UNIVERSITY OF OKLAHOMA GRADUATE COLLEGE

HIERARCHICAL SHRINKAGE IN TREE-BASED MODELS APPLIED TO IMAGE, HIGH-DIMENSIONAL, AND NOISY DATA

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By LUIS EDUARDO VAZQUEZ

Norman, Oklahoma

2025

HIERARCHICAL SHRINKAGE IN TREE-BASED MODELS APPLIED TO IMAGE, HIGH-DIMENSIONAL, AND NOISY DATA

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY THE COMMITTEE CONSISTING OF

Dr. Dimitrios I. Diochnos, Chair

Dr. Richard Veras

Dr. Andrew H. Fagg

Table of Contents

Abstract

Machine learning models often face a trade-off between predictive performance and interpretability—an issue of growing importance as AI systems become increasingly embedded in critical domains such as healthcare and finance. Tree-based models, like decision trees and random forests, are widely used due to their interpretability, but they are also known to overfit, especially in high-dimensional or imbalanced settings. This thesis investigates the use of *hierarchical shrinkage*, a recently proposed post-hoc regularization technique, as a way to improve the generalization of decision tree-based models without compromising interpretability.

The work of Agarwal et al. [1] focused on regression as well as on some tabular datasets for binary classification. This motivated me to extended the evaluation of hierarchical shrinkage to new settings, not explored by Agarwal et al. [1]. First, I tested hierarchical shrinkage on image classification tasks. Our results indicate that, unlike in the tabular setting, hierarchical shrinkage does not improve performance on image data—regardless of whether dimensionality reduction via Principal Component Analysis (PCA) is applied, or not—and this holds for both decision trees and random forests.

Next, I expanded the investigation to a broader range of tabular datasets, including both binary and multi-class problems. Our findings confirm that hierarchical shrinkage consistently improves performance on binary classification datasets and continues to offer improvements, though to a lesser extent, on multi-class datasets. I also observe that hierarchical shrinkage yields more substantial gains on high-dimensional tabular datasets. These trends hold across both individual decision trees and random forests.

I further introduce synthetic label noise and apply confident learning to identify and denoise mislabeled instances. While confident learning leads to improved performance in some cases, especially when the noise is high, our experiments suggest that the impact of confident learning varies depending on the dataset and noise level. Moreover, some of the results that we obtain under noise are conflicting with the results that we obtain in the

no-noise setting, thus indicating that more experimentation is needed here so that some safe conclusions can be drawn.

Overall, this thesis provides a comprehensive evaluation of hierarchical shrinkage under a variety of learning conditions. Our results suggest that hierarchical shrinkage is a promising, interpretable regularization method for decision tree-based models.

Chapter 1

Introduction

1.1 Introduction

Machine learning algorithms have demonstrated exceptional performance across diverse domains, from medical diagnostics to financial forecasting. However, there is often a trade-off between performance and interpretability, a dilemma that has gained increasing attention as artificial intelligence (AI) systems become more integrated into decision-making processes. In high-stakes fields such as healthcare and finance, where algorithmic predictions influence critical outcomes, the ability to understand and trust these models is paramount. A lack of interpretability can hinder adoption, introduce ethical concerns, and create regulatory challenges, making it essential to develop models that are both accurate and explainable.

Interpretability is not only important for transparency but also plays a crucial role in improving machine learning models themselves. Understanding how models arrive at their predictions can lead to better insights, refined feature engineering, and more effective algorithmic improvements. With the rapid expansion of AI applications, the demand for interpretable models is growing, prompting researchers to explore methodologies that balance predictive power with human comprehensibility.

Tree-based models, such as decision trees, have been widely studied and are often re-

garded as inherently interpretable due to their simple, rule-based structure. A decision tree's hierarchical nature allows for clear visualization and straightforward decision rules, making it easier for practitioners and domain experts to understand the reasoning behind predictions. However, in practice, large and complex trees can lose their interpretability, resembling black-box models rather than intuitive decision aids. Additionally, decision trees are highly prone to overfitting, capturing noise instead of meaningful patterns, which limits their generalization to new data.

To address these issues, regularization techniques have been developed to enhance the interpretability and stability of tree-based models. Various approaches, both ad hoc and post hoc, have been introduced to control tree complexity. Pre-pruning methods, such as depth constraints and minimum split requirements, attempt to prevent trees from growing excessively large. Post-pruning, on the other hand, involves simplifying an already trained tree by removing branches that contribute little to predictive performance. Beyond direct modifications to tree structures, regularization-inspired ideas—such as penalizing tree complexity or incorporating cost-complexity pruning—have been used to mitigate overfitting while maintaining robust performance. Ensemble techniques like bagging and boosting have also been employed to reduce variance and improve generalization [2, 16].

To address these issues, regularization techniques have been developed to enhance the interpretability and stability of tree-based models. Various approaches, both ad hoc and post hoc, have been introduced to control tree complexity. Pre-pruning methods, such as depth constraints and minimum split requirements, attempt to prevent trees from growing excessively large. Post-pruning, on the other hand, involves simplifying an already trained tree by removing branches that contribute little to predictive performance. Beyond direct modifications to tree structures,

This thesis explores hierarchical shrinkage, a promising post-hoc regularization technique designed to improve decision trees by imposing structured constraints on model complexity. Hierarchical shrinkage offers a more nuanced approach to regularization by systematically

controlling tree depth and node-level influence in a way that enhances both predictive power and interpretability. By leveraging hierarchical relationships within the tree structure, this method provides an alternative to conventional pruning and penalization techniques, aligning with the broader goal of making machine learning models both effective and transparent.

1.2 Research Questions and Hypotheses

This thesis seeks to address the following research questions:

- Can hierarchical shrinkage improve the performance and generalization of decision tree-based models on datasets that go beyond tabular setting—such as image-based, high-dimensional, and noisy data?
- How does dimensionality reduction via Principal Component Analysis (PCA) interact with hierarchical shrinkage in terms of model accuracy and efficiency?
- Does hierarchical shrinkage benefit both individual decision trees and ensemble methods like random forests, and does its effectiveness vary with dataset characteristics such as binary vs multi-class datasets?

The following hypotheses are tested:

- **H1:** Hierarchical shrinkage will improve classification performance on high-dimensional tabular datasets by mitigating overfitting and enhancing generalization.
- **H2:** Hierarchical shrinkage will not significantly improve performance on image classification tasks, even when PCA is applied.
- **H3:** The benefits of hierarchical shrinkage will be more pronounced for binary classification tasks than for multi-class classification tasks.
- **H4:** Hierarchical shrinkage will remain effective when applied to ensemble methods like random forests.

- **H5:** Applying confident learning to noisy datasets will improve performance, and the combination of confident learning with hierarchical shrinkage will yield more robust models under label noise.

1.3 Contributions

The main contributions of this thesis are summarized below. All code, datasets, and experimental results are publicly available at:

https://github.com/luisvazq98/HierarchicalShrinkage.

- I validate and reproduce the experimental results from the hierarchical shrinkage paper by Agarwal et al. [1], using the `imodels` Python library.
- While the work by Agarwal et al. [1] focused exclusively on tabular data, I extended the analysis to the domain of image classification. Our findings indicate that—contrary to the tabular case—hierarchical shrinkage does not improve performance for image data using decision trees or random forests. This result holds true both with and without applying PCA for dimensionality reduction
- I evaluate hierarchical shrinkage on additional tabular datasets beyond those considered by Agarwal et al. [1]. Our results confirm that HS consistently improves model performance in these settings—especially for binary classification tasks, and to a lesser extent for multi-class datasets. This pattern holds across both decision trees and random forests..
- I study the effect of dimensionality reduction using PCA and analyze how it impacts accuracy, training time, and the interaction with hierarchical shrinkage regularization.
- I incorporate confident learning to address the presence of noisy labels. I introduce synthetic noise into selected datasets and apply confident learning to identify misla-

beled samples. While denoising led to some improvement, especially under high noise levels, the overall impact of confident learning was dataset-dependent and inconclusive.

- I provide detailed comparative analysis against baseline models including standard decision trees, random forests, and their PCA-preprocessed versions—highlighting both predictive accuracy and computational efficiency.

1.4 Outline of the Thesis

The next sections are organized as follows. Chapter 2 details related work in regards to decision trees and regularization methods. Chapter 3 goes over some background information. Chapter 4 showcases the experiments conducted. Chapter 5 provides discussion on the results obtained. Finally, Chapter 6 concludes with final thoughts.

Chapter 2

Related Work

2.1 Related Work

Decision trees have long been a fundamental tool in machine learning and statistical analysis, providing an intuitive, rule-based approach to classification and regression. The broader societal push toward automated decision making in business, healthcare, and engineering fueled their adoption, as they provided transparency and interpretability, qualities essential for high-stakes decision environments. In the 1980s, methods such as the ID3 algorithm and its successors, C4.5 and CART, established formal frameworks for learning decision trees from data, introducing recursive partitioning as a primary mechanism for structure formation[4, 16].

However, as datasets grew in complexity and volume, a major challenge emerged: overfitting. Decision trees, by their very nature, tend to fit training data too closely, capturing noise rather than underlying patterns. This issue became particularly pronounced in the late 1990s and early 2000s, as machine learning applications expanded to domains such as finance, bioinformatics, and image recognition[8]. The need for models that generalized well to unseen data led to the widespread exploration of regularization techniques, which aimed to control model complexity and enhance robustness.

Ensemble methods, such as bagging and boosting, emerged as critical solutions to decision tree overfitting, leveraging multiple models to reduce variance and bias. Random forests, introduced by Breiman in 2001[3], became a powerful alternative by averaging multiple decision trees trained on random subsets of data. Simultaneously, boosting techniques such as AdaBoost and later gradient boosting demonstrated significant performance gains by iteratively refining weak learners [5, 6].

As machine learning models grew increasingly sophisticated, regularization techniques evolved beyond simple pruning strategies to more structured and theoretically grounded methods. The introduction of penalization-based approaches, such as LASSO and ridge regression, influenced the development of regularized decision trees, emphasizing the importance of controlling model complexity in high-dimensional settings [10, 18]. While these methods were originally designed for linear models, their core idea—adding a penalty to discourage complexity—has inspired analogous strategies in tree-based models, such as penalizing tree depth or node splits to prevent overfitting in high-dimensional data. Within this landscape, hierarchical shrinkage has emerged as a promising framework for improving tree-based models[1]. By incorporating structured regularization, hierarchical shrinkage seeks to address the limitations of conventional pruning and penalization techniques by imposing constraints at multiple levels of the model. The development of hierarchical shrinkage reflects ongoing efforts to refine decision tree methodologies in response to modern challenges of ensuring that models remain both powerful and interpretable as machine learning applications continue to expand.

Chapter 3

Background

3.1 Background

This section provides a technical overview of key concepts relevant to this research. It covers decision trees as a fundamental machine learning model and discusses overfitting along with regularization techniques to improve generalization performance.

3.2 Decision Trees

Decision trees are a widely used supervised learning algorithm for both classification and regression tasks. They operate by recursively partitioning the input feature space into disjoint regions based on feature values, ultimately forming a hierarchical structure that can be used to make predictions.

A decision tree is composed of internal nodes representing feature-based decision rules, branches corresponding to outcomes of these rules, and leaf nodes that contain the final predictions. The splitting criteria at each node are chosen to maximize information gain or minimize impurity. Common impurity measures include entropy (used in ID3 and C4.5 [15, 16]) and Gini impurity (used in CART [4]).

Popular decision tree algorithms include:

- **ID3 (Iterative Dichotomiser 3)**: Uses entropy and information gain to determine the best splits [15].
- **C4.5**: An extension of ID3 that supports continuous feature values and handles missing data [16].
- **CART (Classification and Regression Trees)**: Uses the Gini impurity metric for classification and supports regression tasks by minimizing variance [4].

Despite their interpretability and ease of use, decision trees are prone to overfitting, especially when deep trees are grown without constraints. Techniques such as pruning, setting a minimum number of samples per split, and using ensemble methods (e.g., random forests) are commonly employed to mitigate this issue.

3.3 Overfitting and Regularization

Overfitting is a common issue that can occur during the training of a machine learning algorithm. This happens when the algorithm fits too closely to the training data causing the algorithm to not generalize well to unseen data. Regularization techniques are often used to prevent overfitting and which techniques to use depends on the algorithm being used as well as the dataset. Overfitting is particularly problematic for high-capacity models such as deep decision trees or complex neural networks.

Regularization techniques aim to reduce overfitting by imposing constraints on the learning process or modifying the model structure. The choice of regularization technique depends on the type of model being used. Some common regularization methods include:

- Pruning (for decision trees): Involves removing branches that do not contribute significantly to predictive performance, reducing tree depth and complexity.
- L1 and L2 Regularization: Applied in models such as logistic regression and neural networks, where L1 (Lasso) encourages sparsity, and L2 (Ridge) penalizes large weights.

- Early Stopping: Monitors validation performance and halts training when overfitting begins.
- Cross-validation: Ensures model performance is evaluated on multiple subsets of data, providing a better estimate of its generalization capability.

In decision trees, overfitting is typically controlled through various regularization techniques that constrain the model's structure. These include limiting the maximum tree depth, setting a minimum number of samples required to split a node, requiring a minimum number of samples per leaf, and restricting the total number of leaf nodes. Additionally, ensemble methods such as random forests and gradient boosting aggregate multiple trees to enhance robustness and generalization.

By carefully applying regularization techniques, models can maintain high accuracy while ensuring better performance on real-world data.

3.4 Hierarchical Shrinkage as a Regularization Technique

Hierarchical shrinkage is a recently proposed post-hoc regularization method designed to improve the accuracy and interpretability of decision trees and tree-based models. Introduced by Agarwal et al. [1], the method responds to a well-known issue in decision tree learning: as trees grow deeper and more complex, they tend to overfit the training data and become harder to interpret. Traditional approaches to regularization, such as pruning or limiting depth, attempt to prevent this by restricting tree complexity during training. Hierarchical Shrinkage, by contrast, takes a different approach—it modifies a fully grown tree after training, by shrinking the predictions at each node toward those of its ancestors in a structured, hierarchical way.

Notation and Setup. Let the training dataset be defined as, $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where each input $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional feature vector and $y_i \in \mathbb{R}$ is the corresponding response variable. Assume we have trained a regression decision tree $\hat{f}$ on this dataset.

For any new input $\mathbf{x}$, let the sequence of nodes along its path from the leaf to the root be denoted as, $t_L \subset t_{L-1} \subset \cdots \subset t_0$,, where t_L is the leaf node and t_0 is the root. Each t_ℓ for $\ell = 0, \ldots, L$ represents an internal node along this path. Let $N(t_\ell)$ denote the number of training samples falling into node t_ℓ, and let $\hat{\mathbb{E}}_{t_\ell}\{y\}$ denote the empirical mean of the target variable y for the samples in node t_ℓ.

Standard Tree Prediction. In a standard decision tree, the prediction for a point $\mathbf{x}$ is simply the mean response of the leaf node it falls into: $\hat{f}(\mathbf{x}) = \hat{\mathbb{E}}_{t_L}\{y\}$. This prediction can also be expressed as a telescoping sum, which decomposes the leaf prediction into a series of incremental contributions from the root to the leaf:

$$\hat{f}(\mathbf{x}) = \hat{\mathbb{E}}_{t_0}\{y\} + \sum_{\ell=1}^{L} \left(\hat{\mathbb{E}}_{t_\ell}\{y\} - \hat{\mathbb{E}}_{t_{\ell-1}}\{y\} \right). \tag{3.1}$$

This alternative formulation lays the groundwork for hierarchical shrinkage by identifying how much each split in the tree contributes to the final prediction.

Hierarchical Shrinkage Prediction. Hierarchical shrinkage modifies this decomposition by scaling down each incremental term in proportion to the reliability of the parent node. The regularized prediction is given by

$$\hat{f}_\lambda(\mathbf{x}) = \hat{\mathbb{E}}_{t_0}\{y\} + \sum_{\ell=1}^{L} \frac{\hat{\mathbb{E}}_{t_\ell}\{y\} - \hat{\mathbb{E}}_{t_{\ell-1}}\{y\}}{1 + \lambda / N(t_{\ell-1})}. \tag{3.2}$$

Where $\lambda > 0$ is a user-defined shrinkage hyperparameter. The term $1 + \lambda/N(t_{\ell-1})$ acts as a shrinkage factor: it reduces the influence of deeper nodes that are based on fewer samples by shrinking their contributions toward their parent's estimate. As a result, the prediction

becomes smoother and more stable, particularly in branches of the tree with sparse data.

3.4.1 Applicability to Classification

While the previous derivation is presented in the context of regression—where the prediction $\hat{f}(\mathbf{x})$ is a scalar-valued output—the hierarchical shrinkage framework extends naturally to classification tasks. In classification, the label y is categorical, taking values from a finite set of C classes, i.e., $y \in \{1, 2, \ldots, C\}$. Rather than predicting a single scalar, a classification tree at each node maintains an empirical distribution over class labels.

Formally, for any node t_ℓ, the prediction $\hat{\mathbb{E}}_{t_\ell}\{y\} \in \mathbb{R}^C$ becomes a probability vector, where each component $\hat{p}_{t_\ell,c}$ represents the estimated probability of class c based on the training data that falls into node t_ℓ. For example, if 100 samples reach a node and the class counts are: 10 from class 1, 50 from class 2, and 40 from class 3, then: $\hat{\mathbb{E}}_{t_\ell}\{y\} = [0.10, 0.50, 0.40]$. Hierarchical shrinkage is applied component-wise to these class probabilities. For each class $c \in \{1, \ldots, C\}$, the shrunk probability at the leaf node is given by:

$$\hat{p}_{\lambda,c}(\mathbf{x}) = \hat{p}_{t_0,c} + \sum_{\ell=1}^{L} \frac{\hat{p}_{t_\ell,c} - \hat{p}_{t_{\ell-1},c}}{1 + \lambda / N(t_{\ell-1})}, \tag{3.3}$$

where $N(t_{\ell-1})$ is the number of samples in the parent node and $\lambda > 0$ is the shrinkage parameter. This formulation ensures that probabilities estimated at deeper, less reliable nodes are smoothed toward their ancestors' more stable estimates. The final class prediction is then made by selecting the class with the highest shrunk probability, $\hat{y} = \arg\max_c \hat{p}_{\lambda,c}(\mathbf{x})$.

3.4.2 Interpretation

This formula implements a data-adaptive regularization: when a node has few training samples, its contribution to the final prediction is shrunk toward its ancestor's value. This reduces the variance of predictions, especially in deep or data-sparse regions of the tree, while

preserving the original tree structure. The result is a smoother, more stable function that generalizes better without requiring structural pruning. Thus, with a higher lambda value (e.g., 10, 50, 100), the model relies more on higher-level nodes and trusts deeper nodes less. Lower lambda values (e.g., 0.1, 0.01, 0), the final prediction is nearly identical to that of the original decision tree, with a lambda value of 0 meaning no shrinkage is applied. Figure 3.1 illustrates how this shrinkage acts on node-level predictions, attenuating changes introduced by low-confidence splits deeper in the tree.

The motivation behind hierarchical shrinkage is that predictions made at deeper nodes in a tree are often based on fewer training samples and are therefore more uncertain. Rather than simply deleting these branches (as pruning does), hierarchical shrinkage attenuates their influence by shrinking their output toward the predictions of higher-level nodes, which are statistically more reliable. This process creates a smoother, more stable model that retains the structure of the original tree while improving generalization.

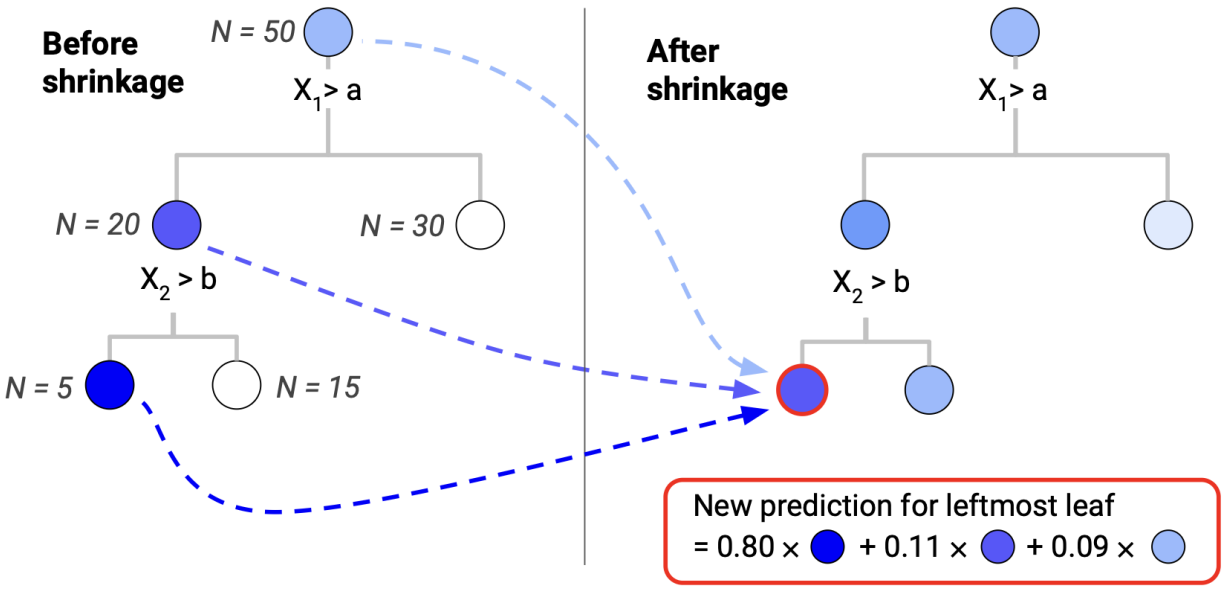


Figure 3.1: Figure taken from Agarwal et al. [1]. As we can see, the prediction of the tree is regularized using information of the examples that are found in the ancestors of the leaf, in the path from the root to the leaf of the tree.

In essence, hierarchical shrinkage offers a principled, post-training adjustment to deci-

sion trees, grounded in statistical shrinkage ideas, that addresses the overfitting issue without compromising interpretability. Unlike pruning, which simplifies trees by entirely removing subtrees, hierarchical shrinkage retains the full structure of the tree and instead scales the contribution of each level in a way that reflects the reliability of the estimates. This preserves the interpretability of decision paths and enables practitioners to trace how each node contributes to a prediction.

The shrinkage factor, which depends inversely on the number of samples in the parent node, allows the model to adaptively trust deeper node estimates less when they are based on small sample sizes. As a result, predictions made in sparsely populated regions of the feature space are pulled toward the more stable, aggregated values of higher-level nodes. This hierarchical adjustment reduces variance without introducing significant bias, leading to better generalization on unseen data.

As demonstrated in the by Agarwal et al. [1], this method consistently improves performance across a range of tabular datasets. Because it modifies only the final predictions and not the learned structure or decision boundaries, hierarchical shrinkage provides an attractive regularization method for scenarios where model transparency and post-hoc interpretability are essential.

3.5 Working with Image Data Sets

Image datasets often require extensive preprocessing to ensure that machine learning models can learn effectively and efficiently. This is because raw images may vary in size, scale, and formats, which can introduce unwanted variability and hinder the model's ability to learn consistent patterns. Images are often reduced in size to reduce the training time of these models due to the often high-dimensionality seen in image data. Preprocessing steps such as resizing, normalization, and centering help standardize the data, and not only enhances model performance but also improves generalization by ensuring that input data is well-

aligned and comparable across the dataset.

Resizing

One of the first preprocessing steps involves resizing images to a uniform dimension. Machine learning models, especially those using decision trees with handcrafted features or deep learning models with convolutional layers, typically require fixed-size inputs. Resizing ensures consistency, allowing models to process images without distortions caused by varying dimensions. In some cases, aspect ratios are preserved using padding to avoid introducing unintended geometric distortions.

Centering

Centering is another key preprocessing step that involves aligning images to a common reference point. This technique is particularly useful in datasets where objects of interest are not consistently positioned across images. Methods such as aligning objects to the center of the image or detecting key points (e.g., centering around the main subject) help reduce variability, making it easier for machine learning models to detect patterns.

Normalization

Normalization is a critical preprocessing step used to scale input features to a consistent numerical range. It improves model stability, accelerates convergence, and prevents certain features from disproportionately dominating the learning process. In this thesis, normalization was applied to both tabular (scalar) and image (tensor) data, with different strategies depending on the data type.

Importantly, all normalization parameters (e.g., mean, standard deviation, minimum, maximum) were computed using only the training data. These parameters were then applied uniformly to normalize the training, validation, and test sets. This avoids information leakage from the test set during preprocessing.

Tabular Data (Scalar Normalization): For tabular datasets, each feature was treated as an independent scalar and normalized across all training samples. The following methods were used:

- **Min-max scaling**:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}. \tag{3.4}$$

  Where x is a feature value, and $x_{\min}, x_{\max}$ are the minimum and maximum of that feature across the training set.

- **Mean subtraction (centering)**:

$$x' = x - \mu. \tag{3.5}$$

  Where μ is the mean of the feature across the training set.

- **Z-score normalization (standardization)**:

$$x' = \frac{x - \mu}{\sigma}. \tag{3.6}$$

  Where μ and σ are the mean and standard deviation of the feature in the training data.

Image Data (Tensor Normalization): For image data, where each input is a 3D tensor of shape (height × width × channels), normalization was applied per channel, treating each color channel (e.g., Red, Green, Blue) independently across all training images. Let $x_{i,j,c}$ represent the pixel value at spatial position (i, j) and channel c. Then:

- **Min-max scaling** (per channel):

$$x'_{i,j,c} = \frac{x_{i,j,c} - x_{\min,c}}{x_{\max,c} - x_{\min,c}}. \tag{3.7}$$

  Where $x_{\min,c}$ and $x_{\max,c}$ are the minimum and maximum values of channel c, computed from the training set.

- **Mean subtraction (centering)**:

$$x'_{i,j,c} = x_{i,j,c} - \mu_c. \tag{3.8}$$

  Where μ_c is the mean of channel c, computed across all training images.

- **Z-score normalization (standardization)**:

$$x'_{i,j,c} = \frac{x_{i,j,c} - \mu_c}{\sigma_c}. \tag{3.9}$$

  Where μ_c and σ_c are the mean and standard deviation of channel c, again computed from training data only.

This channel-wise normalization ensures that each color channel has similar numerical properties, which is particularly important for models that are sensitive to input scale, such as decision trees and neural networks.

Other Preprocessing Techniques

In addition to the fundamental techniques above, other preprocessing methods can further refine image datasets:

- **Grayscale conversion**: For models that do not require color information, converting images to grayscale reduces complexity while retaining key structural details.

- **Data augmentation**: While not a strict preprocessing step, applying transformations such as rotation, flipping, scaling, or contrast adjustments helps artificially increase dataset diversity. This technique improves model robustness by exposing the model to a wider variety of input conditions, reducing overfitting, and enabling it to generalize better to unseen data—especially when the amount of labeled training data is limited.

- **Denoising and filtering**: Techniques such as Gaussian blurring or median filtering can remove noise, making it easier for models to focus on meaningful patterns rather than artifacts.

By applying these preprocessing techniques, image datasets become more structured, reducing unnecessary variations and enhancing feature consistency, which is particularly beneficial when working with tree-based models or hierarchical shrinkage techniques.

Dimensionality Reduction

In many machine learning applications, particularly in image analysis, datasets often contain a large number of features, many of which may be redundant or irrelevant for predictive modeling. Dimensionality reduction techniques aim to transform high-dimensional data into a lower-dimensional representation while preserving as much of the relevant information as possible. Reducing dimensionality not only enhances computational efficiency but also mitigates the risk of overfitting, a common challenge in decision tree learning.

One widely used technique for dimensionality reduction is **Principal Component Analysis (PCA)** [11]. PCA reduces the number of features in a dataset while preserving as much of the original information (or variance) as possible. This is particularly useful when working with high-dimensional data, such as images or datasets with many correlated features.

The main idea behind PCA is to find a new set of axes, called *principal components*, that capture the directions along which the data varies the most. These new axes are constructed so that:

- Each component is a linear combination of the original features.
- The components are orthogonal (i.e., uncorrelated).
- The first principal component captures the most variance in the data, the second captures the next most, and so on.

Mathematically, PCA computes these directions by finding the eigenvectors of the data's covariance matrix. The top k eigenvectors (those with the largest eigenvalues) form a new basis for the data. We then project the data onto this smaller set of axes to get a reduced-dimensional representation.

If we let $\mathbf{X}$ represent the original (centered) data matrix and $\mathbf{W}$ the matrix of top k principal components, the reduced data is, $\mathbf{X}_{\text{PCA}} = \mathbf{XW}$. This transformation results in a compressed version of the data that keeps most of the important structure while discarding noise or redundancy.

In the context of decision trees, this can be particularly beneficial for image-based models, where raw pixel values create extremely high-dimensional inputs. Without dimensionality reduction, decision trees may struggle to generalize due to the curse of dimensionality, which can lead to overly complex trees that memorize training data rather than learning meaningful patterns.

However, a key trade-off with PCA is interpretability. The principal components are linear combinations of the original features and do not correspond to meaningful physical quantities or spatial features (e.g., specific pixels in an image). This can reduce the transparency of the model, especially in domains where feature interpretability is important. Nonetheless, in practice, PCA often provides substantial performance and efficiency benefits.

Beyond PCA, other techniques such as autoencoders [9] and feature selection methods [7] have also been explored to improve decision tree performance in high-dimensional settings. These approaches help streamline the learning process by removing irrelevant features and emphasizing the structural elements most relevant to classification or regression tasks. While

dimensionality reduction tools like t-SNE [19] are useful for visualizing high-dimensional data, they are not suitable for use in supervised learning pipelines with independent test queries, as t-SNE does not learn a parametric mapping that can be applied to new data.

3.6 Hyperparameter Tuning and Model Selection

Selecting the optimal hyperparameters for a machine learning model is critical to achieving high performance and generalization. Decision trees, like many other models, have several hyperparameters—such as tree depth, minimum samples per split, and regularization parameters—that can significantly impact model behavior. To systematically tune these hyperparameters, various validation strategies are employed to assess model performance while mitigating the risk of overfitting.

3.6.1 K-Fold Cross-Validation

One widely used approach is **k-fold cross-validation**, which provides a robust mechanism for model selection by evaluating performance across multiple training-validation splits. In k-fold cross-validation, the dataset is partitioned into k equally sized subsets (folds). The model is trained on $k-1$ folds and validated on the remaining fold. Meaning that this process is repeated across all k folds, with each fold serving as the validation set once. The final performance estimate is then averaged across the k folds to provide a stable estimate of model generalization.

For decision trees, k-fold cross-validation is particularly useful in settings where datasets are not excessively large, as it ensures that all data points contribute to both training and validation. This approach reduces variance in performance estimation and helps guide the selection of hyperparameters that generalize well to truly unseen test data.

3.6.2 2-Way and 3-Way Splits

Another common strategy for model evaluation involves simple train-validation-test splits, which are computationally less expensive than k-fold cross-validation, as the model is trained only once rather than k times. In a 2-way split, the dataset is divided into a training set (used to train the model) and a test set (used to evaluate final performance). While straightforward, this approach does not provide a dedicated validation set for hyperparameter tuning, increasing the risk of overfitting to the test set.

A 3-way split addresses this limitation by introducing an additional validation set, leading to three partitions:

- **Training set**: Used to train the model.
- **Validation set**: Used to tune hyperparameters and select the best model.
- **Test set**: Used for final evaluation of the chosen model.

By ensuring that hyperparameter tuning is performed on the validation set rather than the test set, the 3-way split prevents information leakage and provides a more realistic estimate of model performance on truly unseen data. This strategy is particularly useful for high-dimensional datasets or situations where computational efficiency is a concern, as it avoids the repeated training cycles required in k-fold cross-validation. However, since the final performance is measured on a single test set, this approach does not support statistical analysis (e.g., confidence intervals or variance estimates), whereas k-fold cross-validation provides multiple performance measurements (one per fold), enabling you to compute averages, standard deviations, and even confidence intervals

3.7 Statistical Testing of Model Differences

In order to evaluate whether hierarchical shrinkage (HS) improves model performance over standard decision trees (CART), I follow the statistical testing framework. The goal is

to assess whether the observed difference in accuracy between CART and HS-CART is statistically significant. Since both models are evaluated on the *same* test set S, we can directly apply the method outlined below for comparing two models. For more information, the interested reader may refer to [12, Sec. 5.5].

Let h_1 and h_2 denote the CART and HS-CART models, respectively. Define:

- $\hat{e}_S(h_1)$: empirical error (e.g., 1− accuracy) of CART on the test set.
- $\hat{e}_S(h_2)$: empirical error of HS-CART on the same test set.

We then compute the observed difference: $\hat{d} = \hat{e}_S(h_1) - \hat{e}_S(h_2)$ which provides an unbiased estimate of the true difference d between the errors of the two models.

Assuming that both $\hat{e}_S(h_1)$ and $\hat{e}_S(h_2)$ approximately follow normal distributions, the distribution of $\hat{d}$ is also approximately normal, with variance:

$$\mathrm{Var}(\hat{d}) = \frac{\hat{e}_S(h_1)(1 - \hat{e}_S(h_1))}{n} + \frac{\hat{e}_S(h_2)(1 - \hat{e}_S(h_2))}{n}.$$

where n is the number of test samples in S. The 95% confidence interval for the true difference d is then: $\hat{d} \pm z_N \sqrt{\mathrm{Var}(\hat{d})}$. Where $z_N \approx 1.96$ is selected depending on the desired confidence level, in our case a 95% confidence interval.

Interpretation: If the resulting confidence interval does not contain zero, we can conclude with 95% confidence that the performance difference between CART and HS-CART is statistically significant. Otherwise, the observed difference may simply be due to random fluctuations in the test set.

Interpreting Statistical Significance via p-values. In addition to reporting confidence intervals, an alternative way to interpret statistical significance is through p-values. Specifically, given an observed difference d between two models, we can ask: what is the smallest confidence level (i.e., the largest p-value) for which the interval around d would still exclude

zero? In other words, we seek the smallest z-value such that zero just barely lies within the interval:

$$d \pm z \cdot \sqrt{\frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2}}$$

where e_1 and e_2 are the sample error rates of the two models on the same test set of size $n = n_1 = n_2$. Solving for z in terms of d and the standard error gives the smallest critical value for which the interval would still include 0. The corresponding p-value can then be obtained from the standard normal distribution table. This approach allows for a more fine-grained assessment of significance beyond the fixed 95% level, especially in borderline cases.

3.8 Confident Learning / Denoising

In real-world datasets, especially those derived from large-scale human annotations or automated pipelines, label noise is a pervasive issue that can degrade the performance of machine learning models. Mislabeling can arise due to human error, ambiguity in data, or systematic biases in data collection. Traditional machine learning models, including decision trees, often assume that training labels are perfectly accurate, leading to poor generalization when trained on noisy labels. Addressing this issue is crucial for ensuring both robustness and reliability in learning algorithms.

Confident learning [13] is a principled framework for identifying and correcting noisy labels in datasets. It operates on the assumption that most labeled data points are correctly classified, and it systematically estimates the joint distribution of noisy and true labels to detect mislabeled instances. Cleanlab [14], a widely used Python library for implementing confident learning, automates the process of label error detection, dataset cleansing, and robust learning. By leveraging statistical techniques and confidence-based estimates, Cleanlab helps identify training samples that are likely mislabeled, allowing for data-driven denoising before model training.

For decision trees and tree-based ensembles, integrating confident learning as a preprocessing step can significantly enhance model quality. By reducing the impact of noisy or incorrect labels, this approach leads to better-calibrated predictions and improved interpretability, as the learned decision paths reflect more meaningful patterns rather than overfitting to erroneous annotations. In our experiments, I apply confident learning to preprocess the dataset, ensuring that the learned decision trees operate on a cleaner, more reliable training set, ultimately improving performance and generalization.

Chapter 4

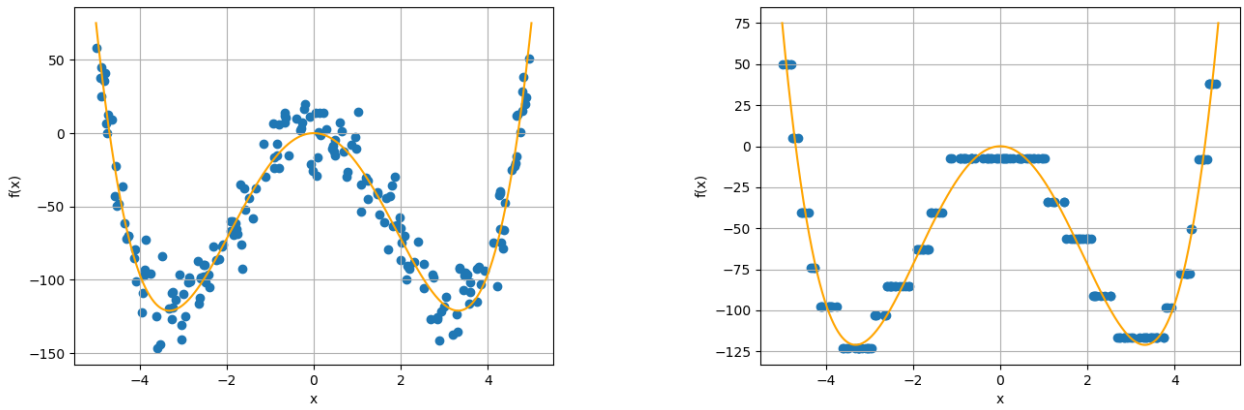
Experiments

4.1 Experiments

The results from the hierarchical shrinkage paper [1] were validated prior to conducting the experiments presented here. All of which, used tabular datasets. From there I conducted various experiments to access the performance of hierarchical shrinkage on other datasets and with different conditions. The datasets selected were popular machine learning datasets in addition to an artificial dataset. I used the GitHub repository imodels [17] to import the models that would be tested.

4.2 Artificial Data

To perform a preliminary sanity check and better understand the behavior of hierarchical shrinkage, I created artificial datasets from two simple, nonlinear functions, $y = x^4 - 22x^2$ and $y = -x^3$. Gaussian noise with a standard deviation of 20 was added to the output variable y to simulate real-world variability. Two models were trained on each dataset: a standard Random Forest regressor and a Random Forest regressor with Hierarchical Shrinkage applied as a post-hoc adjustment. Figures 4.1a and 4.1b display the fitted curves of a Random Forest and Hierarchical Shrinkage-enhanced model on the first synthetic function, $y = x^4 - 22x^2$,



(a) Random Forest

(b) With Hierarchical Shrinkage

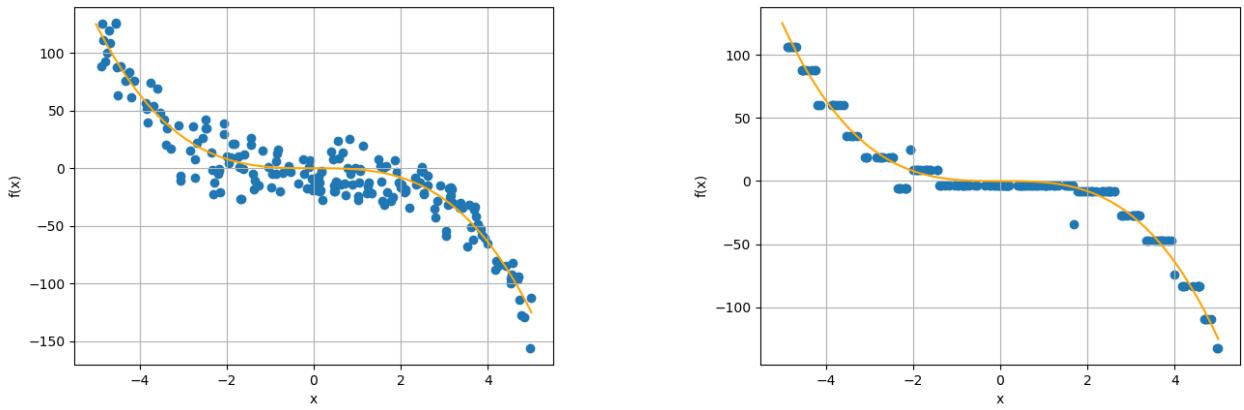
Figure 4.1: Predictions on the artificial function $y = x^4 - 22x^2$ with Gaussian noise. (a) The standard Random Forest exhibits high variance in sparse regions. (b) Hierarchical Shrinkage reduces variance and yields smoother, more reliable predictions.

while Figures 4.2a and 4.2b show the results for the second function, $y = -x^3$. The difference in smoothness between the models is clearly visible: the HS model produces more stable curves that avoid the oscillatory artifacts seen in the standard Random Forest predictions, especially in low-data regions.

These results illustrate that the predictions of the standard Random Forest model are noticeably more erratic, especially in regions with sparse data. In contrast, the Hierarchical Shrinkage-enhanced model produces smoother, more stable predictions that better approximate the underlying function. This supports the hypothesis that hierarchical shrinkage can reduce overfitting and improve generalization by tempering unreliable node-level predictions in tree-based models.

4.3 Datasets

Table 4.1 shows all the datasets used in this study. These datasets were deliberately selected to extend the scope of analysis beyond the datasets that were used by Agarwal et al. [1], which focused exclusively on low-dimensional tabular data with binary classification tasks.



(a) Random Forest

(b) With Hierarchical Shrinkage

Figure 4.2: Predictions on the artificial function $y = -x^3$ with Gaussian noise. Similar to the first experiment, (a) shows unstable predictions by the Random Forest, while (b) demonstrates how Hierarchical Shrinkage smooths the output while retaining the overall trend.

In contrast, this thesis incorporates datasets with characteristics not previously studied under HS, including image data, high-dimensional tabular data, multi-class classification problems, and datasets with injected label noise. This expanded dataset selection allows for a more comprehensive evaluation of the robustness and generalizability of hierarchical shrinkage. Table 4.2 shows the dimensionality of the datasets before and after applying and additionally the imbalance ratio.

Figures 4.4a, 4.4b, 4.11a, 4.5b, 4.6a, and 4.6b illustrate the accuracy of CART and hierarchical shrinkage models as a function of tree complexity, represented by the number of leaf nodes, across four datasets. For CIFAR-10 (Figure 4.4a), accuracy steadily increases for both models as more leaf nodes are added, with CART maintaining a slight edge over hierarchical shrinkage. In contrast, for tabular datasets like Student Performance (Figure 4.5b), hierarchical shrinkage outperforms CART by a noticeable margin at nearly every complexity level, highlighting its strength in stabilizing predictions and mitigating overfitting. A similar trend can be seen in Student Dropout (Figure 4.5a), where hierarchical shrinkage catches up to and slightly surpasses CART as the tree becomes more expressive. Likewise,

	#Instances		Dimensionality		
Dataset	Total	Missing Values	Original	No redundancy	#Classes
CIFAR-10	60000	0	3072	3072	10
Fashion-MNIST	70000	0	784	784	10
Oxford Pets	7390	0	750000*		37
Adult Income	48842	2399	14	14	2
Titanic	891	708 / 179	11	7	2
Credit Card	30000	0	24	22	2
Student Dropout	4424	0	36	36	3
Students Performance	2392	0	14	13	5
Multivariate Gait Data	181800	0	6	6	3
GitHub MUSAE	37700	0	4006	4005	2
Internet Advertisements	3279	918	1558	1558	2

Table 4.1: Information on the datasets used. Note that the Oxford Pets dataset contains images of varying dimensions. To standardize input size, all images were resized to 500×500 using TensorFlow's `tf.image.resize(·)`. A 2-way split of the data was performed. In the absence of a default split (e.g., CIFAR-10 has a standard 50,000:10,000 split), an 80:20 train-test split was used.

for the titanic and credit card dataset (Figures 4.6a and 4.6b), we see similar results. Lastly, for Fashion-MNIST (Figure 4.4b), performance remains close between the models, though hierarchical demonstrates greater stability at moderate complexity.

For the Credit Card dataset 4.10b, both CART and hierarchical shrinkage show rising AUC values as the number of leaf nodes grows, but hierarchical shrinkage consistently outperforms CART at nearly every level of complexity, especially in deeper trees, suggesting that shrinkage mitigates overfitting. In the Student Dropout dataset 4.11a, the AUC for both models improves with tree size, yet hierarchical shrinkage reaches a performance plateau at a higher level, indicating more efficient learning from fewer nodes. The Student Performance dataset 4.11b exhibits the most dramatic benefit from shrinkage: hierarchical shrinkage achieves very high AUC with as few as 10 leaf nodes, while CART starts to decline in performance as the tree becomes more complex. In the Titanic dataset 4.10a, both models improve with more leaves until around 20, after which CART gradually declines while hierarchical shrinkage remains stable or improves slightly. Lastly, in the CIFAR-10 and

Dataset	#Instances	Dimensionality		#Classes	Imbalance
		no PCA	PCA		
CIFAR-10	60000	3072	658	10	6000 : 6000
Fashion-MNIST	70000	784	459	10	7000 : 7000
Ofxord Pets	7390	750000	523	37	200 : 184
Adult Income	48842	14	14	2	37120 : 11722
Titanic	891	7	2	2	549 : 342
Credit Card	30000	22	9	2	23364 : 6636
Student Dropout	4424	36	11	3	2209 : 794
Students Performance	2392	13	7	5	1211 : 107
Gait Data	181800	6	3	3	60600 : 60600
GitHub MUSAE	37700	4006	582	2	27961 : 9739
Internet Ads	3279	1558	7	2	2821 : 458

Table 4.2: Information on the datasets used. The dimensionality in the case of PCA is determined by the top projects that together can explain 99% of the variance.

Fashion-MNIST (Figures 4.12b and 4.12a) datasets the experiments indicate hierarchical shrinkage performing very similar.

We note that in the case of multi-class classification a single graph in the ROC curve does not have an immediate meaning, since when there are more than two classes, then there is no meaning for true positive rate (TPR) and false positive rate (FPR) which would allow us to create the ROC curve graph in a binary classification setting. Nevertheless, one can calculate such graphs when identifying one class as positive and all the other classes as negative. This would result in multiple graphs on the same ROC curve, one graph corresponding for each class in this one-versus-rest calculation. However, one can go one step beyond that and average the values of all these graphs and obtain just a single graph in the ROC curve graph even in the multi-class setting. This is also allowed in the scikit-learn library. In particular, the averaged macro-averaged ROC curve can be obtained that would give us such a graph and that is what we plot in the figures that correspond to ROC curves for datasets with multiple classes. For more information, please see

https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html.

Furthermore, along these lines, scikit-learn also allows one to calculate the AUC for such ROC curves that are the average over all the possible classes and this can be done using the `roc_auc_score` function; for more information, please see

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html.

4.4 PCA

To reduce the computational burden of working with high-dimensional image data, Principal Component Analysis (PCA) was applied to the CIFAR-10 dataset, which originally has 3072 features ($32 \times 32 \times 3$). PCA projects the data onto a set of orthogonal axes (principal components) that capture the directions of maximum variance. As shown in Figure 4.3, the first 100 components retain approximately 99% of the total variance, offering a compact representation of the dataset while preserving most of its informative structure. This dimensionality reduction is intended to improve model training efficiency and to test the interaction between PCA and hierarchical shrinkage.

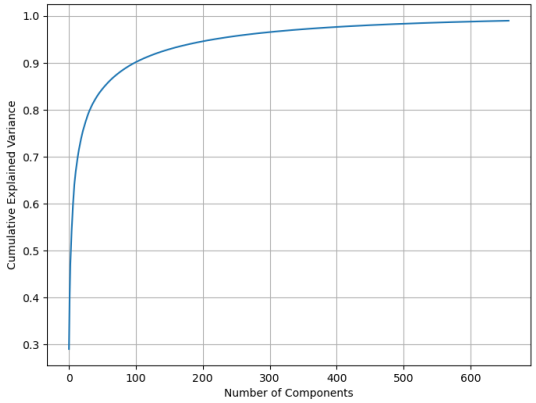


Figure 4.3: Cumulative explained variance on the CIFAR-10 dataset after PCA. The x-axis denotes the number of retained components, while the y-axis shows the total variance captured.

Table 4.3 reports classification accuracy results on the CIFAR-10 dataset for both standard decision trees and trees with hierarchical shrinkage, using the original 3072-dimensional input and a reduced 658-dimensional input after PCA. Without PCA, hierarchical shrinkage slightly under performs the standard tree model, but after PCA is applied, HS achieves a slightly better accuracy.

	Decision Tree	**Hierarchical Shrinkage**
3072 Dimensions	26.75%	26.06%
658 Dimensions	24.76%	25.95%

Table 4.3: Classification accuracy (%) on the CIFAR-10 dataset using Decision Trees and Hierarchical Shrinkage models. The first row shows results using the full 3072-dimensional feature space (raw pixels), while the second row reports results after dimensionality reduction to 658 features using PCA. While performance is similar at full dimensionality, Hierarchical Shrinkage slightly outperforms the standard Decision Tree after PCA.

4.4.1 Impact of PCA and Hierarchical Shrinkage Across Models

To further assess the behavior and generalizability of hierarchical shrinkage (HS), I conducted an extended set of experiments evaluating four variations of decision tree and random forest models across all datasets:

- Vanilla DT / RF: Standard decision tree or random forest with no preprocessing.
- PCA-DT / PCA-RF: PCA applied prior to model training to reduce dimensionality.
- HS-DT / HS-RF: Hierarchical shrinkage applied directly to the base model.
- PCA-HS-DT / PCA-HS-RF: PCA applied first, followed by hierarchical shrinkage during training.

These experiments were motivated by: (1) Does hierarchical shrinkage improve generalization performance over standard tree-based models across diverse datasets? (2) How does dimensionality reduction via PCA affect model accuracy, especially on high-dimensional datasets

such as CIFAR-10 or GitHub MUSAE? The results are shown in Tables 4.4 and 4.5. These tables allow us to compare the relative effect of PCA and HS independently and jointly across datasets of varying types—image, tabular, high-dimensional, multiclass, and noisy. From Table 4.4, we observe that Hierarchical Shrinkage improves the accuracy of decision trees across most tabular datasets, especially Adult Income, Titanic, and Credit Card. The benefit is less pronounced in high-dimensional image datasets, where PCA alone contributes more to accuracy improvements.

Dataset	DT	PCA-DT	HS-DT	PCA-HS-DT
CIFAR-10	27.85(0.9762)	24.45(0.0013)	27.01(0.9123)	25.95(0.0021)
Fashion-MNIST	79.04(0.9239)	74.95(0.0017)	78.06(0.8812)	73.86(0.0025)
Ofxord Pets	26.50(0.02110)	23.75(0.0057)	27.85(0.0187)	24.88(0.0061)
Adult Income	80.93(1.002)	77.54(0.0013)	85.33(0.9568)	82.12(0.0020)
Titanic	77.87(1.0950)	66.43(0.0065)	82.12(1.0124)	73.08(0.0072)
Credit Card	72.33(0.9885)	69.97(0.0028)	82.07(0.9341)	78.18(0.0033)
Student Dropout	74.35(0.9025)	70.93(0.0019)	74.58(0.8810)	72.13(0.0023)
Students Performance	89.56(1.0412)	85.40(0.0022)	93.11(1.0874)	90.05(0.0031)
Multivariate Gait Data	40.28(0.9328)	33.02(0.0026)	43.48(0.9107)	37.39(0.0030)
GitHub MUSAE	71.02(1.9235)	69.33(1.9619)	77.38(0.3833)	75.29(0.01223)
Internet Advertisements	89.23(0.8569)	86.33(0.3319)	91.89(0.7361)	87.89(0.9386)

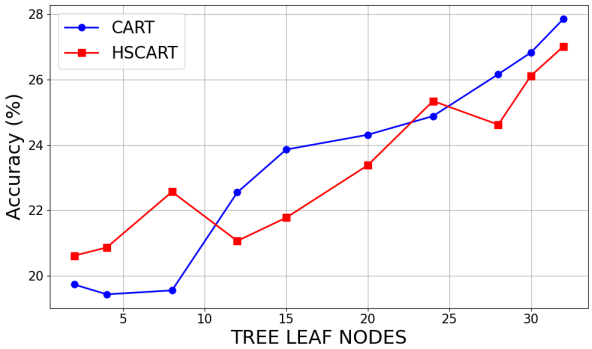
Table 4.4: Average classification accuracy (in %) for all datasets. Standard decision trees (DT), trees after PCA (PCA-DT), trees with Hierarchical Shrinkage (HS-DT), and PCA combined with HS (PCA-HS-DT). Results are averaged over 10 runs; standard deviation is shown in parentheses.

We observe that hierarchical shrinkage shows improvement in model accuracy, particularly as tree size increases. This is especially apparent in binary tabular datasets (Figures 4.7b, 4.6b, and 4.6a). In contrast, for certain image-based datasets (Figures 4.7a), the benefit of HS is less clear or even absent. In multi-class datasets (Figures 4.8a, 4.5b, 4.5a and Figures 4.14a, 4.11b, 4.11a), HSCART provides mixed results. While some plots show gains, others indicate negligible or fluctuating improvements. For the datasets like adult income, student dropout, and student performance (Figure 4.13b, 4.11a, 4.11b) HSCART consistently outperforms or matches CART in terms of AUC, particularly as the number of tree

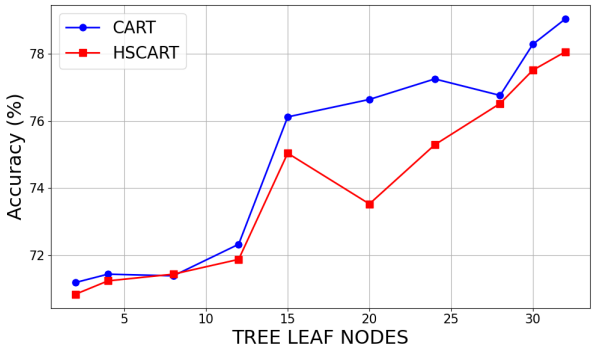
Dataset	RF	PCA-RF	HS-RF	PCA-HS-RF
CIFAR-10	46.63(1.1234)	39.04(0.0020)	44.23(1.0789)	40.11(0.0025)
Fashion-MNIST	87.61(0.9876)	84.74(0.0018)	87.12(0.9452)	82.92(0.0023)
Ofxord Pets	30.24(0.0456)	29.21(0.0098)	32.04(0.0421)	30.29(0.0105)
Adult Income	85.45(1.0451)	83.63(0.0015)	91.34(1.1023)	88.62(0.0027)
Titanic	81.26(.7029)	69.83(0.0071)	85.34(1.1482)	70.24(0.0078)
Credit Card	81.37(1.2894)	78.01(0.0030)	89.92(1.1356)	85.04(0.0034)
Student Dropout	80.01(0.9763)	76.11(0.0021)	81.89(0.9880)	78.47(0.0029)
Students Performance	91.48(1.1345)	87.40(0.0024)	95.98(1.1890)	90.66(0.0031)
Multivariate Gait	42.56(0.9234)	34.64(0.0029)	44.76(0.9428)	38.32(0.0033)
GitHub MUSAE	72.72(0.8246)	70.19(0.9416)	74.24(0.8567)	76.25(1.4962)
Internet Advertisements	90.23(0.9485)	83.47(0.9230)	93.92(0.9284)	91.24(1.3945)

Table 4.5: Average classification accuracy (in %) for all datasets. Standard random forests (RF), after PCA (PCA-RF), random forests with Hierarchical Shrinkage (HS-RF), and PCA combined with HS (PCA-HS-RF). Results are averaged over 10 runs; standard deviation is shown in parentheses.

leaf nodes increases. For high-dimensional datasets (Figures 4.15a, 4.14b), the performance gains from HS are less pronounced. Image dataset (Figure 4.13a), the performance of HS is inconsistent. In some cases, it slightly improves AUC, but in others, CART performs comparably or even slightly better (Figures 4.4a and 4.4b).



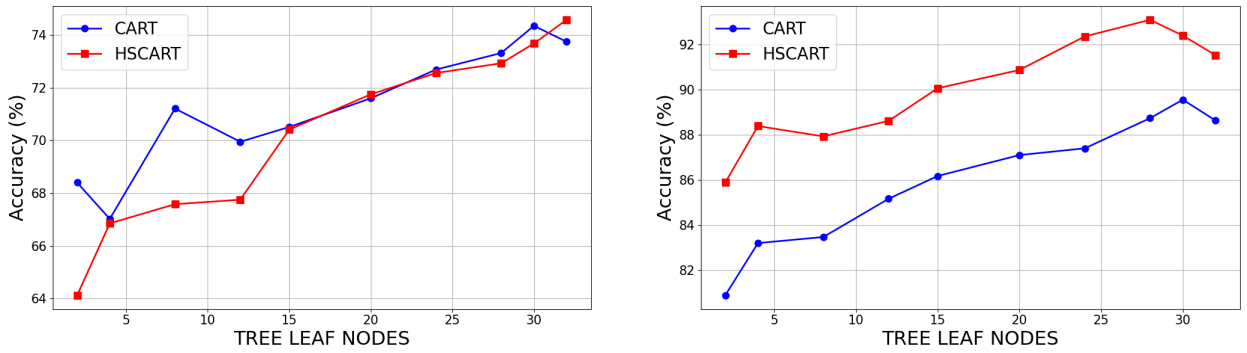
(a) CIFAR-10



(b) FASHION-MINST

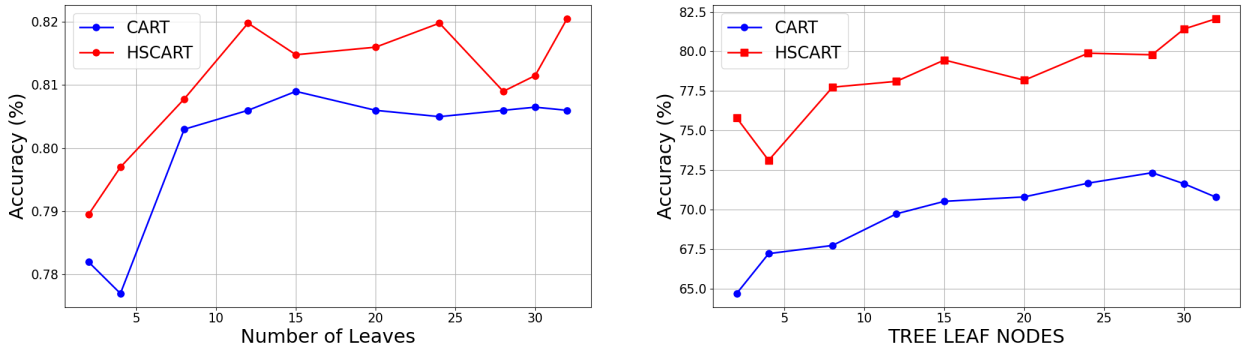
Figure 4.4: Accuracy plots. Where the y-axis is the accuracy value and the x-axis the tree leaf nodes. The left plot is the CIFAR-10 dataset and the right is the Fashion-MNIST dataset.

Additionally, I also recorded the time taken to complete each experiment to highlight



(a) Student Dropout (b) Student Performance

Figure 4.5: Accuracy plots. Where the y-axis is the accuracy value and the x-axis the tree leaf nodes. The left plot is the student dropout dataset and the right the student performance dataset.



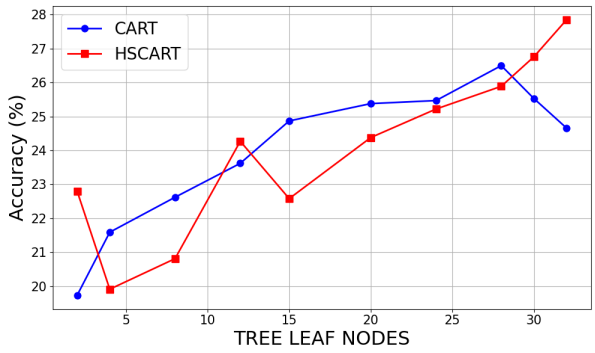
(a) Titanic (b) Credit Card

Figure 4.6: Accuracy plots. Where the y-axis is the accuracy value and the x-axis the tree leaf nodes. The left plot is the titanic and the right the credit card dataset.

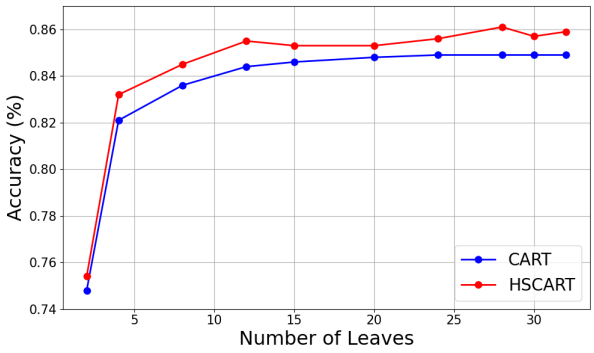
the computational cost of hierarchical shrinkage and dimensionality reduction. While HS provides accuracy improvements, it also incurs additional training time, particularly on image-based datasets. Tables 4.6 and 4.7 report the average runtime (in minutes) over multiple runs.

4.5 Statistical Comparison of CART vs HS-CART

To assess whether the observed differences between CART and HS-CART models are statistically significant, we followed the confidence interval methodology described in Mitchell [12,

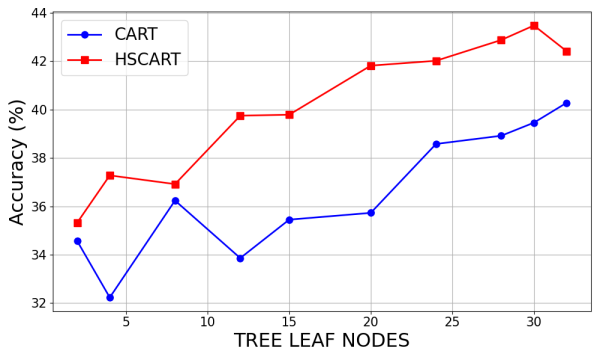


(a) Oxford Pets

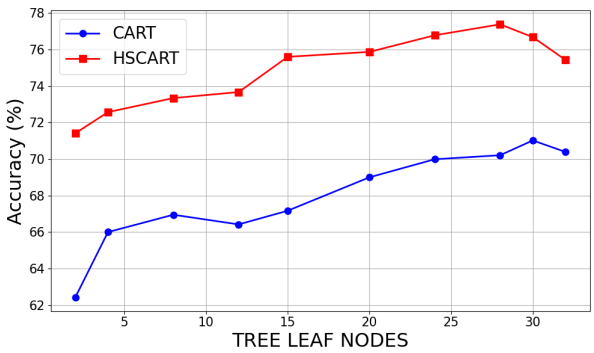


(b) Adult Income

Figure 4.7: Accuracy plots. Where the y-axis is the accuracy value and the x-axis the tree leaf nodes. The left plot is the Oxford Pets and the right the adult income dataset.



(a) Gait Data

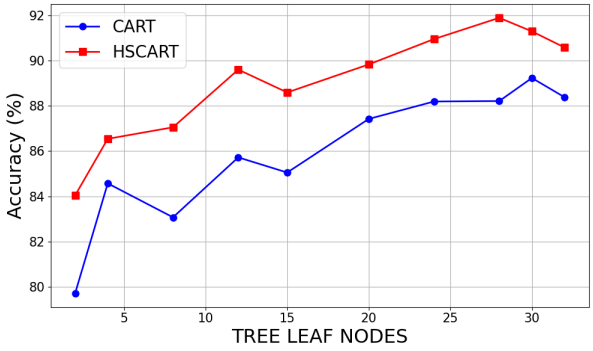


(b) MUSAE

Figure 4.8: Accuracy plots. Where the y-axis is the accuracy value and the x-axis the tree leaf nodes. The left plot is the Gait dataset and the right the MUSAE dataset.

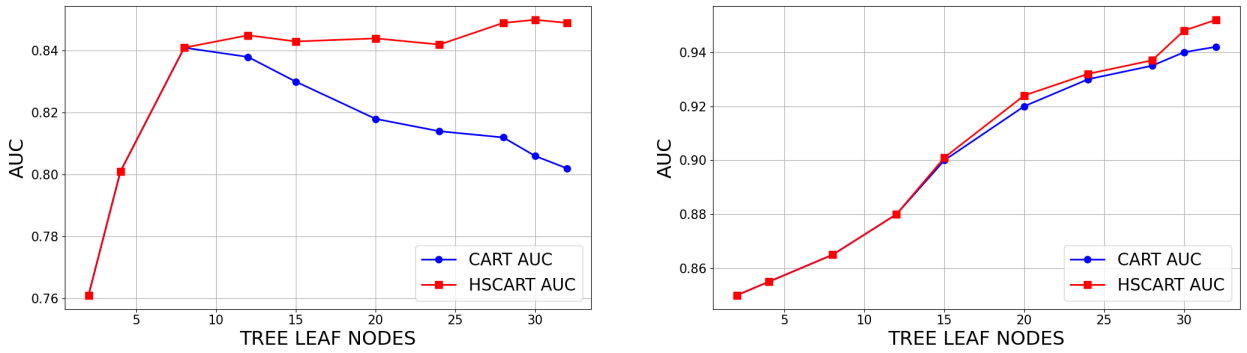
Sec. 5.5]. Table 4.8 summarizes the results for each dataset.

For several datasets—including *Adult Income*, *Credit Card*, *Students Performance*, *GAIT*, *MUSAE*, and *Internet Ads*—the 95% confidence intervals exclude zero, indicating that HS-CART significantly outperforms CART at the 95% confidence level. On other datasets such as *CIFAR-10*, *Fashion-MNIST*, *Oxford Pets*, *Titanic*, and *Student Dropout*, the confidence intervals include zero, suggesting that the performance difference is not statistically significant. These findings reinforce that while hierarchical shrinkage offers notable improvements in structured tabular domains, its gains seem to not be statistically significant in other domains like image datasets.



(a) Internet Ads

Figure 4.9: Accuracy plot for the Internet Ads dataset. Where the y-axis is the accuracy value and the x-axis the tree leaf nodes.



(a) Titanic

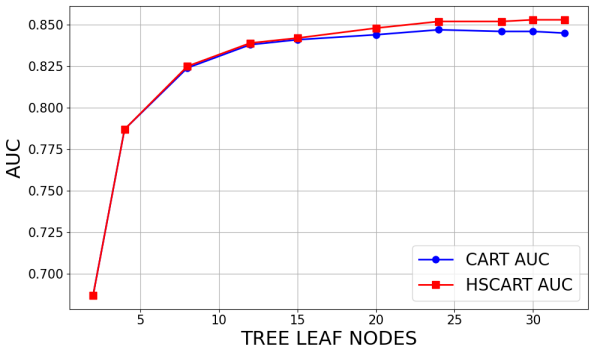
(b) Credit Card

Figure 4.10: AUC plots. Where the y-axis is the AUC value and the x-axis the tree leaf nodes. The left the titanic dataset and the right the credit card dataset.

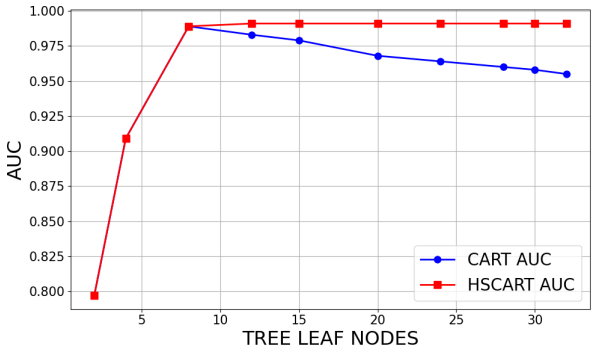
4.6 Tree Structure

Figures 4.16 and 4.17 visualize fully trained HSCART (Hierarchical Shrinkage Classification and Regression Tree) models on the Student Dropout and Student Performance datasets, respectively. These figures demonstrate how hierarchical shrinkage can be applied post hoc to decision trees without altering their structure—only modifying the predictions at the leaf nodes.

The tree in Figure 4.16 shows a moderately deep model with clear branching. Hierarchical shrinkage helps smooth predictions by shrinking them toward the means of their parent nodes, which mitigates overfitting due to small sample sizes in deeper nodes.

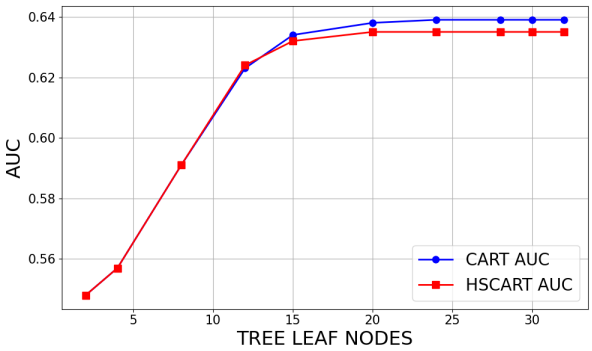


(a) Student Dropout

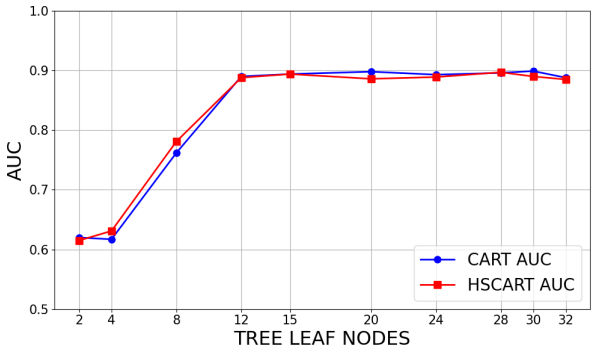


(b) Student Performance

Figure 4.11: AUC plots. Where the y-axis is the AUC value and the x-axis the tree leaf nodes. The left is the student dropout dataset and the right the student performance.



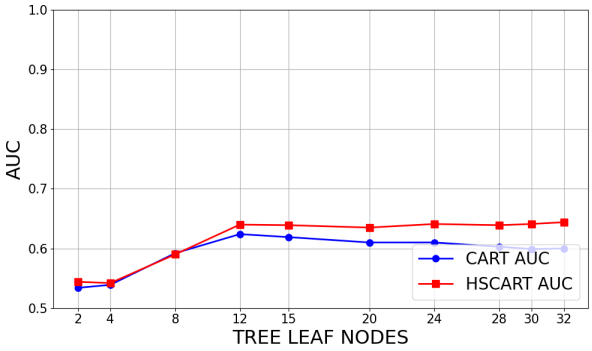
(a) CIFAR-10



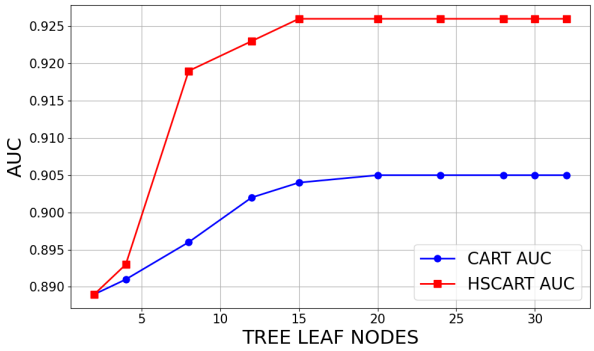
(b) FASHION-MINST

Figure 4.12: AUC plots. Where the y-axis is the AUC value and the x-axis the tree leaf nodes. The left figure is the CIFAR-10 dataset and the right the Fashion-MNIST dataset.

The tree in Figure 4.17 represents a multi-class classification problem with a broader and more balanced structure. This tree shows how HSCART accommodates multiple output classes while still benefiting from the same shrinkage regularization, helping the model generalize in the presence of potentially overlapping or noisy features.

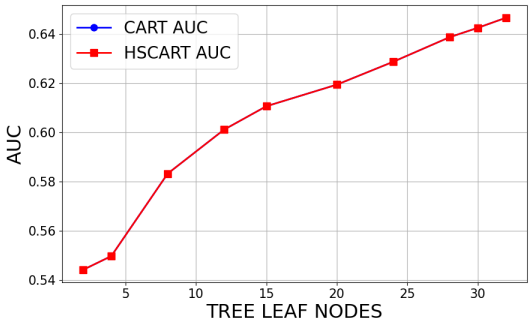


(a) Oxford Pets

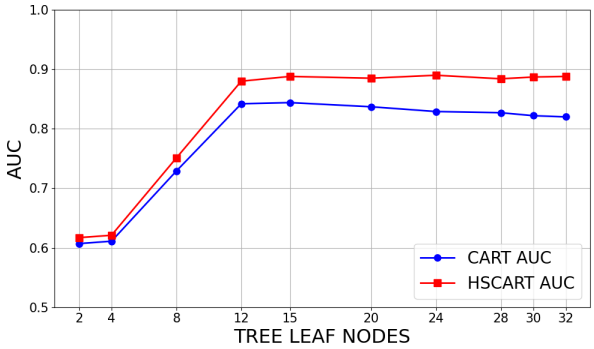


(b) Adult Income

Figure 4.13: AUC plots. Where the y-axis is the AUC value and the x-axis the tree leaf nodes. The left figure is the Oxford Pets dataset and the right the Adult Income dataset.

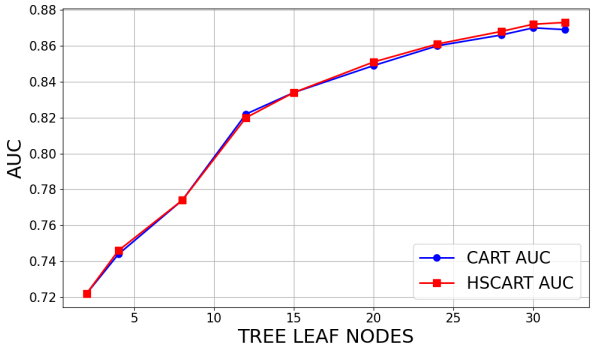


(a) Gait Data



(b) MUSAE

Figure 4.14: AUC plots. Where the y-axis is the AUC value and the x-axis the tree leaf nodes. The left figure is the Gait dataset and the right the MUSAE dataset.



(a) Internet Ads

Figure 4.15: AUC plot for the Internet Ads dataset. Where the y-axis is the AUC value and the x-axis the tree leaf nodes.

Dataset	DT	PCA-DT	HS-DT	PCA-HS-DT
CIFAR-10	4.23	3.10	13.05	11.37
Fashion-MNIST	2.79	3.04	8.35	7.80
Ofxord Pets	4.56	3.90	24.09	23.13
Adult Income	< 1s	< 1s	< 1s	< 1s
Titanic	< 1s	< 1s	< 1s	< 1s
Credit Card	< 1s	< 1s	< 1s	< 1s
Student Dropout	< 1s	< 1s	< 1s	< 1s
Student Performance	< 1s	< 1s	< 1s	< 1s
Gait Data	< 1s	< 1s	< 1s	< 1s
GitHub MUSAE	< 1s	< 1s	< 1s	< 1s
Internet Ads	< 1s	< 1s	< 1s	< 1s

Table 4.6: Time (in minutes) required to perform the experiments shown in Table 4.4. The reported time is the average over ten runs and inside parentheses we see the standard deviation over these ten runs.

Dataset	RF	PCA-RF	HS-RF	PCA-HS-RF
CIFAR-10	4.01	3.72	9.08	5.07
Fashion-MNIST	1.39	3.52	1.72	4.40
Ofxord Pets	2.12	2.10	20.10	18.74
Adult Income	0.02	0.09	0.16	0.24
Titanic	< 1s	< 1s	< 1s	< 1s
Credit Card	0.07	0.09	0.24	0.27
Student Dropout	< 1s	< 1s	< 1s	< 1s
Student Performance	< 1s	< 1s	< 1s	< 1s
Gait Data	< 1s	< 1s	< 1s	< 1s
GitHub MUSAE	0.02	0.04	0.56	0.57
Internet Ads	< 1s	0.04	0.56	0.57

Table 4.7: Time (in minutes) required to perform the experiments shown in Table 4.5. The reported time is the average over ten runs.

Dataset	CART Accuracy (%)	HS-CART Accuracy (%)	Difference (%)	95% Confidence Interval
CIFAR-10	27.85	27.01	-0.84	[-2.08, 0.40]
Fashion-MNIST	79.04	78.06	-0.98	[-2.12, 0.16]
Oxford Pets	26.50	27.85	+1.35	[-1.13, 3.83]
Adult Income	80.93	85.33	+4.40	[+3.37, +5.43]
Titanic	77.87	82.12	+4.25	[-2.18, +10.68]
Credit Card	72.33	82.07	+9.74	[+8.58, +10.90]
Student Dropout	74.35	74.58	+0.23	[-2.92, +3.38]
Students Performance	89.56	93.11	+3.55	[+0.80, +6.30]
Multivariate Gait	40.28	43.48	+3.20	[+2.64, +3.76]
GitHub MUSAE	71.02	77.38	+6.36	[+5.28, +7.44]
Internet Ads	89.23	91.89	+2.66	[+0.21, +5.11]

Table 4.8: Comparison of CART vs. HS-CART performance across datasets. Positive differences indicate higher accuracy for HS-CART. Confidence intervals not containing zero imply statistically significant improvements at the 95% level.

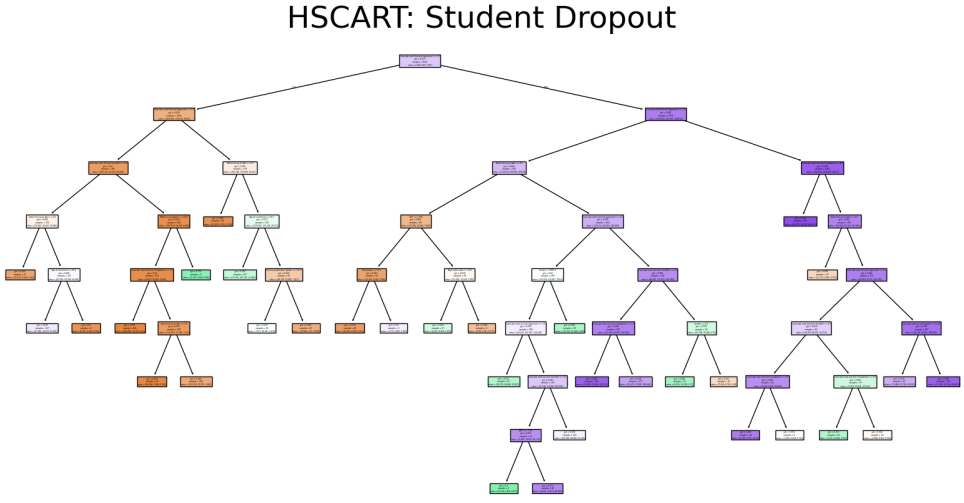


Figure 4.16: Learned Tree of HSCART Model on Student Dropout Data.

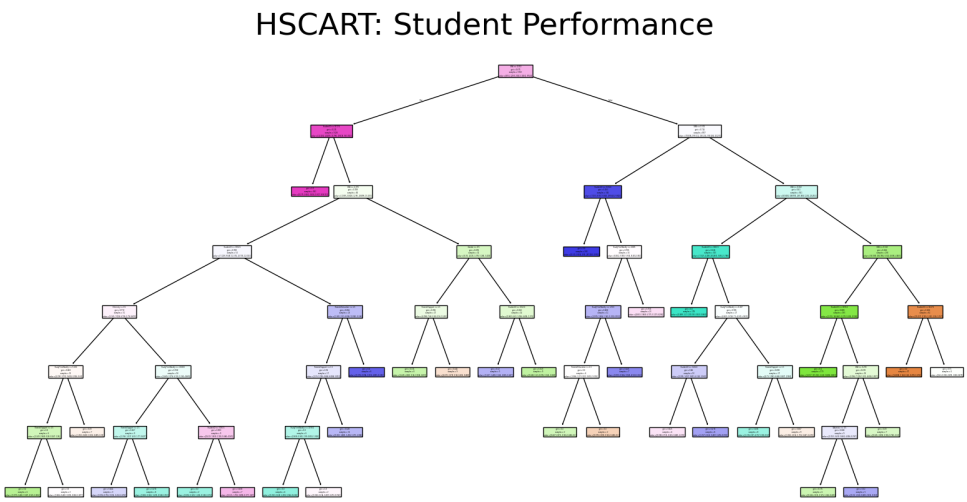


Figure 4.17: Learned Tree of HSCART Model on Student Performance Data.

4.7 Working with Noisy Data and Confident Learning

To evaluate how hierarchical shrinkage performs under conditions of label noise—a common issue in real-world datasets—I artificially introduced noise into the Credit Card and Student Performance datasets. The goal of this experiment was to assess whether hierarchical shrinkage maintains its benefits when the training data contains mislabeled instances.

Given a specified noise ratio (e.g., 30%), a custom function randomly selects a proportion of the labels and corrupts them depending on whether the task is binary or multi-class classification. In binary tasks, selected labels are flipped (i.e., 0 becomes 1 and vice versa). For multi-class classification, each selected label is replaced with a randomly chosen incorrect label from the remaining classes.

To mitigate the impact of this synthetic noise, I applied Confident Learning as a preprocessing step using the `cleanlab` Python package. This method estimates which data points are likely to be mislabeled and enables either correction or removal of those samples prior to training. After denoising, models with and without hierarchical shrinkage were retrained,

and their performance compared using both accuracy and AUC.

Figure 4.22 shows AUC results for the Credit Card dataset under various levels of label noise (1% to 49%). HSCART generally performs as well or slightly better than CART, especially after denoising. Accuracy results (Figure 4.24) show a similar trend, with hierarchical shrinkage producing slightly flatter curves—consistent with its role in stabilizing predictions.

For the Student Performance dataset, the most promising improvement is seen in Figure 4.28, where HSCART maintains stable AUC values as the number of leaf nodes increases. However, when analyzing the remaining results (Figures 4.18, 4.19, 4.26, 4.21, 4.20), we observe only marginal improvements or inconsistent behavior.

Box plots before and after denoising (Figures 4.32, 4.33, 4.30, 4.31) show that HSCART generally has tighter interquartile ranges and slightly better medians, especially at higher noise levels. However, these effects are subtle.

4.7.1 Inconsistent Results

With that being said, many of the results seem to be inconsistent with previous results gathered, possibly suggesting some issue in the experimental setup or implementation of these experiments. As an example, if we analyze the credit card dataset we can notice discrepancies. From Figure 4.24, we can see that the for a noisy level of 1%, the accuracy consistently hovers a little above 80% as the number of max leaf nodes increases. In contrast, if we look at Figure 4.6b, which shows the accuracy *without* any noise added, we see that the performance is somehow worse. Therefore, further investigation is required before drawing strong conclusions regarding the effectiveness of confident learning when combined with hierarchical shrinkage. Future work should revisit these experiments to verify implementation correctness and potentially explore alternative denoising configurations.

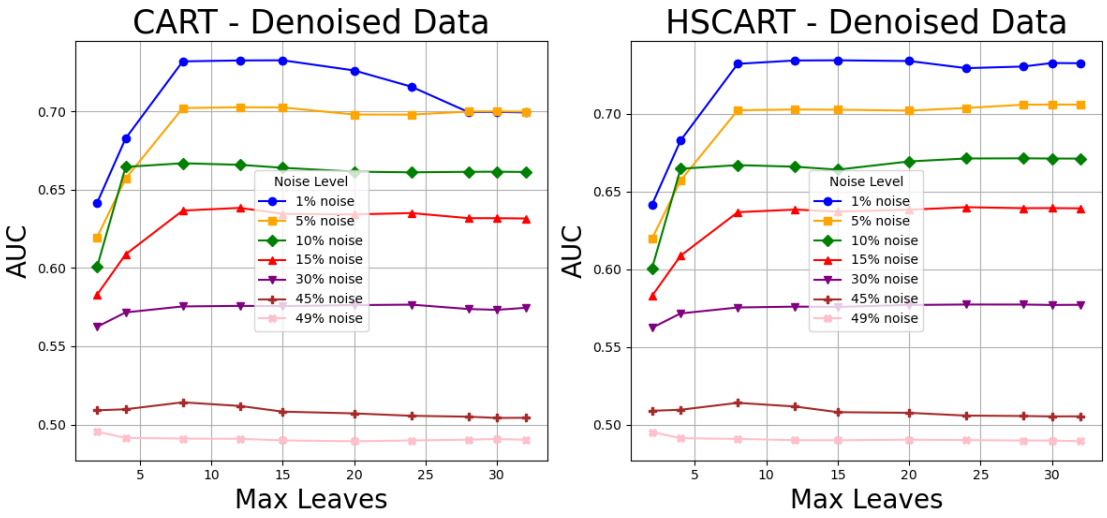


Figure 4.18: Effect of increasing label noise on AUC scores for the Credit Card dataset after applying confident learning (denoising the noisy data). The left figure (CART) is a standard decision tree, while the right figure (HSCART) is a decision tree model with hierarchical shrinkage.The y-axis shows the AUC values while the x-axis shows the max leaf nodes.

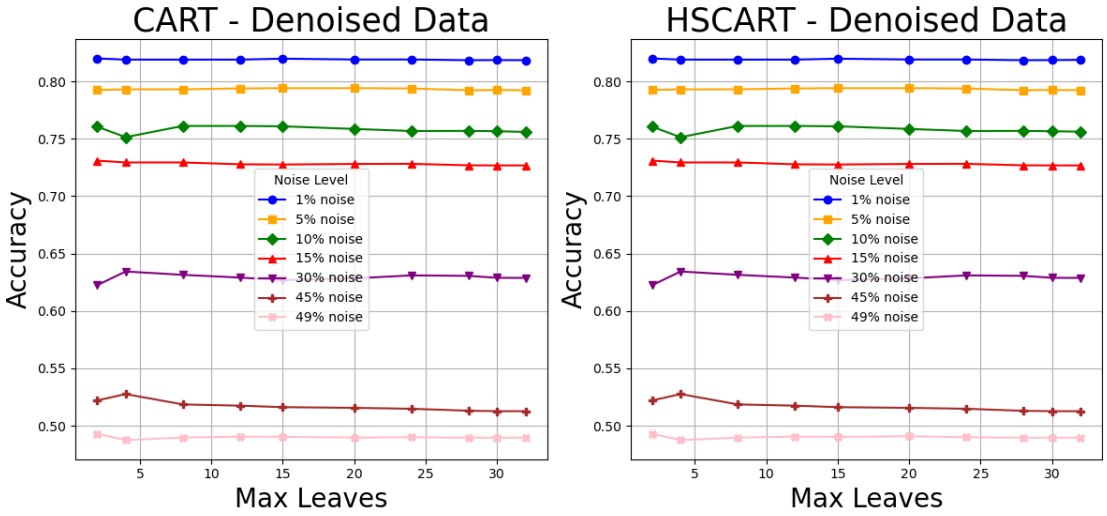


Figure 4.19: Effect of increasing label noise on accuracy for the Credit Card dataset after applying confident learning (denoising the noisy data). The left figure (CART) is a standard decision tree, while the right figure (HSCART) is a decision tree model with hierarchical shrinkage. The y-axis shows the accuracy values while the x-axis shows the max leaf nodes.

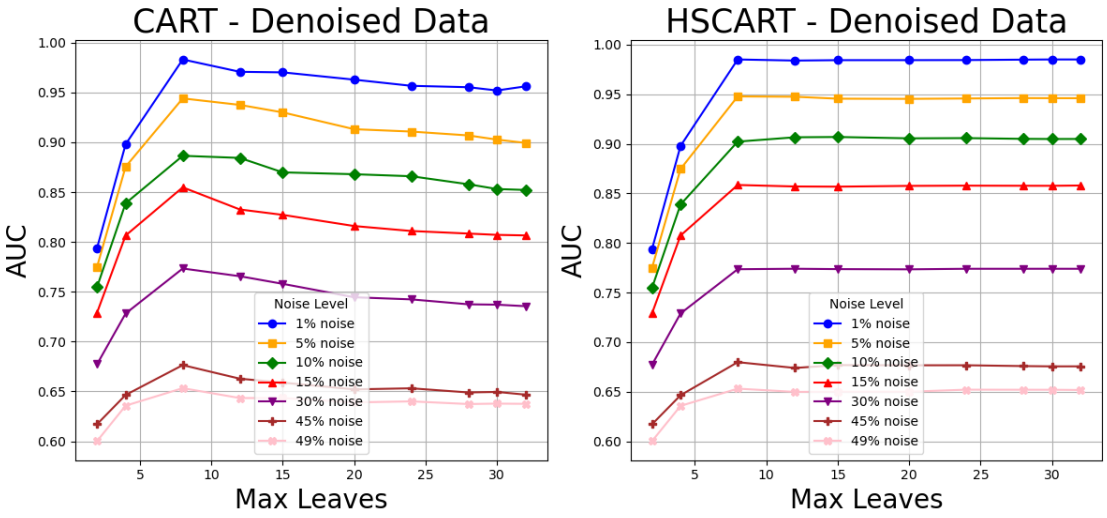


Figure 4.20: Effect of increasing label noise on accuracy for the Student Performance dataset after applying confident learning (denoising the noisy data). The left figure (CART) is a standard decision tree, while the right figure (HSCART) is a decision tree model with hierarchical shrinkage. The y-axis shows the AUC values while the x-axis shows the max leaf nodes.

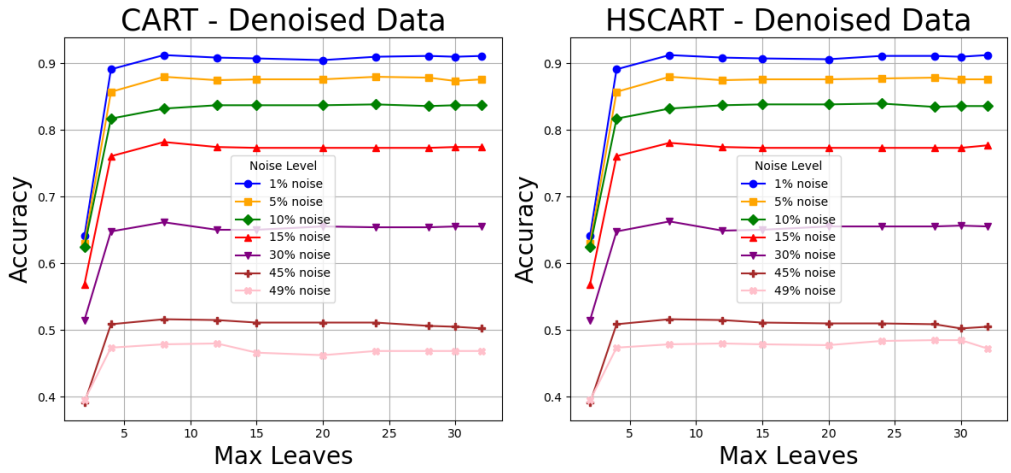


Figure 4.21: Effect of increasing label noise on accuracy for the Student Performance dataset after applying confident learning (denoising the noisy data). The left figure (CART) is a standard decision tree, while the right figure (HSCART) is a decision tree model with hierarchical shrinkage. The y-axis shows the accuracy values while the x-axis shows the max leaf nodes.

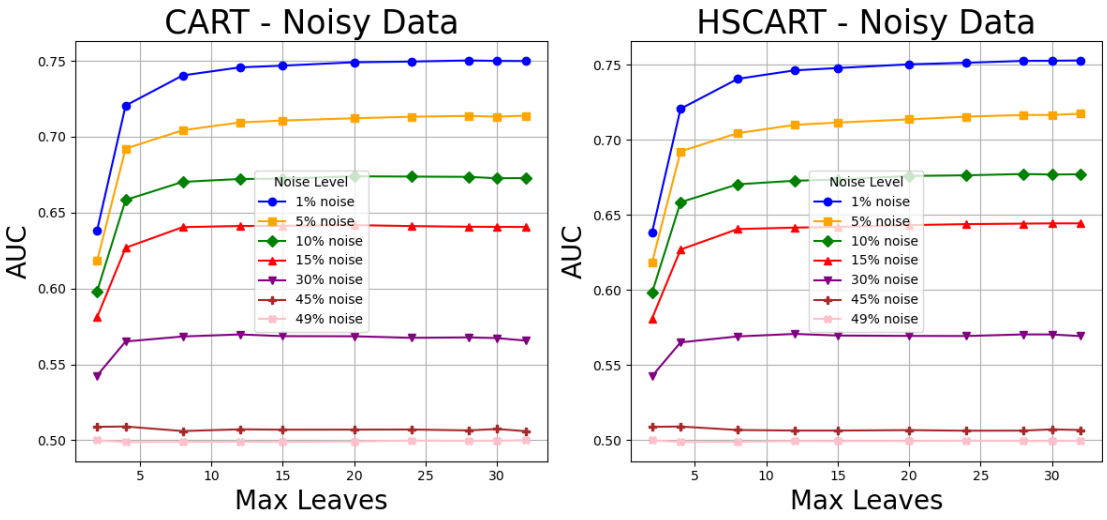


Figure 4.22: Credit Card Noisy

Figure 4.23: Effect of increasing label noise on AUC for the Credit Card dataset before applying confident learning (denoising the noisy data). The left figure (CART) is a standard decision tree, while the right figure (HSCART) is a decision tree model with hierarchical shrinkage. The y-axis shows the AUC values while the x-axis shows the max leaf nodes.

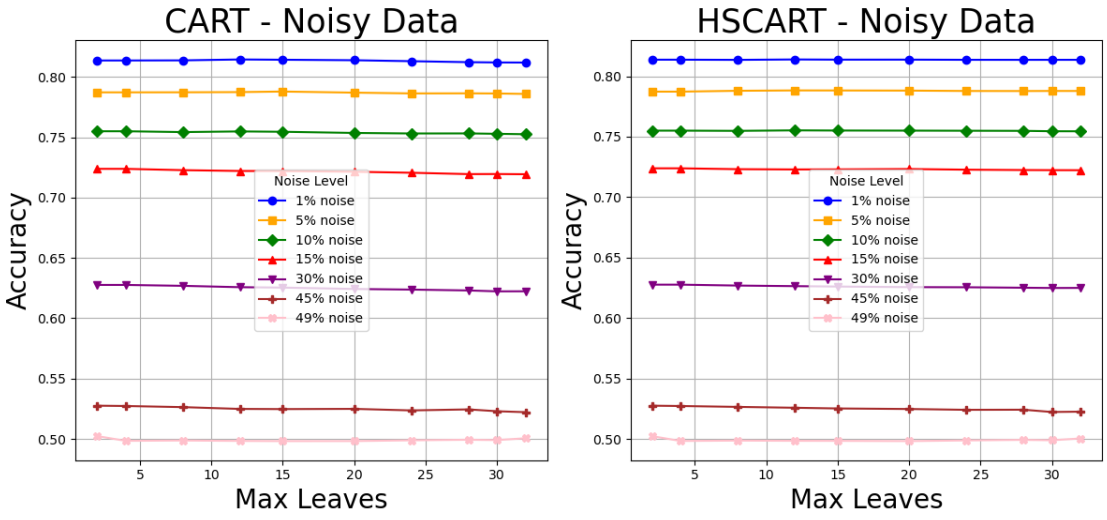


Figure 4.24: Credit Card Noisy

Figure 4.25: Effect of increasing label noise on accuracy for the Credit Card dataset before applying confident learning (denoising the noisy data). The left figure (CART) is a standard decision tree, while the right figure (HSCART) is a decision tree model with hierarchical shrinkage. The y-axis shows the accuracy values while the x-axis shows the max leaf nodes.

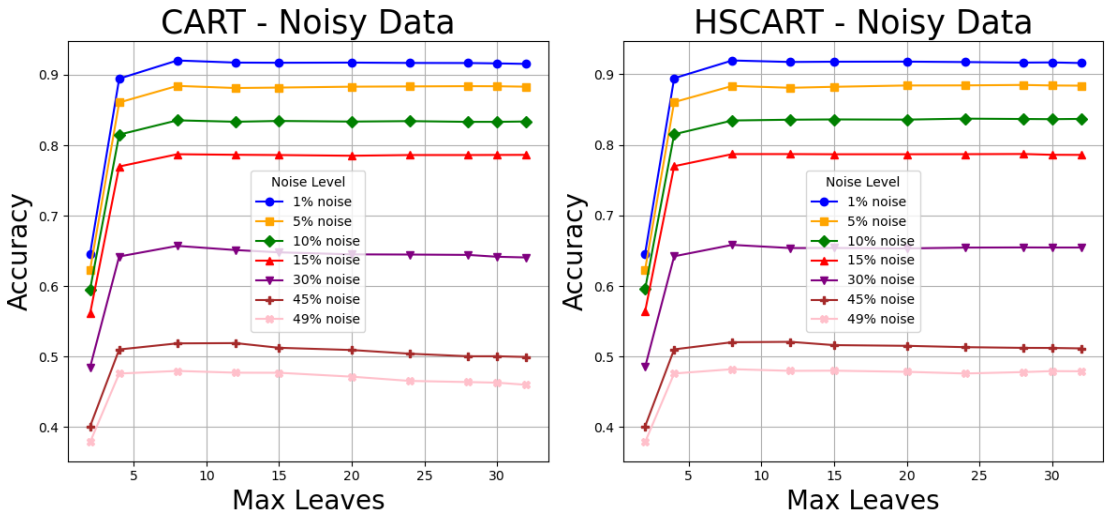


Figure 4.26: Student Performance Noisy

Figure 4.27: Effect of increasing label noise on accuracy for the Student Performance dataset before applying confident learning (denoising the noisy data). The left figure (CART) is a standard decision tree, while the right figure (HSCART) is a decision tree model with hierarchical shrinkage. The y-axis shows the accuracy values while the x-axis shows the max leaf nodes.

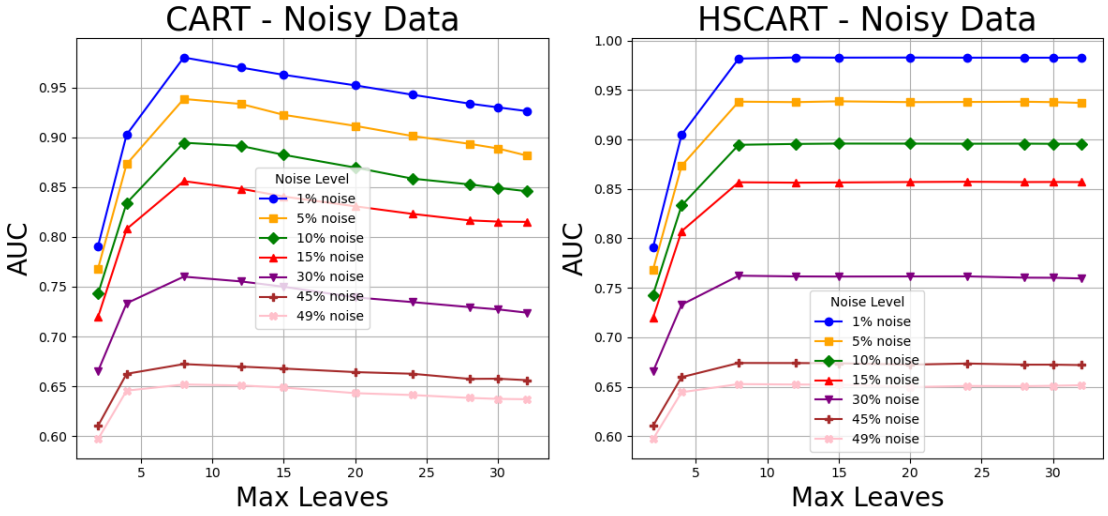


Figure 4.28: Student Performance Noisy

Figure 4.29: Effect of increasing label noise on AUC for the Student Performance dataset before applying confident learning (denoising the noisy data). The left figure (CART) is a standard decision tree, while the right figure (HSCART) is a decision tree model with hierarchical shrinkage. The y-axis shows the AUC values while the x-axis shows the max leaf nodes.

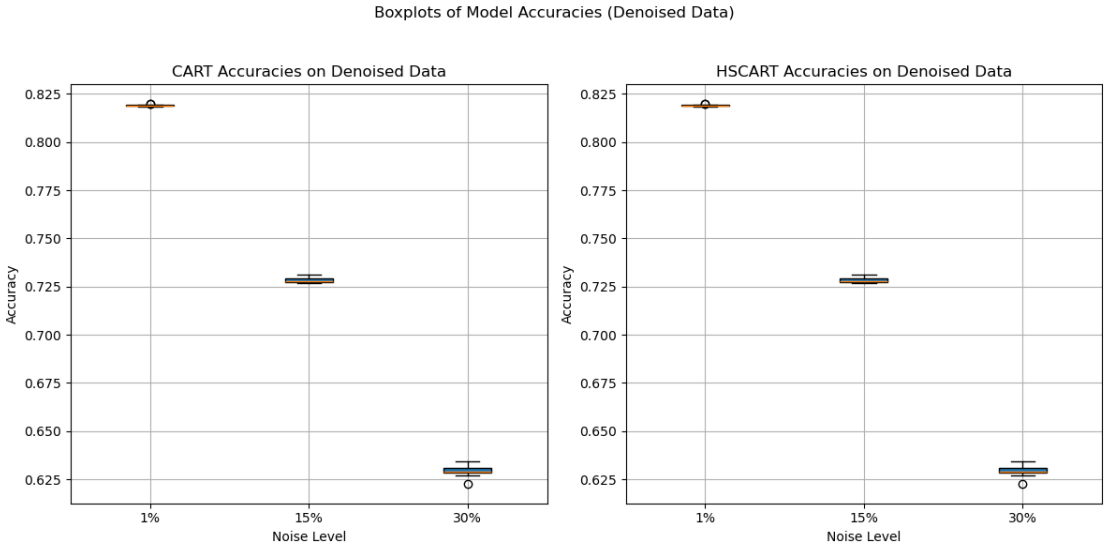


Figure 4.30: Box plots of model accuracies on the Credit Card dataset after denoising. The figure compares CART and HSCART performance across different synthetic noise levels (1%, 15%, and 30%).

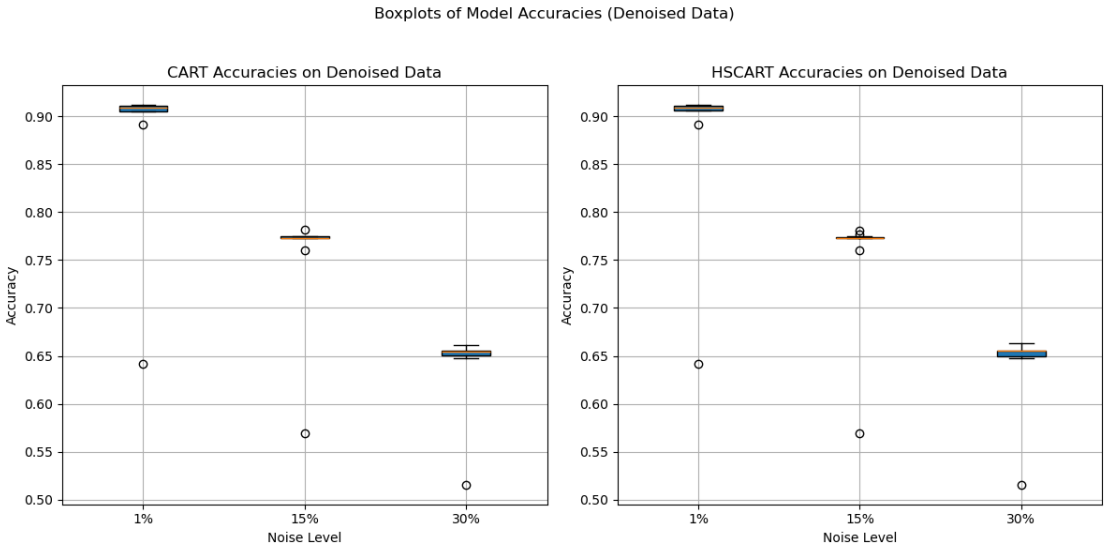


Figure 4.31: Box plots of model accuracies on the Student Performance dataset after denoising. The figure compares CART and HSCART performance across different synthetic noise levels (1%, 15%, and 30%).

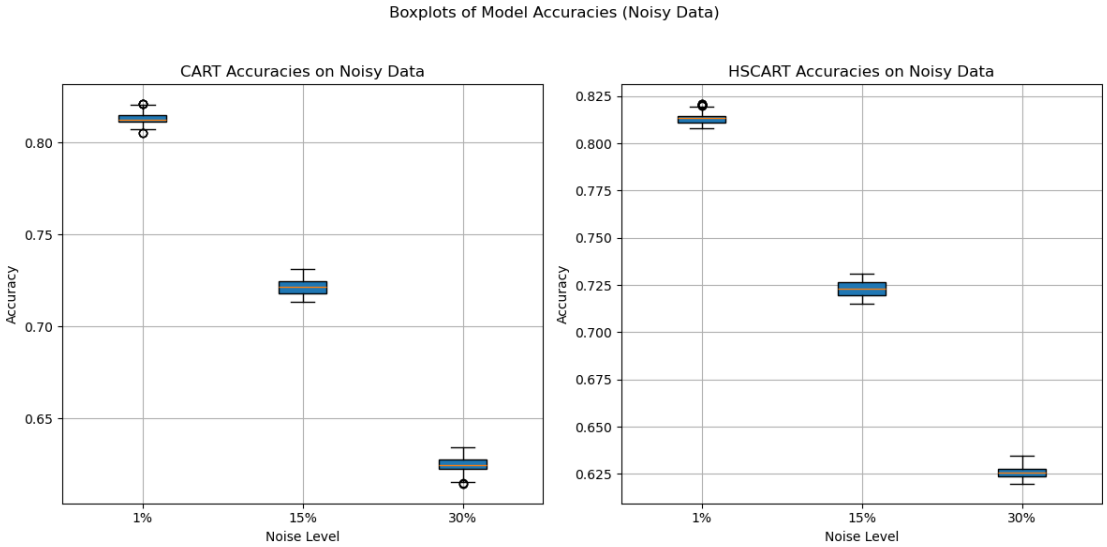


Figure 4.32: Box plots of model accuracies on the Credit Card dataset before denoising. The figure compares CART and HSCART performance across different synthetic noise levels (1%, 15%, and 30%).

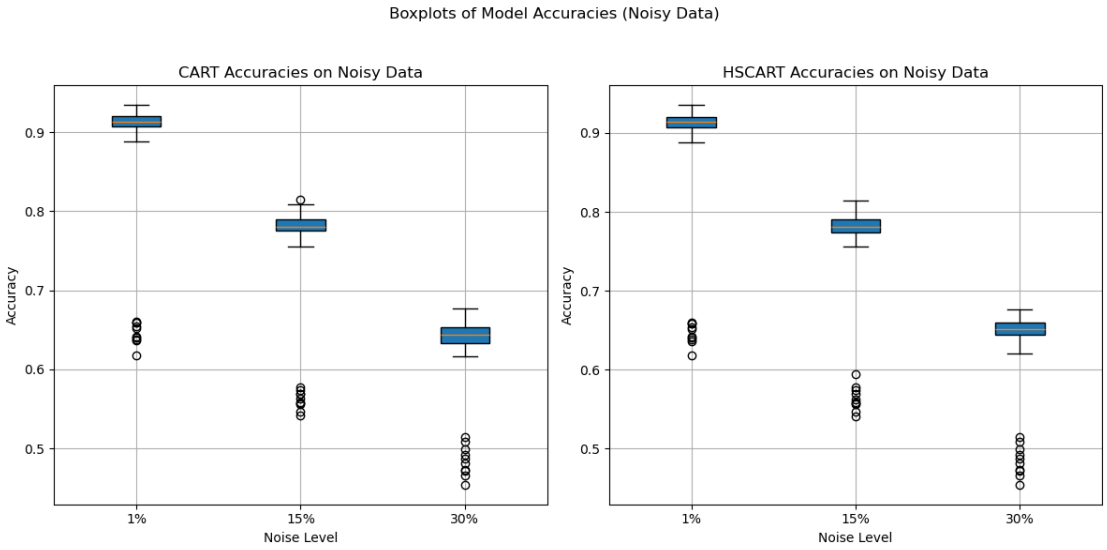


Figure 4.33: Box plots of model accuracies on the Student Performance dataset before denoising. The figure compares CART and HSCART performance across different synthetic noise levels (1%, 15%, and 30%).

Chapter 5

Discussion

5.1 Discussion

I conducted various experiments to access the performance of hierarchical shrinkage on other datasets not used by Agarwal et al. [1] and with different conditions. The subsections that follow will focus on the experiments we conducted. The experiments conducted provide a comprehensive analysis of the performance of hierarchical shrinkage applied to decision trees and random forests across a diverse range of datasets. This thesis focused on evaluating hierarchical shrinkage's effectiveness in improving model generalization, reducing overfitting, and maintaining interpretability, with a particular emphasis on comparing its performance to traditional decision trees and random forests, both with and without dimensionality reduction through Principal Component Analysis (PCA).

5.2 Artificial Data

Before evaluating hierarchical shrinkage on real-world datasets, I conducted a controlled experiment using artificially generated data. The purpose of this experiment was to perform a sanity check and observe the behavior of hierarchical shrinkage in a simplified setting, where the ground truth is known and easily visualized. This also served as the first application of

hierarchical shrinkage on data other than the tabular datasets used by Agarwal et al. [1].

Artificial datasets were generated using two simple nonlinear functions (shown in Equations 4.2 and 4.2), with added Gaussian noise to simulate real-world variability. Two random forest models were trained on this data: one standard model and one with hierarchical shrinkage applied post-training. As shown in Figures 4.1 and 4.2, the predictions of the standard random forest model appear much more erratic compared to those of the hierarchical shrinkage model.

5.3 Summary of Findings on Real-World Datasets

The experiments conducted in this thesis aim to evaluate the extent to which hierarchical shrinkage generalizes beyond the datasets originally tested [1]. Based on the results across a diverse collection of datasets, we find that hierarchical shrinkage shows indications of improvements in the performance of decision tree-based models on several high-dimensional tabular datasets, particularly in binary classification settings. These findings provide support for **H1** and **H3**, suggesting that hierarchical shrinkage can generalize well across structured, tabular domains. Additionally, we observe that these improvements are preserved when hierarchical shrinkage is applied to random forests, offering partial support for **H4**. The observed accuracy gains in these settings typically ranged between 2–5% compared to baseline models.

In contrast, for image classification tasks such as CIFAR-10 and Oxford Pets, hierarchical shrinkage did not yield consistent improvements—even after applying PCA to reduce dimensionality. Based on these experiments, we interpret the results as supporting **H2**, which perhaps indicates that hierarchical shrinkage is less effective for image-based learning tasks. This may be due to the inherently different structure of image data, where decision trees already struggle to capture meaningful spatial patterns.

Our experimental findings suggest that the improvements gained by hierarchical shrink-

age are statistically significant when working with high-dimensional tabular data, but no conclusion could be drawn (in a statistical significant way) regarding the image datasets, as well as on a tabular dataset (Titanic) where the dimension as well as the number of instances was low.

Regarding dimensionality reduction, we observed that PCA was effective in reducing computational cost and, in some cases, improved accuracy—particularly when paired with hierarchical shrinkage on high-dimensional tabular data. However, these benefits were not universal and often depended on the dataset's structure and noise level. These findings suggest that PCA may serve as a useful complement to hierarchical shrinkage in specific scenarios, though its interaction with HS is dataset-dependent.

Finally, in experiments involving synthetic label noise, the use of confident learning was inconsistent and no definite conclusions can be drawn. These results do not offer support for **H5**, and additional experiments will need to be conducted to properly test this hypothesis.

Taken together, these findings point to hierarchical shrinkage as a promising regularization technique for decision trees and random forests in structured data settings, with its effectiveness influenced by data dimensionality, label noise, and task complexity.

5.3.1 Performance Across Datasets

The results, summarized in Tables 4.4 and 4.5, indicate that, based on the set of experiments conducted across eleven datasets, models incorporating hierarchical shrinkage tended to perform better than their unregularized counterparts. This trend was especially notable in tabular datasets such as Adult Income, Titanic, and Credit Card, where hierarchical shrinkage improved accuracy by an average of 3–5%. We interpret this as an indication that hierarchical shrinkage is beneficial in structured, low-dimensional settings where overfitting is more pronounced.

In settings where PCA was applied prior to model training, performance was slightly reduced but remained comparable in most cases. This suggests that dimensionality reduction

helped eliminate noisy or redundant features, while hierarchical shrinkage further stabilized the model's predictions.

For image datasets such as CIFAR-10 and Fashion-MNIST, the benefits of hierarchical shrinkage were less pronounced. Based on the experiments conducted in this domain, models with HS showed marginal or no improvements over their baselines. However, PCA was effective at reducing input dimensionality (e.g., from 3072 to 658 for CIFAR-10 while retaining 99% of the variance), and when combined with hierarchical shrinkage, resulted in modest gains in some cases. These results suggest that while HS may be more effective on tabular data, it may offer limited benefits in high-dimensional image classification tasks.

5.3.2 Impact of Dimensionality Reduction (PCA)

PCA was applied as a preprocessing step in several experiments to assess its impact on model complexity and performance. As observed in Tables 4.4 and 4.6, PCA not only reduced the input dimensionality but also led to faster training times, especially on high-dimensional datasets such as CIFAR-10, Oxford Pets, and GitHub MUSAE. However, while PCA improved computational efficiency, it occasionally resulted in a slight drop in accuracy, likely due to the loss of some discriminative information. This trade-off between dimensionality reduction and performance was mitigated when combined with hierarchical shrinkage, which further improved model regularization.

5.3.3 Handling Noisy Data with Confident Learning

Label noise is a common challenge in real-world datasets, and to assess the resilience of our models to noisy labels, I incorporated confident learning via the Cleanlab library to identify and correct mislabeled instances. Figures 4.18 and 4.19 show results on the Credit Card dataset, highlighting the impact of denoising on both AUC and accuracy metrics. However, due to the inconsistent results shown in Section 4.7.1, no definite conclusions can be reached and additional experiments will need to be carried out.

5.3.4 Time Complexity and Computational Efficiency

An analysis of computational efficiency, shown in Tables 4.6 and 4.7, highlights the trade-offs associated with hierarchical shrinkage. While hierarchical shrinkage introduces an additional layer of regularization that increases training time, the overhead remains manageable in most cases. For large-scale datasets, such as CIFAR-10 and Oxford Pets, the additional time required for hierarchical shrinkage was offset by the computational efficiency gained through PCA, maintaining reasonable training times. In contrast, smaller tabular datasets (e.g., Titanic and Credit Card) showed negligible differences in time requirements across all models.

5.3.5 HS for Multi-Class and Imbalanced Datasets

Based on the experiments conducted on multi-class classification tasks, such as Student Performance and Multivariate Gait Data, models that incorporated hierarchical shrinkage demonstrated modest improvements in overall accuracy. These improvements were typically in the range of 1–3% over standard decision trees and random forests. Given the added complexity of distinguishing between more than two classes, we interpret these gains as an indication that hierarchical shrinkage may contribute to more stable decision boundaries in multi-class settings. However, the benefit appeared less pronounced compared to binary classification tasks.

In datasets with pronounced class imbalance — such as Credit Card and Internet Advertisements — hierarchical shrinkage yielded more consistent improvements, typically between 2–4% in accuracy compared to unregularized models, as shown in Tables 4.4 and 4.5. We interpret this as an indication that hierarchical shrinkage can help mitigate overfitting to the majority class, possibly by reducing the variance in decision paths that involve sparsely represented outcomes. That said, additional analysis would be necessary to determine whether these improvements stem specifically from better minority class performance, or from a general stabilizing effect across the tree.

5.3.6 Comparison Between Decision Trees and Random Forests

Across the experiments conducted in this thesis, and as shown in Tables 4.4 and 4.5, random forests generally outperformed individual decision trees, particularly on high-dimensional datasets such as CIFAR-10 and GitHub MUSAE. This result is consistent with the ensemble-based nature of random forests, which combine multiple decision trees to reduce variance and improve generalization.

When hierarchical shrinkage was applied to random forests, additional gains in performance were observed in several cases — especially on datasets with noisy labels and high feature dimensionality. These improvements were typically modest, but they suggest that hierarchical shrinkage may offer an additional layer of regularization even when applied to ensemble models.

Although random forests are computationally more expensive than single decision trees, the observed accuracy improvements often justified this cost, particularly when combined with shrinkage and dimensionality reduction. However, further investigation is needed to assess how these benefits scale with larger datasets or deeper forest architectures.

Chapter 6

Conclusion

6.1 Conclusion

This thesis investigated the application of hierarchical shrinkage (HS) as a post-hoc regularization technique for decision trees and random forests, with the goal of improving predictive performance while maintaining interpretability. The motivation for this study stems from the well-known trade-off between model accuracy and transparency—particularly relevant in high-stakes domains such as healthcare and finance, where black-box predictions may be difficult to trust or validate. While decision trees are often considered interpretable, they can easily overfit, especially in high-dimensional or noisy environments. Hierarchical shrinkage seeks to address this issue by introducing structured regularization that smooths predictions without altering the underlying tree structure.

The experiments conducted across a range of tabular, image-based, and artificially generated datasets suggest that hierarchical shrinkage is most effective in structured tabular settings, particularly in binary classification and imbalanced data scenarios. In these cases, hierarchical shrinkage consistently improved model accuracy by 2–5%, both when applied to single decision trees and when extended to random forests. These findings support the idea that HS can generalize beyond the datasets originally explored in prior work. In fact, our

experimental findings suggest that the improvements gained by hierarchical shrinkage are statistically significant when working with high-dimensional tabular data, but no conclusion could be drawn (in a statistical significant way) regarding the image datasets, as well as on a tabular dataset (Titanic) where the dimension as well as the number of instances was low.

Dimensionality reduction using Principal Component Analysis (PCA) was also evaluated as a preprocessing step. The results indicate that PCA can reduce training time and occasionally enhance accuracy, particularly for high-dimensional tabular datasets. When combined with hierarchical shrinkage, PCA sometimes amplified performance gains; however, the interaction between these two techniques varied depending on the dataset and was not universally beneficial.

In contrast, hierarchical shrinkage provided little to no performance improvement on image datasets such as CIFAR-10 and Oxford Pets. This was true both before and after PCA-based dimensionality reduction. These results suggest that the underlying limitations may lie not in the regularization method itself, but in the suitability of decision trees for image classification tasks. Additional work is needed to more thoroughly assess HS in visual domains and potentially combine it with models that better exploit spatial structure.

Experiments on synthetic datasets further confirmed the stabilizing effect of hierarchical shrinkage. Even in the presence of Gaussian noise, HS led to smoother and more accurate function approximations, suggesting its robustness in controlled scenarios. Similarly, experiments involving synthetic label noise showed that confident learning improved performance under noise, and that pairing it with HS led to additional robustness in some settings—although these effects varied across datasets.

In summary, the findings from this thesis highlight hierarchical shrinkage as a promising regularization method for interpretable machine learning, particularly in low-dimensional, structured, and noisy settings. Its performance on image data remains inconclusive and warrants further investigation. Future work may explore its integration with more expressive models, evaluate its impact on interpretability in real-world applications, and develop

diagnostic tools to better predict when and where hierarchical shrinkage is likely to be effective.

Bibliography

[1] Abhineet Agarwal, Yan Shuo Tan, Omer Ronen, Chandan Singh, and Bin Yu. Hierarchical Shrinkage: Improving the accuracy and interpretability of tree-based models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 111–135. PMLR, 2022. URL https://proceedings.mlr.press/v162/agarwal22b.html. Accessed: 2023-10-02.

[2] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[3] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

[4] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Chapman & Hall/CRC, 1984.

[5] Yoav Freund and Robert E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. doi: 10.1006/JCSS.1997.1504. URL https://doi.org/10.1006/jcss.1997.1504.

[6] Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[7] Isabelle Guyon and André Elisseeff. Introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[8] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009. URL https://hastie.su.domains/ElemStatLearn/. Accessed: 2023-09-12.

[9] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. doi: 10.1126/science.1127647.

[10] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[11] Ian T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics, 2nd edition, 2002. ISBN 9780387954424.

[12] Tom M. Mitchell. *Machine learning*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. ISBN 978-0-07-042807-2. URL https://www.worldcat.org/oclc/61321007.

[13] Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. Confident Learning: Estimating Uncertainty in Dataset Labels. *Journal of Artificial Intelligence Research (JAIR)*, 70: 1373–1411, 2021.

[14] Curtis G. Northcutt, Jonas Mueller, and Anish Athalye. Cleanlab: The standard data-centric AI package for data quality and machine learning with noisy labels. https://github.com/cleanlab/cleanlab, 2021. Accessed: 2025-02-01.

[15] J. Ross Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986.

[16] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

[17] Chandan Singh, Keyan Nasseri, Yan Shuo Tan, Tiffany Tang, and Bin Yu. imodels: a python package for fitting interpretable models, 2021. URL https://doi.org/10.21105/joss.03192. Accessed: 2023-11-18.

[18] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[19] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.