Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics and Operational Research

Feature selection with different importance metrics on a classification tree construction algorithm

Devi Slobbe (570674)

Supervisor:	RM Badenbroek
Second assessor:	MH Akyuz
Date final version:	30th June 2023

Abstract

We replicate the work of Verwer and Zhang (2017) on an integer linear program method (DTIP) to construct optimal classification trees of different depths for different data sets with and without warm start solutions. We then look at the impact of feature selection on this algorithm based on two different importance metrics. We find that there are no big differences between the performance of feature selection based on the Gini- or Permutation importance measure in terms of accuracy. We show that only a few features are enough for the optimization algorithm be able to construct the classification trees without losing too much accuracy and we can easily identify those features. In particular, using only the first most important feature, the algorithm can already reach substantially high accuracies and the marginal gain of adding features is low. We also demonstrate that in some cases, the DTIP algorithm with feature selection reduces the solving time and for more complicated problems, feature selection can increase the classification accuracies gathered within a time limit significantly. For a medium size data set or for a large data set with a depth smaller than 4, feature selection can even construct more accurate trees than the CART algorithm where the DTIP without feature selection failed to do so.

1 Introduction

In the world of machine learning, there is a tremendously growing demand for transparent and easily interpretable models. Decision trees can mainly be used to analyze data and reveal underlying patterns that are easily interpretable by humans, which is uncommon in the field of machine learning (Gilpin et al., 2019). Furthermore, decision trees are the base of most of the common ensemble methods such as bagging, random forests and boosting. In addition to this, the use of Big Data has exploded. This leads to the desire for machine learning models that can handle big data sets, which is a property of the decision tree. Consequently, decision trees have gained popularity in the last years taking into account that the world has become data-driven.

The increasing popularity of the use of decision trees goes hand in hand with the research done on this subject and in particular on methods that find optimal decision trees on larger data sets in a small amount of time. This is also due to the fact that decision trees are applicable to all sorts of industries such as finance, health and technology. Greedy based heuristics such as Classification and Regression Trees (CART) and Iterative Dichotomiser 3 (ID3) are widely used to construct decision trees. However, these trees frequently appear to be sub-optimal (Verwer & Zhang, 2017). Verwer and Zhang (2017) claimed to be the first researchers that have come up with a method that encodes decision tree learning entirely in an integer program. They build optimal decision trees by simultaneously taking into account all split decisions. Their method is called DTIP. They compare the results of their DTIP method both with the classification and regression method from scikit-learn (CART) and with DTIP solved by CPLEX with starting trees learned from CART (DTIPs). Verwer and Zhang (2017) do mention that there existed already formulations to construct decision trees, for example by relating the problem to structured prediction with latent variables.

However, as already stated, a lot of research has been done on decision trees in the past years. For example, Menickelly, Gunluk, Kalagnanam and Scheinberg (2016) present a novel Mixed-Integer programming formulation to construct optimal decision trees of specified size.

They particularly focus on building small decision trees with a maximum depth of 3. Moreover, Bertsimas and Dunn (2017) also use a Mixed-Integer formulation to construct classification trees. Their research deviates from other papers due to the fact that their method can generate optimal classification trees with hyperplanes (multivariate splits). In this method, they use a complexity penalty in the optimization of the tree and analyze the effects of giving warm start solutions to the solver. However, both Bertsimas and Dunn (2017) and Menickelly et al. (2016) make use of a Mixed-Integer formulation in contrast to the pure integer formulation Verwer and Zhang (2017) have set up. The integer formulation of Verwer and Zhang (2017) can be preferred in terms of simplicity and flexibility. The method is more straight forward and easy to implement and therefore it is also mathematically uncomplicated to incorporate any other desires in the construction of decision trees.

Feature importance plays a big role within the decision tree problem. By reducing the noise of unimportant features, the algorithm can more efficiently go through the search space of the model. Numerous of papers explain the concept and results of feature selection. As stated by Iqbal, Rahaman and Nabil (2012), decision tree methods require large training sets to learn accurately because the algorithms recursively partition the data set that leaves very few instances in the lower levels of the tree. Also, feature selection can be helpful for data sets with many features to keep the decision tree interpretable.

Constructing decision trees while taking into account feature importance not only in the branching phase has already been done in prior research. For example, Ruggieri (2019) uses a complete search algorithm enumerating all distinct decision trees with feature subsets built by a greedy top-down decision tree induction algorithm, minimizing classification error. Furthermore, Zhou, Zhang, Zhou, Guo and Ma (2021) propose an algorithm with a preprocessing step to pre-filter some redundant or irrelevant features before decision tree construction to improve the classification accuracy and accelerate the model construction procedure. They thus split up the process in a filtering and constructing phase. None of these papers use an integer formulation containing a pre-process selecting features based on an importance measure, which can lead to less computation times and the ability to handle larger data sets. The trade-off between accuracy and complexity of the tree can be more balanced in this way. The most informative features will reduce the risk of overfitting which further improves the model's generalization ability. Particularly, Verwer and Zhang (2017) show that for one of the data sets in their research, their method attains no more than a 0.12 classification accuracy after a 30 minute time limit. Incorporating feature selection in their formulation might be a method to improve the computability of this analysis and ensure the model can run to optimality.

Prior studies show the differences between different importance measures with regards to decision trees and ensemble methods. The most widely used variable importance measures are the impurity importance and the Permutation importance. The Permutation variable importance is defined as the decrease in the classification accuracy when a single feature value is randomly shuffled. For the impurity importance, a split with a large decrease of impurity is considered important and as a consequence variables used for splitting at important splits are also considered important. For classification, the impurity is usually measured by the Gini impurity, which ranks the importance of the feature by the contributing value of a split in the tree (Breiman, 2001).

As investigated by Nembrini, Knig and Wright (2018), the Gini impurity measure is sometimes biased, favoring splits on variables with more categories or distinct numerical values. They also mention that Permutation variable importance measures are sensitive to the correlation structure of the predictor variables. In addition to this, the Permutation importance metric can give high importance to features that may not be predictive on unseen data when the model is overfitting. Permutation-based feature importance, on the other hand, avoids this issue, since it can be computed on unseen data (Pedregosa et al., 2011). As stated by Pedregosa et al. (2011), different metrics might lead to significantly different feature importance values. This holds in particular for classification problems on imbalanced data sets. The optimal importance measure thus depends on the data set and characteristics of the features, which shows the importance of examining multiple measures.

In this paper, we firstly replicate the work of Verwer and Zhang (2017). We go into depth on the data and methods used to retrieve the results and compare our findings with the ones of Verwer and Zhang (2017). Thereafter, we give an answer to the following research question: *Does incorporating feature importance selection in the integer formulation of Verwer and Zhang lead to more accurate and computationally efficient decision trees?* To gather an answer to this question, we make use of the following sub-questions:

- What are the different importance measures we can use for determining importance?
- Which importance measure results in the best performance?
- How many important features should be incorporated in the decision tree?
- Does feature selection based on the best importance measure result in a more accurate and computationally efficient classification decision tree?

In our analysis, we first gather potential importance measures from relevant literature. Thereafter, we use the CART algorithm to calculate the feature importance scores for both metrics. We then select a number of features based on both measures. We look at the difference in performance of both metrics and at the overall effect of feature selection on the DTIP. Lastly, we compare the accuracies of DTIP with feature selection to the CART method. For this research, we use the "Diabetes", "Bank" and "Iris" classification data sets which are also used by Verwer and Zhang (2017).

In Section 2, we give more information on the data sets used in our research and the associated data transformations. Thereafter, in Section 3, we explain the methodology for replicating the work of Verwer and Zhang (2017) on the tree construction algorithm and how we incorporated feature selection into their method. Lastly, in Section 4 we present our results and in Section 5 we provide a concise conclusion to our research.

2 Data

In this section, we explain which data sets we use and where they are gathered. We also go into depth on the transformation of the data and the distribution of the data rows across different target variables.

As already stated above, we use the “Iris”, “Diabetes” and “Bank” data sets. These data sets are all gathered from the UCI Machine Learning Repository (Dua & Graff, 2017). The “Iris” data set has 4 features and 150 samples with 3 target variables which all refer to a type of iris plant. The “Diabetes” data set provides information on diagnostic measurements to predict the presence of diabetes. It has 8 features, 768 samples and a binary target variable. The “Bank” data set originates from a Portuguese banking institution that recorded direct phone call marketing campaigns. The set exists of 4521 samples on 17 categorical and numerical features and has a binary target variable. We use the same data sets when analyzing the impact of feature importance.

To use the data sets to construct classification trees, we must ensure all data points are numerical. We decide to use binary encoding to transform the categorical features of the “Bank” data set into binary features. This is particularly done to avoid the feature to have a numeric order. Within a large data set, it might also contribute to the computational efficiency of the algorithms. This results in an extended numerical data set with 49 features, opposed to the 51 features Verwer and Zhang (2017) address.

Just like explained in Verwer and Zhang (2017), we perform a non-linear transformation on the data. We assign every unique value of every feature to a unique integer, only maintaining the ordering of these values. The most frequently occurring feature value is mapped to 0.

The steps taken to perform this transformation are stated below and are executed independently per feature:

1 Sorting all of the feature values

2 Mapping all of the unique feature values to integers while combining successive values to the same integer if they have the same target

3 Subtracting the most frequently occurring feature value from the mapped integer values

As stated in Verwer and Zhang (2017), this transformation has multiple purposes. Firstly, the thresholds in decision nodes can now be represented by integer values which allows MIP solvers to branch on these values. Additionally, mapping the most frequent occurring feature value to 0 reduces the number of non-zero coefficients in the linear constraints. Lastly, the size of big M values used in the constructing algorithm is reduced because the ranges of feature values are centered around 0. All of these reasons result in an increased computational efficiency for the MIP solver.

However, some mistakes were found in the example of the data transformation done by Verwer and Zhang (2017). For example, the Sepwid values of 3.5 and 3.6 are mapped to different integer values while they are successive and share the same target value. In addition to this, the mapping of feature values to 0 is only correct for the SepWid column. For example, for the PetWid feature values, 0.2 is most frequently present in the data set. This value should be mapped to 0 and in their example, it is mapped to -3.0.

We performed the data transformation on all data sets. The feature values of the first few rows from the Iris data before and after our data transformation are shown in Table 1

When looking at the distribution of the data rows across different target variables for the

SepLen	SepWid	PetLen	PetWid	SepLen	SepWid	PetLen	PetWid
5.1	3.5	1.4	0.2	1	5	0	0
4.9	3.0	1.4	0.2	-1	0	0	0
4.7	3.2	1.3	0.2	-1	2	0	0
4.6	3.1	1.5	0.2	-1	1	0	0
5.0	3.6	1.4	0.2	0	5	0	0

Table 1: First few rows of Iris data set, original (left) and transformed (right).

different data sets, shown in Figure 1, we see that the "Iris" data set is perfectly in balance. In contrast to this, the "Diabetes" and "Bank" data sets are highly imbalanced. This once again displays the importance of analyzing different importance metrics for feature selection, as mentioned in Section 1.

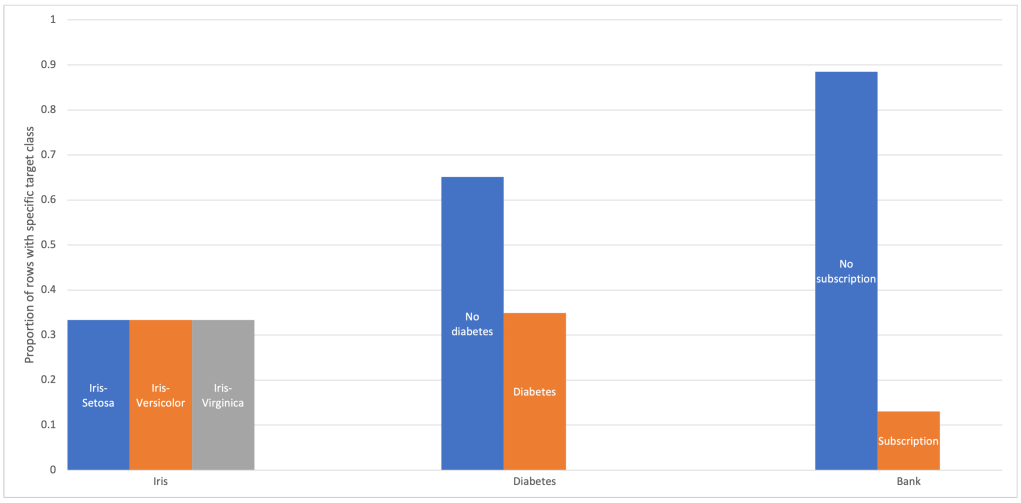

Figure 1: Distribution of the data rows across different target variables for the different data sets.

3 Methodology

In this section, we firstly elaborate on the construction algorithm used by Verwer and Zhang (2017) and the mistakes we captured. We then go into depth on the incorporation of feature selection within this model.

3.1 Notation

We use the DTIP model stated in the paper of Verwer and Zhang (2017) to construct classification trees for the transformed data sets "Iris", "Diabetes" and "Bank". In Table 2 and 3 the notation of sets, constants and variables we use in the formulation are shown.

Symbol	Type	Definition
R	set	set of rows in data file
N	set	set of nodes in tree, excluding leaf nodes
L	set	set of leaves in tree
F	set	set of unique features in data
D	set	set of depths in tree
T	set	set of unique target values in data
P	set	set of pairs of leafs with same parent

Table 2: Summary of notation of sets used in formulation.

Symbol	Type	Definition
$d(j)$	constant	depth of node $j \in N \cup L$ of tree; the root node has depth 0
$v_{r,i}$	constant	feature value for data row $r \in R$ and feature $i \in F$
$t(r)$	constant	target value of data row $r \in R$
LF, UF	constant	minimum, maximum feature value over all features
$f_{i,j}$	variable	binary decision variable, feature $i \in F$ is used in decision rule of node $j \in N$
c_j	variable	integer decision variable, threshold of decision rule of node $j \in N$
$d_{h,r}$	variable	binary decision variable, path of data row $r \in R$ goes right (0) or left (1) at depth $h \in D$
$p_{l,t}$	variable	binary classifier prediction of leaf $l \in L$ and target $t \in T$
e_r	variable	continuous prediction error for data row $r \in R$

Table 3: Summary of notation of constants and variables used in formulation.

In addition to this, a variable called $dlr(h, j, r)$ is used. This is a variable that encodes the path from the root to internal or leaf node $j \in N \cup L$. It equals 1 if a certain path (going towards node $j \in N \cup L$ at height $h \in D$) is taken by row $r \in R$ since it is defined as follows:

$$dlr(h, j, r) = \begin{cases} d_{h,r} & \text{if the path to node } j \text{ goes } \textbf{left} \text{ at depth h} \\ (1 - d_{h,r}) & \text{if the path to node } j \text{ goes } \textbf{right} \text{ at depth h.} \end{cases}$$

This is one of the aspects of great importance and value in the proposed method of Verwer and Zhang (2017). By introducing this binary variable to return the path directions, they use $|D|$ variables per observation instead of $2^{|D|}$. This increases the maximum data size the formulation can handle.

Furthermore, we use big-M formulations to encode certain paths taken by the data rows. The big-M variables used in the formulation are calculated as follows:

$$M_r = \max\{(v(r, i) - LF) | i \in F\} \tag{1}$$

$$M_r^{'} = \max\{(UF - v(r, i)) | i \in F\} \tag{2}$$

3.2 Integer formulation

In the mathematical formulation of Verwer and Zhang (2017), a few mistakes are made. We elaborate on those going forward. We show the formulation we use, explain the objective value and constraints and go into depth on the differences with the formulation of Verwer and Zhang (2017). If an equation contains any differences, it is followed by a (*).

Within this model, we minimize the classification error for all data rows. This classification error is defined in Equation (11). The objective function is defined as follows:

$$\min \sum_{r \in R} e_r \tag{3}$$

The first set of constraints ensure that every internal node uses one feature in their decision rule. For every internal node $j \in N$, the sum over all binary variables $f_{i,j}$ that tell if feature i is used in decision node j, should be equal to 1:

$$\sum_{i \in F} f_{i,j} = 1, \quad \forall j \in N \tag{4}$$

Big-M constraints (5) and (6) assure that if row r takes the left (right) branch at depth $d(j)$ of node j, denoted by $d_{d(j),r}$, and row r takes the path to node j, then the corresponding feature value has to be smaller (greater) than the threshold. However, we adapted the formulation shown in Verwer and Zhang (2017). We changed the index of the first summation to include depth 0 (root node) and exclude the depth of node j since it is about the path *to* node j and we do not consider the decision taken *at* node j here. Consequently, we had to adapt constraints (5) and (6) by respectively subtracting or adding another big-M to make it feasible again. Furthermore, we added 1 to the left side of constraints (6) to ensure that if the value is equal to the threshold, it is infeasible to take the right path. An example is shown after stating the equations:

$$\sum_{0 \leq h < d(j)} M_r dlr(h,j,r) + M_r(d_{d(j),r} - 1) + \sum_{i \in F} v_{r,i} f_{i,j} \leq M_r d(j) + c_j, \quad \forall j \in N, r \in R (*) \tag{5}$$

$$M_r^{'}(d_{d(j),r}) + M_r^{'} d(j) + \sum_{i \in F} v_{r,i} f_{i,j} \geq c_j + 1 + \sum_{0 \leq h < d(j)} M_r^{'} dlr(h,j,r), \quad \forall j \in N, r \in R (*) \tag{6}$$

The use of these constraints can be shown by an example. Suppose row r takes the left branch at node j. Since node j is reached, $dlr(h,j,r) = 1$ for all $h \in D$ ranging from 0 to $d(j)$. Variable $d_{d(j),r}$ is equal to 1 since the left branch is chosen. This reduces Equations (5) into:

$$M_r d(j) + \sum_{i \in F} v_{r,i} f_{i,j} \leq M_r d(j) + c_j. \tag{7}$$

If we substract $M_r d(j)$ on both sides, this constraint forces that the used feature value is smaller than the threshold.

The same holds for Equations (6). Suppose row r takes the right branch at node j. In this case, $dlr(h,j,r)$ should still equal 1 for all h ranging from 0 to $d(j)$. Furthermore, variable $d_{d(j),r}$ is equal to 0 since the left branch is chosen. $M_r^{'} d(j)$ can be subtracted from both sides and the following inequality remains:

$$\sum_{i \in F} v_{r,i} f_{i,j} \geq c_j + 1. \tag{8}$$

This inequality should be met for row r to feasibly take the right branch at node j.

Furthermore, the following constraints secure that every leaf node has 1 prediction type:

$$\sum_{1 \leq t \leq y} p_{l,t} = 1, \quad \forall l \in L \tag{9}$$

Constraints (10) force that if a row ends in a leaf l, the error e_r for that row is 1 if the leaf prediction type is different from the rows target. For these constraints, we also changed the indexes to make sure we incorporate the root node depth level. An example is shown to illustrate the use of the constraints:

$$\sum_{0 \leq h < k} dlr(h,l,r) + \sum_{t \neq t(r)} p_{l,t} \leq e_r + k, \quad \forall l \in L, r \in R(*) \tag{10}$$

For example, if a row ends up in leaf l, $dlr(h,l,r)$ equals 1 for every h ranging from 0 to k, since l is reached. Therefore, the first summation equals k in that case. If we subtract k from both sides of the equations, we are left with the following inequality:

$$\sum_{t \neq t(r)} p_{l,t} \leq e_r. \tag{11}$$

Since we are minimizing e_r for every row r, e_r will be set equal to $\sum_{t \neq t(r)} p_{l,t}$, which is the definition of the classification error.

To strengthen the formulation, we use constraints (12) such that the threshold values are between the minimum and maximum feature values.

$$\sum_{i \in F} LFf_{i,j} \leq c_j \leq \sum_{i \in F} UFf_{i,j}, \quad \forall j \in N \tag{12}$$

For constraint (13), $(l,l') \in P$ is defined as a pair of leaf nodes with the same parent. We use this constraint to make sure every leaf node has an unique target variable.

$$p_{l,t} + p_{l',t} \leq 1, \quad \forall (l,l') \in P, t \in T(*) \tag{13}$$

We changed the equation from an equality into an inequality. This is particularly done since one of our data sets ("Iris") has a target variable with 3 different classes. In that case, it can also be the case that both leaf l and l' do not predict a certain target class.

To evaluate the performance of our method, we compare it to the performance of the CART algorithm. We also use the CART algorithm to give the outcomes of this algorithm as warm start variables to the DTIP.

Some of the trees cannot directly be used in our formulation. Since Verwer and Zhang (2017) have not specified how they translated the CART trees to feasible solutions for the DTIP, we came up with our own method. It is possible that the CART algorithm constructs a noncomplete binary tree. Leaf nodes with the same parents also appear to have the same class label. To be able to use this tree in our DTIP, we add missing internal- and leaf nodes to create a complete tree. For the leaf nodes with the same parent and same class label, we change the parent's threshold to UF to make sure all rows go to the left leaf node and change the class label of the right leaf node to the first target class. We also set the thresholds for the added internal nodes to UF, and let the most left leaf node have the same class label as the previous leaf node. In this way, we make sure that the accuracy is not affected by adding those nodes. We also make sure that the added leaf nodes with same parents do not have the same targets. Those changes to the trees are inserted manually into the DTIP warm start variables.

3.3 Feature importance

To investigate whether feature selection improves the DTIP method, we look at two different feature importance metrics. We use the impurity importance (measured by the Gini impurity) and the Permutation importance, already mentioned in Section 1. We gather the importance measures for the different features from scikit-learn after performing the CART algorithm. According to the Pedregosa et al. (2011), the Permutation- and Gini importance are calculated in scikit-learn as stated in respectively Algorithm 1 and 2. As can be seen in Algorithm 1, the Permutation feature importance is the decrease in classification accuracy when a single feature column is randomly shuffled. In this way, the importance of the corresponding feature is high if it contributes significantly to the classification accuracy of the tree. In addition, in Algorithm 2 can be seen that a feature has a high Gini importance measure if in splits where this feature is used, a bundle of rows that go into this split node and have multiple different targets are split into groups with rows where the targets are less diverging.

Algorithm 1 Calculating the Permutation importance

```
Input: classification tree model $m$, data set $D$, number of repetitions $W$ ($default = 5$)
Compute classification accuracy $s$ of $m$ on $D$
for Each feature $i \in F$ (column of $D$) do
    for Each repetition $w$ in [1,W] do
        Randomly shuffle column $i$ of $D$ to generate a corrupted version of $D$ named $\tilde{D}_{w,i}$.
        Compute accuracy score $s_{w,i}$ of model $m$ on corrupted data $\tilde{D}_{w,i}$.
    end for
    Compute importance $imp_i$ for feature $i$ defined as:
        $$imp_i = s - \frac{1}{W}\sum_{w=1}^{W} s_{w,i}$$
end for
```

We gather both importance measures for the different features. Thereafter, we try different maximum numbers of features based on the importance values and only provide those corresponding feature columns to our DTIP program. To investigate the difference between the two metrics, we both look at the selected features and the consequences for the accuracy of the DTIP with feature selection. In addition to this, we also examine the total effect of this feature selection on accuracy and solving time.

Algorithm 2 Calculating the Gini importance

```
Input: classification tree model $m$, data set $D$
for Each feature $i \in F$ (column of data set $D$) do
    for Each node $j \in N$ where $f_{ij} = 1$ do
        Let $p_{j,t}$ be the proportion of observations with target class $t \in T$ in node $j \in N$.
        The Gini impurity in node $j \in N$ is calculated as follows:
            $Gini = \sum_{t \in T} p_{j,t}(1 - p_{j,t})$
        Calculate the Gini impurity decrease after splitting on node $j \in N$.
    end for
    Take the weighted sum of all Gini impurity decreases of nodes $j \in N$ where feature $i \in F$
    is used, with the weights being the the number of samples in each split. This is the Gini
    importance measure.
end for
Normalize all Gini importance measures to sum up to 1 over all features.
```

4 Results

In this section we present our findings. First, we elaborate on the outcomes of the CART and DTIP method with and without warm start solutions. Thereafter, we look at the results of incorporating feature selection in the DTIP method. All of the results are gathered after implementing the algorithms on a computer with a RAM of 8 GB and a processor clock speed of 1.4 GHz.

4.1 CART, DTIP and DTIPs

We perform the CART algorithm on the transformed data set using the Python module scikit learn. We measure the performance based on the classification accuracies of the tree. We saw that the CART algorithm creates mostly incomplete trees. To illustrate this we show a visualization the CART tree of depth 5 of data set "Iris". In Figure 2, we see that the left side of the tree is completely empty and there are only 2 leaf nodes of depth 5.

We also implement the DTIP model stated in Section 3 and use the regression trees gathered from the CART algorithm as warm starts for our DTIP to analyze the differences (indicated as DTIPs). To be able to use them as warm start solutions, we first created complete trees as explained in the end of Section 3.2.

For all of the methods, we use a maximum time limit of 30 minutes. We found classification accuracies for the different depths, stated in Table 4.

In Table 4, we see that for the trees constructed with the DTIP and DTIPs algorithms that run to optimality within the time limit, we find the same accuracies as Verwer and Zhang (2017) do. For the algorithms that do not run to optimality, the accuracies are also approximately equal, with a maximum difference of around 2.5 percentage points when not considering the DTIP tree for the "Bank" data set of depth 5. For this case, the accuracy we derived is around 50 percentage points higher. This can be explained by the fact that, when looking at the branch and bound tree of this algorithm, we see that only after 25 minutes the algorithm reaches a

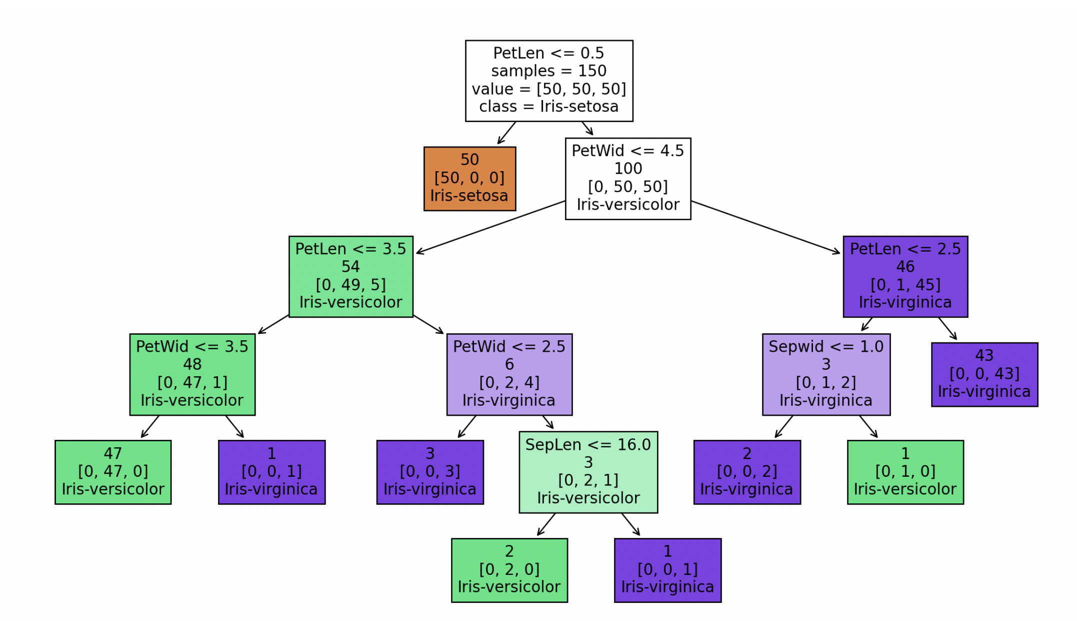

Figure 2: A CART classification tree of maximum depth 5 on the Iris data set. For internal nodes, the decision rule is shown. For both internal and leaf nodes, the number of rows that go through or end up in the corresponding nodes is shown. In addition to this, we also see the division of targets of those rows, indicated as an array of values. Lastly, we also see the most prevalent target variable. For leaf nodes, this is the classification prediction.

Data set	**Method**	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
Iris	CART	0.6667*	0.96*	0.9733	0.9933	1.0*
	DTIP	$\underline{0.6667*}$	$\underline{0.96*}$	$\underline{0.9933*}$	$\underline{1.0*}$	$\underline{1.0*}$
	DTIPs	$\underline{0.6667*}$	$\underline{0.96*}$	$\underline{0.9933*}$	$\underline{1.0*}$	$\underline{1.0*}$
Diabetes	CART	0.75*	0.75	0.7578	0.7904	0.819
	DTIP	$\underline{0.75*}$	0.7708*	0.7865*	0.7943	0.7331
	DTIPs	$\underline{0.75*}$	0.7708*	0.7760	0.7995*	0.8294*
Bank	CART	0.8848	0.9005*	0.9031	0.9109	0.9202
	DTIP	$\underline{0.8929*}$	0.8938	0.8969	0.1155	0.6158
	DTIPs	$\underline{0.8929*}$	0.9005*	0.9038*	0.9111*	0.9204*

Table 4: Classification accuracies of three regression methods with trees of depths 1–5. Underlined values show that the method has run to optimality within 30 minutes (for the DTIP and DTIPs) and values with an * indicate that the accuracy of a method is the highest for a certain depth.

significant breakthrough in which the accuracy raises from 0 to 56.4% immediately. Most likely, the algorithm of Verwer and Zhang (2017) would have reached such a branch a little after the time limit.

When looking at the CART classification accuracy, we see that especially for the "Diabetes" data set, the CART tree accuracies differ slightly from the outcomes of Verwer and Zhang (2017). However, the maximum absolute difference is only around 2.2 percentage points. This can be explained by the fact that their example of the data transformation differs from ours.

We see that with the "Iris" data, CART finds the optimal trees for depths 1, 2 and 5. For the "Diabetes" data, CART also finds the optimal tree for depth 1. Lastly, for the "Bank" data, CART finds the optimal tree for depth 2. In almost all of the other cases, the DTIP and DTIPs outperform the trees constructed with CART. Only for trees of depth 4 and 5 on the "Bank" data, the DTIP finds very low classification accuracies in the time limit of 30 minutes. Here you can see the difficulty of CPLEX in solving large instances. However, the accuracies when the DTIP starts with warm solutions (DTIPs) are always equal to or higher than the CART accuracy.

4.2 Feature importance

We analyzed the different feature importance values for the different data sets and depths. All of the values can be found in Appendix A. When looking at both the Gini- and Permutation importance values, we see that for all of the data sets there are some differences among the ordering of features based on the two different metrics. We did not see a clear relation between the target distribution displayed in Figure 1 and the difference in orders for the different data sets. For the "Iris" data set, only trees of depth 3 and 4 give different orders. The difference of those orders is shown in Table 5. Along the Y-axis of Table 5, the features of data set "Iris" are ordered based on the Gini Importance metric. Along the X-axis, they are ordered based on the Permutation Importance metric. If the order would be the same, a perfect diagonal line with black cells would appear. We see here that the order of features slightly differs across the two importance metrics.

To examine the differences for the "Diabetes" and "Bank" data sets, we look at Figures 3 and 4 respectively. Here, the rank difference is shown. The rank of a certain feature based on the Permutation importance measure is subtracted from the corresponding rank based on the Gini importance measure. For the "Diabetes" data set, we see that for all depths, the ranks are approximately the same because all of the lines are centered around zero. What stands out here is the fact that feature 6, the BMI of a person, is considered more important based on the Gini importance measure as opposed to the Permutation importance measure. This can have several reasons, such as a non-linear relationship between the feature and the target variable or interaction effects with other variables. However, this is not the case for feature 6, so we can not find a clear reason for this occurrence.

For the "Bank" data set, we also see that the ranks are very similar. Without the two outliers, the rank differences only range from -4 up onto 3. Those outliers can be explained by the fact that for many features, the importance measures equal zero. When two features have the same importance value, the features are automatically sorted based on their index.

d	**Perm.** / **Gini**	**PetLen** (0.52)	**PetWid** (0.11)	**SepLen** (0.00)	**SepWid** (0.00)
$d = 3$	**PetWid** (0.95)		■		
	PetLen (0.05)	■			
	SepLen (0.00)			■	
	SepWid (0.00)				■

d	**Perm.** / **Gini**	**PetLen** (0.57)	**PetWid** (0.13)	**SepWid** (0.01)	**SepLen** (0.00)
$d = 4$	**PetWid** (0.94)		■		
	PetLen (0.05)	■			
	SepLen (0.01)				■
	SepWid (0.00)			■	

Table 5: Features of data set "Iris" ordered based on the Gini (rows)- and Permutation (columns) importance values gathered from the CART tree of depth 3 and 4. The importance values are stated in between brackets.

Accordingly, when the feature importance value for a feature is a little above zero for one metric and zero for the other metric, it can appear as an outlier if the feature index is high. This is exactly what happens for feature 26 in the tree of depth 3 and feature 36 in the tree of depth 4.

4.3 Incorporation of feature importance in DTIP

In this section, we elaborate on the findings when incorporating feature importance into the DTIP algorithm. We first look at the difference in accuracy between the two metrics. Thereafter, we elaborate on the marginal gain in accuracy of adding features and look at the exceptional case of constructing a tree of depth 4 or 5 on the "Bank" data set. Lastly, we go into depth on the overall effect of incorporating feature importance on solving time of the DTIP.

4.3.1 Difference in accuracy between using Gini- or Permutation metric

To investigate if the difference in order of features based on the importance metrics, shown in Section 4.2, has a significant impact on the feature selection procedure, we use both metrics and compare the performances. For the "Iris" data set, we look at all cases where we select 1 up to 3 of the 4 features based on the importance measures. For the "Diabetes" data set, we also select 1 up to 7 of the 8 features. However, for the "Bank" data set, for solving purposes, we only look at the cases where we add 6 features at the time, ranging from 1 to 43 of the 49 features. All of the corresponding classification accuracies and solving times can be found in Appendix B.

As expected regarding the minor differences in feature order based on the two importance

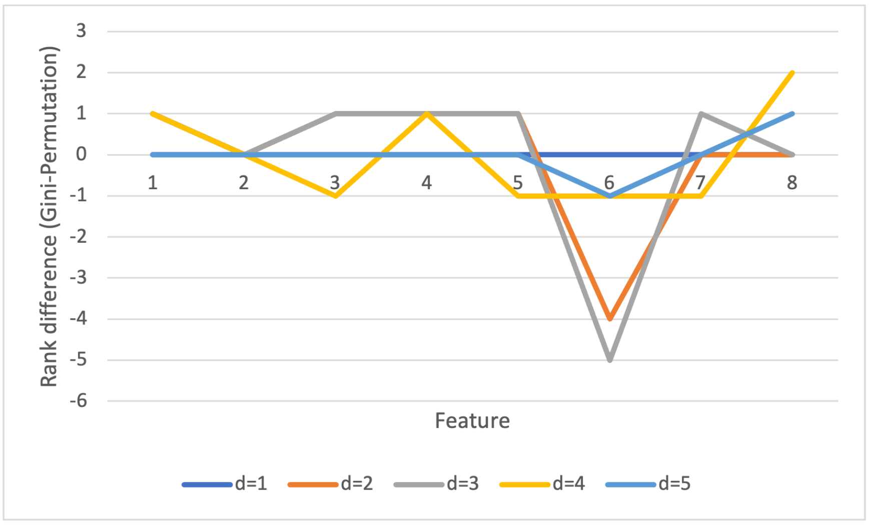

Figure 3: The order differences for the "Diabetes" data set, calculated as the order rank of a feature based on the Gini importance measure minus the order rank based on the Permutation importance measure. The lines correspond to the different depths of the tree.

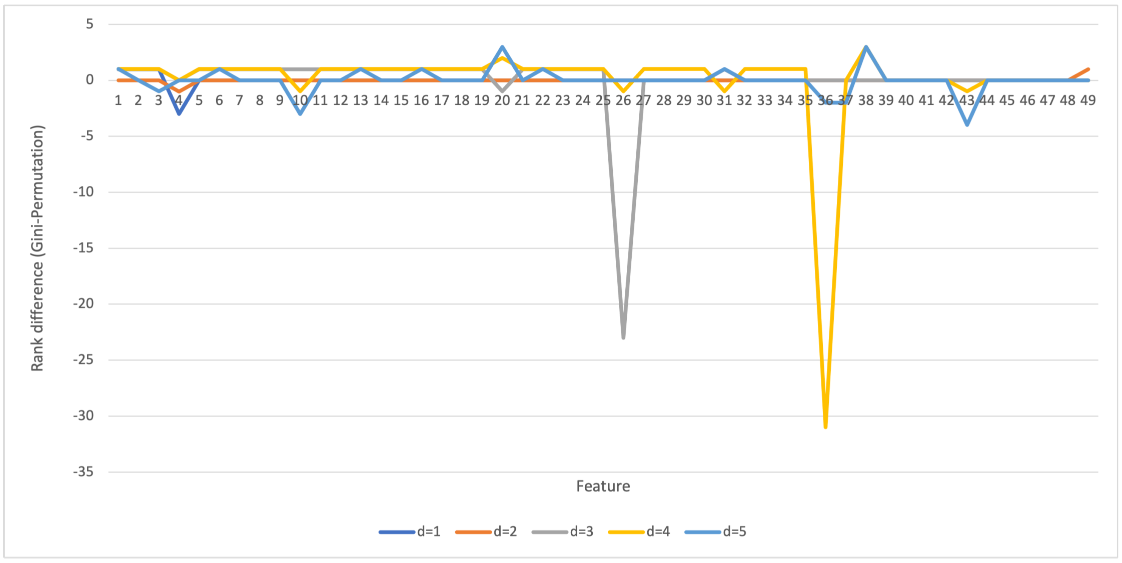

Figure 4: The order differences for the "Bank" data set, calculated as the order rank of a feature based on the Gini importance measure minus the order rank based on the Permutation importance measure. The lines correspond to the different depths of the tree.

Data set	Depth	Gini	Permutation
Iris	$d = 1$	0 pp.	0 pp.
	$d = 2$	0.67 pp.	0.67 pp.
	$d = 3$	3.33 pp.	4 pp.
	$d = 4$	4.67 pp.	4.67 pp.
	$d = 5$	4.67 pp.	4.67 pp.
Diabetes	$d = 1$	0 pp.	0 pp.
	$d = 2$	1.82 pp.	1.82 pp.
	$d = 3$	2.87 pp.	2.87 pp.
	$d = 4$	3.39 pp.	3.39 pp.
	$d = 5$	-2.73 pp.	-2.73 pp.
Bank	$d = 1$	0.17 pp.	0.79 pp.
	$d = 2$	0.06 pp.	0.09 pp.
	$d = 3$	0.2 pp.	0.2 pp.
	$d = 4$	-78.19 pp.	-78.19 pp.
	$d = 5$	-28.4 pp.	-28.44 pp.

Table 6: Difference in accuracy, measured in percentage points, between using 1 feature for the DTIP and using all features.

measures, we see very small differences in the performance measured by classification accuracy among those two measures. To analyze this difference, we look at the summary statistics of differences in accuracy percentage points between the Permutation- and Gini importance measure among all the data sets, with different depths and different numbers of features. For the “Iris” data set, only the tree of depth 3 contains differences, resulting in a mean of differences in accuracy of 0.04 percentage points. The mean of differences in accuracy for the “Diabetes” data set corresponds to 1.25 percentage points. Lastly, the mean for the “Bank” data set is 5.27 percentage points. However, this mean is biased upwards since there are three very high outliers due to the DTIP of the “Bank” data set which is remarkably slow. We will go into depth on this occurrence later in this section. The median of 0.065 percentage points is more informative here. These low values indicate that the difference in accuracy between the two importance measures is almost negligible.

4.3.2 Marginal gain of adding features

We found that in most of the situations, the marginal gain in accuracy of adding features is very low for almost all of the cases. When looking at Table 6, we see that in all of the cases, the difference in accuracy of trees constructed with the DTIP that uses only 1 feature and the DTIP without feature selection is notably small. The negative values observed for the bigger data sets and high tree depths show that with more complicated problems, giving only 1 feature to the solver can sometimes be even more accurate within the time limit due to the solution speed. We go more into depth on this occurrence in the next section. Apparently, the DTIP is able to build accurate trees for every depth with only using the first important feature.

The percentage points shown in Figure 6 are gained gradually across the addition of features. As an example, we look at the accuracies for the DTIP trees with feature selection of trees with depth 2 and 3 on the “Bank” data set shown in Figure 5. We look at this situation as this is

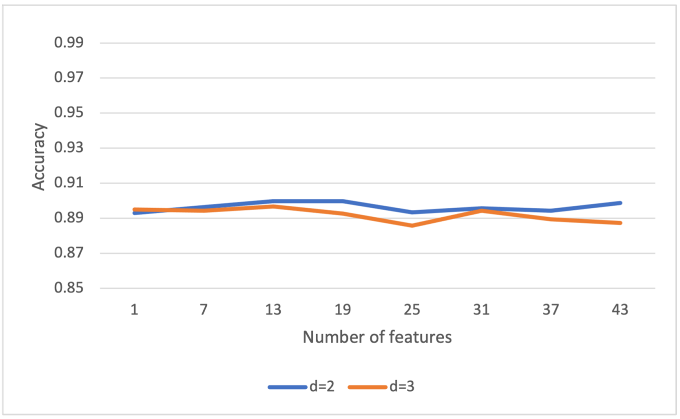

Figure 5: Classification accuracy of DTIP with feature selection on data set “Bank” with time limit of 30 minutes. Permutation importance is used to construct trees with maximum depth of 2 and 3. Only the tree with depth 2 using 1 feature has run to optimality.

representative for all of the cases. We have only displayed the results of using the Permutation importance, since the differences in accuracy of both metrics was at maximum 1 percentage point. The figure shows a horizontal line, indicating the marginal gain in accuracy of adding features is minimal.

When using the DTIP with fewer features than present in the data set, we can reach the same accuracies as the DTIP with all of the features in almost all of the cases. Those findings show us that there are a lot of unimportant feature variables which are not needed to reach the highest possible accuracy.

4.3.3 Constructing trees of depth 4 and 5 for “Bank” data set

For the construction of a tree of depth 4 for the “Bank” data set, we saw in Table 4 that the DTIP method could only get to a tree of accuracy 11.5%. As shown in Figure 6, we see that only 1 important feature is needed to build a tree of depth 4 with an accuracy of around 0.9 within the time limit. We also see that in most of the cases, there is no significant difference between the performance of the Gini importance metric and the Permutation importance metric. However, when selecting 37 or 43 important variables, there is a massive difference between the accuracy after selecting based on the Gini- and Permutation importance. The extra features added based on the order according to the Gini importance make the problem more difficult, while the features according to the Permutation importance do not show this phenomenon. For the tree of depth 5, both importance measures give an accuracy of around 90% when using only 1 feature, while the DTIP without feature selection only comes to an accuracy of 61.6% within the time limit.

4.3.4 Effect on solving time

We found that most of the times, only a small subset of features is needed to arrive at the maximum classification accuracy. We now investigate whether using only a small subset of those features is beneficial in terms of solving time. We saw that for some of the situations, the solving time of the DTIP using feature selection based on the Gini- and Permutation importance

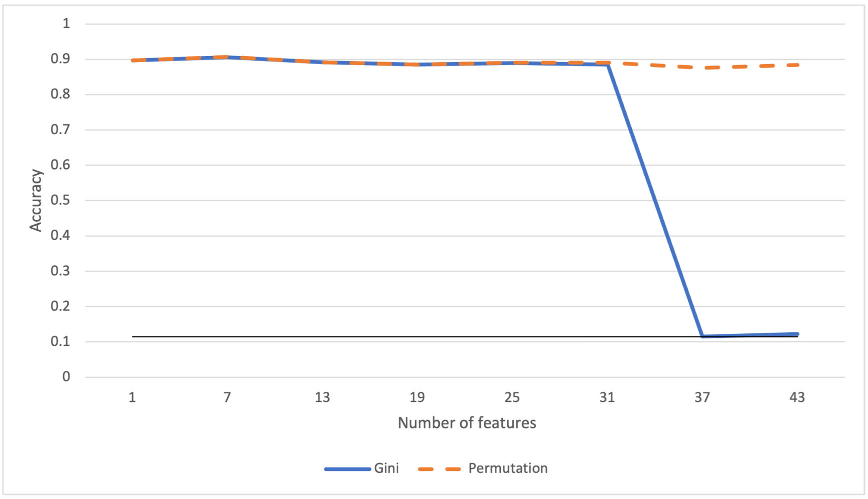

Figure 6: Classification accuracy of the tree of depth 4 constructed with the DTIP algorithm for the "Bank" data set with a time limit of 30 minutes. The number of features selected is stated on the X-axis, using the two different importance measures indicated by the color of the lines. The black line indicates the accuracy found by the DTIP method without feature selection.

metrics can diverge substantially. Nevertheless, this is different for every data set and depth of the trees so no metric stands out as a clear winner in terms of solving time.

The benefit of feature selection in terms of solving time is seen when creating a tree of depth 2 and 3 based on the "Diabetes" data set or a tree of depth 2 on the "Bank" data set. For the other cases, the solving time is either approximately the same as for the DTIP method without using feature selection or cannot be investigated due to the attainment of the 30 minute time limit.

For a tree of depth 2 based on "Diabetes" data, the DTIP method without feature selection was not able to run to optimality. Within the time limit of 30 minutes, the DTIP model is able to find a tree with accuracy 0.7708. By using the Gini importance measure to select only the first most important feature, we can construct a tree of depth 2 with an accuracy of 0.7526 in around 1 second. In Figure 7, we see the classification accuracy of the DTIP for a tree with a maximum depth of 2 with feature selection based on the Gini importance, plotted against the corresponding solving time. The data points correspond to the different number of features selected. As is shown, the accuracy of 0.7708 is already reached within 1200 seconds, using 6 important features. This shows that feature selection can decrease solving time of the DTIP while maintaining the accuracy.

For a tree of depth 3 for the "Diabetes" data, using either the Gini- or Permutation importance measure, the DTIP with only using the first important feature runs to optimality in around 50 seconds, gathering an accuracy of approximately 75.8%. The DTIP without feature selection only reaches an accuracy of around 78.7% in 30 minutes. In addition to this, the DTIP with using only the first important feature for a tree of depth 2 for the "Bank" data reaches an accuracy of around 89.3% in 0.03 or 38.76 seconds using the Permutation or Gini measure respectively. In this case, the DTIP without feature selection reaches approximately the same accuracy, but it reaches the 30 minutes time limit to get there. We can thus state that in some

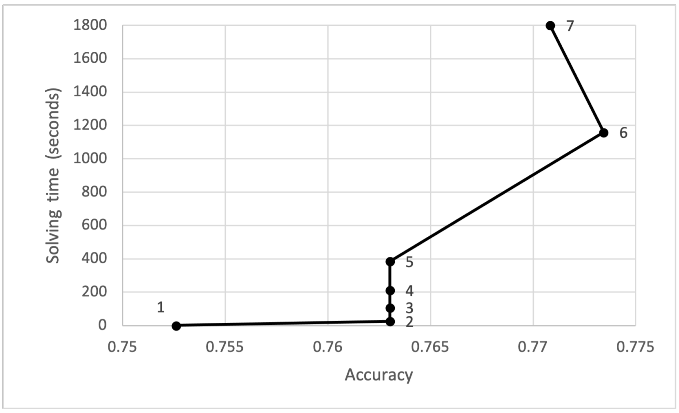

Figure 7: The accuracy and solving times of the DTIP algorithm for a tree of depth 2 with different numbers of important features based on the Gini importance measure on data set “Diabetes”.

cases, the resolution time can be extremely reduced with using feature selection without losing too much accuracy.

4.3.5 CART and DTIP with feature selection

We have now shown the effect of feature selection on the DTIP algorithm. To see if using DTIP with feature selection in the end leads to more accurate trees than CART in the time limit of 30 minutes, we look at Table 7. In this table, we compare the CART and DTIP accuracies within this time limit with the maximum accuracy reached by the DTIP with feature selection using the Gini importance metric. We show only the results based on the Gini importance metric since we have seen in Section 4.3.1 that the differences are small. In addition, since for the “Bank” data set, we saw that selecting either 1 or 7 features led to the maximum accuracy, we also considered the cases where we select 2 to 6 features. All of the results are in Appendix B.3. When looking at Table 7, we see that for the “Iris” data set, a small data set with few features, the DTIP without feature selection is already able to beat the CART algorithm. However, for the medium size data set “Diabetes” with 8 features, the DTIP with feature selection beats both the DTIP without feature selection and CART significantly. For the large data set “Bank”, we see that for depth 1, where the DTIP problem runs to optimality already, the DTIP without feature selection is optimal. For depth 2 and 3, the DTIP with feature selection improves the accuracy of the DTIP algorithm and therefore beats the CART algorithm. However, when the depth of the tree is bigger and the problem becomes more difficult, the DTIP with feature selection beats the DTIP without feature selection but it still not able to build more accurate trees than the CART algorithm in the time limit.

These conclusions are only based on the maximum values of the DTIP with feature selection over all numbers of features. It would be beneficial to know beforehand how many features are needed to create trees with maximum accuracy. When looking at the last column of Table 7, we see that for a tree of depth 1, for all data sets, only the first important feature is needed to get the maximum accuracy. For depth 2, only the first two important features are needed to get the maximum accuracy. For depth 3, the first three features are needed to get to the maximum accuracy. We thus do see a pattern. However, future research on more data sets is needed to

		CART	**DTIP**	**DTIPfs**	
				min	max
Iris	$d = 1$	**0.6667 (4)**	**0.6667 (4)**	**0.6667 (1)**	**0.6667 (1)**
	$d = 2$	**0.96 (4)**	**0.96 (4)**	0.9533 (1)	**0.96 (2)**
	$d = 3$	**0.9733 (4)**	**0.9933 (4)**	0.96 (1)	**0.9933 (3)**
	$d = 4$	0.9933 (4)	**1.0 (4)**	0.9533 (1)	**1.0 (3)**
	$d = 5$	**1.0 (4)**	**1.0 (4)**	0.9533 (1)	**1.0 (3)**
Diabetes	$d = 1$	0.75 (8)	0.75 (8)	0.75 (1)	**0.75 (1)**
	$d = 2$	0.75 (8)	0.7708 (8)	0.7526 (1)	**0.7734 (2)**
	$d = 3$	0.7578 (8)	0.7865 (8)	0.7578 (1)	**0.7878 (3)**
	$d = 4$	0.7904 (8)	0.7943 (8)	0.7604 (1)	**0.8190 (5)**
	$d = 5$	0.819 (8)	0.7331 (8)	0.75 (7)	**0.8216 (3)**
Bank	$d = 1$	0.8848 (49)	**0.8929 (49)**	0.8912 (1)	0.8912 (1)
	$d = 2$	0.9005 (49)	0.8938 (49)	0.8932 (1)	**0.9013 (2)**
	$d = 3$	0.9031 (49)	0.8969 (49)	0.7864 (37)	**0.9044 (3)**
	$d = 4$	**0.9109 (49)**	0.1155 (49)	0.1152 (37)	0.9067 (4)
	$d = 5$	**0.9202 (49)**	0.6158 (49)	0.1152 (13)	0.9051 (2)

Table 7: Classification accuracies for trees of different depths constructed with the CART, DTIP or DTIP with feature selection (DTIPfs) algorithms. For the DTIPfs, the minimum and maximum accuracy over all number of features are stated. This accuracy is based on the Gini importance. For the DTIP(fs), the time limit is set to 30 minutes. The minimum number of features needed to get to this accuracy is noted in between brackets and the highest accuracies over the 3 methods are in bold.

verify these findings and find the best way to choose a number of features to maximize accuracy.

5 Conclusion

In this research, we have firstly replicated the work of Verwer and Zhang (2017) about the performance of a classification tree algorithm called DTIP and using trees constructed with CART as warm start solutions for this DTIP algorithm. From the CART algorithm, we gathered incomplete trees and leaf nodes with the same parent that share the same target. We changed those trees accordingly to be able to use them as warm start solutions in the DTIP. For the trees that run to optimality, we get the same accuracies for the DTIP and DTIPs as Verwer and Zhang (2017) do. The trees for which the algorithms have not run to optimality are also approximately equal. In almost all of the cases, the DTIP and DTIPs outperform the CART algorithm but we see that for large data sets such as the "Bank" data set, trees with depths 4 and 5 become too hard to construct and the DTIP lacks to build accurate trees.

After replicating the work of Verwer and Zhang (2017), we tried to get an answer to the following question: *Does incorporating feature importance selection in the integer formulation of Verwer and Zhang lead to more accurate and computationally efficient decision trees?* To answer this question, we have looked at the Gini importance measure and the Permutation importance measure. We examined the difference in performance of those two metrics and the overall effect of feature selection on the classification accuracy and solving time. The order of features based on these two metrics appear to be quite similar for all of the data sets and tree depths. As a result, the difference in accuracy of the DTIP with feature selection based on the two metrics is minimal. We do see a big difference in performance when building a tree of depth 4 on the "Bank" data set, where additional features based on the Gini importance measure make the problem too difficult to solve (the accuracy stays below 15%), while this does not happen when using the Permutation importance. In terms of solving time, we see substantial differences between the two metrics in some cases. However, this is different for every data set and depth of the trees, so no metric stands out as a clear winner in terms of solving time.

When we look at the overall effect of feature importance on the performance of the DTIP, we see that the difference in accuracy between trees that use only 1 feature and trees that use all features is less than 5 percentage points for all cases. Thus, with only using the first most important feature, the DTIP can already construct very accurate trees. In addition to this, the marginal gain in accuracy of adding features is minimal. Furthermore, we found that for most of the cases, the solving time of the DTIP with feature selection is almost the same as the solving time of the DTIP without feature selection. However, we found three trees of depth 2 and 3 on large data sets in which the DTIP is not able to run to optimality while the DTIP with feature selection is able to find optimal trees with an accuracy difference of maximal 2 percentage points.

On top of that, for very complicated problems, incorporating feature selection can lead to a massive increase in accuracy compared to the DTIP without feature selection. When comparing the DTIP with feature selection to the CART algorithm, we saw that for the small data set, the DTIP without feature selection is already able to beat the CART algorithm and the solving times are already low. For the medium size data set "Diabetes", feature selection increased the accuracy of the DTIP without feature selection and the performance beats that of CART. For the large data set, "Bank", for smaller depths ($d < 4$), feature selection can sometimes ensure the CART accuracies are defeated. Despite the remarkable improvement the feature selection procedure has on the accuracies when creating trees of higher depth, the DTIP still fails to beat the CART algorithm in these situations. The above conclusions are based on the number of features that maximized accuracy. We saw a pattern in which for depth 1, 2 and 3 only the first 1, 2 and 3 features were selected accordingly to achieve the maximal accuracy. However, future research based on more data sets is needed to find these numbers for different depths or an optimal threshold value in terms of the importance metric. In this way, we know beforehand how many features to select to achieve this maximal accuracy.

To see if the small size of the RAM (8GB) or the relatively low clock speed of the processor (1.4 GHz) of the computer which we used to perform the analysis has an impact on the results, future research should be conducted on the use of different devices with a larger memory or a better processor. As already mentioned, for future research on this subject, to enhance the generalizability of the findings, this research should be done on more data sets. In this way, more characteristics of data sets can be linked to the performance to investigate relationships. The effect of, for example, imbalanced data sets or correlated features can then be analyzed more thoroughly. Lastly, it would be interesting to look at other importance metrics to select the features.

References

Bertsimas, D. & Dunn, J. (2017). Optimal classification trees. *Machine Learning*, *106*. doi: 10.1007/s10994-017-5633-9

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5-32. doi: 10.1023/A:1010950718922

Dua, D. & Graff, C. (2017). *UCI machine learning repository.* Retrieved from `http://archive.ics.uci.edu/ml`

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M. & Kagal, L. (2019). Explaining explanations: An overview of interpretability of machine learning.

Iqbal, M., Rahaman, M. & Nabil, S. (2012). Construction of decision trees by using feature importance value for improved learning performance. In (Vol. 7664, p. 242-249). doi: 10.1007/978-3-642-34481-7_30

Menickelly, M., Gunluk, O., Kalagnanam, J. & Scheinberg, K. (2016). Optimal generalized decision trees via integer programming.

Nembrini, S., Knig, I. R. & Wright, M. N. (2018). The revival of the Gini importance? *Bioinformatics*, *34*(21), 3711-3718. Retrieved from `https://doi.org/10.1093/bioinformatics/bty373` doi: 10.1093/bioinformatics/bty373

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Ruggieri, S. (2019). Complete search for feature selection in decision trees. *Journal of Machine Learning Research*, *20*(104), 1–34. Retrieved from `http://jmlr.org/papers/v20/18-035.html`

Verwer, S. & Zhang, Y. (2017). Learning decision trees with flexible constraints and objectives using integer optimization. In D. Salvagnin & M. Lombardi (Eds.), *Integration of ai and or techniques in constraint programming* (pp. 94–103). Springer Science+Business Media. (Accepted author manuscript; International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems : CPAIOR 2017 ; Conference date: 05-06-2017 Through 08-06-2017) doi: 10.1007/978-3-319-59776-8_8

Zhou, H., Zhang, J., Zhou, Y., Guo, X. & Ma, Y. (2021). A feature selection algorithm of decision tree based on feature weight. *Expert Systems with Applications*, *164*, 113842. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0957417420306515` doi: https://doi.org/10.1016/j.eswa.2020.113842

A Feature importance values

A.1 Iris

Feature		**1**	**2**	**3**	**4**
$d = 1$	Gini	0.00	0.00	1.00	0.00
	Perm.	0.00	0.00	0.32	0.00
$d = 2$	Gini	0.00	0.00	0.56	0.44
	Perm	0.00	0.00	0.42	0.29
$d = 3$	Gini	0.00	0.00	0.05	0.95
	Perm.	0.00	0.00	0.52	0.11
$d = 4$	Gini	0.01	0.00	0.05	0.94
	Perm.	0.00	0.01	0.57	0.13
$d = 5$	Gini	0.01	0.01	0.55	0.42
	Perm.	0.01	0.01	0.58	0.14

Table 8: Both importance measures of different features per maximum depth of tree of "Iris" Data Set.

A.2 Diabetes

Feature		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
$d = 1$	Gini	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000
	Perm.	0.000	0.169	0.000	0.000	0.000	0.000	0.000	0.000
$d = 2$	Gini	0.000	0.795	0.000	0.000	0.000	0.205	0.000	0.000
	Perm	0.000	0.169	0.000	0.000	0.000	0.000	0.000	0.000
$d = 3$	Gini	0.007	0.775	0.000	0.000	0.000	0.181	0.037	0.000
	Perm.	0.000	0.151	0.000	0.000	0.000	0.000	0.021	0.000
$d = 4$	Gini	0.006	0.644	0.024	0.024	0.016	0.147	0.070	0.069
	Perm.	0.005	0.181	0.005	0.007	0.004	0.038	0.015	0.040
$d = 5$	Gini	0.005	0.531	0.034	0.019	0.042	0.154	0.101	0.113
	Perm.	0.005	0.161	0.009	0.006	0.017	0.051	0.037	0.054

Table 9: Both importance measures of different features per maximum depth of tree of "Diabetes" Data Set.

A.3 Bank

Feature		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	Gini	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000
	Perm.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
	Gini	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Perm.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
$d = 1$	Gini	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Perm.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
	Gini	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Perm.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		41	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	
	Gini	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
	Perm.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	

Table 10: Both importance measures of different features for classification tree on "Bank" Data Set with a maximum depth of 1.

Feature		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	Gini	0.000	0.000	0.000	0.571	0.000	0.000	0.000	0.000	0.000	0.000
	Perm.	0.000	0.000	0.000	0.030	0.000	0.000	0.000	0.000	0.000	0.000
		11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
	Gini	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.035
	Perm.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.008
		21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
$d = 2$	Gini	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Perm.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
	Gini	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Perm.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		41	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	
	Gini	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.393	
	Perm.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.030	

Table 11: Both importance measures of different features for classification tree on "Bank" Data Set with a maximum depth of 2.

Feature		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	Gini	0.000	0.000	0.000	0.613	0.000	0.000	0.000	0.000	0.000	0.000
	Perm.	0.000	0.000	0.000	0.060	0.000	0.000	0.000	0.000	0.000	0.000
		11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
	Gini	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.029
	Perm.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.004
		21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
$d = 3$	Gini	0.000	0.000	0.000	0.000	0.000	0.009	0.000	0.000	0.000	0.000
	Perm.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
	Gini	0.021	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Perm.	0.005	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		41	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	
	Gini	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.328	
	Perm.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.020	

Table 12: Both importance measures of different features for classification tree on "Bank" Data Set with a maximum depth of 3.

Feature		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	Gini	0.000	0.000	0.000	0.540	0.000	0.000	0.000	0.000	0.000	0.022
	Perm.	0.000	0.000	0.000	0.040	0.000	0.000	0.000	0.000	0.000	0.002
		11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
	Gini	0.000	0.000	0.008	0.000	0.000	0.000	0.000	0.000	0.000	0.025
	Perm.	0.000	0.000	0.002	0.000	0.000	0.000	0.000	0.000	0.000	0.008
		21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
$d = 4$	Gini	0.000	0.000	0.000	0.000	0.000	0.028	0.000	0.000	0.000	0.000
	Perm.	0.000	0.000	0.000	0.000	0.000	0.004	0.000	0.000	0.000	0.000
		31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
	Gini	0.018	0.000	0.000	0.000	0.000	0.016	0.000	0.013	0.000	0.000
	Perm.	0.002	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.000	0.000
		41	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	
	Gini	0.000	0.000	0.050	0.000	0.000	0.000	0.000	0.000	0.280	
	Perm.	0.000	0.000	0.005	0.000	0.000	0.000	0.000	0.000	0.021	

Table 13: Both importance measures of different features for classification tree on "Bank" Data Set with a maximum depth of 4.

Feature		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$d = 5$	Gini	0.000	0.024	0.004	0.505	0.000	0.040	0.000	0.000	0.000	0.019
	Perm.	0.000	0.004	0.000	0.054	0.000	0.013	0.000	0.000	0.000	0.002
		11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
	Gini	0.000	0.000	0.013	0.000	0.000	0.005	0.000	0.000	0.000	0.021
	Perm.	0.000	0.000	0.002	0.000	0.000	0.001	0.000	0.000	0.000	0.010
		21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
	Gini	0.000	0.005	0.000	0.000	0.000	0.024	0.000	0.000	0.000	0.000
	Perm.	0.000	0.001	0.000	0.000	0.000	0.004	0.000	0.000	0.000	0.000
		31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
	Gini	0.015	0.000	0.000	0.000	0.000	0.014	0.005	0.011	0.000	0.000
	Perm.	0.003	0.000	0.000	0.000	0.000	0.002	0.000	0.003	0.000	0.000
		41	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	
	Gini	0.000	0.000	0.056	0.000	0.000	0.000	0.000	0.000	0.239	
	Perm.	0.000	0.000	0.004	0.000	0.000	0.000	0.000	0.000	0.018	

Table 14: Both importance measures of different features for classification tree on "Bank" Data Set with a maximum depth of 5.

B Feature selection accuracies

B.1 Iris

Depth	**Features**	**Accuracy**	**Solving time**
$d = 1$	1	0.6667	0.055 s
	2	0.6667	0.189 s
	3	0.667	0.231 s
$d = 2$	1	0.9533	0.083 s
	2	0.96	0.167 s
	3	0.96	0.454 s
$d = 3$	1	0.96	0.206
	2	0.9867	0.482 s
	3	0.9933	2.085 s
$d = 4$	1	0.9533	1.23 s
	2	0.9867	3.668 s
	3	1.0	9.051 s
$d = 5$	1	0.9533	7.319 s
	2	0.9867	5.473 s
	3	1.0	5.473 s

Table 15: Accuracies and solving times of the DTIP on the "Iris" data set with feature selection using the Gini importance measure.

Depth	**Features**	**Accuracy**	**Solving time**
$d = 1$	1	0.6667	0.032 s
	2	0.6667	0.15 s
	3	0.6667	0.204 s
$d = 2$	1	0.9533	0.067 s
	2	0.96	0.15 s
	3	0.96	0.482 s
$d = 3$	1	0.9533	0.238
	2	0.9867	0.328 s
	3	0.9933	1.359 s
$d = 4$	1	0.9533	0.721 s
	2	0.9867	2.69 s
	3	1.0	4.132 s
$d = 5$	1	0.9533	8.086 s
	2	0.9867	7.339 s
	3	1.0	5.887 s

Table 16: Accuracies and solving times of the DTIP on the “Iris” data set with feature selection using the Permutation importance measure.

B.2 Diabetes

Depth	Features	Accuracy	Solving time
$d = 1$	1	0.75	0.213 s
	2	0.75	0.556 s
	3	0.75	0.66 s
	4	0.75	0.557 s
	5	0.75	0.803 s
	6	0.75	0.789 s
	7	0.75	1.068 s
$d = 2$	1	0.7526	1.201 s
	2	0.7734	78.807 s
	3	0.7734	134.586 s
	4	0.7734	524.991 s
	5	0.7734	407.129 s
	6	0.7734	1786.441 s
	7	0.7734	1800 s (time limit)
$d = 3$	1	0.7578	50.717 s
	2	0.7826	1800 s (time limit)
	3	0.7878	1800 s (time limit)
	4	0.7852	1800 s (time limit)
	5	0.7773	1800 s (time limit)
	6	0.7760	1800 s (time limit)
	7	0.7799	1800 s (time limit)
$d = 4$	1	0.7604	1800 s (time limit)
	2	0.8008	1800 s (time limit)
	3	0.7760	1800 s (time limit)
	4	0.7969	1800 s (time limit)
	5	0.8190	1800 s (time limit)
	6	0.7943	1800 s (time limit)
	7	0.7995	1800 s (time limit)
$d = 5$	1	0.7604	1800 s (time limit)
	2	0.7982	1800 s (time limit)
	3	0.8216	1800 s (time limit)
	4	0.7930	1800 s (time limit)
	5	0.7878	1800 s (time limit)
	6	0.7826	1800 s (time limit)
	7	0.75	1800 s (time limit)

Table 17: Accuracies and solving times of the DTIP on the “Diabetes” data set with feature selection using the Gini importance measure.

Depth	Features	Accuracy	Solving time
$d = 1$	1	0.75	0.213 s
	2	0.75	0.556 s
	3	0.75	0.66 s
	4	0.75	0.557 s
	5	0.75	0.803 s
	6	0.75	0.789 s
	7	0.75	1.068 s
$d = 2$	1	0.7526	1.243 s
	2	0.7630	25.929 s
	3	0.7630	106.674 s
	4	0.7630	209.617 s
	5	0.7630	383.998 s
	6	0.7734	1158.708 s
	7	0.7708	1800 s (time limit)
$d = 3$	1	0.7578	55.705 s
	2	0.7695	1800 s (time limit)
	3	0.7734	1800 s (time limit)
	4	0.7760	1800 s (time limit)
	5	0.7734	1800 s (time limit)
	6	0.7747	1800 s (time limit)
	7	0.7864	1800 s (time limit)
$d = 4$	1	0.7604	1800 s (time limit)
	2	0.7721	1800 s (time limit)
	3	0.7956	1800 s (time limit)
	4	0.7943	1800 s (time limit)
	5	0.7799	1800 s (time limit)
	6	0.7734	1800 s (time limit)
	7	0.8021	1800 s (time limit)
$d = 5$	1	0.7604	1800 s (time limit)
	2	0.8060	1800 s (time limit)
	3	0.7760	1800 s (time limit)
	4	0.8099	1800 s (time limit)
	5	0.6979	1800 s (time limit)
	6	0.7904	1800 s (time limit)
	7	0.8138	1800 s (time limit)

Table 18: Accuracies and solving times of the DTIP on the "Diabetes" data set with feature selection using the Permutation importance measure.

B.3 Bank

Depth	Features	Accuracy	Solving time
$d = 1$	1	0.8912	1.271 s
	7	0.8912	14.068 s
	13	0.8912	21.635 s
	19	0.8912	24.363 s
	25	0.8912	32.012 s
	31	0.8912	51.711 s
	37	0.8912	57.221 s
	43	0.8912	69.339 s
$d = 2$	1	0.8932	38.757 s
	7	0.9013	1800 s (time limit)
	13	0.9013	1800 s (time limit)
	19	0.9000	1800 s (time limit)
	25	0.8998	1800 s (time limit)
	31	0.8943	1800 s (time limit)
	37	0.8952	1800 s (time limit)
	43	0.8956	1800 s (time limit)
$d = 3$	1	0.8949	1800 s (time limit)
	7	0.8870	1800 s (time limit)
	13	0.8872	1800 s (time limit)
	19	0.8861	1800 s (time limit)
	25	0.8925	1800 s (time limit)
	31	0.8910	1800 s (time limit)
	37	0.7864	1800 s (time limit)
	43	0.8856	1800 s (time limit)
$d = 4$	1	0.8974	1800 s (time limit)
	7	0.9058	1800 s (time limit)
	13	0.8916	1800 s (time limit)
	19	0.8854	1800 s (time limit)
	25	0.8890	1800 s (time limit)
	31	0.8848	1800 s (time limit)
	37	0.1152	1800 s (time limit)
	43	0.1228	1800 s (time limit)
$d = 5$	1	0.8998	1800 s (time limit)
	7	0.1154	1800 s (time limit)
	13	0.1152	1800 s (time limit)
	19	0.1154	1800 s (time limit)
	25	0.2820	1800 s (time limit)
	31	0.1155	1800 s (time limit)
	37	0.1155	1800 s (time limit)
	43	0.1152	1800 s (time limit)

Table 19: Accuracies and solving times of the DTIP on the "Bank" data set with feature selection using the Gini importance measure.

Depth	**Features**	**Accuracy**	**Solving time**
$d = 1$	1	0.8850	2.149 s
	7	0.8912	13.36 s
	13	0.8912	12.045 s
	19	0.8912	21.797 s
	25	0.8912	100.964 s
	31	0.8912	21.24 s
	37	0.8912	27.065 s
	43	0.8912	101.806 s
$d = 2$	1	0.8929	0.032 s
	7	0.8963	1800 s (time limit)
	13	0.8998	1800 s (time limit)
	19	0.8998	1800 s (time limit)
	25	0.8934	1800 s (time limit)
	31	0.8956	1800 s (time limit)
	37	0.8943	1800 s (time limit)
	43	0.8987	1800 s (time limit)
$d = 3$	1	0.8949	1800 s (time limit)
	7	0.8943	1800 s (time limit)
	13	0.8967	1800 s (time limit)
	19	0.8925	1800 s (time limit)
	25	0.8856	1800 s (time limit)
	31	0.8943	1800 s (time limit)
	37	0.8894	1800 s (time limit)
	43	0.8872	1800 s (time limit)
$d = 4$	1	0.8974	1800 s (time limit)
	7	0.9069	1800 s (time limit)
	13	0.8912	1800 s (time limit)
	19	0.8852	1800 s (time limit)
	25	0.8907	1800 s (time limit)
	31	0.8905	1800 s (time limit)
	37	0.8755	1800 s (time limit)
	43	0.8837	1800 s (time limit)
$d = 5$	1	0.9002	1800 s (time limit)
	7	0.1155	1800 s (time limit)
	13	0.1155	1800 s (time limit)
	19	0.1152	1800 s (time limit)
	25	0.6959	1800 s (time limit)
	31	0.1155	1800 s (time limit)
	37	0.1155	1800 s (time limit)
	43	0.1155	1800 s (time limit)

Table 20: Accuracies and solving times of the DTIP on the "Bank" data set with feature selection using the Permutation importance measure.

Depth	Features	Accuracy	Solving time
$d = 1$	2	0.8912	10.683 s
	3	0.8912	15.969 s
	4	0.8912	8.548 s
	5	0.8912	22.125 s
	6	0.8912	16.408 s
$d = 2$	2	0.9013	91.726 s
	3	0.9013	69.868 s
	4	0.9013	424.808 s
	5	0.9009	1800 s (time limit)
	6	0.9013	1800 s (time limit)
$d = 3$	2	0.9040	1800 s (time limit)
	3	0.9044	1800 s (time limit)
	4	0.9042	1800 s (time limit)
	5	0.9033	1800 s (time limit)
	6	0.9027	1800 s (time limit)
$d = 4$	2	0.8008	1800 s (time limit)
	3	0.9055	1800 s (time limit)
	4	0.9067	1800 s (time limit)
	5	0.9064	1800 s (time limit)
	6	0.8971	1800 s (time limit)
$d = 5$	2	0.9051	1800 s (time limit)
	3	0.9033	1800 s (time limit)
	4	0.8952	1800 s (time limit)
	5	0.6457	1800 s (time limit)
	6	0.1152	1800 s (time limit)

Table 21: Accuracies and solving times of the DTIP on the "Bank" data set with feature selection using the Gini importance measure, looking at the cases where 2 to 6 features are selected.

C Programming code

In this Section, a short description of the code used in our research is given. A lot of comments are also present in the code itself. One part of the code is written in Python, and the other one is written in Java. In this appendix, we go into depth on both parts and explain what the different classes or methods do.

C.1 Python code

The python code (JavaAndPython/python/CART.py) exists of 7 different methods. It is used to construct CART trees with the scikit-learn package. The methods and their functionality are listed below in Table 22:

Name	**Functionality**
prediction-errors	A method to write the prediction errors of trees of 5 different depths to a csv file for a data set.
paths	A method to write the 5 path matrices for the different depths of a data set to a csv file.
tree-data	A method to print the classification accuracy, threshold array, feature and class matrix and to get a visualization of the tree.
visualize-tree	A method to get a visualization of the tree.
bfs	A method to get a breadth first traversal of a given tree.
get-path-matrix	A method to retreive the path matrix of a tree.
importance-scores	A method to print the Gini and Permutation importance scores of a certain tree for all features.

Table 22: The methods in the CART.py file and their functionality.

For all three data sets, we call the *prediction-errors* and *paths* methods to write the prediction errors and path matrices of the trees for a data set to a csv file. This csv file will later be used for the warm start solutions of the DTIP. Thereafter, for all three data set and different depths, we call the *importance-scores* and *tree-data* methods to print the importance scores to analyze them, and use the tree data for the warm start solutions and to visualize the trees.

C.2 Java code

The java code (JavaAndPython/IProgram/src) consists of 7 overall classes. The classes and their functionality are listed below in Table 23:

Name	**Functionality**
TransformData	A class to read feature matrices and target columns and transform the data.
WriteCSV	A class to write the transformed data to a csv file.
Matrices	A class to calculate the values for variables M, M', UF and LF based on the feature data of a data set.
Main	A class to implement the DTIP integer formulation and run it on the different data sets for different depths.
WarmStartMatrices	A class to construct the matrices for the warm start solutions. They are constructed either from a csv file or by copying them from the Python code into arrays.
WarmStart	A class to implement the DTIP integer formulation with warm start solutions and run it on the different data sets for different depths.
FeatureImportance	A class to select features based on the Gini- and Permutation importance and run the DTIP with the corresponding features for different data sets and depths.

Table 23: THe classes in the JavaDTIP(fs) file and their functionality.

For all three data sets, we firstly use the *TransformData* class to transform the data. Then, we use the *WriteCSV* class to write this transformed data to a csv file which is used throughout the rest of the research. To run the *Main* class for all data sets to retreive the DTIP solutions, we use the *Matrices* class to get certain values needed to run it. We then use the *WarmStart* and *WarmStartMatrices* classes to retreive the warm start results for all data sets and in the end, the *FeatureImportance* class is used to analyze the impact of feature selection on the DTIP for all data sets and features.