Enhanced Mixed Integer Programming-based Classification and Regression Trees with Advanced Feature Engineering Techniques

Author:

Mingyu Zhang

Supervisor:

Dr. Hamideh Anjomshoa

School of Mathematics and Statistics

The University of Melbourne

2023

Thesis submitted for the degree of Master of Science

Abstract

Classification and Regression Trees (CART) is a widely adopted machine learning methodology known for its interpretability and ability to accommodate non-linear relationships between predictor variables and target outcomes. However, the traditional CART algorithm employs a greedy, local optimization approach, which may not yield globally optimal decision trees. Mixed Integer Programming (MIP) has been proposed as an alternative framework for constructing decision trees, offering global optimization capabilities to address the limitations of conventional CART methods.

The objective of this research project is to develop an advanced MIP-based CART model that integrates multiple enhancements to boost performance and efficiency, ultimately delivering a more accurate and efficient decision tree learning approach. The proposed MIP-based CART model employs the following techniques: (1) Warm Start strategy for expediting the MIP solver, (2) Principal Component Analysis (PCA) for dimensionality reduction, (3) Random Forest Feature Selection method for identifying the most informative features, (4) K-Means clustering for generating additional features that encapsulate complex data patterns, and (5) LASSO regularization for encouraging sparsity and refining feature selection.

The findings highlight that our proposed enhancements effectively achieve competitive performance within limited time constraints, which holds significant practical value. In the majority of scenarios, our proposed methods surpass CART in building depth-2 trees under strict time restrictions, a desirable outcome when interpretability is crucial.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Hamideh Anjomshoa, for your exceptional guidance, unwavering support, and mentorship throughout this research project. In times when I faced challenges and felt overwhelmed, your invaluable advice, encouragement, and patience helped me navigate and overcome these difficulties. The completion of this research project would not have been possible without your dedication. Throughout my graduate studies, you not only cared about my academic progress but also fostered my overall personal growth. When I faced challenges in my coursework, you proactively offered assistance, invited me to academic study group, and generously shared your insights on career development, including providing advice and sharing job opportunities during my job search. I extend my sincerest thanks to you.

I am also deeply grateful to my parents for their unwavering support, both emotionally and financially. Without them, I would not be where I am today. Their unconditional love and enduring encouragement made all of this possible, and I want to thank them wholeheartedly.

Additionally, I would like to express my gratitude to my friends who have been a constant source of support, assistance, and motivation during my time as a graduate student. I am particularly grateful to Zixuan Guo, Runyu Chen, and Bingqing Zhang, who not only patiently listened to my relentless discussions about my research project but also contributed their insights and ideas. Their friendship and encouragement played a pivotal role in my growth, and I am truly thankful for their presence in my life.

Contents

Chapter 1

Introduction

1.1 Motivation

Decision trees are a popular machine learning technique for classification and regression tasks. Unlike other predictive models, decision trees do not rely on strong parametric assumptions, making them applicable to a wide range of datasets. Furthermore, the nature of decision tree algorithms involves recursively partitioning the sample space into disjoint groups, resulting in a tractable and interpretable model. The ability to explain why a prediction was made is crucial in various fields where understanding the underlying decision-making process is essential.

The Classification and Regression Trees (CART) algorithm, similar to many other decision tree methods, adopts a local optimisation approach to construct the tree structure. It searches for an optimal solution at each split without considering the implications of previous or future splits. Consequently, a suboptimal split decision at the top of the tree will have a permanent effect on its child nodes, with no way to reverse it. Although the CART algorithm produces reasonably accurate results, it does not guarantee a globally optimal decision tree. It was designed this way because finding the optimal decision tree is an *NP-complete* problem [14], and at the time Breiman *et al.* proposed this method (1984), it was considered computationally infeasible to find the optimal solution.

The advancements in computational resources and optimisation techniques over the years have made it feasible to tackle *NP-complete* problems, such as the optimal decision tree, using MIP-based approaches, which addressing the limitations of the traditional CART method by formulating the problem as a MIP to find the globally optimal tree. By guaranteeing an optimal solution, MIP-based CART can provide higher predictive accuracy and reduce overfitting, thus leading to more robust models.

1.2 Outline

The thesis is structured starting with the Introduction, followed by Preliminaries, Methodology, Results and Discussion, and finally, the Conclusion. The Preliminaries chapter provides an overview of the CART algorithm, highlights its limitations, and introduces the OCT algorithm with MIP as an alternative for achieving optimal decision trees.

In the Methodology chapter, we first present the MIP formulation for the OCT algorithm, then propose several methods for improving the performance of the OCT algorithm, such as incorporating Warm Start, PCA as a dimension reduction technique, Random Forest Feature Selection, K-Means as extra features, and LASSO regularisation.

The Results and Discussion chapter presents the results of our experiments, comparing the performance of the improved OCT algorithms with that of the original OCT and CART.

The Conclusion chapter summarises our findings, discusses the practical implications of the research, and acknowledges the limitations of the study. We also suggest potential directions for future research.

Chapter 2

Preliminaries

In this chapter, we provide an overview and discuss the limitations of the CART algorithm, then introduce the mixed-integer programming (MIP) approach for constructing optimal decision trees.

2.1 An overview of CART

The CART algorithm employs a recursive binary splitting method to construct the tree. It begins with the root node, which encompasses all observations, and searches for the optimal split to divide the data points into two offspring nodes, with the objective of minimizing the sum of the impurity measures of the two nodes. The impurity measure evaluates the homogeneity of classes within a node; typical impurity measures include the Gini Index and Cross Entropy for classification tasks, as well as Mean Square Error (MSE) and Mean Absolute Error (MAE) for regression tasks [18].

CART handles both continuous and categorical predictors. For continuous predictors, the algorithm typically searches for the best split threshold by sorting the unique values of the predictor and computing the impurity for each possible split between those values. For categorical predictors, different strategies can be employed, such as splitting the categories into two disjoint sets or using a one-versus-all approach for each category. Depending on the implementation, some CART algorithms may convert categorical predictors into binary dummy variables via One-Hot

Encoding and treat them as continuous predictors [18].

The size of the tree is important: too large a tree might fit the training data perfectly but does not generalise well to make a good prediction for test data, and too small a tree might fail to unearth a strong split hidden under a weak one, and perform poorly on making a prediction for unseen data as well. Therefore, finding the balance between model accuracy and complexity is critical in determining the optimal tree size. There are two main approaches to control the growth of the tree: user-defined parameters and pruning techniques.

1. **User-defined Parameters:** This approach involves setting constraints on the tree growth directly during the tree-building process. The tree will stop growing if any of these constraints are met. Some common user-defined parameters include:

   - Minimum number of points in a terminal node: The tree will not split a node further if the node contains the number of data points less than this value.
   - Maximum depth of the tree: The tree will not grow further beyond this specified depth.
   - Minimum improvement threshold in impurity measure: The tree will not split a node if the improvement in the impurity measure is less than the specified threshold.

2. **Pruning Techniques:** The main idea of pruning is to prune out the weak branches (remove some of the child nodes and merge the corresponding predictor space) after growing a significantly large tree (larger than we need), which leaves only the important splits. Recall that impurity measure is utilised to guide the accuracy of the tree, now in order to prevent a tree from overfitting, a complexity parameter denoted by α (≥ 0) is added to the equation to penalise the complexity of the tree. A larger α results in more penalties for the tree size, leading to a smaller tree. Conversely, a smaller α imposes less penalty on a complex tree, resulting in a larger tree size. For a given α, the weakest link pruning is used to identify the tree with the highest accuracy, the

algorithm starts by calculating the accuracy change over each split if instead merging its child nodes, then a non-terminal node with the smallest change in accuracy will be pruned. This pruning process continues until all splits exhibit an accuracy improvement exceeding α. The complexity parameter α is chosen by cross-validation.

After completing building the tree structure, a prediction is assigned to each terminal node. In classification tasks, the most frequently occurring class label within the node is designated as the predicted class label, and any new data points that traverse the structure and land in that node will receive the corresponding class prediction. Meanwhile, in regression tasks, the average value of the response variables present in each leaf serves as the prediction.

Common performance metrics for evaluating CART models include accuracy, precision, recall, F1 score, and area under the receiver operating characteristic (AUC ROC) curve for classification problems, and mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and R-squared for regression problems [18, 5, 21]. These metrics can be used to assess the quality of the model, its predictions, and its generalisation performance on unseen data. Cross-validation techniques can further help in model evaluation and selection by providing an unbiased estimate of the model's performance [18].

To provide a concrete example, we implemented the CART algorithm on the well-known IRIS dataset obtained from the UCI Machine Learning Repository [11]. The dataset comprises 150 samples from three distinct iris flower species, with each contributing one-third of the dataset. For each flower, four attributes are recorded: sepal length, sepal width, petal length, and petal width. The objective is to classify the flower species and predict a given iris flower's species based on these four measurements. Figure 2.1 shows a fully grown tree using the CART algorithm without any restrictions. The algorithm starts by identifying a split on the petal width to completely separate out fifty "setosa" samples, the split stops for this node because the class label in it is identical. Then another split is made on petal width, which separates most of "versicolor" and "virginica" species. In this case, the splits continue until the class labels for all terminal nodes are the same, in another word, the node is pure. Figure 2.2 is the corresponding splitting in the feature space and class labels assigned to each space. The CART algorithm perfectly classifies the three species in 5 depths, but we all know by now that it is overfitted.

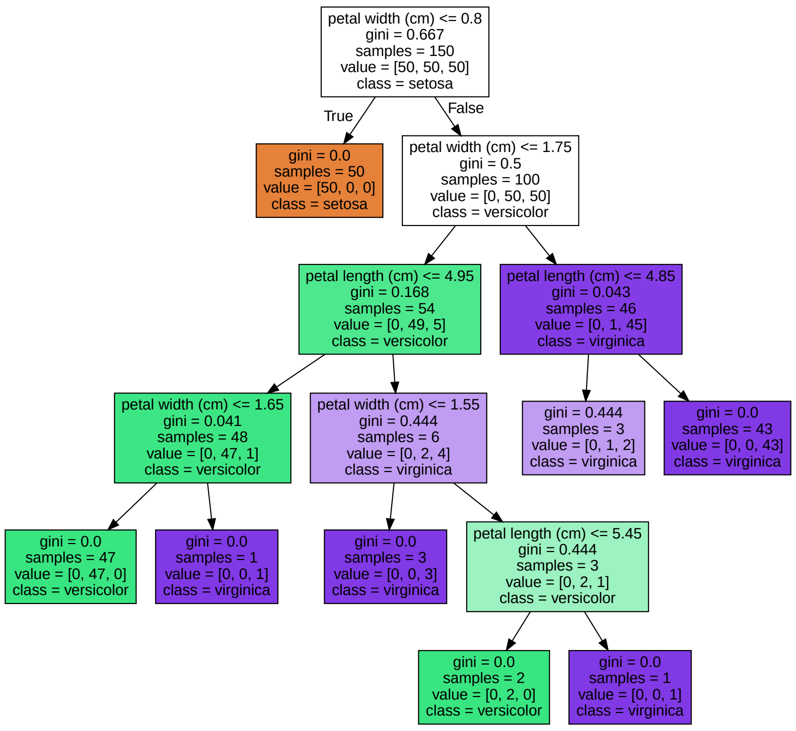

Figure 2.1: A fully grown tree using CART algorithm for the IRIS dataset.

In Figure 2.1, we observed several weak splits. For instance, the right-most branching of the second depth is weak in which the generated child nodes have an even larger summed Gini index, and the left-most branching of the third depth is also weak because the total accuracy improvement is negligible, there is only one point accuracy improved with this branch. Figure 2.3 shows a refined tree after pruning with the complexity parameter $\alpha = 0.01$. As we can see, all weak splits are removed without a significant sacrifice in accuracy (Figure 2.4).

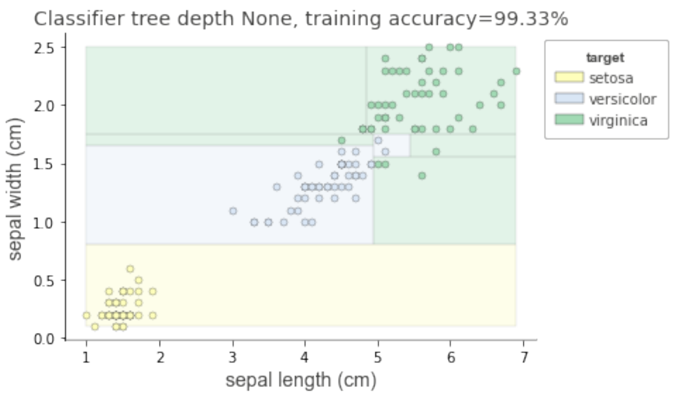

Figure 2.2: A visualisation of the fully grown tree using CART in the feature space.

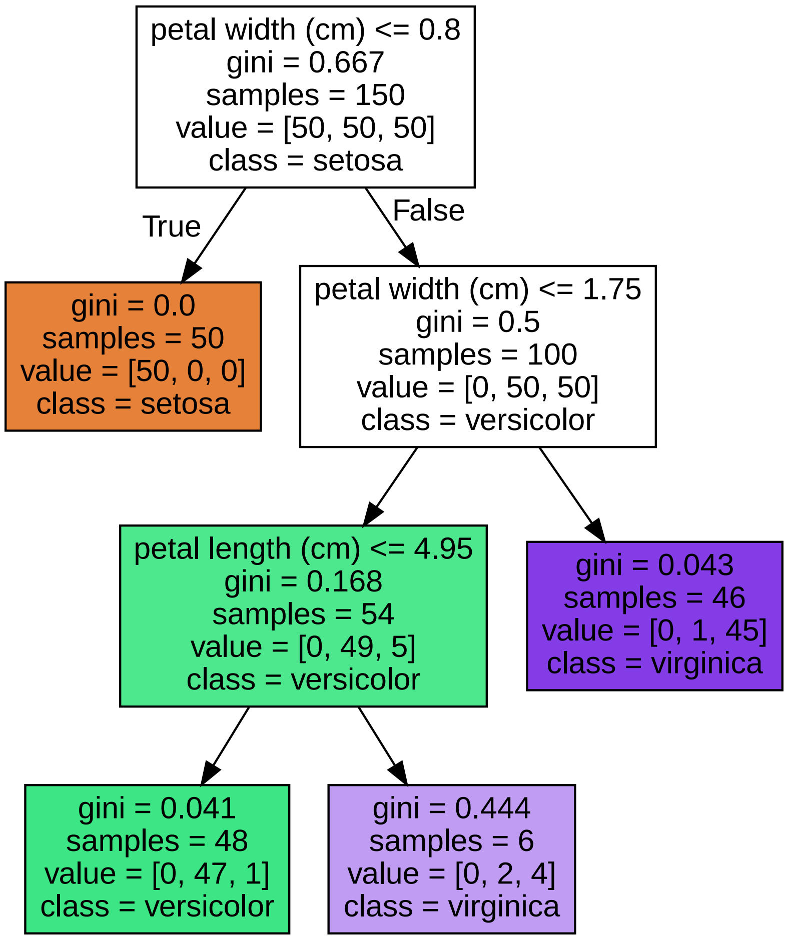

Figure 2.3: A pruned tree for the IRIS dataset.

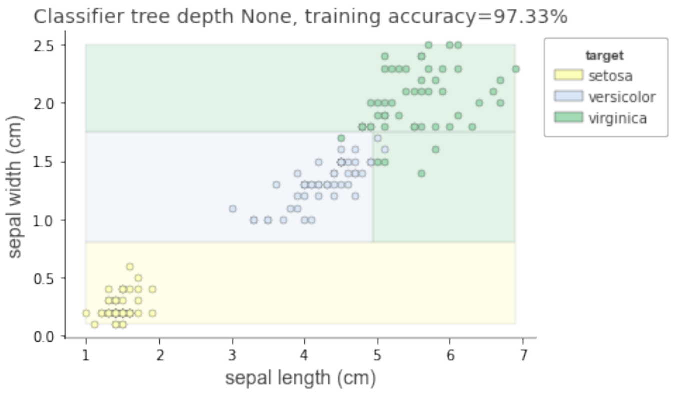

Figure 2.4: A visualisation of the pruned tree in the feature space.

2.2 Limitation of CART

Although CART is widely adopted in the practice due to its simplicity and interpretability, it has some limitations. A key limitation, shared with other well-known decision tree algorithms such as C4.5 [27] and ID3 [26], is their greedy and short-sighted approach. Trees are constructed recursively, with each split decision made to achieve local optimality; however, a bad split at the top of the tree will propagate down and there is no way to undo it [18], therefore the greedy algorithms produce suboptimal trees, which might be far from global optimal. Moreover, the top-down induction approaches of CART and the other methods are not always robust, they do not know how large to grow the tree, resulting in overfitting [17], and additional pruning step is required to prevent it from happening. In this respect, small decision trees are often more desirable; however, it leads to another disadvantage: unsatisfactory accuracy. More advanced machine learning algorithms such as random forests [6] and gradient boosting [13] was introduced to cater for better performance in term of accuracy at the cost of interpretability. Hence, building small decision trees with state-of-art performance has become the next million dollars question.

Another drawback of CART is that it is not inherently invariant to imbalanced datasets, and its performance can be negatively affected by class imbalance. This is because CART uses impurity measures like Gini Index or Cross Entropy to decide the best splits at each node. When there is a significant class imbalance, these impurity measures tend to favor splits that benefit the majority class, leading to a bias towards split that improve the classification of the majority class at the expense of the minority class.

Moreover, because CART can be affected by modest adjustments in the training data, various trees may be constructed even with slight alterations in the dataset.

2.3 Optimal Decision Trees with MIP

In recent years, research on improving the accuracy and robustness of decision trees in real-world applications is prosperous. The shortcomings of CART have prompted researchers to explore alternative methods for constructing decision trees, with a particular focus on formulating the problem as a Mixed-Integer Programming (MIP) formulation. MIP-based decision trees address some of the limitations of CART, such as suboptimality and overfitting, by seeking globally optimal solutions and incorporating regularisation techniques.

Mixed-Integer Programming (MIP) is a mathematical optimisation framework that involves both integer and continuous variables. MIP is especially relevant to decision tree construction for the following reasons. First of all, the discrete nature of constructing a decision tree makes it possible to utilise discrete optimisation to formulate the problem: we can think of constructing decision trees as a combination of several discrete decisions: choosing which variable to split on at each branch node; picking label for each leaf node; deciding which leaf node each data point goes to; whether each data point is correctly classified or not. The MIP approach allows us to build decision trees in one step, with each split decision being made given all the information of previous and future splits.

Another advantage of MIP-based formulation is its flexibility. In a classification problem, it is natural to use the misclassification error rate to evaluate the performance of the model, however, it does not work well in CART analysis when constructing the tree, because of the top-down stepwise approach [7], while in MIP-based formulation, it is advocated by multiple authors [4, 1, 29, 30, 33]. MIP brings extra flexibility in targeting different tasks. Bertsimas and Dunn and Aghaei et al. introduce extra regularisation term in the objective function to balance model accuracy and interpretability [4, 2], Aghaei et al. propose a regulariser in their objective function to penalise discrimination towards sensitive predictor variable such as gender and race[1].

The tremendous improvement in hardware and optimisation solvers such as GUROBI [15] and CPLEX [9] contributes as another reason to prompt research in optimal decision trees. It is known that building optimal decision trees is an NP-hard problem [20]. Breiman et al. mentioned in their book, decision trees are built in such a greedy heuristic approach because of the computational limitation at that time, it was computationally impossible to search through all possible branches during a time limit[7].

There are a number of researches published recently that contributes to the development of optimal decision trees harnessing the MIP techniques. Given the fixed depth of the decision tree, Bertsimas and Dunn [4] encode each discrete decision that is required to construct a decision tree and assign a label to the leaf node as a binary decision variable, the objective function is to minimise the misclassification error. The model can tackle both continuous and categorical predictor variables. Due to a large number of decision variables, the optimal solution becomes hard to certify for moderate and large datasets. Based on the work by Bertsimas and Dunn, Verwer and Zhang [30] proposed a new formulation that has fewer decision variables and constraints, specifically the size of the model is independent of the data size. The training time significantly decreases compared with the previous model, however, fewer variables and constraints lead to weaker LP relaxation, which therefore results in a slight improvement in accuracy.

In addition to the aforementioned research, other techniques to build optimal decision trees using MIP formulations have also been proposed. For instance, the 1-norm-SVM method [33] introduces a support vector machine (SVM)-based formulation for constructing decision trees, which can provide a more robust and accurate model. This approach uses an SVM-like loss function with an additional 1-norm regularisation term to control the tree's complexity, resulting in a more balanced trade-off between model accuracy and interpretability. Another interesting approach is the flow-based optimal decision tree (ODT) method [2], which leverages network flow optimisation techniques to build decision trees. By representing the tree construction process as a flow problem, this method can efficiently solve large-scale instances while maintaining strong LP relaxations. The combinational branching decision rules technique [17] introduces a novel way of constructing decision trees by allowing multiple variables to be combined in a single split. This approach can lead to more compact and interpretable trees by considering interactions between variables during the tree construction process. The column generation approach [10] tackles the challenge of constructing optimal decision trees for large datasets by dynamically generating decision variables during the optimisation process. This method can significantly reduce the number of variables and constraints in the MIP formulation, potentially leading to faster convergence and more efficient solutions.

The development of optimal decision trees using MIP formulations has attracted significant research interest, driven by the advantages of MIP techniques and the continuous improvement in optimisation solvers and hardware. Several approaches have been proposed to tackle different aspects of the tree construction process, and further research in this area has the potential to improve the quality and interpretability of decision tree models.

Chapter 3

Methodology

3.1 MIP formulation

A significant portion of the research efforts discussed above are built upon the Optimal Classification Tree model proposed by Dimitris Bertsimas and Jack Dunn [4]. Their work pioneered the application of a Mixed Integer Optimisation (MIO) framework related to CART, offering a globally optimal solution in contrast to the local optimisation approaches commonly employed in earlier methods. The framework has the ability to incorporate regularisation techniques to prevent overfitting and can be easily extended to incorporate additional constraints, such as expert knowledge, cost-sensitive learning, and multi-class classification.

In order to fully comprehend the foundations of our proposed improvements and the context in which they were developed, it is crucial to provide a comprehensive review of the MIP formulation introduced by Bertsimas and Dunn. This review will not only enhance our understanding of the OCT algorithm but also serve as a solid foundation for the subsequent sections, where we explore various strategies to enhance the performance and efficiency of the algorithm.

The optimal classification tree (OCT) is inspired by the discrete nature of CART procedure, and its Mixed-Integer Programming (MIP) formulation focuses on addressing several key questions:

- Should a node branch or not?

Set	
$\mathcal{T}_\mathcal{B}$	set of all branching nodes
$\mathcal{T}_\mathcal{L}$	set of all leaf nodes
$\mathcal{I}$	set of data points
$\mathcal{F}$	set of features
$\mathcal{K}$	set of labels
$\mathcal{A}_\mathcal{L}$(t)	set of ancestors of node t following a left branch from the root
$\mathcal{A}_\mathcal{R}$(t)	set of ancestors of node t following a right branch from the root
Index	
i	data point index, $i \in \mathcal{I}$
f	feature index, $f \in \mathcal{F}$
t	node index, $t \in \mathcal{B} \cup \mathcal{L}$
$p(t)$	parent node of node t
k	class label index, $k \in \mathcal{K}$
Constant	
D	the predetermined tree depth, the root node has depth 0
N_{min}	minimum leaf node size
x_f^i	value of feature f of data point i
Y_{ik}	One-Hot Encoding of label y, 1 if y_i = k, -1 otherwise
$\hat{L}$	baseline accuracy by predicting the most occurred class label
α	complexity parameter
Decision variable	
d_t	1 if apply branch on node t
a_{tf}	1 if feature f is used at branching node t
c_{tk}	1 if class label k is assigned to leaf node t
z_{it}	1 if data point i is at leaf node t
l_t	1 if node t contains at least one data point
b_t	threshold value for branching node t
L_t	the number of mis-classified data point at leaf node t

Table 3.1: Summary of notation.

- If branching occurs, which feature should be used for the split?
- If branching stops, which label should be assigned to the resulting leaf node?
- After constructing the tree structure, how should data points be assigned to leaf nodes while respecting the established tree structure?

The MIP formulation is as follows.

Objective

$$\min \quad \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_{\mathcal{L}}} L_t + \alpha \sum_{t \in \mathcal{T}_{\mathcal{B}}} d_t \tag{3.1}$$

Constraints

$$\sum_{f\in\mathcal{F}} a_{ft} = d_t \qquad \forall t \in \mathcal{T}_\mathcal{B} \tag{3.2}$$

$$0 \leq b_t \leq d_t \qquad \forall t \in \mathcal{T}_\mathcal{B} \tag{3.3}$$

$$d_t \leq d_{p(t)} \qquad \forall t \in \mathcal{T}_\mathcal{B}\backslash\{1\} \tag{3.4}$$

$$z_{it} \leq l_t \qquad \forall t \in \mathcal{T}_\mathcal{L} \tag{3.5}$$

$$\sum_{i\in N} z_{it} \geq N_{min} l_t \qquad \forall t \in \mathcal{T}_\mathcal{L} \tag{3.6}$$

$$\sum_{t\in\mathcal{L}} z_{it} = 1, i = 1, \dots, n \tag{3.7}$$

$$\sum_{f\in\mathcal{F}} a_{mf} x_f^i < b_t + M_1(1 - z_{it}), i = 1, \dots, n \qquad \forall t \in \mathcal{T}_\mathcal{B}, m \in \mathcal{A}_\mathcal{L}(t) \tag{3.8}$$

$$\sum_{f\in\mathcal{F}} a_{mf} x_f^i \geq b_t - M_2(1 - z_{it}), i = 1, \dots, n \qquad \forall t \in \mathcal{T}_\mathcal{B}, m \in \mathcal{A}_\mathcal{R}(t) \tag{3.9}$$

$$N_{kt} = \frac{1}{2}\sum_{i=1}^{n}(1 + Y_{ik}) z_{it}, k = 1, \dots, K \qquad \forall t \in \mathcal{T}_\mathcal{L} \tag{3.10}$$

$$N_t = \sum_{i=1}^{n} z_{it} \qquad \forall t \in \mathcal{T}_\mathcal{L} \tag{3.11}$$

$$\sum_{k\in\mathcal{K}} c_{tk} = l_t \qquad \forall t \in \mathcal{T}_\mathcal{L} \tag{3.12}$$

$$L_t \geq N_t - N_{kt} - M(1 - c_{kt}), k = 1, \dots, K \qquad \forall t \in \mathcal{T}_\mathcal{L} \tag{3.13}$$

$$L_t \leq N_t - N_{kt} + M c_{kt}, k = 1, \dots, K \qquad \forall t \in \mathcal{T}_\mathcal{L} \tag{3.14}$$

$$L_t \geq 0 \qquad \forall t \in \mathcal{T}_\mathcal{L} \tag{3.15}$$

$$z_{it}, l_t \in \{0, 1\}, i = 1, \dots, n \qquad \forall t \in \mathcal{T}_\mathcal{L} \tag{3.16}$$

$$a_{ft}, d_t \in \{0, 1\}, f = 1, \dots, p \qquad \forall t \in \mathcal{T}_\mathcal{B} \tag{3.17}$$

The objective function (3.1) aims to minimise the total number of misclassified data points across all leaf nodes. In this equation, $\hat{L}$ serves as a normalisation term, representing the foundational accuracy achieved by forecasting the predominant class label, and α controls the trad-off between accuracy and interpretability, a large α penalises complex tree, which will force small tree structure.

Constraints (3.2), (3.3), (3.4) are used to set up the tree structure. Constraint (3.2) forces that only one feature is used for applying a branch at branching node t, constraint (3.3) is valid because as the value of the predictable variables $\mathbf{x}$ is normalised to the 0-1 interval, we just need to investigate b_t in this range. Constraint (3.4) enforces the tree's hierarchical structure, it makes sure a branch is executed at node t if its parent node applied a branch. Constraints (3.5), (3.6), and (3.7) manage the data points allocation to leaf nodes, they prevent the label assigned to an empty leaf node, and make sure that each leaf node contains at least N_{min} data points, if it is not empty. Constraint (3.7) forces each data point to fall in only one of the leaf nodes.

Constraint (3.8) and (3.9)enforce the necessary divisions in the tree's structure when allocating points to leaves: If a data point i is assigned to leaf node t, for all of its ancestors following a left branch in the path from the root node to leaf node t, the feature value x_f^i will be less than the threshold b_t, and vice versa. One advantage of these formulations is that it allows early termination of branching, which is the case when $d_t = 0$, both $\mathbf{a}_t$ and b_t are force to be 0, $0 \geq 0$ will be always true, which forces all instances to follow a right branch, eventually end in a leaf node.

Constraint (3.8) employs a strict inequality which can lead to the problem being infeasible or unbounded, a small ϵ is added to the left-hand side, and to prevent numerical instabilities by introducing the small ϵ, the authors aim to make it as big as possible by assigning each feature f a distinct ϵ_f. The value is calcuated by first sorting the feature values, and then taking the difference between the feature and its adjacent value, for instance, if {180, 181, 182} are the values of feature 'Height', 'x < 182' is equivalent to '$x \leq$ 181'. The largest value of M_1 and M_2 can also be specified: As previously mentioned, both $\mathbf{a}_t^T\mathbf{x}_i \in [0,1]$ and $b_t \in [0,1]$, the value of $\mathbf{a}_t^T(\mathbf{x}_i + \epsilon) - b_t$ can be at most $1 + \epsilon_{max}$, where $\epsilon_{max} = max_f\{\epsilon_f\}$. Similarly, the value of $b_t - \mathbf{a}_t^T x_i$ cannot exceed 1, thus, constraints (3.8) and (3.9) become:

$$\sum_{f \in \mathcal{F}} a_{mf}(x_f^i + \epsilon_f) < b_t + (1 + \epsilon_{max})(1 - z_{it}), i = 1, \ldots, n \quad \forall t \in \mathcal{T_B}, m \in \mathcal{A_L}(t) \tag{3.18}$$

$$\sum_{f \in \mathcal{F}} a_{mf}x_f^i \geq b_t - (1 - z_{it}), i = 1, \ldots, n \quad \forall t \in \mathcal{T_B}, m \in \mathcal{A_R}(t) \tag{3.19}$$

Constraints (3.10) and (3.11) are designed to calculate the count of data points with label k and the overall quantity of data points in node t, respectively. The ideal label to forecast is the one with the highest frequency among all data points allocated to the leaf node: $c_t = \arg, \max k \in \mathcal{K} Nkt$. Constraint (3.12) is employed to ensure a unique class prediction is made at each leaf node containing data points.

Constraint (3.13), (3.14), and 3.15) are the linearisation of the misclassificaiton loss in each leaf node:

$$L_t = N_t - \max_{k \in \mathcal{K}} N_{kt} = \min_{k \in \mathcal{K}} N_t - N_{kt} \tag{3.20}$$

Where again M can be replaced by the size of the dataset, thus they become

$$L_t \geq N_t - N_{kt} - n(1 - c_{kt}), k = 1, \ldots, K \qquad \forall t \in \mathcal{T}_{\mathcal{L}} \tag{3.21}$$

$$L_t \leq N_t - N_{kt} + nc_{kt}, k = 1, \ldots, K \qquad \forall t \in \mathcal{T}_{\mathcal{L}} \tag{3.22}$$

$$L_t \geq 0 \qquad \forall t \in \mathcal{T}_{\mathcal{L}} \tag{3.23}$$

The authors propose using a tree obtained from the Classification and Regression Trees (CART) algorithm as the initial feasible solution for the mixed-integer optimisation (MIO) problem in OCT. By using the CART tree structure and decision rules as a starting point, the MIO solver can begin its search from a reasonably good solution and potentially converge to the optimal solution faster. However, due to the large number of decision variables and heavy dependency on dataset size, the OCT is less scalable than other heuristic method like CART, and is reported to be computational expensive for large data set (more than thousands instance).

3.2 Enhancing MIP Performance

Despite the demonstrated effectiveness of the MIP approach in constructing optimal classification trees, its computational complexity poses a significant challenge, particularly when dealing with large datasets or high-dimensional feature spaces. The complexity of the MIP formulation is $O\left(2^D(N + F)\right)$, where D represents the tree depth, N denotes the number of data points, and F signifies the number of features. As the tree depth and data dimensions increase, solving the MIP problem becomes increasingly computationally demanding. To overcome these challenges, we first explore the concept of "warm start" in Section 3.2.1. Subsequently, we introduce three feature selection techniques in Section 3.2.2 to improve the efficiency of the MIP solver, facilitating faster convergence to optimal solutions. Moreover, we present the integration of K-Means Clustering as new features to assist the MIP solver in identifying optimal solution more efficiently.

3.2.1 Warm Start

In pursuit of improving the performance of the MIP formulation for constructing optimal classification trees, we introduce the concept of "warm start" to the algorithm. A "warm start" in optimisation refers to providing the solver with an initial feasible solution that can help speed up the optimisation process [31]. This is particularly useful for large-scale mixed-integer programming problems, such as the ones we encounter in decision tree optimisation. The primary goal of employing a warm start is to accelerate the MIP solver's convergence to an optimal solution by providing it with a strong feasible solution.

In our implementation, we use the Classification and Regression Trees (CART) algorithm to provide an initial feasible solution, which we then pass it to the mixed-integer programming (MIP) solver. The CART algorithm's branching rules and instance assignments are translated into a feasible solution for the MIP solver as follows:

1. Train a classification tree using the CART algorithm to the same depth and minimum samples split as the optimal decision tree.

2. Convert the CART decision tree's splitting rules into corresponding variables for the MIP problem. For each branch node, set the corresponding decision variables $a_{f,t}$ and d_t according to the feature used for splitting and whether the node is split or not.

3. Assign the predicted class labels from the CART decision tree to the leaf nodes in the MIP problem by setting the appropriate decision variables c_{tk} and l_t.

By starting with a feasible and strong solution, the MIP solver can more effectively prune the search space, as it already has an upper bound on the objective function value. Moreover, the solver can focus on refining the initial solution and exploring promising branches of the search tree, ultimately reaching the optimal solution faster than without the warm start.

In order to provide a tangible illustration, we conducted two experiments on the breast cancer dataset, which comprises 569 data points, 30 features, and two distinct labels. One experiment employed a warm start, while the other did not. To expedite the process, we set the maximum tree depth to 2 in both cases. The experiment results are shown in Figure 3.1. The MIP model without a warm start, shown on the left, failed to converge within the 1-hour time limit. In contrast, the MIP model with a warm start, displayed on the right, converged in just 1539 seconds. This experiment underscores two key observations: (i) the MIP model greatly benefits from utilizing a warm start, and (ii) the MIP solver primarily devotes its time to proving the optimality of a solution, even though a satisfactory solution can be identified within a relatively short time frame. Motivated by these findings, we explore methods that facilitate the discovery of satisfactory solutions in a more efficient manner in the following sections.

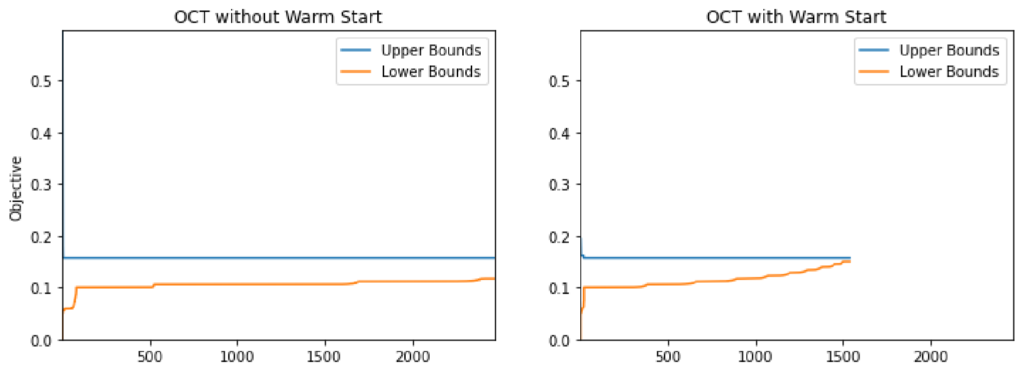

Figure 3.1: Comparison of upper and lower bounds over time for MIP-based CART model with and without Warm Start on the breast cancer dataset.

The warm start approach, although useful in many optimisation problems, does have some limitations and drawbacks that may affect its performance in certain situations. The effectiveness of the warm start approach is largely dependent on the quality of the initial solution provided. If the initial solution is far from optimal, the warm start may not significantly improve the convergence speed or solution quality. Moreover, there is a risk that the warm start approach could lead the optimisation algorithm to a suboptimal solution, especially under the strict time limit.

3.2.2 Feature Selection

In machine learning and optimisation tasks, selecting the most pertinent features from the original dataset is an essential preprocessing step. Suppose we have a dataset with 100 features, some of which are irrelevant or redundant. Without feature selection, a machine learning algorithm, like a decision tree or support vector machine, may use all 100 features to make predictions, potentially leading to overfitting, increased computational time, and reduced interpretability. By applying a feature selection approach, we can identify a subset of the most relevant features, say 20, to build a more accurate and efficient model. The reduced feature set enables the model to generalise better to new data, speeds up the training process, and facilitates a more interpretable model, as only the most important features are considered. In this section, we discuss several feature selection techniques that have been applied to our MIP-based CART model.

Feature selection techniques play a crucial role in a wide range of machine learning tasks and have been extensively explored in the literature [16, 8]. In the context of classification tasks, feature selection techniques are primarily calssified into three classes: filter methods, wrapper methods, and embedded methods [23].

Filter approaches evaluate the significance of each feature based on its statistical characteristics, including correlation or mutual information, and operate independently from the learning process [25]. Wrapper methods, in contrast, make use of the learning algorithm as an evaluation tool to assess the value of a feature subset using a search strategy [23]. There are several examples of wrapper techniques, such as forward selection, backward elimination, and recursive feature elimination (RFE). Forward selection starts with no features and incrementally adds the best feature at each stage. In contrast, backward elimination begins with all features and gradually discards the least relevant feature during each step. RFE is a more aggressive approach, which fits the learning algorithm multiple times with different subsets of features, assigning an importance score to each feature, and recursively eliminating the least important features.

Embedded techniques integrate feature selection directly into the learning algorithm, often incorporating regularisation terms in their optimisation problems [34], popular embedded methods include LASSO (Least Absolute Shrinkage and Selection Operator) [28], Ridge regression [19], and Elastic Net [34]. LASSO is a linear regression method that introduces an L1 penalty term, which results in some feature weights being exactly zero, effectively selecting a subset of pertinent features. Ridge regression, on the other hand, incorporates an L2 penalty term, which does not necessarily set feature weights to zero but shrinks them toward zero, providing some form of feature selection. Elastic Net incorporates LASSO and Ridge regression penalties, balancing the strengths of both methods.

In recent years, a significant number of research works have employed feature selection techniques to improve the performance of various machine learning algorithms in diverse applications, such as bioinformatics [12], and image processing [32]. Feature selection has found its way into optimisation models, like mixed-integer programming (MIP), where it helps reduce the model's variables and constraints, thereby facilitating faster convergence to optimal solutions [4].

In the subsequent subsections, we will discuss the application of several feature selection techniques specifically for our MIP-based CART problem. These techniques include: (i) Random Forest Feature Importance, a wrapper-based approach that uses feature importances from the Random Forest algorithm to assess and choose the most pertinent features; (ii) Principal Component Analysis (PCA), a technique for linear dimensionality reduction that transforms high-dimensional data into a lower-dimensional subspace while keeping the fundamental structure of the data; and (iii) Incorporating Feature Selection and Regularisation within the Optimal Classification Tree, an embedded method that selects a subset of the most important features during the tree building process using regularisation techniques. By applying these feature selection methods to the MIP-based CART problem, our objective is to enhance the model's performance and efficiency.

Random Forest Feature Importance

Random Forest is an ensemble learning approach that builds numerous decision trees and aggregates their results to make predictions. To create multiple decision trees, Random Forest relies on bootstrapping, a process that involves sampling with replacement from the original dataset to create multiple new datasets. Bagging, or Boostrap Aggregating, is then employed to aggregate the predictions of multiple decision tree, resulting in a more accurate and stable model. In the context of MIP-based CART models, using Random Forest Feature Importance for feature selection can potentially improve computational efficiency without sacrificing prediction accuracy.

The accuracy of a Random Forest model can be assessed through the out-of-bag (OOB) error rate, which is computed using instances not present in the bootstrapped dataset. The OOB error rate is defined as $\frac{1}{n}\sum_{i=1}^{n} I(\hat{Y}_i \neq Y_i)$, where n denotes the number of excluded instances, $\hat{Y}_i$ represents the predicted label for instance i, and Y_i is the true label. OOB data not only serves as a tool for evacuating the model's accuracy but also aids in determining the importance of individual features.

To estimate the importance of a particular feature X_j, we examine the impact of altering its values within the OOB data on the model's predictive accuracy. We achieve this by randomly permuting the values of X_j among the OOB samples. If X_j plays a crucial role in the model, the accuracy should deteriorate as a result of these permutations. By calculating the decline in prediction accuracy due to these random permutations and averaging the decrease over all trees, we can obtain a measure of the importance of feature X_j. This method, called Permutation Feature Importance (PFI), offers an unbiased estimate of feature importance but may require a larger sample size to produce stable results compared to the alternative method Mean Decrease Impurity (MDI) that will introduce next. The PFI is defined by the following formulation:

$$PFI(x_f) = \frac{1}{T}\sum_{t=1}^{T}(acc(t) - acc_{perm(f)}(t)), \tag{3.24}$$

where $acc(t)$ is the accuracy of tree t on the out-of-bag (OOB) samples, $acc_{perm(f)}(t)$ is the accuracy of tree t on the OOB samples with the values of feature x_f permuted, and T is the total number of trees.

Another measure of feature importance is Mean Decrease Impurity (MDI), it is based on the impurity reduction achieved by each feature when used for splitting. The importance of a feature x_f is computed as the sum of the impurity reduction, weighted by the number of samples that reach the node, across all trees in the forest:

$$MDI(x_f) = \sum_{t=1}^{T} \sum_{i \in I_f^t} w_i^t \Delta impurity(i), \tag{3.25}$$

where T is the total number of trees, I_f^t is the set of nodes in tree t that split on feature x_f, w_i^t is the fraction of samples that reach node i in tree t, and $\Delta impurity(i)$ is the reduction in impurity achieved by the split at node i.

Both PFI and MDI provide an importance score for each feature, which can be used for feature selection in conjunction with the MIP-based CART model. By retaining only the most important features, we can potentially improve the computational efficiency of the MIP-based CART model without sacrificing prediction accuracy. The algorithm 1 is employed in finding the optimal features for the MIP model.

Algorithm 1 Finding Optimal Features using Random Forest

```
1: Split the data into 80% training and 20% testing sets
2: Train a Random Forest classifier using the training set
3: Calculate the feature importance for each feature in feature list based on $Gini\_importance$
4: Sort the features by feature importance and remove a percentage $p$ (10 in default) of the least important features
5: Train a new Random Forest classifier on the training set with reduced features and evaluate its accuracy on the test set
6: Repeat step 2 to 5 until there are no features left
7: return The best feature subset based on the test accuracy
```

Random Forest Feature Importance offers a valuable visualisation for understanding the significance of features in predicting the target variable, as illustrated in the left-hand side of Figure 3.2. However, this method has its limitations, demonstrating in the right-hand side of Figure 3.2, where the permutation importance plot indicates that none of the features are significant. This misleading result arises due to the collinearity present among the features in the dataset. Therefore, it is important to be familiar with the dataset when interpreting the results of Random Forest Feature Importance, especially when collinearity is present within the feature set.

To address the issue of collinearity, we propose Principal Component Analysis (PCA) in the next section, which transforms the original features into a set of linearly uncorrelated components.

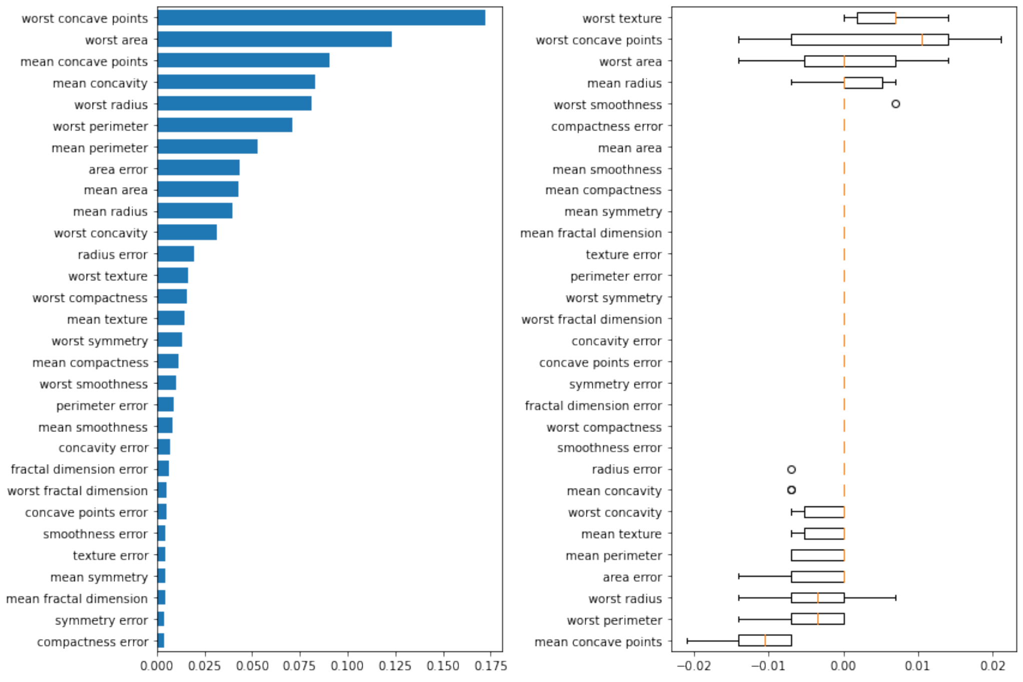

Figure 3.2: Feature importance on the breast cancer dataset based on Mean Decrease Impurity and Permutation Feature Importance.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a prominent linear dimensionality reduction method, which aims to transform high-dimensional datasets into a lower-dimensional representation while preserving the inherent structure of the data as much as possible. We call PCA a linear dimensionality reduction technique because it project data points onto a lower-dimensional space, where the relationships between the data points are linearly related to their relationships in the original higher-dimensional space. Moreover, PCA can handle multicollinear features because the projections are linearly independent and uncorrelated. PCA has been extensively applied in various research areas such as bioinformatics and spectroscopy, where the datasets used typically exhibit a large number of features ($p \gg n$). Applying PCA to our MIP-based CART problem can potentially increase the efficiency of the MIP solver by shrinking the amount of variables and constraints in the MIP formulation, thereby facilitating faster convergence to optimal solutions.

In PCA, the amount of variability in the data and the linear relationships between the features are quantified by the covariance matrix Σ. Let $\mathbf{X}$ be a $n \times p$ data matrix, where n is the number of data points and p is the number of features. The $\mathbf{X}_i$ are i.i.d. $\sim (0, \Sigma)$ ($\mathbf{X}$ is centered in order to obtain a zero mean PC). Projecting high-dimensional data onto a lower-dimensional subspace involves finding the linear combinations

$$\mathbf{Y} = a_1 X_1 + \cdots + a_p X_p = \mathbf{a}^T \mathbf{X}, \tag{3.26}$$

where $\mathbf{X} = (X_1, \ldots, X_p)^T$ is the feature vector of the dataset, such that the variance of the projection is maximised. The problem can be represented by an optimisation problem:

$$\max \quad \mathrm{Var}(\mathbf{a}^T \mathbf{X}) \tag{3.27}$$

$$\text{s.t} \quad \sum_{j=1}^{p} a_j^2 = 1 \tag{3.28}$$

The problem is typically solved by eigendecomposition. As $\mathrm{Var}(\mathbf{a}^T\mathbf{X}) = \mathbf{a}^T \boldsymbol{\Sigma} \mathbf{a} = \mathbf{a}^T \boldsymbol{\Gamma} \boldsymbol{\Lambda} \boldsymbol{\Gamma}^T \mathbf{a}$, where $\boldsymbol{\Sigma}$ is the covariance matrix, $\boldsymbol{\Gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_p)$ is the matrix of eigenvectors, and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$ is a diagonal matrix containing the eigenvalues, it can be proved that $\mathbf{a}$ that maximises the variance of the projection is equal to the eigenvector with largest eigenvalue. Therefore, $\mathbf{Y} = \mathbf{a}^T \mathbf{X} = \boldsymbol{\Gamma}^T \mathbf{X}$ is the PC transformation we are looking for, and $Y_1, Y_2, \ldots, Y_p$ are called the first, second, …, p-th principal component of $\mathbf{X}$. The eigenvalue decomposition of the covariance matrix yields orthogonal eigenvectors, ensuring that the PCs are linearly independent and uncorrelated,which are basic assumptions for many machine learning applications. The linear independence between PCs can be proved by:

$$\begin{aligned}
\mathrm{Cov}(\gamma_k^T\mathbf{X}, \gamma_j^T\mathbf{X}) &= \gamma_k^T\mathbf{\Sigma}\gamma_j \\
&= \boldsymbol{\gamma}_k^T\mathbf{\Gamma}\mathbf{\Lambda}\mathbf{\Gamma}^T\gamma_j \\
&= \begin{pmatrix} 0 & \cdots & \overset{\rightarrow k-th}{1} & \cdots & 0 \end{pmatrix} \mathbf{\Lambda} \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \leftarrow j - th \\
&= 0
\end{aligned}$$

The eigenvectors and eigenvalues of the covariance matrix provide valuable information about the underlying structure of the data. The eigenvectors represent the directions of maximum variance of the data, while the eigenvalues indicate the magnitude of the variance along these directions.

Another important property of constructing $\mathbf{Y}$ using eigenvalue decomposition is that the variability explained by the PCs are in descending order: $Var(Y_1) \geq Var(Y_2) \geq \cdots \geq Var(Y_p) \geq 0$. To determine the optimal number of PCs to select, we can use the following equation as a measure of how well the first q PCs explain the variation in the data:

$$\psi_q = \frac{\sum_{j=1}^{q} \lambda_j}{\sum_{j=1}^{p} \lambda_j} \tag{3.29}$$

In practice, a scree plot is often employed to visualise the contribution of each PC to the total variance explained. For illustration purposes, a scree plot resulting from PCA applied to the breast cancer dataset is presented in Figure 3.3. The blue curve in the figure indicates the proportion of variance explained by each individual principal component, while the orange curve depicts the cumulative explained variance. In this particular example, retaining the first six principal components would be a reasonable choice, as they account for approximately 90% of the total variance cumulatively. With respect to feature selection, we set a threshold of 95% for the cumulative proportion of variance explained to determine the number of PCs to keep as features for training our MIP-based CART model, the algorithm is described in Algorithm 2. Note that it is essential to standardise the data matrix X. Standardisation ensures that all features have equal importance and influence on the PCA, as PCA is affected by the scale of the features. Without standardisation, features with larger magnitudes could dominate the principal components, leading to biased results that do not accurately reflect the relationships among the original features.

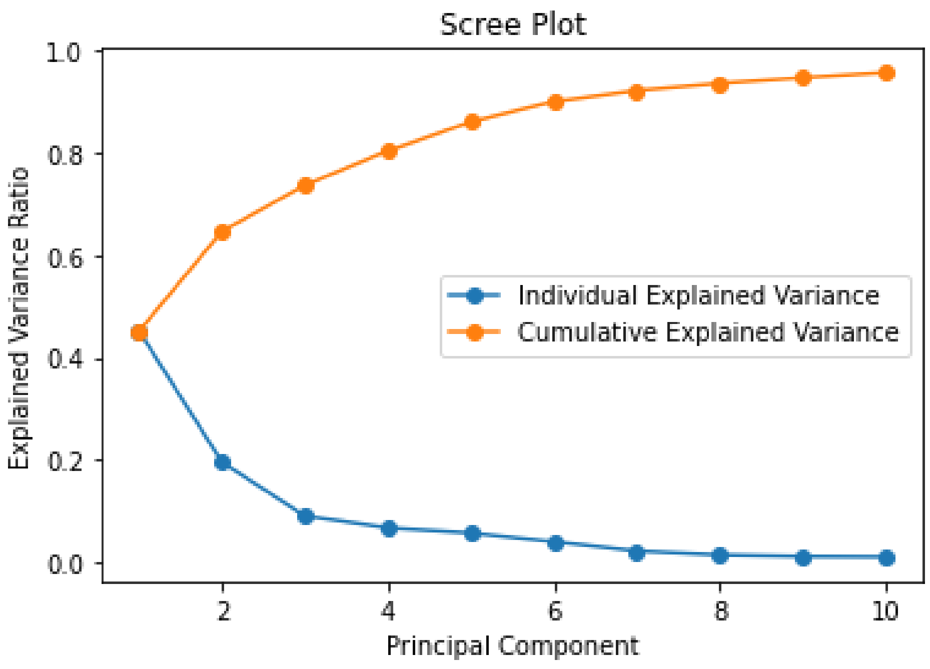

Figure 3.3: The scree plot on the breast cancer dataset.

Applying PCA as a data preprocessing method prior to inputting data into the MIP model can introduce certain disadvantages. PCA transforms the initial features into a novel set of orthogonal attributes, which are linear combinations derived from the original features. As a result, the optimal classification tree built on these transformed features may present challenges in terms of interpretability. Moreover, applying PCA for feature selection may not always be effective, as the separation between groups might not align with high-variance principal components, potentially leading to the loss of valuable discriminative information when low-variance components are omitted [22]. One potential solution is discussed in the

Algorithm 2 Principal Component Analysis (PCA)

```
Require: Standardised data matrix $X$
1: Calculate the covariance matrix $\Sigma = \frac{1}{n-1} X^T X$
2: Compute the eigenvalues $\lambda_i$ and eigenvectors $v_i$ of $\Sigma$
3: Sort eigenvectors $\gamma_i$ in descending order of their corresponding eigenvalues $\lambda_i$
4: Choose the top $k$ eigenvectors $\gamma_1, \dots, \gamma_k$ that meets the total variation explained
   threshold, where $k$ is the desired dimensionality
5: Form the projection matrix $\Gamma$ with columns $\gamma_1, \dots, \gamma_k$
6: Project the original data onto the subspace formed by the chosen eigenvectors:
   $Y = \Gamma^T X$
7: return $Y$
```

next section.

Incorporating Feature Selection and Regularisation within the Optimal Classification Tree

An alternative approach to feature selection is to incorporate it directly within the objective of the MIP formulation 3.1. By introducing a binary variable r_f for each feature $f \in \mathcal{F}$, where $r_f = 1$ if the feature f is selected and 0 otherwise. We also introduce a regularisation parameter, denoted as β, to manage the overall sparsity of the model, encouraging the algorithm to build a tree with fewer features, leading to a simpler and more interpretable model. Larger values of β will place more emphasis on reducing the number of selected features, whereas smaller values will allow the tree to use more features if it leads to better performance. We enable the MIP-based CART algorithm to perform feature selection and regularisation simultaneously. The modification to the MIP formulation is simple, which is one of the advantage of formulating decision tree using MIP technique. The objective function now becomes

$$\min \quad \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_{\mathcal{L}}} L_t + \alpha \sum_{t \in \mathcal{T}_{\mathcal{B}}} d_t + \beta \sum_{f \in \mathcal{F}} r_f. \tag{3.30}$$

And additional constraints need to added:

$$a_{ft} \leq r_f \qquad \forall f \in \mathcal{F}, t \in \mathcal{T}_{\mathcal{B}} \tag{3.31}$$

$$r_f \in 0, 1, f = 1, \ldots, p \tag{3.32}$$

The constraints ensure that the branching on feature f can only happen if the feature is selected. This modification encourages the selection of a subset of the most relevant features, which improves the model's generalisation ability, interpretability, efficiency, and robustness.

Compared to PCA, this approach has the advantage of selecting relevant features during the tree construction process, leading to more interpretable models, as the original features are preserved. Unlike PCA, this method can handle linear and non-linear relationships between features. However, tuning the regularisation parameter β can be more challenging than selecting the number of principal components in PCA.

In our experiment, considering the time-consuming nature of exploring various β values in our MIP model, we opted for selecting best β from 0, 0.1, and 0.01 based on the test accuracy.

3.2.3 K-Means Clustering

In addition to employing data preprocessing techniques to speed up computation in the MIP model, we propose a novel approach involving the integration of K-Means clustering features into the original dataset. This method aims to group similar data points into distinct clusters, thereby facilitating the MIP model's ability to arrive at solutions more efficiently and maintain the complexity of the model within manageable bounds. Once the clustering is performed, the cluster membership can be treated as an additional feature in the MIP model, enabling the model to leverage the information gained from clustering.

Incorporate K-Means clustering in Operational Research has been applied to various problems, such as network restoration and elective surgery planning, Laporte et al. employed K-Means clustering in a two-phase matheuristic for water distribution network restoration, using the algorithm to identify potential Steiner vertices in their optimisation model, which aided in constructing a shorter network [24]. In the elective surgery planning problem described in [3], Anjomshoa et al. incorporated K-Means clustering to create surgery groups based on similarities in resource demands (e.g., surgery duration and hospital stay length). This approach allowed them to enhance the accuracy and efficiency of their mixed integer programming model while managing model complexity and considering downstream effects. In our context, the integration of K-Means clustering features can aid the MIP model in identifying patterns within the data, thereby improving its ability to make accurate and efficient prediction.

K-Means clustering is a partitioning method that divides a dataset into K distinct, non-overlapping clusters based on the Euclidean distance between data points and cluster centroids, the greater the distance between data points, the lower their similarity. The algorithm seeks to minimise the within-cluster variation, which is typically measured using the the sum of squared distances between data points and their corresponding cluster centroids.

Given a dataset $x_1, x_2, ..., x_n$, where $x_i \in \mathbb{R}^p$ and a pre-defined number of clusters K, the algorithm starts by randomly initializing K centroids, $c_1, c_2, ..., c_K$. The algorithm then iteratively performs two steps until convergence:

1. Allocation step: For every data point x_i, it is allocated to the cluster corresponding to the nearest centroid c_j such that:

   $$S_j = \{x_i : \|x_i - c_j\|^2 \leq \|x_i - c_k\|^2 \ \ \forall k \neq j\},$$

   where $\|\cdot\|$ denotes the Euclidean norm and S_j defines the set of data points allocated to the j-th cluster.

2. Modification step: The centroids are recalculated by computing the average of the data points belonging to each cluster:

   $$c_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i,$$

   where $|S_j|$ denoting the quantity of data points in cluster j.

The algorithm converges when the positions of the centroids exhibit minimal changes between consecutive iterations or when a predetermined termination condition is satisfied. Note that K-Means is sensitive to the initial centroid positions and can converge to local minima. Therefore, in our experiment, we ran the algorithm 10 times with different initialisations and choose the solution with the lowest within-cluster variation.

Silhouette Analysis is an important step in our K-Means clustering approach, as it helps to establish the best cluster number (K) to be used in the K-Means algorithm. By identifying the best number of clusters, we ensure that the MIP model benefits from the most informative and compact representation of the dataset, thus improving its efficiency and performance.

Silhouette Analysis assesses clustering quality by comparing the resemblance of each data point to its corresponding cluster with that of other clusters. The Silhouette Coefficient is used to quantify this similarity and ranges from -1 to 1; higher values signify improved clustering quality. The silhouette coefficient is calcualted by:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}, \tag{3.33}$$

where a_i denotes the mean distance between x_i and all other points inside its cluster, and b_i signifies the minimal mean distance between x_i and all other points in any other cluster. The algorithm 3 illustrates how to use Silhouette Analysis to find the best cluster number.

Algorithm 3 Finding Best Cluster Number using Silhouette Analysis

```
Require: Standardised dataset
1: for each K in the possible values do
2:     Perform K-Means clustering on the standardised dataset
3:     Calculate the Silhouette Coefficient $s_i$ for each data point $x_i$
4:     Compute the average Silhouette Coefficient for all data points in the dataset
5: end for
6: Choose the K with the highest average Silhouette Coefficient as the best cluster
   number
7: Perform K-Means clustering with the optimal K value
```

Chapter 4

Results & Discussion

This chapter presents the outcomes and analysis of the experiments designed to evaluate the performance of our proposed improvements to the Optimal Classification Trees (OCT) algorithm. The experiments encompass various enhancements, including OCT with Warm Start, OCT with PCA as a dimension reduction technique, OCT with Random Forest Feature Selection, OCT with K-Means as extra features, and OCT with LASSO regularisation added to the objective function. By implementing these improvements, we aim to demonstrate the potential of achieving comparable or superior model performance within a shorter time frame, which is of paramount importance in practical applications.

The experiments were conducted on high dimension datasets, such as Connectionist Bench Sonar, QSAR Biodegradation, Statlog Project German Credit, and Cylinder Bands, with the primary evaluation metric being classification accuracy. Additionally, we examine the relationship between tree depth and model performance, with a specific focus on low-depth trees where interpretability is essential.

This chapter is organised into two main sections: Experiment Setup in section 4.1 and Results and Analysis in section 4.2. The Experiment Setup section provides an overview of the methodology employed in our experiments, discussing the datasets, hyperparameters, and evaluation techniques used. The Results and Analysis section dives into the performance of our proposed improvements, comparing them to both the original OCT and CART algorithms. We also analyze the effectiveness of individual techniques and their applicability under different problem scenarios.

Our findings underscore the potential of our proposed enhancements in addressing the challenges of time constraints and interpretability while maintaining or improving model performance. The insights gained from this analysis will serve as a foundation for future research in the development and application of improved MIP-based decision tree algorithms.

4.1 Experiment Setup

The experiments were conducted on four datasets, including Connectionist Bench Sonar, QSAR Biodegradation, Statlog Project German Credit, and Cylinder Bands, to assess the performance of the proposed improvements for Optimal Classification Trees (OCT) developed by Bertsimas and Dunn [4]. Table 4.1 summarises the characteristics of these datasets. We investigated tree depths ranging from 2 to 5 and evaluated the classification accuracy of each method. The methods employed in the experiments encompass the standard CART algorithm, OCT, and our proposed enhancements: OCT with Warm Start, OCT with PCA for dimensionality reduction, OCT with random forest feature importance for data preprocessing, OCT with K-Means for additional features, and OCT with LASSO incorporated into the objective function during the MIP problem-solving phase.

The experiments were executed using Python 3.9.7 on a computer equipped with a 1.4 GHz Intel Core i5 CPU and 8 GB of memory. To ensure consistent class distribution across all dataset splits and minimise potential biases, stratified sampling was utilised to divide each dataset into training (50%), validation (25%), and test sets (25%). This approach maintains the overall class balance and prevents under-representation or complete omission of classes in the training, validation, or test sets.

The Classification and Regression Trees (CART) algorithm, implemented through the scikit-learn library with default hyperparameters, served as a baseline for comparison.

OCT models and the proposed improved OCT models were trained for each dataset, with tree depths varying from 2 to 5. The hyperparameters α and β were selected from the set $0, 0.1, 0.01$ to optimise performance on the validation set. To account for potential biases resulting from a single train-validation-test dataset split, each experiment was conducted in triplicate, employing three distinct seeds for each dataset split. The tuned hyperparameters were used to construct the final models, and the corresponding average out-of-sample accuracies were reported to ensure the robustness of the results.

A critical aspect of applying MIP formulations to real-world problems is the time constraint associated with model training. While Bertsimas and Dunn allowed for extensive time limits in their work to find the optimal solution, practical applications often prioritise obtaining satisfactory results within shorter time frames. Our proposed improvements aim to demonstrate that similar results can be achieved even with limited time allocated for model training. Experiments were conducted with time constraint of 5 minutes to evaluate the performance of the improved OCT approach under different time-sensitive condition. These time limit facilitates the assessment of the trade-off between model accuracy and computational efficiency, providing a deeper understanding of the practical implications of the proposed techniques. The implementations can be found at `https://github.com/mingyu19950413/MIP-based-CART`.

Dataset	N	F	K
Connectionist_Bench_Sonar	208	60	2
Qsar_biodegradation	1055	41	2
Statlog_Project_German_Credit	1000	61	2
cylinder_bands	277	494	2

Table 4.1: Dataset characteristics.

4.2 Results and Analysis

Table 4.2 presents the mean out-of-sample prediction accuracy on the test set for various models. In the majority of cases, our proposed improvements demonstrate comparable or superior performance to the original OCT model. Table 4.3 displays the percentage increase or decrease in the mean out-of-sample prediction accuracy on the test set compared to CART. Generally, a significant improvement in mean accuracy is observed for low-depth trees (depth 2).

The OCT with Warm Start (OCT WS) method produces promising results, often achieving comparable or better accuracy than the original OCT. As expected, providing a strong feasible solution as a starting point leads to faster convergence while maintaining solution quality. The true strength of incorporating a warm start should become more evident when the time constraint is relaxed, making it a valuable improvement for practical applications.

For the OCT with PCA (OCT PCA) method, significant accuracy improvements are observed in specific cases, such as the Connectionist_bench_sonar dataset with tree depths of 2, 3, and 4. PCA performs well with datasets containing a large number of features, indicating that it can effectively reduce the dimensionality of the feature space without compromising the model's performance. However, OCT with PCA is not consistently robust across all datasets, as seen in the QSAR Biodegradation dataset. This limitation may arise due to PCA being a linear dimensionality reduction technique, which fails to capture nonlinear relationships within the dataset. As such, a thorough understanding of the dataset is recommended before applying PCA as a data preprocessing technique.

The incorporation of Random Forest Feature Selection (OCT RFFS) yields mixed results, outperforming the original OCT in certain cases but underperforming in others. This outcome suggests that the effectiveness of this technique may be problem-specific and warrants further investigation.

The addition of K-Means as extra features (OCT KM) demonstrates notable improvements in specific instances, such as Statlog_Project_German_Credit and cylinder_bands with tree depth 4, where the accuracy surpasses that of other OCT variants. This finding implies that K-Means clustering can serve as a valuable feature engineering technique for enhancing OCT performance.

The OCT model with LASSO added to the objective function (OCT LASSO) achieves the highest accuracy in several cases and generally performs well with high-depth trees. Given that the regularisation hyperparameter β is not fine-tuned, this result highlights the potential of combining LASSO regularisation with OCT for improved performance.

Dataset	Depth	OCT	OCT WS	OCT PCA	OCT RFFS	OCT KM	OCT LS
Connect_sonar	2	0.6859	0.6859	**0.7692**	0.6346	0.6859	0.7051
Connect_sonar	3	0.6667	0.6667	**0.7756**	0.6346	0.6538	0.6795
Connect_sonar	4	0.6538	0.6923	**0.7692**	0.6218	0.6154	0.7244
Connect_sonar	5	0.7115	**0.7179**	0.7115	0.6218	0.6795	0.7115
Qsar_bio	2	0.7424	**0.7576**	0.7374	0.7197	0.7386	**0.7576**
Qsar_bio	3	**0.7601**	0.7576	0.7109	0.7374	0.7260	0.7487
Qsa_bior	4	**0.7860**	**0.7860**	0.7247	0.6919	0.7083	0.7399
Qsar_bio	5	**0.7109**	**0.7109**	0.6705	0.7008	0.6591	0.6894
German_credit	2	0.6947	0.7027	0.7000	0.7000	**0.7253**	0.7040
German_credit	3	0.6987	0.6987	**0.7000**	**0.7000**	0.6853	0.6947
German_credit	4	0.7040	0.7053	0.6880	0.7000	**0.7120**	0.7000
German_credit	5	**0.7000**	**0.7000**	0.6973	**0.7000**	**0.7000**	**0.7000**
cylinder_bands	2	0.6429	0.6429	**0.6619**	0.6429	0.6286	0.6429
cylinder_bands	3	**0.6571**	0.6524	0.6476	0.6429	**0.6571**	0.6333
cylinder_bands	4	0.6381	0.6643	0.6238	0.6429	0.7071	**0.7571**
cylinder_bands	5	**0.6476**	**0.6476**	0.6381	0.6429	0.6095	0.6429

Table 4.2: Experimental results on various datasets and tree depths.

The findings highlight that our proposed enhancements effectively achieve competitive performance within limited time constraints, which holds significant practical value. In the majority of scenarios, our proposed methods surpass CART in building depth-2 trees under strict time restrictions, a desirable outcome when interpretability is crucial. Nonetheless, the efficacy of specific techniques, such as PCA and Random Forest Feature Selection, depends on the problem at hand and requires further exploration to identify the conditions under which they are most effective. These outcomes emphasise the need to comprehend the characteristics of the underlying dataset and the interplay between features when employing various strategies to bolster OCT performance.

Dataset	Depth	OCT	OCT WS	OCT PCA	OCT RFFS	OCT KM	OCT LS
Connect_sonar	2	**0.0**	**0.0**	**12.1**	-7.5	**0.0**	**2.8**
Connect_sonar	3	-11.1	-11.1	**3.4**	-15.4	-12.8	-9.4
Connect_sonar	4	-11.3	-6.1	**4.3**	-15.7	-16.5	-1.7
Connect_sonar	5	-3.1	-2.2	-3.1	-15.3	-7.4	-3.1
Qsar	2	2.6	**4.7**	1.9	-0.5	2.1	**4.7**
Qsar	3	-4.4	-4.8	-10.6	-7.3	-8.7	-5.9
Qsar	4	-1.8	-1.8	-9.5	-13.6	-11.5	-7.6
Qsar	5	-12.3	-12.3	-17.3	-13.6	-18.7	-15.0
German_credit	2	-2.1	-0.9	-1.3	-1.3	**2.3**	-0.8
German_credit	3	-0.9	-0.9	-0.8	-0.8	-2.8	-1.5
German_credit	4	-0.3	-0.1	-2.5	-0.8	**0.8**	-0.8
German_credit	5	-1.5	-1.5	-1.9	-1.5	-1.5	-1.5
cylinder_bands	2	**6.3**	**6.3**	**9.4**	**6.3**	3.9	**6.3**
cylinder_bands	3	-0.4	-1.1	-1.8	-2.5	-0.4	-4.0
cylinder_bands	4	2.7	**6.9**	0.4	3.4	**13.8**	**21.8**
cylinder_bands	5	**0.7**	**0.7**	-0.7	**0.0**	-5.2	**0.0**

Table 4.3: Percentage improvement compared to CART on various datasets and tree depths.

Chapter 5

Conclusion

In this research project, we set out to explore and develop improvements to the Optimal Classification Trees (OCT) algorithm, with the aim of enhancing its performance and efficiency, particularly under time constraints. Through a series of experiments and analyses, we proposed several methods for improving OCT performance, such as incorporating Warm Start, PCA as a dimension reduction technique, Random Forest Feature Selection, K-Means as extra features, and LASSO regularisation. Our findings indicate that these improvements can lead to satisfactory performance within a shorter time frame, with our proposed methods maintaining or outperforming CART in most cases, especially when constructing trees with a depth of 2.

The proposed improvements not only enhance the performance of the original OCT algorithm but also demonstrate the potential for combining various techniques, such as dimensionality reduction, feature selection, and regularisation, to address real-world classification problems. By exploring and comparing different approaches, this research project provides valuable insights into the conditions under which specific techniques are most effective, contributing to a better understanding of the relationships between features and the performance of OCT-based models.

The practical implications of this research extend to various industries and scenarios where interpretable and efficient classification models are required. By improving the OCT algorithm's performance and efficiency, decision-makers and practitioners can benefit from more accurate and interpretable models, leading to betterinformed decision-making processes. Moreover, the proposed improvements can be particularly valuable in time-sensitive applications, where obtaining a satisfactory solution within a short time frame is of great importance.

Despite the promising results, this study has several limitations. First, the success of certain techniques, such as PCA and Random Forest Feature Selection, is problem-specific and may not be consistently robust across all datasets. This limitation highlights the importance of understanding the underlying dataset and its features before applying specific techniques. Second, the computational constraints imposed during the experiments may not fully capture the potential performance of the proposed improvements when the time constraint is relaxed.

Building on the findings and limitations of this study, future research could explore alternative techniques for improving OCT performance, such as leveraging other dimensionality reduction or feature selection methods. Additionally, researchers could investigate the applicability of the proposed improvements to a broader range of datasets or problem domains, as well as the effects of fine-tuning the hyperparameters of the improved OCT algorithms. Furthermore, examining the effectiveness of combining different techniques we proposed in conjunction could lead to the discovery of even more powerful and efficient OCT variants, for instance, incorporating warm start together with the dimension reduction techniques, transforming the original features into a set of linearly uncorrelated components by PCA, which can then be used in the Random Forest Feature Selection model. Lastly, developing novel methods to address the identified limitations, such as exploring nonlinear dimensionality reduction methods like kernel-based Principal Component Analysis or adaptive time constraints, could further enhance the practicality and performance of the OCT algorithm.

Bibliography

[1] S. Aghaei, M. J. Azizi, and P. Vayanos. Learning optimal and fair decision trees for non-discriminative decision-making. *CoRR*, abs/1903.10598, 2019.

[2] S. Aghaei, A. Gomez, and P. Vayanos. Learning optimal classification trees: Strong max-flow formulations, 2020.

[3] H. Anjomshoa, I. Dumitrescu, I. Lustig, and O. J. Smith. An exact approach for tactical planning and patient selection for elective surgeries. *European Journal of Operational Research*, 268(2):728–739, 2018.

[4] D. Bertsimas and J. Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[6] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[7] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen. *Classification and Regression Trees.* The Wadsworth statistics/probability series. Chapman and Hall/CRC, 1984.

[8] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers Electrical Engineering*, 40(1):16–28, 2014. 40th-year commemorative issue.

[9] I. I. Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[10] S. Dash, O. Günlük, and D. Wei. Boolean decision rules via column generation. *CoRR*, abs/1805.09901, 2018.

[11] D. Dua and C. Graff. UCI machine learning repository, 2017.

[12] R. Díaz-Uriarte and S. A. D. Andres. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3, 2006.

[13] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[14] A. Géron. Hands-on machine learning with scikit-learn and tensorflow. *Tools, and Techniques to build intelligent systems*, pages 177–189, 2017.

[15] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.

[16] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[17] O. Günlük, J. Kalagnanam, M. Li, M. Menickelly, and K. Scheinberg. Optimal decision trees for categorical data via integer programming. *Journal of Global Optimization*, 81(1):233–260, 2021.

[18] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2009.

[19] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000.

[20] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[21] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning.* Springer, 2013.

[22] I. T. Jolliffe. *Principal Component Analysis.* Springer, 2002.

[23] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273–324, 1997. Relevance.

[24] G. Laporte, M. Rancourt, J. Rodríguez-Pereira, and S. Silvestri. Optimizing access to drinking water in remote areas. application to nepal. *Computers and Operations Research*, 140, Apr. 2022.

[25] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[26] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[27] S. L. Salzberg. C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16(3):235–240, 1994.

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

[29] S. Verwer and Y. Zhang. *Learning Decision Trees with Flexible Constraints and Objectives Using Integer Optimization*, book section Chapter 8, pages 94–103. Lecture Notes in Computer Science. 2017.

[30] S. Verwer and Y. Zhang. Learning optimal classification trees using a binary linear program formulation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1625–1632, 2019.

[31] L. A. Wolsey. *Integer Programming.* Wiley-Interscience, New York, 1998.

[32] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys (CSUR)*, 35(4):399–459, 2003.

[33] H. Zhu, P. Murali, D. T. Phan, L. M. Nguyen, and J. R. Kalagnanam. A scalable mip-based method for learning optimal multivariate decision trees. *Advances in Neural Information Processing Systems*, 33:1771–1781, 2020.

[34] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.