

[41] M. B. Richman, "Rotation of principal components," *J. Climatol.*, vol. 6, no. 3, pp. 293–335, 1986.