ERASMUS UNIVERSITEIT ROTTERDAM
Erasmus School of Economics
Bachelor Thesis Double degree Bsc^2 in Econometrics and Economics:
Business Analytics and Quantitative Marketing

Optimal Decision Trees in Economics

A Framework to Construct Binary Decision Trees with Categorical Data

---

M.J.F. Korf

Student ID: 455752
Supervisor: Prof. Dr. S.I. Birbil
Second Assessor: Dr. O. Karabag

July 5, 2020

Abstract

Decision trees are a class of machine learning algorithms that are praised for their ease of interpretability and versatility. Recently, an integer programming formulation using categorical variables has been devised by Günlük et al. (2018), called Optimal Decision Trees (ODT), which has good computational and classification results. We scrutinise their findings and devise a framework to construct decision trees more thoroughly and investigate its suitability for economic contexts. We find that the ODT technique proposed by Günlük et al. (2018) shows good performance and computational efficiency, except for the dataset imbalance correction factor. Also, feature selection can be advantageous and K-fold cross-validation improves the validity of the accuracy estimates. When applying the framework to economic contexts, one has to beware that the model needs re-estimation frequently. We conclude that ODT is a good technique that can handle classification problems well, also in economic contexts.

Keywords: Decision Trees, IP Formulation, Machine Learning, Cross-validation, Feature Selection

Contents

1 Introduction

Machine learning techniques have entered our daily lives more and more over the past few years. Smart image recognition and natural language processing software, for instance, have changed life in myriad ways. As for the business world, there are also numerous opportunities to improve efficiency and accuracy of the work done, for example with assigning credits scores to potential borrowers in the financial sector. However, professionals are often hesitant to implement new techniques that replace well-known and tested methods, if they do not fully understand the new ones themselves. Also, machine learning techniques do not always caters to the specific needs of the application area, such as economics. A class of machine learning techniques that is often praised for its ease of interpretability and versatility is decision trees, although this ease is perhaps only in a relative manner to other highly complicated black-box algorithms. Since even with this class of algorithm, it can become too difficult to understand for field experts who would need to apply it. Or, as Quinlan (1987) puts it: "If a decision tree that measures up very well on the performance criterion is nevertheless totally incomprehensible to a human expert, can it be described as knowledge?"

To maximise the impact of the decision tree method and empower the professionals that would use them, we investigate the following question in this paper:

How can we provide accurate decision trees that are suitable for an economic context?

We will first focus on obtaining accurate decision trees following and improving upon the methods provided by Günlük et al. (2018). After-which, we investigate ways for improvement of the suitability in economic contexts, for example by improving the interpretability.

The primary aim of the decision tree method is to find a tool that can correctly classify data. For example, when one knows a couple of characteristics of a loan applicant, can one predict whether he or she is likely to default on his or her loan? A decision tree often employs binomial classification and categorical data. The former means that it identifies a sample as one of two options, denoted with -1 and 1 respectively, representing whether or not a customer repays its loan, for instance. Categorical data comes in the form that a particular attribute has a fixed number of different options, e.g. you either own a home, rent a home or live at your parents. Only one can be true for each sample, or in this case each loan applicant. In a binomial decision tree with categorical data, multiple decision with a fixed number of options are made to classify a sample into one of two options.

Moreover, we discuss the suitability of decision trees for economics, by looking at issues that arise specifically in an economic context. Namely, a model needs to have a proper handling of exogenous shocks and needs to be interpretable in an economic setting.

We use data from the UCI machine learning repository (Dua & Graff, 2017), LIBSVM (Chang & Lin, 2011) and the FICO Explainable Machine Learning Challenge (Fair Isaac Corporation, 2018). The last one includes data on loan applications. We will also look into the other non-economic datasets to improve our methodology, as described in the paragraph below.

There are multiple classes of decision trees and various optimisation methods. A recent one proposed by Günlük et al. (2018) is Optimal Decision Trees (ODT). This method provides an integer programming formulation for decision trees with binomial classification and multinomial data. They improved the computational efficiency of the integer programming formulation by exploiting the categorical nature of the input data. We try to emulate this formulation and propose a comprehensive framework to construct decision trees with ODT. Herein, we incorporate the IP formulation and its suggested improvements by Günlük et al. (2018). Also, we investigate possible extensions regarding a priori dataset assessment, feature selection, tree topology, K-fold cross-validation and validity assessment. Subsequently, we discuss issues that arise specifically in economic contexts, namely the validity over time and interpretability.

We find that the ODT technique proposed by Günlük et al. (2018) has good performance and computational efficiency, except for their suggested dataset imbalance correction factor. Our five-step framework provides a systematic system for constructing decision trees, which incorporates ODT and our improvements, as we find that our extensions can be fruitful additions. Namely, well-executed feature selection is advantageous and K-fold cross-validation can improve the validity of the accuracy estimates. Furthermore, when applying the framework to economic contexts, one has to be aware that the model needs re-estimation frequently, if it does not handle exogenous shocks well. We conclude that ODT is a good technique that can handle classification problems well, also in economic contexts.

This research contributes to the suitability and through that to more wide-spread application of machine learning algorithms. Such applications will increase efficiency and accuracy of choices made by professionals and hence allow for more informed and better decisions, which has limitless potential to increase efficiency in our economy and society. Also, we try to improve the computational efficiency to contribute to the research area. Machine learning methods have the potential to transform economics and improve its empirical research methods. Our study helps to identify how this can be achieved by providing a data classification tool and a clear framework to construct it, namely ODT.

In this report, we will first discuss the literature on decision trees and its application to economics in Section 2. Then, we will discuss the theoretical framework of decision trees and their optimisation in Sections 3.1 to 3.3. This is followed by a our extensions to this framework in Section 3.4 and a discussion of the framework's suitability for economics in Section 3.5. Subsequently, we will present results of numerical experiments in Section 4, followed by a conclusion and discussion in Section 5.

2 Literature Review

It is the explosion of available data, the enormous advances in computing power and the increased interest in data science that make for rapid development of the machine learning field in recent years. In economics, where we often employ empirical data to construct models, implementing findings of machine learning techniques can have substantial benefits. As Athey (2018) puts it: "Machine learning will have a dramatic impact on the field of economics." We will now discuss literature on machine learning, and decision trees in particular. Afterwards, we elaborate on current research conducted regarding the application to economics.

2.1 Machine Learning and Decision Trees

Machine learning is the discipline that develops algorithms for dataset analysis, with the main focus on prediction, classification, and clustering or grouping tasks (Athey, 2018). In this paper, we focus on one supervised classification algorithm, namely decision trees. Supervised means that each data point has a label and we set out to teach the algorithm to recognise the label of a data point given its other characteristics.[1] An algorithm that classifies data points is also known as a classifier:

$$\phi(S_n) : S_n \mapsto \hat{\vec{y}} \quad \text{with} \quad \varepsilon(\phi) = \frac{1}{n}\sum_{i=0}^{n} \min(1, |\hat{y}_i - y_i|),$$

where $\phi(S_n)$ is the classifier that can be interpreted as a function that maps the dataset S_n to an estimated classification vector $\hat{\vec{y}}$. This estimator has its error, $\varepsilon(\phi)$, which is equal to the percentage of false classifications.[2] The principal aim of supervised machine learning is to find the classifier with the 'best' error, i.e. an unbiased one with low variance. The dataset will most often have a natural error rate, like how something can meet all criteria, but still not be it. The best classifier has an error with minimal variance and an expected value equal to the natural error rate in the data.

With machine learning algorithms, there is a difference between in-sample and out-of-sample performance. When the former is much better than the latter, there is so-called overfitting. This can be countered by regularisation, which is the process of restraining the fit of the in-sample data to increase the out-of-sample performance (Mullainathan & Spiess, 2017). With decision trees this can be done by tweaking the tree depth.[3] Namely, the more depth a tree has, the more it can be moulded into the in-sample data points and improve its classification performance on them. However, this generally results in deterioration of out-of-sample performance. As we regularise more, we do a worse job at approximating the in-sample variation, but for the same reason, the wedge between in-sample and out-of-sample fit will typically decrease. As a corollary, simpler models tend to perform better for out-of-sample data (Varian, 2014).

[1] Unsupervised algorithms, on the other hand, work with unlabelled data.

[2] Notice that the classifier error equation, '$min(1, |\hat{y}_i - y_i|$' equals 0 when the estimated and realised y are the same, and zero otherwise.

[3] A parameter used for regularisation is called a regulariser, which is tree depth in this case.

The difference between supervised machine learning and econometric methods is that the former focuses on predicting the dependent variable, whereas the latter on estimating parameters, e.g. β in OLS. Machine learning methods often outperform conventional econometric methods for out-of-sample prediction of the dependent variable. Namely, as compared to regression models, they are particularly good at finding interdependencies in data (Mullainathan & Spiess, 2017).

Decision Trees have been a prevalent class of machine learning models. Breiman et al. (1984) discussed multiple techniques to solve a tree to optimality given a specific topology. They distinguished between classification and regression trees, collectively known as CART. The former labels a certain sample and the latter assigns a real value to it. In this paper, we focus on classification trees.

In general, decision trees techniques comprise two stages: the building procedure and the classification procedure (Kotsiantis, 2013). In the former, the optimal tests are determined to classify the data using a subset of the samples. During the latter, the remaining samples are classified using the tree to test its accuracy. These subsets are referred to as the training set and the testing set.

In the building procedure, we want to translate a problem into a useful decision tree. For this, there are three things to construct: (a) a tree topology, (b) the binary tests applied at each decision node, and (c) assignment of the labels to each leaf node (Günlük et al., 2018). Solving a decision tree to optimality only involves step (b) and (c).

2.2 Application in Economics

Varian et al. (2014) showed that decision trees and other machine learning techniques could be incorporated into economics to improve the scientific discipline and specifically its empirical research. For example, with data on the survivors of the Titanic, a decision tree could identify that the elderly and children had different survival odds from an age variable. They showed that a conventional logistic regression did not easily find this pattern when one was not explicitly looking. Hence, a decision tree can offer valuable complementary insights alongside conventional econometric techniques.

Furthermore, machine learning allows economists to view their models and theories as algorithms, which can be tweaked and estimated frequently (Athey, 2018). This approach is far more potent than only observing what happens when a certain endogenous variable increases ceteris paribus. As such, more complex relations and models can be investigated. This tweaking is essentially model selection, which has also been part of economic research. However, machine learning excels in this as it is data-driven.

Moreover, Zurada (2010) looked into machine learning models that could assess whether a loan application should be approved or denied and found that decision trees provide the most interpretable model, as compared to other machine learning methods. As such, decision trees are well suitable for economics because they are one of the most interpretable machine learning classes that currently exist.

3 Theoretical Framework

In the following, we present a theoretical framework to construct optimal decision trees given a dataset. Firstly, we provide an explanation of decision trees and then an integer programming formulation to find the optimal one for a given dataset. We incorporate the improvements suggested by Günlük et al. (2018) in this formulation. Afterwards, we improve upon this framework ourselves by proposing extensions. Lastly, we investigate the framework's suitability in economic contexts.

3.1 Decision Trees

A Decision Tree (DT) is a tree-like graph that can be used as an auxiliary tool for decision making. We apply a DT to construct a classification model for samples. A DT consists of nodes, tests and labelling, as we see in the example in Figure 1.[4]

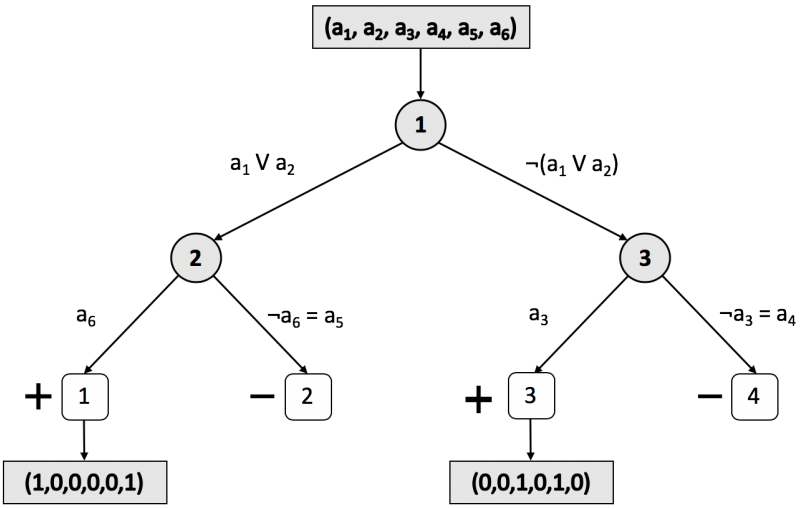

Figure 1. Example binomial decision tree with categorical data respectively

Here we can see that a DT consists of decision and leaf nodes. The round nodes depict the former, where the tree directs a sample downwards based on a logical test until a sample eventually ends up at a leaf node, i.e. one the square nodes. These leaf nodes either have positive labels (node 1 and 3) or negative ones (node 2 and 4). When a sample ends up at a positive leaf, the DT classifies that sample as positive. The tree decides which sample ends up at which leaf by looking at the features of the sample. In this example, there are two groups of features, $g_1 = \{a_1, a_2, a_3, a_4\}$ and $g_2 = \{a_5, a_6\}$. We can see that when the sample has a_3 from the former choice and a_5 from the latter choice, it will end up at leaf node 3 with a positive label. With a certain tree specification, a tree can direct a sample to a leaf node with the same labelling as its empirical value or not, i.e. the sample is classified correctly or not.

[4]This example is similar to one given in Günlük et al. (2018).

The tree that has the highest classification accuracy on a given training sample is called an Optimal Decision Tree (ODT). The general procedure to obtain an ODT is as follows:

(i) Transform the data to fit into the DT context.

(ii) Choose a tree topology.

(iii) Optimise the tree

(iv) Test the classification capabilities of the tree

(v) Accept the tree and end, or reject and start over at the step (ii) or (iii)

These steps provide a framework for constructing an ODT and we will elaborate on each step. We will refer to this framework as ODT as well.

For the first step, we need some mathematical notation to execute it systematically. Following the notation of Günlük et al. (2018), we consider a sample with n observations of the form $S_n = \{(g_1^i, g_2^i, \dots, g_t^i, y^i) : i \in 1, 2, \dots, n\}$, where $g_j^i \in G_j$ for $j = 1, 2, \dots, t$ and $y \in \{-1, +1\}$. Here g_j^i is the realised attribute at choice j of sample i and G_j is the set of all options at choice j. For example, g_j^i could be $Blue$ and $G_j = \{Blue, Grey, Brown\}$ when considering eye colour. In total, the sample has $|\bigcup_{j=1}^{t} G_j| = d$ number of different features spread across t different groups. For example, there are 12 features spread over four groups in Figure 2.

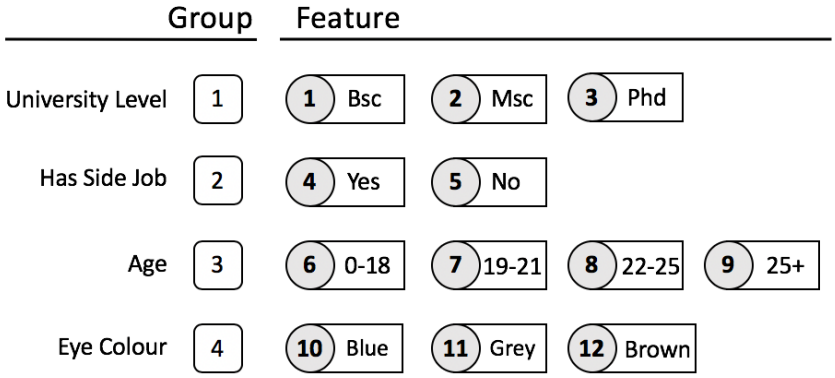

Figure 2. Example of groups and features for a dataset on university students

For computational ease, we transform the data in such a way that we obtain a binary array a_i of length d with the realised features, i.e. one when the sample has feature j and zero otherwise.[5] Also, let $A = [\vec{a}_i, \dots, \vec{a}_n]'$ be the realised observation matrix, or simply realisation matrix. With this notation, evaluating the sample at each decision node simplifies to a binary test. Note that this works well with categorical data and needs some modifications to handle real-valued data. In the latter case, one can employ bucketing of the data and use the different buckets as binary variables. For example, when considering age, one could replace the continuous variable using four binary ones which indicate whether someone's age falls into one of the following buckets: *0-18, 19-21, 22-25* and *25+*, see also the example in Figure 2.

[5] See also the example sample on top of and below of Figure 1.

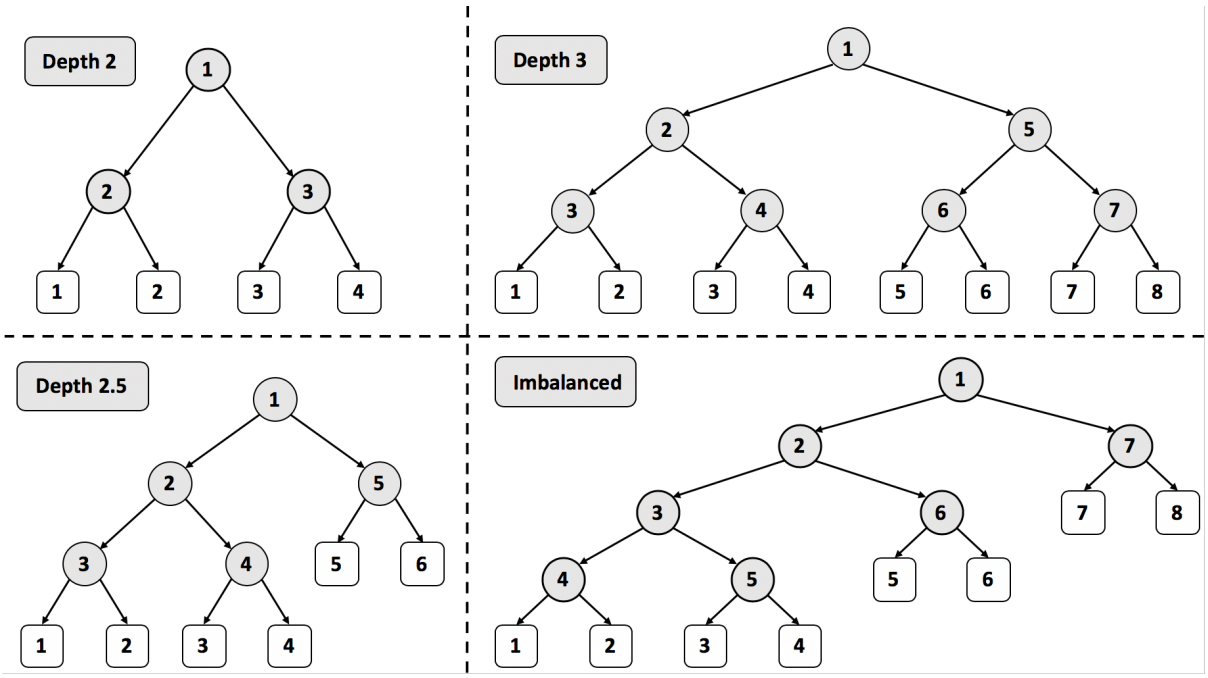

Figure 3. Possible tree topologies for ODT

With the data suitably formatted, we choose a tree topology in step (ii). The tree topology influences the structure of the tests and hence the possible accuracy of a tree. In this paper, we use a procedure that optimises a tree given a certain topology. As such, to find the optimal topology, multiple trees have to be made and compared after the procedure. In this research, we work with binary trees, i.e. every node has two children or none, such as in Figure 1. For our study, we will consider four tree topologies with increasing complexity: a *Depth 2*, a *Depth 2.5*, a *Depth 3* and an *Imbalanced* tree, shown in Figure 3.

3.2 Integer Programming Formulation

Given a tree topology, we aim to find the optimal set of tests at each decision node in step (iii). In order to do so, we need to formulate a decision tree mathematically.

Let $K = \{1, 2, \ldots, |K|\}$ be the set of decision nodes and $B = \{1, 2, \ldots, |B|\}$ be the set of leaf nodes. Then, we create variables to denote which groups and features are used for testing at a specific decision node. Let v_g^k, $z_j^k \in \{0, 1\}$ be whether group g is selected for branching at decision node k, and whether feature j is (one of) the selected feature(s). We want that, at each decision node, we decide based only on one group of choices, i.e. that each decision node only branches on one group. Therefore we add the following restrictions:

$$\sum_{g \in G} v_g^k = 1 \quad \forall k \in K, \tag{1}$$

$$z_j^k \leq v_{g(j)}^k \quad \forall j \in J, \forall k \in K, \tag{2}$$

where $g(j)$ is the group corresponding to feature j. In these equations, (1) ensures that each

node branches on only one group and (2) that a node can only branch on a certain feature if it also branches on its corresponding group. Note that this allows for branching on multiple features per group. Combining these restriction yields the set S of different possible tests given a certain tree topology, i.e. all possible combinations of $(\vec{v}, \vec{z})$:

$$S = \left\{ (\vec{v}, \vec{z}) \in \{0,1\}^{|K|\times|G|} \times \{0,1\}^{|K|\times d} : \ (\vec{v}, \vec{z}) \ satisfies \ (1) \ and \ (2) \right\}. \tag{3}$$

With this set, we have a collection of all possible tests and we can check each one to find the best ones, i.e. the optimal $(\vec{v}, \vec{z})$. For this check, we need to know which sample is directed to which leaf given certain tests. To this end, we introduce $c_b^i \in \{0,1\}$, which depicts whether sample i is routed to leaf b given $(\vec{v}, \vec{z})$. Also, let $L(i,k) = \sum_{j\in J} a_j^i z_j^k = 1 - R(i,k) \quad \forall k \in K, \forall i \in I$, whether sample i will be directed to the left at decision node k, and $R(i,k) = 1 - L(i,k)$ to the right respectively. With these variables, we can add restrictions to ensure that when sample i is routed to leaf node b, all the tests at the decision nodes it passes, lead this sample to this particular leaf b. Specifically, when sample i is routed to leaf 2 in the *Depth 2* tree in Figure 1, we need to have $L(i,1) = 1$ and $R(i,2) = 1$ and $c_2^i = 1$ and $c_b^i = 0$ for $b \in B \setminus \{2\}$. This leads to the following equations:

$$c_b^i \leq I_{k\in K^L(b)} L(i,k) + I_{k\in K^R(b)} R(i,k) \quad \forall k \in K, \forall i \in I, \tag{4}$$

$$\sum_{b\in B} c_b^i = 1 \quad \forall i \in I, \tag{5}$$

where I_C is the indicator function which yields 1 when C is true and 0 otherwise. $K^L(b)$ and $K^R(b)$ are the sets of decision nodes, at which a sample needs to branch to the left to arrive at leaf b, or the right respectively. We only allow for samples to be routed to one leaf at a time and hence have equation (5). These equations provide the possible sample routing of the DT given certain tests, i.e. $(\vec{v}, \vec{z})$:

$$Q(z,v) = \left\{ c \in \{0,1\}^{N\times|B|} : \ c \ satisfies \ (4) \ and \ (5) \right\}. \tag{6}$$

With the possible tests at the decision nodes (S) and routing of the samples (Q), we want to optimise these tests and routing in order to classify most samples correctly.[6] Remember that each sample has a realised y value of -1,1 and that each leaf assigns a label -1,1 to each sample. As such, let B_+ and B_- be the set of leaves with positive and negative labels, respectively. Also, let I_- and I_- be the set of samples that have a positive y or a negative one, respectively. Then, as objective function, we sum the total number of correct positive and negative labelling. This yields the integer programming formulation for optimising a DT:

[6] In machine learning terminology: $\hat{\phi} = \arg\min_{\phi}(\varepsilon(\phi_i))$.

$$\begin{aligned} \max \quad & \sum_{i \in I_+} \sum_{b \in B_+} \mathrm{c}_b^i + \sum_{i \in I_-} \sum_{b \in B_-} \mathrm{c}_b^i \\ \text{s.t.} \quad & (z, v) \in \mathrm{S} \\ & \mathrm{c} \in \mathrm{Q(z, v)} \ , \end{aligned} \tag{7}$$

where S imposes restrictions (1) and (2), and $Q(z, v)$ restrictions (4) and (5). Put into words, the equation says: maximise the number of correct classifications such that we have valid tests at each node and tests direct each sample to exactly one leaf node. So, when the IP tries to reclassify a false positive to a different leaf node, it tweaks the tests to direct the sample to a different leaf node. As such, it also alters the routing of the other samples and thus changes the tree formulation. The algorithm tries to find the optimal of such formulations, where here optimal means most correct classifications. The integer program in (7) can be solved with software, such as the IBM ILOG CPLEX Optimisation Studio and Python, which we use in this research.

After the optimisation, we test the tree's classification capabilities in step (iv). We assess how well the constructed DT can classify new data. For this, we compute an accuracy rate in order to evaluate the constructed tree. This metric is calculated by dividing the objective value by the training sample size.

With this metric, we make a judgement call in step (v) whether or not to accept the tree. There is no clear cut-off value for what makes a tree accurate, this is highly context-dependent.[7] If one rejects the tree, one can go back to step (ii) and try a different tree topology. Otherwise, one can conclude that the dataset cannot be described accurately with a decision tree and end the procedure.

3.3 Improving the Formulation

Considering the optimisation in step (iii), the formulation in (7) is of a simple form and can be extended upon to improve computational efficiency and classification accuracy. For example, one can apply a linear relaxation, omit redundant variables, strengthen the formulation, employ anchoring, account for ordering of numerical values and dataset imbalance (Günlük et al., 2018). We will discuss these techniques in the following, in order to arrive at an improved formulation.

First of all, a linear relaxation of variables in (7) would yield a much less complex optimisation problem (Fisher, 1981). Namely, a major computational disadvantage in optimisation is the number of integer or binary variables, as these increase the complexity of the problem. As such, reducing the number of binary variables in our formulation improves computational efficiency. Instead of using a Lagrangian-like penalty in our objective value, Günlük et al. (2018) showed that only the z_j^k variables that are not adjacent to leaf nodes need to be declared binary, in order for the other variables to stay integral in an optimal solution. Hence, we can apply a linear relaxation on all c_b^i, v_g^k and some z_j^k. This reduces the number of binary variables in the model drastically.

[7] We elaborate on this in Section 3.4.5

Secondly, we can further reduce the complexity of the problem when we alter the constraints in such a way that the IP formulation becomes tighter (Trick, 2005). In other words, when the constraints enclose the possible set more tightly, the software spends less time optimising infeasible solutions, as these are not considered. Günlük et al. (2018) found that we can replace equation (4) with the following tighter ones:

$$\sum_{b \in B^L(k)} c_b^i \leq L(i,k) \qquad \forall k \in K, \forall i \in I, \tag{8}$$

$$\sum_{b \in B^R(k)} c_b^i \leq R(i,k) \qquad \forall k \in K, \forall i \in I, \tag{9}$$

where $B^R(k)$ and $B^L(k)$ are the set of leaf nodes b that can be reached by branching to the right at node k, or the left respectively. Equation (8) and (9) are stricter than equation (4). Hence, the computational efficiency of the problem should be improved.

Moreover, we can apply anchoring to reduce the number of possible tests, i.e. reduce the size of S. This can be done because most binary tree topologies have many symmetrical nodes, i.e. the sub-tree at its right child has the same topology as the one at his left child. This means that when one inverts the binary test at such a node and all its children, one ends up at a tree that is the mirror image of the original. In practice, these additional specifications do not add value to the problem and hence do not need to be evaluated. Hence, we can assign a single feature per group to always branch to the left in order to break the symmetry. These are called anchor features. To this end, we add the following restrictions to the model:

$$z_j^k(g) = v_g^k \qquad \forall k \in K^{Sym}, \forall g \in G, \tag{10}$$

where K^{Sym} is the set of symmetrical decision nodes that are not adjacent to leaves nodes. Note that when one imposes anchoring on nodes next to to leaves, samples with the anchor features are forced to obtain a positive label, which leads to unnecessary erroneous classifications.

Another point for improvement of the formulation is to take the ordinal nature of features in some groups into account. When considering age, for example, one is often interested in whether being older than a certain age is of influence. Conversely, when considering income, whether someone making more than a certain amount of money per year is more likely to buy a product, for instance. However, the formulation in (7) only handles data as categorical. So, we have to adjust it to also adequately handle continuous data and ordered categorical data. Therefore, we add some restrictions to take this ordering into account:

$$z_j^k \geq z_{j+1}^k - w_g^k \qquad \forall j,\ j+1 \in J(g) \tag{11}$$

$$z_j^k \geq z_{j+1}^k - (1 - w_g^k) \qquad \forall j,\ j-1 \in J(g) \tag{12}$$

$$w_g^k \in \{0,1\} \qquad \forall k \in K,\ \forall g \in G^{Num}, \tag{13}$$

where G^{Num} is the set of groups that have numerical features and w_g^k is an indicator variable of

whether we include values greater than or less than a certain one at decision node k and group g. Equation (11) and (12) can be interpreted as constraining the features of a certain group to be ordered with w_g^k indicating whether the order is increasing or decreasing.

For groups with continuous data points, one can employ bucketing to transform these into ordered categorical features in the group, as aforementioned. One way to do this, is to use ten quantiles, which we employ. These transformed variables can then be used in the methodology presented here.

Furthermore, Günlük et al. (2018) proposed to change the objective function when either True Positive Rates (TPR) or True Negative Rates (TNR) are more important. Namely, the objective function in (17) optimises correct classifications, minimising both false positives and false negatives. However, in certain contexts minimising one of the two is far more important than the other. For example, when diagnosing someone with cancer, a false positive will likely prompt a doctor to test the patient more thoroughly, which will eventually conclude that the patient is healthy nonetheless. This may be a bit of a hassle for the patient temporarily. A false negative, however, can have detrimental effects on the long-term health of the patient. As such, it would be best to minimise only the false negatives in this case. We do this by, in (17), interchanging the objective function with (14) and adding the constraint in (15).[8] In other cases, when we want to maximise the TPR, one changes all I_+ and B_+ into I_- and B_- in these two equations, and vice versa.

$$\max \quad \sum_{i \in I_+} \sum_{b \in B_+} \mathrm{c}_b^i \tag{14}$$

$$\sum_{i \in I_-} \sum_{b \in B_-} \mathrm{c}_b^i \geq \lceil (0.95)|I_-| \rceil \ . \tag{15}$$

Moreover, a corrective constant C in the objective function can account for dataset imbalance. When the data has far more positive observations, the model is not given a lot of incentive to identify the patterns that distinguish a negative from positive classification. Rather it will search for features that define the dominant classification, which is not exactly the same. As such, by adding this corrective constant, the model puts more weight into the samples with the 'recessive' y-value. This balances the incentive for the algorithm to find patterns in each class and hence should improve the predictive performance of the algorithm, following this rationale. To correct for the imbalance of the dataset, we use the following value of C:

$$C = \frac{\%\ Positive\ Observations}{\%\ Negative\ Observations}\ , \tag{16}$$

which equals 1 when the dataset is balanced.

Finally, we add all these additional restrictions to the integer programming formulation to arrive at an improved version:

[8] Note that Günlük et al. (2018) mistakenly used "1-0.95" instead of "0.95" in their restriction.

$$
\begin{aligned}
\max \quad & \sum_{i\in I_+}\sum_{b\in B_+} \mathrm{c}_b^i + \mathrm{C}\sum_{i\in I_-}\sum_{b\in B_-} \mathrm{c}_b^i \\
\text{s.t.} \quad & \sum_{g\in G} v_g^k = 1 && \forall k \in K \\
& z_j^k \le v_g^k && \forall j \in J,\ \forall k \in K \\
& \sum_{b\in B^L(k)} c_b^i \le L(i,k) && \forall i \in I,\ \forall k \in K \\
& \sum_{b\in B^R(k)} c_b^i \le R(i,k) && \forall i \in I,\ \forall k \in K \\
& z_j^k(g) = v_g^k && \forall k \in K^{Sym},\ \forall g \in G \\
& z_j^k \ge z_{j+1}^k - w_g^k && \forall j,\ j+1 \in J(g) \\
& z_j^k \ge z_{j+1}^k - (1 - w_g^k) && \forall j,\ j-1 \in J(g) \\
& w_g^k \in \{0,1\} && \forall k \in K,\ \forall g \in G^{Num} \\
& c_b^i, v_g^k \in [0,1] && \forall k \in K,\ \forall g \in G,\ \forall i \in I,\ \forall b \in B \\
& z_j^k \in [0,1] && \forall k \in K^{Leaves},\ \forall j \in J \\
& z_j^k \in \{0,1\} && \forall k \in K \setminus K^{Leaves},\ \forall j \in J,
\end{aligned}
\tag{17}
$$

which we use in the optimisation in step (iii) to reproduce the results from Günlük et al. (2018).[9]

3.4 Extending the Framework

In the sections above, we have presented a five-step outline to construct an ODT. This technique and its improvements were outlined by Günlük et al. (2018). We will now propose extensions and improvements to their techniques by examining our framework at each step to increase its effectiveness.

3.4.1 Dataset Assessment

First, we look at the actions undertaken in step (i), namely the formatting and assessment of the data. We investigate some a priori assessment methods of a dataset's suitability for the ODT technique. Namely, some datasets lend themselves better to forecasting than others. Therefore, the highest obtainable accuracy is not only dependent on the optimisation technique but also on the dataset used.

Let's illustrate this by an example: consider a sample $S_i = \{(g_1^i, \ldots, g_t^i), y^i\}\ \forall i \in N$ that is completely white noise, e.g. Gaussian noise: $S_i \sim N(\vec{\mu}, \Sigma)\ \forall i \in N$. When applying ODT on this data, the resulting decision tree will always perform poorly. This is because the algorithm tries

[9] Note that the TPR or TNR are not included in formulation 17. This is because those are only relevant in specific contexts.

to find patterns in a training set which do not exist because every $\vec{y}$ and X are independent. Then it tries to classify other completely random data, which does not adhere to the spurious patterns found in the training set. This is an extreme example but shows that certain datasets are better for ODT than others. Hence, we want to know how we can assess the suitability of datasets for ODT a priori. For this we have the following two hypotheses:

1. *Datasets with a high sparsity of the realisation matrix (A) will perform poorly on ODT.*
2. *Datasets whose realisation matrix (A) exhibits multicollinearity are less suitable for ODT optimisation.*

Hypothesis 1 is based on the fact that this sparsity is related to the construction of the tree and bucketing of features. Namely, data comes in the form of a $\{\vec{x} = (g_1^i, \ldots, g_t^i) : i \in 1, 2, \ldots, N\}$. These are vectors of length $|G|$, the number of groups. We transform this into a binary array of length d, the number of features. So, when there are relatively many features per group, the realisation vector per i or the realisation matrix for all i will have a higher sparsity. We will investigate how this relates to accuracy and computational efficiency. If it has a negative effect for example, we can advise to bucket similar features to reduce the number of features.

The second hypothesis stems from the fact that when the realisation matrix A exhibits multicollinearity, the algorithm has less information to adjust itself. We can assess multicollinearity by looking at the rank of the realisation matrix A. This provides information about how many features are linearly independent of others. That is, whether there exist features that are a linear combination of other features, suggesting the redundancy of the former. Note that because we consider categorical data, we have that the last feature of each group is linearly dependent on the other features from its group. This is because one can only have a single realised feature per group. However, we cannot omit this feature from the model because this would impede the branching possibilities per group. Taking the above into account, we obtain a formula for the number of redundant features per group:

$$\#\text{ Redundant Features} = |J| - (|G| - 1) - Rank(A), \tag{18}$$

where redundant features can be features with no observations or that can be expressed as a linear combination of other features. This equation can indicate whether a database has redundant features, as $|G|$ and $|J|$ are known and $Rank(A)$ can be computed. There were quite a few datasets with these redundant features employed by Günlük et al. (2018). Also, note that when the number of training samples is less than the rank of A, this will give errors in the optimisation when we apply linear relaxation.

3.4.2 Feature Selection

Moreover, feature selection can be performed a priori to improve our ODT framework. Good feature selection, namely, reduces computation time, the risk of overfitting and can increase the accuracy of a machine learning model, such as ODT in our case (Chandrashekar & Sahin, 2014).

These benefits arise because when we optimise a model with fewer features, the number of v_g^k and z_j^k variables constructed is reduced, hence the size of the MILP problem is smaller and thus easier to solve. Also, having fewer features means that the tree's ability to mould itself too heavily into the training set is diminished, and hence overfitting becomes less problematic. These two positive effects can lead to a higher classification accuracy. However, we must be aware of the fact that poorly executed feature selection can lead to the deletion of relevant variables, which could have detrimental effects on the classification accuracy and the validity of the constructed tree.

Feature selection is the process of choosing which features to include in your analysis. As such, we are interested in identifying the most relevant features for the algorithm. Those features have high predictive power on the dependent variable. However, the data might well be cluttered with unrelated variables or a seemingly unrelated variable in the data could have high predictive power. Conversely, some con-founders could be excluded, which reduces the predictive power of the algorithm. These difficulties arise because the relations between the variables are not known a priori. In fact, the salient aim is to find the patterns between the features and the dependent variable. As such, omitting variables based on assumptions and hypotheses a priori defeats the purpose of the algorithm. Therefore, we reckon that the most suitable feature selection strategy is a prudent one: only omit groups of features when multiple metrics say that they are irrelevant, i.e. statistically independent. This is related to the backward subset selection method for feature selection.

We have identified three possible techniques to evaluate the importance of features: a Chi-squared test, a Gini Impurity Reduction metric and an analysis of correlations. The first of which is a univariate technique, which looks at each feature in isolation, and the latter two multivariate, where the interdependencies between features are taken into account to calculate each feature's importance.

Firstly, we consider the Chi-squared test. For this, we assume that the realisations of each feature are normally distributed and hence we can use the Chi-squared test statistic to test the independence:

$$\chi^2_{c,j} = \sum_{i=1}^{n} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}} \quad \forall j \in J,$$

where $O_{i,j}$ is the observed realisation of feature j for sample i, $E_{i,j}$ the expected value given that the feature is normally and independently distributed, and c the degree of freedom of the Chi-squared statistic. A high score on this test indicates a significant relationship with the dependent variable. Hence, these should be included in the dataset.

Secondly, we consider the Gini Impurity Reduction metric. This is calculated using the Gini Impurity statistic, which calculates the chance of false classifications of a randomly drawn sample from a dataset, based on the proportions of observations:

$$\mathrm{G}(j) = \sum_{i=1}^{n} p_{i,y}(1 - p_{i,y}) \quad \forall j \in J,$$

where $p_{i,y}$ is the probability that a randomly drawn observation i falls into classification category y. In our case, we have two classification categories and, as such, $p_{i,y}$ is 0.5 for balanced datasets. However, we have more data at our disposal than the observed classification, we have far more ways to group the samples, namely by all their features. We can investigate how much the Gini Impurity decreases when we split the dataset in two for either having a feature or not. This percentage drop is the Gini Impurity Reduction Metric. The features that yield the highest reduction can be deemed as the most relevant.

The third technique considers the correlations between all the features and the dependent variable. As such, we construct the correlation matrix and investigate which features have a high correlation with y and can thus be deemed more relevant in the analysis.

$$Cov(X) = E[(X - E[X])^2],$$

$$Corr(X) = (diag(Cov(X)))^{-\frac{1}{2}} \times Cov(X) \times (diag(Cov(X))^{-\frac{1}{2}},$$

where $X = [A, \vec{y}]$. Note that zero correlation is a necessary condition for statistical independence but not a sufficient one. Hence, we can only deduce from this technique that certain features should be included and others potentially not.

Given our prudent approach, we are interested in the variables that perform poorly on these tests and are thus likely to be independent of the dependent variable. Given the IP formulation of our decision tree, we cannot omit single features because then samples may not have a realisation for the corresponding group, whereas the algorithm needs this in order to be executed. Hence, we are interested in groups of which all features are independent of $\vec{y}$. Practically, this also also makes more sense because when a particular group of attributes affects an outcome, not every instance of it has to. When hair colour has an effect on getting hired for a job, being blond for example may be highly relevant, but the other hair colours not. A solution to this could be to create a new feature called 'non-blond' that is 1 when a person is not blond, and 0 otherwise. However, because of the interdependence of the features and the model's purpose of discovering these relations with the model, we recommend to not alter too many variables a priori. However, when every feature in a group scores badly on all three of the aforementioned tests, omission of the group would likely not deteriorate the classification accuracy. Hence, we can omit the group and enjoy all the benefits of having fewer variables in our model.

3.4.3 Considering Tree Topology

In the second step, devising different labelling of leaf nodes or having other tree topologies is not much advantageous for our framework. We argue that the former is equivalent to the latter and that the latter is often not beneficial.

Firstly, alterations to the leaves labelling do not provide potentially better trees given a

topology, but rather change the tree topology. If, for example, a decision node leads to two leaves with positive labelling, the decision at that node is redundant: the sample will always end up with a positive label. Hence, you could interchange the subtree of the decision node and its two leaves with a single positively labelled leave, as this is practically the same tree. As such, tweaking the labelling changes the tree topology, for trees with only paired leaves. However, labelling tweaking is also not the best strategy to pursue, when this is not the case. This is because a tree with different labelling at leaf b can also be constructed by increasing the tree topology and letting the algorithm allocate the optimal labelling. That is, we add two leaves as children to leaf b, one positively labelled and one negatively. Then, the optimisation can assign a test at the decision node adjacent to the leaves that every sample is directed to the new positive label. As such, the tree is practically transformed into a tree with a different labelling. Whether this different labelling is advantageous for a certain dataset can better be assessed by the optimisation algorithm than by ourselves a priori. Hence, we conclude that altering the labelling is not a worthwhile endeavour, but one should increase the tree topology instead.

Furthermore, this reasoning concerning trees and their subtrees extends to choosing the tree topology. In theory, a more extensive tree topology with full nodes, i.e. each leaf is at the same depth like the *Depth 3* tree, has all smaller trees as its subtrees. For example, the *Depth 3* tree in Figure 3 can be transformed into the *Depth 2.5* and *Depth 2* tree when assigning certain tests in the optimisation. Specifically, if tests are assigned such that decision node 6 directs every sample to leaf 5 and decision node 7 every sample to leaf 8, the *Depth 3* tree can practically be viewed as a *Depth 2.5* tree. This implies that a larger tree topology has a strictly higher training accuracy than any of its subtrees. As such, the *Depth 3* tree has a strictly higher training accuracy than the *Depth2.5* and *Depth 2* trees for the topologies in Figure 3. This suggests that we should choose the biggest topology possible. However, overfitting on the training sample becomes problematic, i.e. a high training accuracy and a subsequent low classification accuracy. Namely, a larger tree is more sensitive to the input data and starts to recognize spurious relations, whereas we are mainly interested in the classification accuracy. Also, the computational time increases drastically.

All in all, choosing different labelling can better be achieved by increasing the tree topology in the ODT framework. Also, one should make a trade-off between the advantages of a more extensive tree topology, i.e. higher achievable accuracy and more different possible 'eventual topologies', and its disadvantages, i.e. computational complexity and overfitting. The *Depth 3* tree seems to be the best middle ground, as by the results of Günlük et al. (2018).

3.4.4 K-Fold Cross-Validation

Now, we look at an improvement to the accuracy calculation in the fourth step of the ODT framework. We consider a technique that increases the validity of a model's calculated accuracy. This is important because this metric is used for model assessment and, as such, determines whether we discard or accept the model.

First, let's look at how we calculate the accuracy of a single trained DT, which is also called the generalisation ability or the classification accuracy of the model. We calculate this metric by testing how well the model can classify new data points. However, as this is a supervised machine learning model, we need to choose beforehand which samples we use for training and which for testing, i.e. specify the training and the testing set. This split can be quite important, as the algorithm finds different ODTs for different training sets and hence can construct different ODTs with different aptitudes with the same data. So, the problem of the validity of the accuracy metric is not in its calculation but rather how we partition our dataset for training and testing.

Hence, we should account for the possible different ways of partitioning to ensure the validity of this generalisation apptitude metric. Günlük et al. (2018) employed a rather inefficient manner to handle these issues, namely random subsampling (Kohavi et al., 1995). They partitioned the data into a training and testing set with the rule $n_{train} = min(600, 0.9n)$ and the remaining samples constitute the testing set, which they used that to construct and test an ODT. They repeated this three times with a randomly shuffled dataset. Then, they averaged the classification accuracy of the three trees. This way, they increased the validity of the model produced, as 'unlucky' ways of partitioning can be crowded out by better ones. However, random subsampling is an inefficient technique because samples in the training partition are not independent of those in the testing one (Kohavi et al., 1995).

Algorithm 1 K-fold cross-validation

- *Shuffle the dataset S randomly*
- *Partition S into K folds $P = \left[P_1, \ldots, P_K\right]$ **

for $k \in K$ **do**

- *Train a classifier ϕ_k on $T = S \setminus P_k$*
- *Test the classifier ϕ_k on fold P_k*
- *Retain the classification accuracy $1 - \varepsilon(\phi_k)$ and discard the model*

end

- *Summarise the skill of the model:* $1 - \varepsilon_K = \frac{1}{K} \sum_{k=0}^{K} (1 - \varepsilon(\phi_k))$

** In theory, K can be any integer. In practice, 5 and 10 are chosen the most because they have a good trade-off between the variance and bias of the classifier error.*

An accuracy computation technique that solves this issue is K-fold cross-validation, see Algorithm 1. This technique trains and tests on all samples, and as such resolves the independence problem with random subsampling. This method uniformly at random partitions the dataset into K folds, and K times trains a tree on K-1 folds and test on the one group not used for training, see Algorithm 1. This way, every sample is used in constructing the tree and in testing.

This algorithm is better than random subsampling because its classifier error has lower variance and bias (Rodriguez, Perez, & Lozano, 2009). The error $\varepsilon(\phi)$ of random subsampling is biased because of the dependence issue, whereas the K-fold cross-validation classifier error estimator is unbiased when the partitioning is done stably. That is, each partition is a good representation of the original dataset (Kohavi et al., 1995). The variance of the estimator is also reduced significantly, due to the ten independent constructed trees. However, most real-life datasets are not stable and we can analyse the variance of the K-fold cross-validation error by:

$$Var[\hat{\varepsilon}_K] = Var[\varepsilon] + Var[\delta_k],$$

where ε is the variance inherent to the dataset and δ_K the variance caused by the partitioning. The former is irreducible and part of all estimated classifiers, whereas we want to minimise the latter with our cross-validation technique (Rodriguez et al., 2009). Rodriquez et al. (2009) found that the variance of the estimators is largely due to the variation in the training sets between the folds, instead of the variation in the testing set.

Rodriquez (2009) advised using $K = 5$ or $K = 10$ because it has a smaller bias and is less computationally intensive than $K = n$. Kohavi et al. (1995) advised $K = 10$ for model validation and to use stratified partitions to increase the steadiness of the partitions.

3.4.5 Model Validity Assessment

In the last step of the framework, we check whether the tree is accurate enough and, based on this, decide whether to accept the tree. This description leaves quantitatively minded people quite unsatisfied because what does "accurate enough" precisely mean?

A key aspect of what makes a decision tree valid is whether it has good generalisation ability. That is, it can classify new data without making too many errors. In the domain of machine learning, a benchmark on whether to deem a model good, is to compare its classification accuracy with the currently conventional assessment technique for the dataset. So, for example, when a decision tree can diagnose breast cancer more accurately than the average cardiologist, we can deem the tree accurate and accept it. This benchmark is thus highly context-dependent and would require extensive research into the application domain.

In scientific research, on the other hand, one often works with significance levels of 5 percent. That is, we consider something accurate when there is at most a five per cent chance of yielding false positives. Note that our classification accuracy rate also includes false negatives, hence it is higher than only this five per cent accuracy level. Consequently, we choose the latter benchmark as it is conservative and the former falls outside of the scope of this research.

3.5 Discussing Suitability for Economics

In the following, we discuss the suitability of the ODT Framework for economic datasets. Apart from the benefits that the computational and validity improvements from the previous sections have in any field, we discuss issues specifically related to economic contexts.

3.5.1 Validity over Time

When we have decided in step (v) that a model is accurate, it is not certain that this validity assessment will hold true over time (Risselada, Verhoef, & Bijmolt, 2010). Namely, circumstances can change, which can cause a deterioration of the performance of the model on new data. This raises the question of what the "expiration date" of an ODT is. We want to know how long a DT can be used before it needs revision or tuning, or even be drastically replaced by a new one. This is relevant because it determines whether a company that wants to incorporate ODT in their operations, can solicit a data scientist every so often or should have one permanently on its payroll to ensure the quality and usability of the algorithm.

This expiry date phenomenon is especially relevant in economics, where we often consider many variables exogenous or consider a variable "ceteris paribus". The economic environment can change, for example by an exogenous shock, like a pandemic or a change in monetary policy by the ECB. When such an exogenous shock happens, one should reexamine the model. What entails an external shock for a model is dependent on the dataset used and what variables are considered endogenous in the model.

The model can change in two ways: the current model just needs new data to be tweaked, for example, a couple of features have changed. Or the model is significantly changed, one needs to erase certain characteristics and add new ones, choose a new topology etc. Because exogenous shocks often happen unpredictably and sometimes quite often in economic settings, one should re-examine the DT frequently.

3.5.2 Interpretability

Furthermore, we investigate the interpretability of the resulting optimal decision trees. We look at the soundness of the tests at the decision branches and in what way tree can be interpreted by a professional in the field related to the data used, such as the lending market for home mortgages for the FICO data.

The level of interpretability of a tree is dependent on the data employed, namely the decisions made by a tree should adhere to some logic in the linked field. In the case of the FICO data, the tests at the decision nodes need to be related to the financial trustworthiness of the applicant for the bank to be able to provide a sound explanation for denial or acceptance of a loan application. Namely, a bank needs to substantiate its decisions to its customer and the outside world, regardless of whether that decision is made by a human or a machine. Therefore, we can look into tweaking the model in such a way to improve the understanding and soundness of the classifications. For example, by deleting illogical tests or tweaking some restrictions. This will lead to a decrease in training accuracy but not necessarily in testing accuracy, but it will likely be affected when one alters many tests.

Considering the aforementioned, we propose to extend the fifth step of the framework by an investigation of the validity and logic of the test at each decision node. Herein, a trade-off needs to be made between accuracy and interpretability.

4 Results

In the following, we present an application of the theoretical framework on nine datasets.[10] With this, we investigate the effectiveness of the methods and their improvements as mentioned in Section 3.1 to 3.3. For those, we compare our results with those of Günlük et al. (2018). Afterwards, we examine the effectiveness of the extensions proposed in section 3.4. Also, we illustrate our discussion in 3.5 with an example.

4.1 Description Datasets

We work with nine datasets for which we construct Optimal Decision Trees, see Table 1. They are sourced from the UCU machine learning repository (Dua & Graff, 2017), LIBSVM (Chang & Lin, 2011) and the FICO Explainable Machine Learning Challenge (Fair Isaac Corporation, 2018). We use these datasets because Günlük et al. (2018) used them as well, and we want to compare our results to theirs.[11]

Table 1
Descriptive statistics of the used datasets

Dataset	# Samples	% Positive	# Features	# Groups
bc	683	65	90	9
krkp	3196	52	73	36
mush	8124	52	116	21
ttt	958	65	27	9
monks-1	432	50	17	6
votes	435	61	48	16
heart	267	79	44	22
student	395	67	114	30
heloc	2052	38	230	23

Compared to Günlük et al. (2018), we altered the *heloc* and *mush* dataset. For the former, we first edited the data to exclude observations that had missing values, reducing the sample size from 9871 to 2052. For the latter, we removed ten features from each $J(g)$ which did not have a single realisation, which also caused one entire group to be omitted. We also note that the *heloc* and *heart* dataset are quite imbalanced as for the percentage of positives.

4.2 Improved Formulation

First of all, we present our computational results as for the improved formulation and investigate whether the improvements in section 3.2 increase performance and computational efficiency. In other words, we compare the added components of (17) with the simple formulation in (7). Unless specified otherwise, we employed $n_{train} = min(600, 0.9n)$, $n_{test} = n - n_{train}$, a ten-minute time limit and random subsampling with three repeats. To have a feeling of what the numbers mean, see Figure 4 for a printed tree of the *mush* dataset.

[10] The python code and the datasets can be found at: https://github.com/mart505/ODT/.

[11] We did not use the a1a dataset used by Günlük et al. (2018), as this was too time-consuming given our Python implementation.

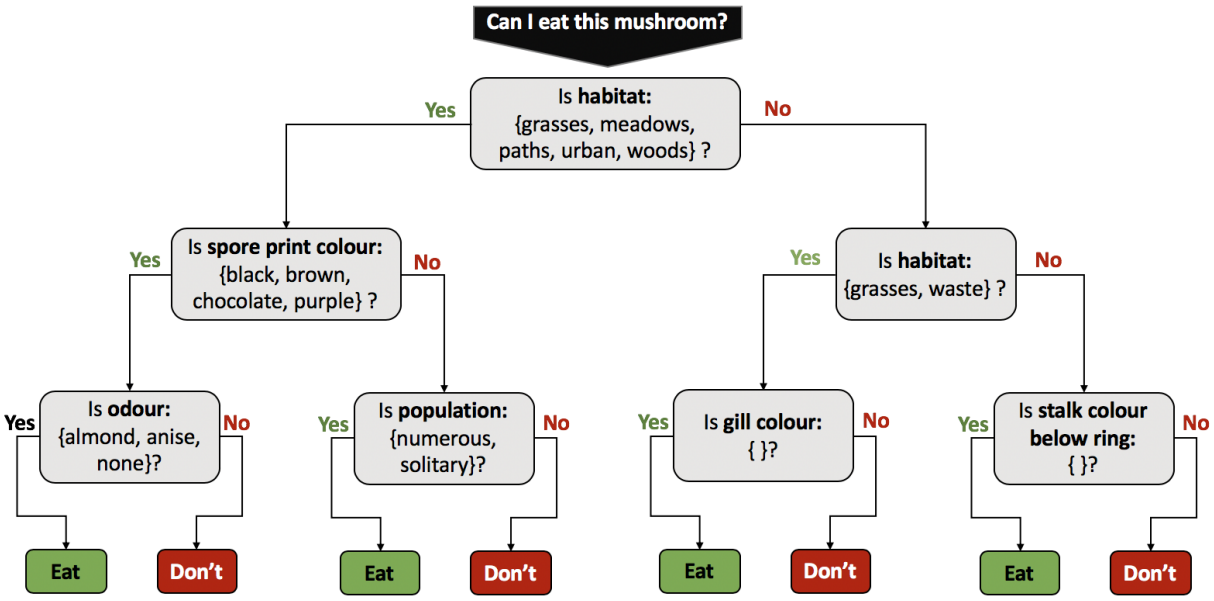

Figure 4. ODT for mush dataset with depth 3 topology and classification accuracy of 99.60 percent

Table 2
Average ODT solving time and nodes solved using different combination of techniques, using random subsampling with five repeats and 200 observations

Data-set	Nothing		No Anchor		No Relax		No Strength		All	
	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes
bc	82.24	3624	450.03	11565	32.12	770	450.65	11674	341.45	10361
krkp	494.37	2283	430.17	9883	183.11	6978	388.36	2774	124.20	2061
mush	2.91	0	4.68	3	6.15	28	5.31	1	4.15	5
ttt	450.04	12332	386.13	14731	182.24	11978	341.20	6647	41.96	1800

Firstly, we examine the effects of linear relaxation, strengthening, and anchoring. In Table 2, we see that not all of these extensions to the original IP problem yield an upgrade to every dataset. For example, the mushroom data (*mush*) has worse computational performance with the improvements. However, we observe that combining all the extensions generally yields faster solving times. This is in line with the theory discussed in Section 3.3. Furthermore, we investigate the effects of the improvements regarding numerical features and the dataset imbalance correction factor in Table 3 and 4 respectively. We see that numerical features adjustments do not make that many changes to the classification accuracy, but do reduce the solving time for datasets with many observations. Also, it is notable that the training accuracy drops for most trees, whereas the classification accuracy increases. Possibly, the restrictions for numerical features allow for less overfitting, as there are fewer options to branch per group with these constraints. The imbalance correction factor, however, deteriorates both the training and classification accuracies significantly, hence we do not include this in further optimisations.

These findings are in line with Günlük et al. (2018), apart from the correction factor C due to the absence of experiments for this in their paper. Hence, we continue with the improved formulation in (17) for the rest of our optimisations without the imbalance correction factor.

Table 3
ODT Results with (n) and without (c) restrictions for numerical features, using random subsampling with five repeats, time in seconds and test and train in percentage

Dataset	n/c	Depth 2		Depth 2.5		Depth 3		Imbalanced	
		test	**train**	**test**	**train**	**test**	**train**	**test**	**train**
bc	n	91.97	95.00	92.77	95.56	93.17	95.61	90.76	95.83
	c	93.57	96.89	91.16	98.06	93.17	97.83	92.77	98.22
heloc	n	66.20	67.28	67.60	67.33	66.22	66.22	66.67	66.67
	c	66.23	70.72	64.20	71.39	66.27	70.94	65.60	71.17
student	n	73.33	70.05	74.17	71.27	71.67	71.08	74.17	70.80
	c	70.83	75.31	70.83	76.62	69.17	76.24	69.17	76.43
		time	**nodes**	**time**	**nodes**	**time**	**nodes**	**time**	**nodes**
bc	n	20	35	270	1269	491	1.320	600	864
	c	10	70	520	4942	600	2064	600	1924
heloc	n	210	784	600	335	600	15	600	67
	c	268	697	600	399	600	460	600	93
student	n	44	46	600	1848	600	1145	600	620
	c	52	229	600	1906	600	1807	600	744

Table 4
ODT Results with and without imbalance correction factor C for the imbalanced datasets, using random subsampling with five repeats, time in seconds and test and train in percentage

Dataset	C	Depth 2		Depth 2.5		Depth 3		Imbalanced	
		test	**train**	**test**	**train**	**test**	**train**	**test**	**train**
bc	1.86	92.17	95.08	92.17	95.50	92.17	95.50	93.37	95.75
	1	92.17	95.08	92.17	95.50	92.77	95.50	93.98	95.75
ttt	1.86	68.30	69.83	70.53	72.83	75.42	78.67	74.16	74.83
	1	72.25	65.78	77.75	72.77	78.75	74.58	82.17	78.91
votes	1.56	95.45	95.91	94.32	96.68	95.45	96.93	95.45	97.19
	1	97.73	95.91	95.45	96.68	95.45	92.07	98.86	97.57
heart	3.76	62.96	71.67	71.60	76.39	75.31	77.78	70.37	79.86
	1	79.01	79.44	71.60	82.78	70.37	82.92	67.90	83.75
heloc	0.61	65.27	67.22	62.87	66.39	61.90	65.17	61.27	65.78
	1	65.47	68.83	64.27	68.72	63.93	67.56	64.90	68.67
student	2.03	57.50	61.13	61.67	66.01	51.67	65.63	60.00	67.70
	1	69.17	70.42	61.67	71.64	68.33	71.64	66.67	71.74
		time	**nodes**	**time**	**nodes**	**time**	**nodes**	**time**	**nodes**
bc	1.86	18	17	356	1471	600	798	600	824
	1	21	32	368	1484	600	698	600	774
ttt	1.86	10	0	231	1849	430	4525	600	2239
	1	9	0	196	1607	354	5072	600	2400
votes	1.56	4	0	76	2215	182	2562	600	8096
	1	4	0	81	2244	169	2492	600	7963
heart	3.76	5	0	66	1942	236	7915	600	5756
	1	5	0	65	1981	600	7392	600	5612
heloc	0.61	331	250	600	356	600	0	600	48
	1	252	1223	600	178	600	18	600	102
student	2.03	60	64	600	1710	600	818	600	681
	1	47	50	600	1635	600	993	600	709

With the accuracy and computational efficiency confirmed, we examine the classification and computational performance of this improved formulation. When looking at Table 5, we see the performance of the IP formulation in (17), and their associated solving time in Table 6.[12] We see that the performance is highly dependent on the dataset employed. For the *bc*, *krkp*, *votes* and *monks-1* datasets, the algorithm gives a highly accurate tree with classification accuracies of above 90 percent. The *heloc*, *ttt*, *heart* and *student* datasets each yield mediocre accuracy of each below 80 percent, and heloc even as low as 68 percent. An algorithm that is only correct for two-thirds of the time, cannot be used in practice. We can also clearly see in Table 5 that the *Depth 3* tree most often provides the best classification accuracy, followed by the *imbalanced* tree. The *Depth 2.5* or *2* trees, on the other hand, rarely yield the most accurate tree. Also, although the heloc dataset exhibits decreasing training accuracy with increases tree size, generally the training accuracy improves with tree size. This is in line with our reasoning in Section 3.4, that bigger trees always have at least the same training accuracy. The exception of heloc can be explained by the fact that it often exceeded the optimisation time limit. Hence, it is logical that the training accuracies get worse, as the software did not optimise the DTs yet fully. Next to that, the fact that the classification accuracy generally improves as well suggests that this increase in tree size does not cause overfitting with these datasets.

Table 5
*ODT classifier accuracy using random subsampling with five repeats and a ten-minute time limit, * indicating an exceeder and bold the best classification accuracy per dataset, test and train in percentage*

Dataset	**Depth 2**		**Depth 2.5**		**Depth 3**		**Imbalanced**	
	test	**train**	**test**	**train**	**test**	**train**	**test**	**train**
bc	94.22	92.80	95.42	94.10	**95.66**	94.00	94.94*	94.97*
heloc	67.80	70.13	67.52*	70.23*	**68.03***	69.37*	67.62*	69.80*
krkp	86.71	87.83	88.44	89.77	**92.58**	93.53	92.50*	93.67*
mush	99.00	99.43	99.37	99.93	99.57	99.97	**99.68**	100
ttt	69.16	70.97	71.96	75.03	73.30*	77.50*	**76.65***	78.40*
monks-1	71.36	78.25	76.82	83.81	81.82	89.43	**99.55**	99.79
votes	96.36	96.06	**96.36***	96.47*	95.91*	96.93*	95.91*	97.34*
heart	73.33	80.08	71.11	87.50	**74.07**	82.58	73.33*	83.08*
student	68.00	70.48	68.00	72.00	**68.50**	71.72	67.00	72.12

Table 6
Average solving time for ODT in Table 5

Dataset	**Depth 2**	**Depth 2.5**	**Depth 3**	**Imbalanced**
bc	21.68	164.94	600.00	600.00
heloc	187.05	600.00	600.00	600.00
krkp	21.42	321.01	591.36	600.00
mush	21.11	228.31	263.69	39.92
ttt	13.50	155.06	600.00	600.00
monks-1	2.30	14.61	24.28	48.38
votes	2.93	600.00	600.00	600.00
heart	7.43	71.76	587.61	600.00
student	0.36	557.72	600.00	600.00

[12]Note, we set C=1 for all datasets because of the results in Table 4.

4.3 Extended Framework

In the following, we present numerical results regarding the suggested improvements in Section 3.4, namely data assessment, feature selection and K-fold cross-validation.

First, we consider the data assessment and look at the sparsity and rank of the observation matrices. In Table 7, we see the sparsity and the rank of the realisation matrix A for each dataset and the highest obtained classification accuracy. From these data points, we do not see a link between rank or sparsity with classification accuracy, contrary to hypothesis one. Neither do we see a relationship between the 'Rank/Features' percentage and accuracy.[13] The difference between the number of features and the rank can be explained by equation 18 and is illustrated by the reduced row echelon form of the realisation matrix in Appendix A. However, we do not observe a link between the number of redundant features and best testing accuracy.

Table 7
Descriptive statistics on dataset, their realization matrix and ODT classifier performance, Rank/Features, Sparisty and Best Test in percentage

Dataset	# Samples	# Features	Rank	Rank/Features	Sparsity	Best Test
bc	683	90	80	89	91.01	95.66
heloc	2052	230	181	79	90.00	68.03
krkp	3196	73	38	52	50.68	92.58
mush	8124	126	86	68	82.55	99.68
ttt	958	27	19	70	66.67	76.65
monks-1	432	17	12	71	64.71	99.55
votes	435	48	33	69	66.67	96.36
heart	267	44	23	52	50.00	74.07
student	395	114	82	72	73.68	68.50

Furthermore, we investigate the relation between rank of the sample realisation matrix and the classification accuracy of the tree, see Table 8. Here, 'normal sampling' is the same technique as we used before, i.e. randomly drawing samples from the data without repetition. With 'full rank sampling', we enforced that we first pick those samples that create a full rank sample realisation matrix before selecting other samples. In this case, for the first 181 samples, we only added them to the sample if they increased the rank of A. After we reached full rank, we choose the samples with 'normal sampling' again.

Table 8
Comparison of ODT classifier accuracy for normal sampling and full rank sampling, with best performance in bold and train and test in percentage

Sample Size	Normal Sampling			Full Rank sampling		
	rank	test	train	rank	test	train
181	125	69.63	**66.60**	181	**76.00**	63.10
200	137	**76.00**	63.70	181	71.00	**66.00**
300	143	**74.00**	64.50	181	73.00	**65.90**
600	155	**68.03**	**69.37**	181	66.00	59.00

[13] Note that this ratio indicates the percentage of redundant features in each dataset.

Table 9
ODT perfomance on the monks-1 dataset trained on various subsets of all groups as a demonstration of feature selection, time in seconds and train and test in percentage

Feature Selection Method	Name	# Groups	Time	Test	Train
normal	monks-1	6	40.00	100.00	100.00
corr	monks-12	1	0.07	64.29	75.90
corr + best	monks-16	2	0.22	80.95	74.10
corr + second best	monks-13	2	0.24	71.00	75.00
corr + top 2	monks-17	3	0.78	78.00	74.62
excluding worst & best	monks-14	4	19.65	100.00	99.74
excluding worst 2	monks-18	4	24.02	76.19	83.85
excluding worst	monks-15	5	3.66	100.00	99.74

Contrary to our hypothesis, we do not see an improvement in classification accuracy with choosing full rank. We even see a decrease in most cases. This is perhaps because full rank sampling ensures that the training sample has information on all features, but not every feature is as important for the DT. So, by forcing the training sample to have information on all, and therefore also on less important features, we omit information on more important features, those that have high predictive power. Following this rationale, when all features are of importance for the prediction, then full rank sampling could improve the performance of ODT.

Furthermore, we explored the effects of feature selection on the solving time and classification accuracy, see Table 9. Here, we employed the feature importance tests as outlined in Section 3.4.2, which are displayed in the Figures in Appendix B, C and D. We consider the *monks-1* dataset because we know that we can construct a tree with 100 percent classification accuracy with the dataset, hence any classification accuracy lower than 100 percent demonstrates a poor feature selection. It is interesting that when one chooses the features only based on the correlation metric, the classification accuracy drops significantly. Perhaps it could be that selecting features based on correlation only, does not take into account that certain combinations of features do correlate with the dependent variable, but we should include the features that have a direct correlation. Omitting the worst group according to both the Gini Impurity Reduction and Chi-squared Metric yields a tree with the same predictive capabilities as with no feature selection, whilst reducing computational time significantly. Remarkably, when you delete the group with the most important feature according to this last criterion, the ODT still yields 100 percent. Feature selection is perhaps more of an art than a science. One does not know which features are significant and having specific combinations of features can yield better effects than others. Therefore a prudent approach is advised.

Moreover, our remarks on tree topology can be illustrated by the ODT for the mush dataset in Figure 4. On the left side of the tree, no matter what features a sample has, it will always obtain a negative classification. Therefore, the tree reduces to a simpler topology, as in Figure 5. This illustrates that tweaking tree topology can also be achieved by the optimisation algorithm itself, where it can choose the optimal tree topology between all the proper subtrees of the topology chosen in step (ii). However, there is a trade-off between this benefit, and the risk of overfitting and computational efficiency loss.

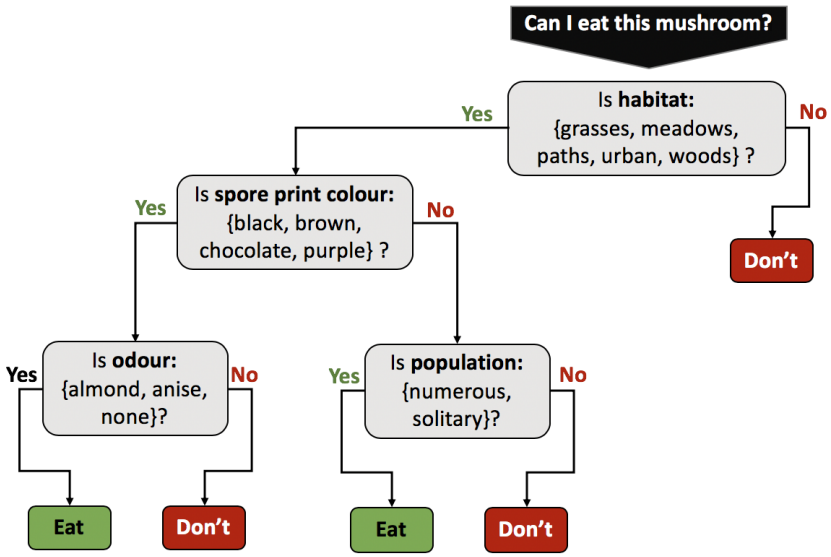

Figure 5. Simplified ODT for mush dataset with depth 3 topology and classification accuracy of 96.60 percent

Table 10
ODT classifier statistics using either 5-fold cross-validation or random subsampling with 5 repeats, avg, min and max in percentage

Dataset	Sampling	Train/Test	Data Stats			
			var	avg	min	max
bc	5-fold	train	0.1588	94.73	94.32	95.25
		test	2.5523	94.29	91.91	95.62
	subsampling	train	0.2016	94.51	93.77	94.87
		test	4.1558	94.45	92.70	97.81
heart	5-fold	train	0.0028	79.40	79.34	79.44
		test	0.0443	79.40	79.25	79.63
	subsampling	train	0.5951	79.72	78.87	80.75
		test	14.7460	75.19	72.22	81.48
votes	5-fold	train	0.0413	96.26	95.98	96.55
		test	2.6424	94.25	91.95	95.40
	subsampling	train	0.1404	96.32	95.98	96.84
		test	4.3599	94.71	91.95	96.55
student	5-fold	train	0.9213	70.95	69.62	72.15
		test	10.0945	66.58	62.03	69.62
	subsampling	train	2.0329	71.39	69.62	73.10
		test	29.9631	66.08	59.49	73.42
ttt	5-fold	train	0.4843	71.24	70.66	72.45
		test	11.9594	67.12	60.94	68.75
	subsampling	train	0.2642	71.28	70.50	71.67
		test	4.2046	66.67	63.54	68.75
heloc	5-fold	train	0.6805	67.02	66.12	67.93
		test	1.5332	66.03	64.20	67.47
	subsampling	train	0.0723	66.28	66.02	66.62
		test	1.0000	65.07	64.07	66.07
monks-1	5-fold	train	0.0755	78.41	77.97	78.61
		test	1.2204	75.23	74.42	77.01
	subsampling	train	0.1008	78.20	77.68	78.55
		test	1.5854	76.09	74.71	78.16

Subsequently, we looked into K-fold cross-validation and its effect on the estimated accuracies, see Table 10. We applied $K = 5$ because this was more computationally efficient than $K = 10$. To compare both sampling methods objectively, we chose $n_{train} = 0.8n$ with five repeats for random subsampling. In the Table, we observe that 5-fold cross-validation estimates outperform those of subsampling as for estimator variance. Certain datasets yield large estimator variance reductions, whilst others exhibit a modest gain. Interestingly, the datasets with large n show an increase in estimator variance, namely mush, heloc, ttt and krkp. Perhaps it could be the case that with larger datasets, the independence inefficiency with random subsampling is less problematic, so the variance of the random subsampling estimator is reduced.

4.4 Illustration Suitability Economics

To illustrate the interpretability of a constructed ODT in an economic context, we investigate one for the heloc dataset. This dataset provides information on loan applications and as such, the tree tries to predict whether or not a new loan application will have a high chance of being repaid. This is called good risk performance, or bad respectively. In Figure 6, we see that the constructed tree is quite straightforward and intuitive to understand. The accuracy is, on the other hand, sub-par as it is far beneath the threshold of 5 percent, as argued in Section 3.5. This would prohibit the tree to be implemented in a professional setting.

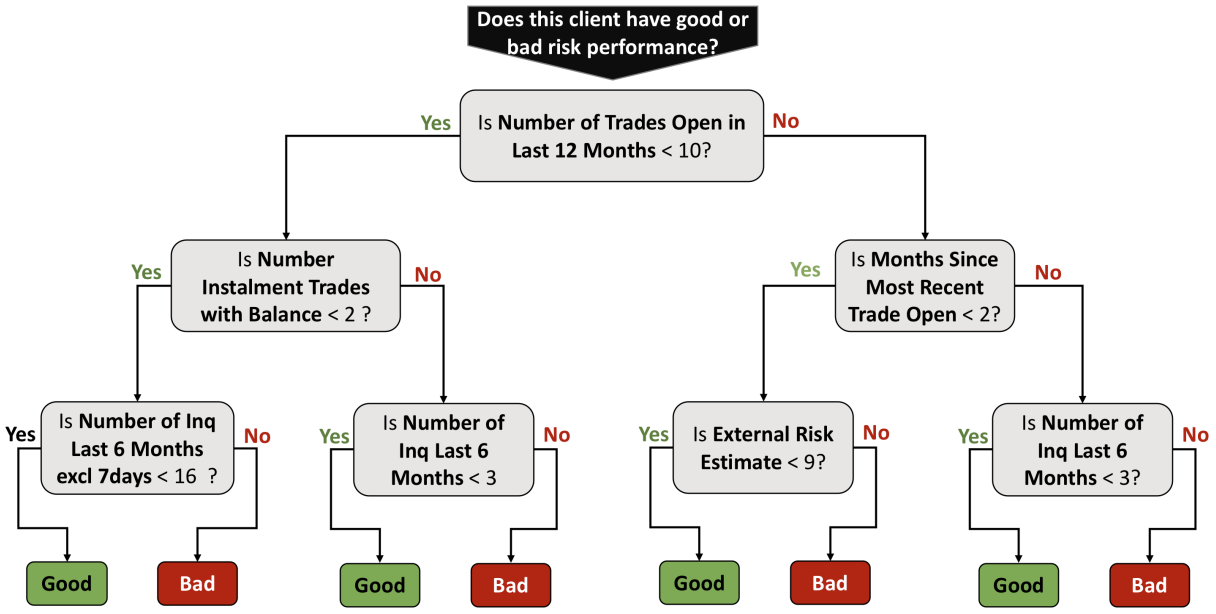

Figure 6. ODT for heloc dataset with depth 3 topology and classification accuracy of 68.00 percent

5 Conclusion and Discussion

In this research, we aimed to answer the following question: *How can we provide accurate decision trees that are suitable in an economic context?*

We outlined a five-step framework to construct optimal decision trees, inspired by (Günlük et al., 2018). Herein, we incorporated their IP formulation and improvements into the framework and proposed extensions: a priori dataset assessment, feature selection and K-fold cross-validation. We also examined tree topology and validity assessment. Furthermore, we discussed issues that arise when applying ODT in an economic setting.

We found that the improved IP formulation for decision trees as proposed by Günlük et al. (2018) is an accurate and computationally efficient method. Only their proposed dataset imbalance corrective factor deteriorated performance and should be excluded. Regarding the dataset assessment, we found that neither sparsity nor multicollinearity of the realisation matrix affects performance, rejecting both our hypotheses for this topic. Furthermore, feature selection can improve computational efficiency significantly, but one has to be prudent with the omission of features as it is not always conspicuous which features are the most important. Subsequently, K-fold cross-validation decreases the variance of the accuracy estimates and outperforms random subsampling in this respect. It, therefore, provides a more valid performance measure for the ODT framework. Considering tree topology, we argued that labelling tweaking is not a beneficial endeavour and one should consider a moderate tree size. Regarding validity assessment, one should use a five percent accuracy level, when one lacks a profound understanding of the application area. Lastly, when applying the framework to economic contexts, one has to be aware that the model needs re-estimation frequently to account for exogenous shocks. Also, one should incorporate an additional step to the framework in which one checks the logic of the tests at each node. Hence, our framework is a great tool to construct accurate decision trees, which are suitable in an economic context.

However, when examining the aptitude of our framework, it is not only the usage the right optimisation technique but also the dataset that determines performance. For example, datasets differ as for data collection quality, i.e. validity issues or measurement errors, and the presence of conspicuous underlying patterns that the algorithm can detect and use to predict. Bad data will produce bad trees, irrespective of the sophistication of the IP formulation used. Therefore, more research into good dataset assessment for ODT would be insightful.

Another point of discussion is the suitability of ODT for continuous or numerical data. The technique is optimised to leverage the categorical aspect of features. Devising a method that takes advantage of the continuous nature of variables should perform better on those datasets. However, when one has both continuous and categorical data, further research can be conducted into which technique is optimal. Also, the choice for the cut-off points for real-valued data in ODT can influence performance. With continuous independent variables, we have applied a ten quantile approach because of the ease of programming, but this is not the only manner to do so. More research on bucketing could lead to a potentially better model.

Next to interpretability, excellent pattern recognition is the key selling point of decision trees. Therefore, they are quite skill-full at classification problems. However, decision trees are not suitable for parameter estimation, which is often of interest in economic contexts. As such, a decision tree provides insights into the interdependencies of variables, but not into the relative importance of variables. We have tried to gain information on this using feature selection techniques, but parameter estimation remains best investigated using econometric methods. Further research could be conducted into which applications ODT could be most useful in economics.

Another issue that will become more relevant with increased usage of techniques such as ODT in society is the question of what to do with morally controversial but statistically significant factors. For example, the sex of a person or their ethnicity could have high predictive power. Should one then exclude these based on principals or include them based on the statistics? A question that falls outside of the scope of this paper, but is highly relevant for professionals implementing machine learning techniques.

Furthermore, a substantial limitation of this research was the lack of time and computing power available. Therefore, we set a ten-minute time limit for every optimisation, which was often exceeded. For example, the *heloc* dataset almost always passed this boundary, and also many others. This caused us to draw conclusions from data of non-optimal trees, which could be erroneous. Hence, we are curious to see the results of an examination with more generous equipment and time.

Also, another limitation was that we only investigated the variance of the ODT classifier and not its bias, due to time constraints. This because one then needs to know the natural error rate of the datasets employed, which is information that we did not have on our datasets. Further research into the bias of the ODT estimator would be interesting as one can then assess the validity of the models more meticulously.

References

Athey, S. (2018). The impact of machine learning on economics. In *The Economics of Artificial Intelligence: An Agenda* (pp. 507–547). University of Chicago Press.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and Regression Trees*. CRC press. doi: https://doi.org/10.1201/9781315139470

Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, *40*(1), 16–28.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*, 27:1–27:27. (Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`)

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

Fair Isaac Corporation. (2018). *Explainable machine learning challenge.* Retrieved 15-05-2020, from `https://community.fico.com/s/explainable-machine-learning-challenge`

Fisher, M. L. (1981). The lagrangian relaxation method for solving integer programming problems. *Management Science*, *27*(1), 1–18.

Günlük, O., Kalagnanam, J., Menickelly, M., & Scheinberg, K. (2018). Optimal decision trees for categorical data via integer programming. Retrieved from `http://www.optimization-online.org/DB_FILE/2018/01/6404.pdf`

Kohavi, R., et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI* (Vol. 14, pp. 1137–1145). Montreal, Canada.

Kotsiantis, S. B. (2013). Decision trees: a recent overview. *Artificial Intelligence Review*, *39*(4), 261–283.

Mullainathan, S., & Spiess, J. (2017). Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, *31*(2), 87–106.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-machine Studies*, *27*(3), 221–234.

Risselada, H., Verhoef, P. C., & Bijmolt, T. H. (2010). Staying power of churn prediction models. *Journal of Interactive Marketing*, *24*(3), 198–208.

Rodriguez, J. D., Perez, A., & Lozano, J. A. (2009). Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(3), 569–575.

Trick, M. (2005). Formulations and reformulations in integer programming. In R. Barták & M. Milano (Eds.), *Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming* (pp. 366–379). Springer, Berlin, Heidelberg.

Varian, H. R. (2014). Big data: New tricks for econometrics. *Journal of Economic Perspectives*, *28*(2), 3–28.

Zurada, J. (2010). Could decision trees improve the classification accuracy and interpretability of loan granting decisions? In *2010 43rd Hawaii International Conference on System Sciences* (pp. 1–9).

A Realisation Matrix in RREF

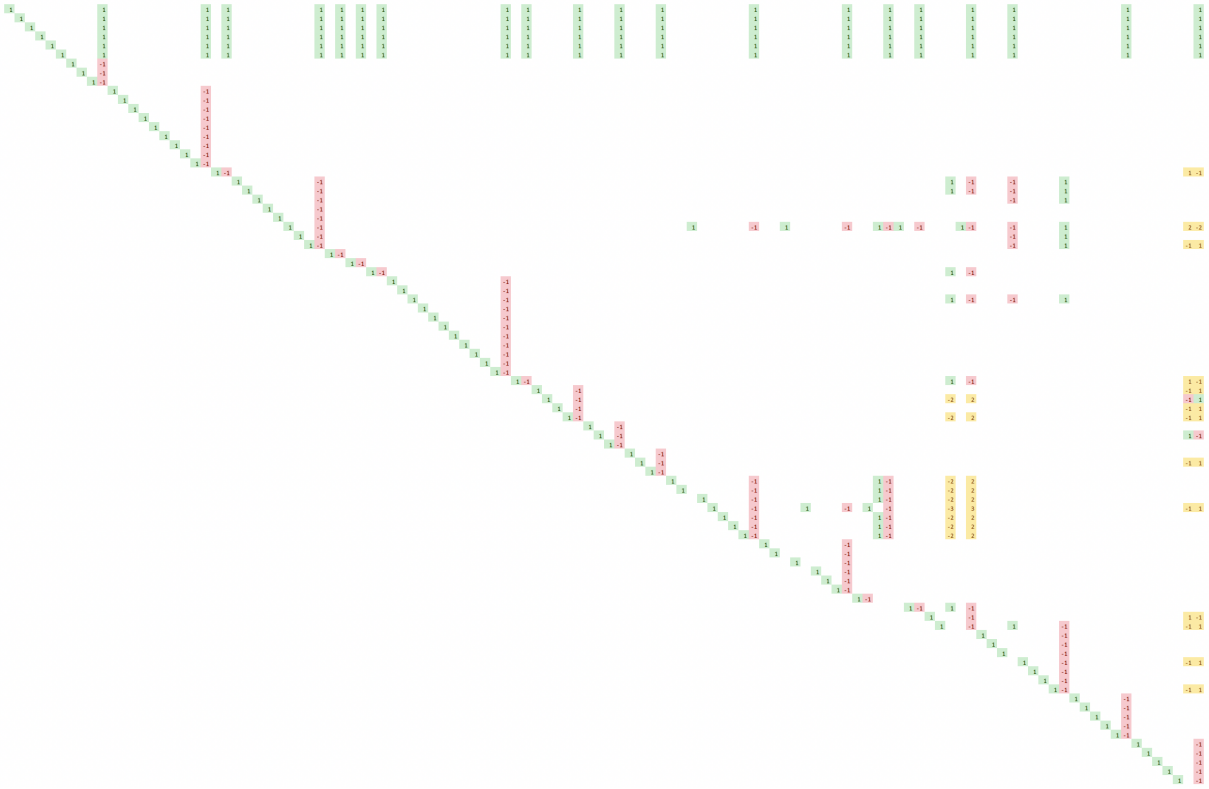

Figure 7. Reduced row echelon form of the realisation matrix for cleaned mushroom (mush) dataset; green, red and yellow are 1,-1 and other nonzero numbers respectively

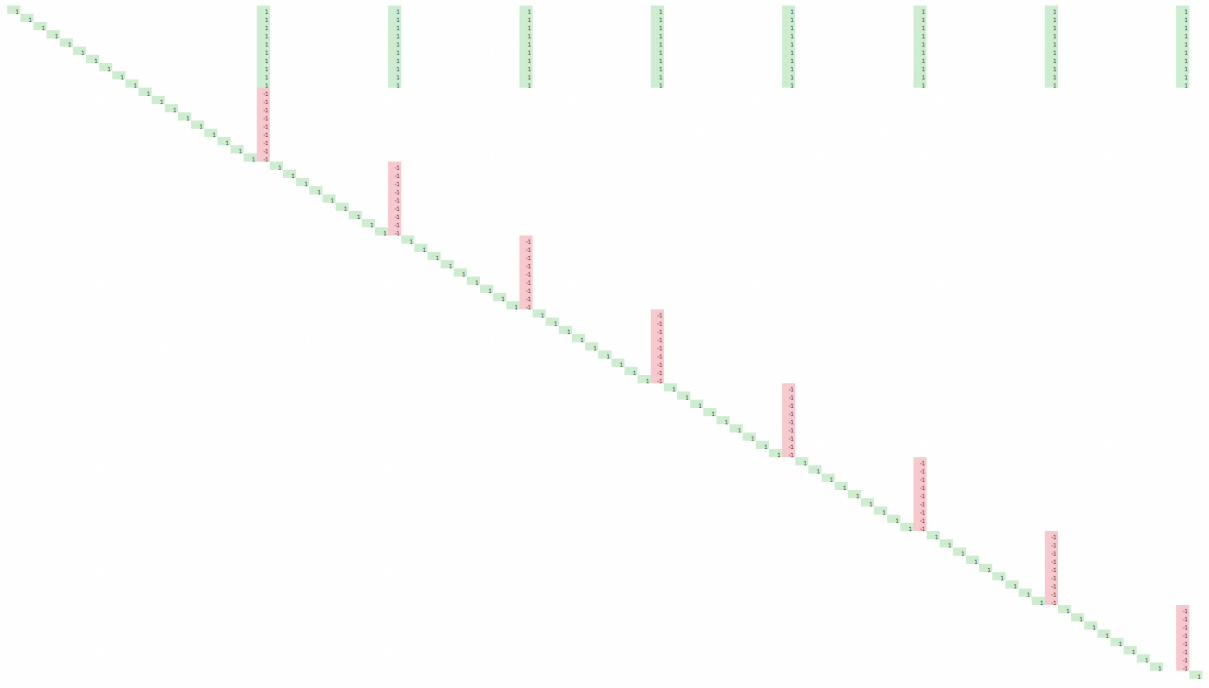

Figure 8. Reduced row echelon form of the realisation matrix for bc dataset; green and red are 1,-1 respectively

B Chi-squared Best Features

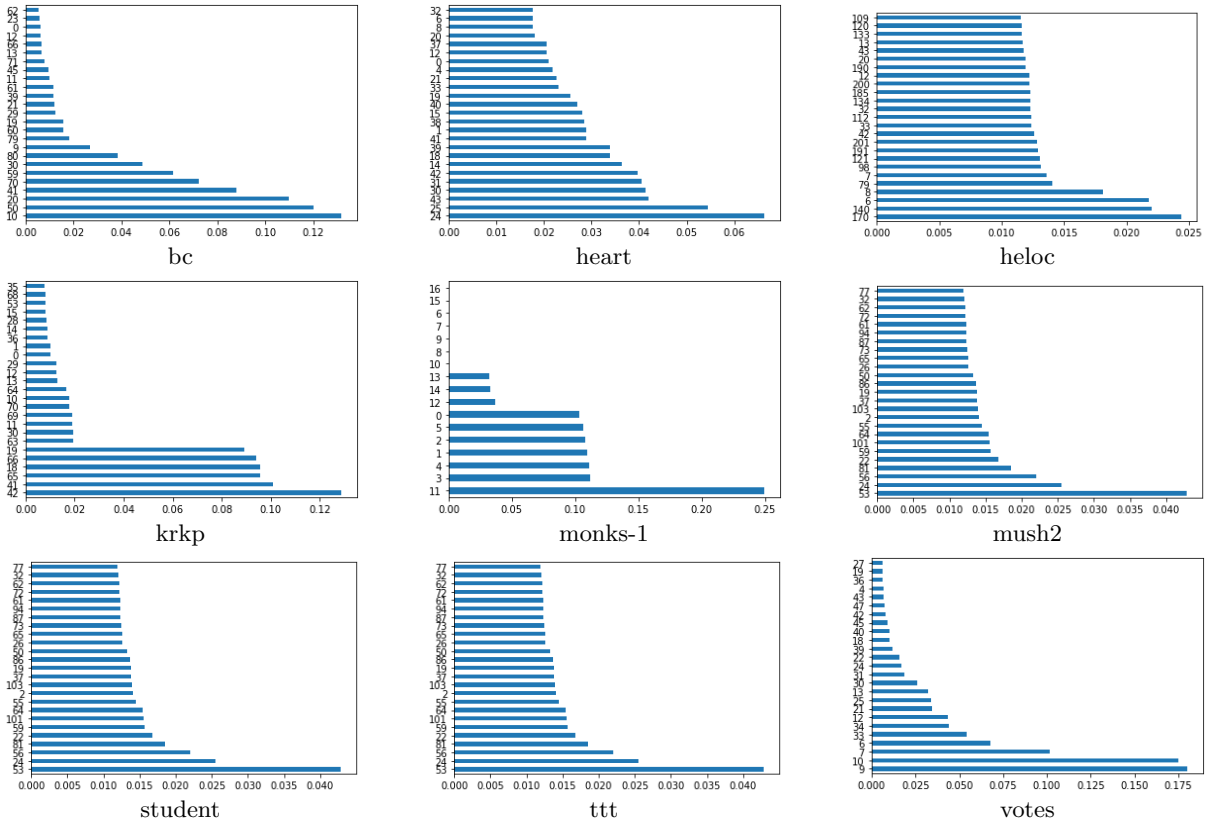

Figure 9. Top 25 features based on the Chi-squared metric for each dataset

C Impurity Reduction Best Features

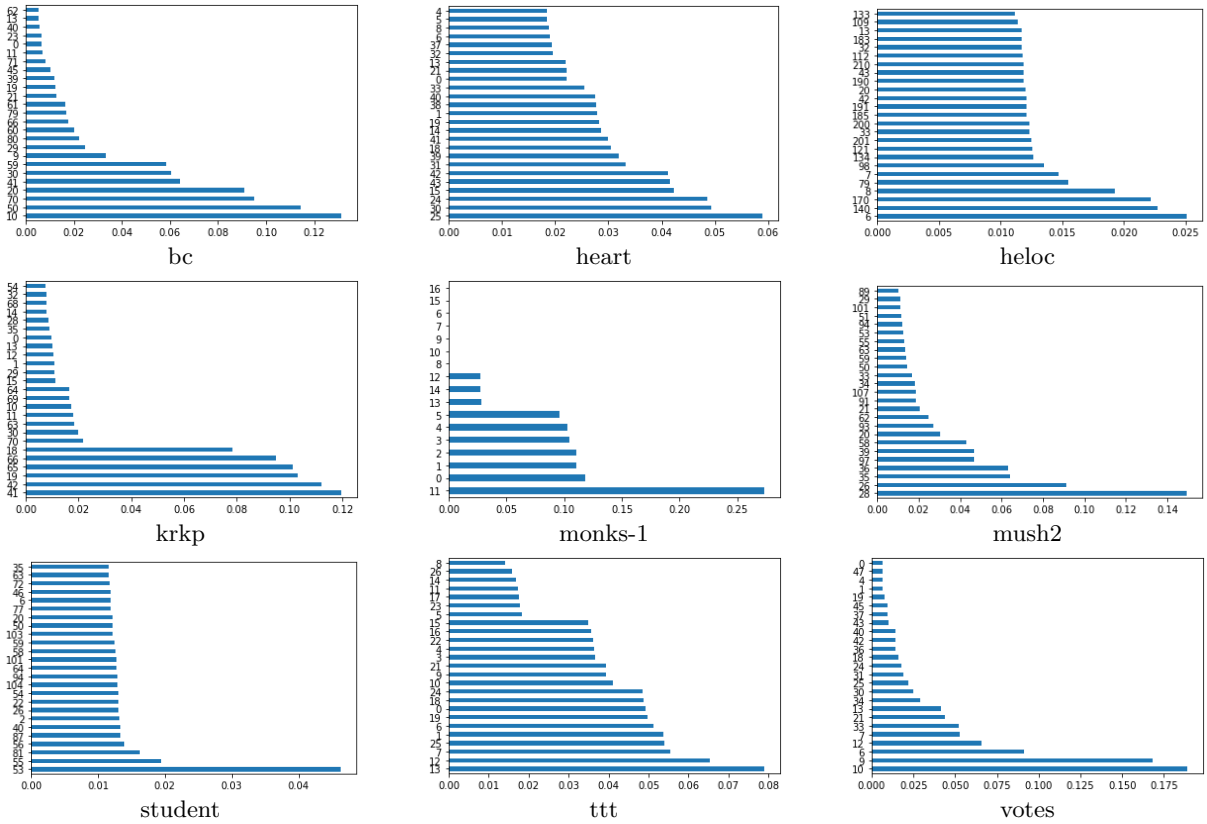

Figure 10. Top 25 features based on the Gini Impurity Reduction metric for each dataset

D Feature Correlations

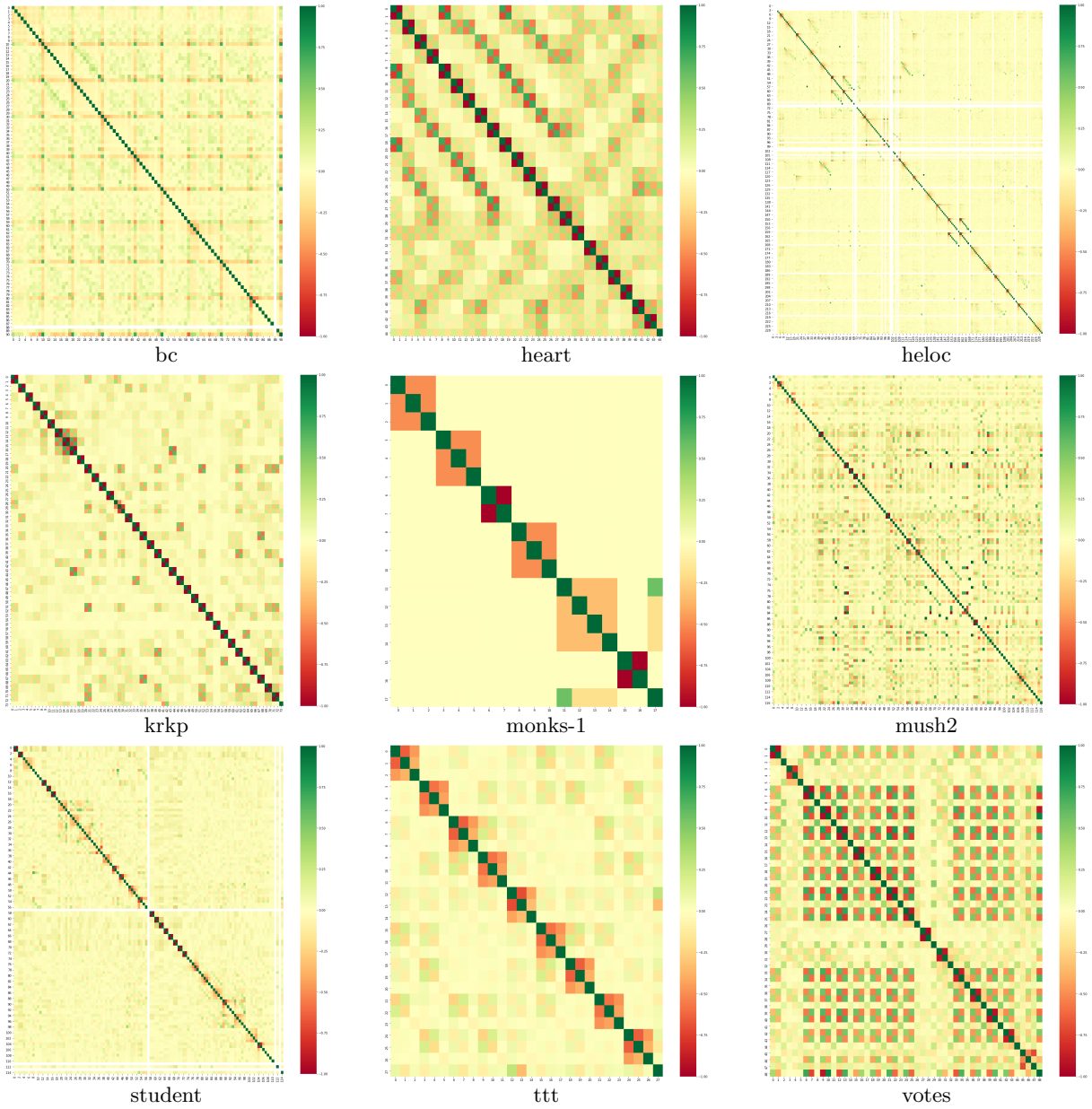

Figure 11. Correlations between features and dependent variables of each dataset