Erasmus University Rotterdam

Erasmus School of Economics

Bachelor Thesis Econometrics and Operations Research

Enhancing Gender Equality in Machine Learning: Optimal Discrimination-Aware Decision Trees using Integer Optimization

Matthijs Nijeboer (579637)

Supervisor:	Dr. R.M. Badenbroek
Second assessor:	Dr. M.H. Akyuz
Date final version:	June 30, 2023

Abstract

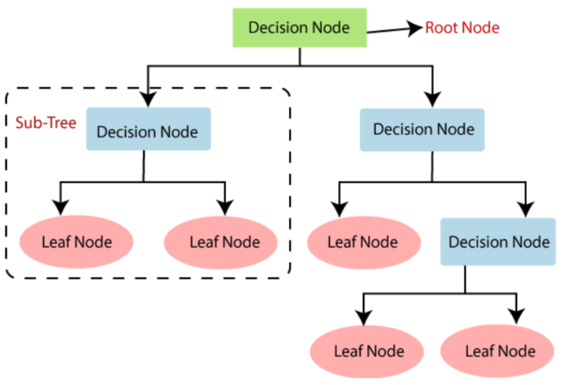
Gender discrimination in decision trees is profoundly relevant for society. By addressing and resolving this issue, we can advance towards a more equal society. Overcoming gender bias in decision-making processes fosters inclusivity, ensuring that individuals are treated fairly and are offered equal opportunities. We replicate the results of Verwer and Zhang (2017), and extend their integer programming formulation to account for discrimination-awareness, by adding constraints which bound discrimination explicitly. We test the performance of our adapted formulation in the context of gender discrimination, focusing on the accuracy-discrimination trade-off. With our adapted formulation we are able to build decision trees with significantly decreased gender discrimination, while having near-optimal accuracy.

1 Introduction

Decision trees are a popular tool for supporting decision making, and are widely used for classification and regression problems. For example, a regression tree can be used to predict the price of a stock, based on market indicators. A classification example could be predicting whether a patient has a certain disease or not, based on their symptoms. These are only two examples, but show that decision trees are a versatile tool, which can be applied in many different fields to solve a wide range of problems. Apart from their applicability, decision trees are also transparent, meaning non-experts are easily able to use and understand them.

A decision tree can be seen as a series of decisions to be made, presented in a tree-like structure. Figure 1 shows what this structure looks like.

Figure 1: General structure of a decision tree



The tree is traversed from top to bottom, where in each decision node of the tree, a condition is evaluated based on an object's data. This evaluation then leads to the decision of proceeding either in the left or the right sub-tree. When one eventually arrives in one of the leaf nodes, the object can be classified accordingly.

The topic of decision trees has been studied since the late 20th century. Among the first scientific research papers are for example Breiman et al. (1984) and Quinlan (1986), who introduce the concept and optimization of learning decision trees. Hyafil and Rivest (1976) show that this problem is NP-complete. Therefore greedy heuristics like CART (Breiman et al. (1984)), ID3 (Quinlan (1986)), and C4.5 (Salzberg (1993)) have been developed. These heuristics determine splits in a top-down fashion. The advantage of using such heuristics is their computational efficiency, however, Bertsimas and Dunn (2017) address some of their drawbacks. Each split in the tree is determined by an optimization problem which provides a local optimum, not taking into account future splits. In this way, the underlying characteristics of the data set are not captured well, and this can result in trees which are not good at classifying new data points.

Furthermore, top-down approaches have a limitation in the fact that they require pruning to design trees which generalize well. A penalty on tree complexity can prevent the selection of weaker splits, however, it might be that strong splits are concealed behind these weaker splits. Therefore, a top-down approach may not find the best tree when the complexity penalty is too high. This problem is usually solved by growing the tree as deep as possible, and then pruning back up using the complexity penalty.

Another way to solve the problem of strong splits hiding behind weak splits, is using look-ahead heuristics like IDX (Norton (1989)), LSID3 and ID3-k (Esmeir and Markovitch (2007)). As its name suggests, a look-ahead heuristic also tries to optimize the tree based on deeper trees which are rooted at the current node. However, the drawback of such a method is the danger of overfitting, leading to weak generalization.

The drawbacks of the classical heuristics, together with the immense increase in computational power of mixed-integer optimization (MIO) solvers, are the main motivations for trying to learn optimal decision trees using mixed-integer optimization. Verwer and Zhang (2017) manage to outperform CART for data sets up to 1000 observations and a tree depth up to five, but for larger data sets the classification accuracy of the best solution found within a reasonable time frame drops significantly. However, they do find that using a feasible CART solution as starting point for the MIO solver always results in an improved solution. This idea of using a so called warm start is also used by Bertsimas and Dunn (2017). In their paper, the authors provide a different formulation than Verwer and Zhang (2017), which in general also improves CART. In particular, they find that mixed-integer optimization solvers are able to practically solve decision tree problems for data sets where the number of observations is in the thousands. Verwer and Zhang (2019) address the issue of having to create variables and constraints for each observation in the data set. They manage to formulate the decision tree problem such that its size is for the most part independent of the number of observations in data set, resulting in a much smaller number of variables and constraints. This, in turn, results in faster solving time and more accurate solutions.

The most recent work in this field aims to improve computation times, and incorporate aspects like fairness and discrimination. Newest insights are for example from Aghaei et al. (2021), Carrizosa et al. (2021) and Blanquero et al. (2021).

The goal of this thesis is to learn good discrimination-aware decision trees, which can classify objects with high accuracy. In particular, we focus on limiting gender discrimination. As a first step, we replicate and verify the work of Verwer and Zhang (2017), who try to find optimal decision trees using integer optimization. Then we extend their integer optimization model to account for discrimination, and test the resulting model on three data sets. We manage to obtain almost the exact results of Verwer and Zhang (2017), with only a few small deviations in accuracy. Furthermore, we find that our adapted formulation for discrimination-aware decision tree learning finds solutions with drastically decreased discrimination, while having close-to-optimal accuracy.

In Section 2, we discuss the used data sets and data transformation in detail. Thereafter, we introduce the decision tree learning formulation of Verwer and Zhang (2017) in Section 3, and add discrimination awareness to their formulation in Section 4. Finally, we report our results in Section 5 and provide our conclusions together with suggestions for future research in Section 6.

2 Data

For reproducing the results of Verwer and Zhang (2017), we need to conduct the same experiments as they do, using the same data sets. For their classification experiments, Verwer and

Zhang (2017) use three data sets, named "Iris", "Diabetes" and "Bank". The "Diabetes" data set comes from the Pima Indian Diabetes database, and "Bank" is from Moro et al. (2014). We obtained these data sets from the UCI machine learning repository (Dua and Graff (2017)). For our extension on discrimination-aware decision trees we use another banking data set called "Japanese Credit Screening", which contains data on people who were or were not granted credit by a Japanese banking institution. We refer to this data set as "Credit". As a second data set to test discrimination-aware decision trees we use a data set called "Heart Disease", which contains data on people who were or were not diagnosed with a heart condition. This is a large database consisting of four smaller databases, each one corresponding to an institution where the data was collected. In our testing, we use the Cleveland database. Finally, we use a data set called "Titanic", which contains data on passengers of the Titanic ship, and whether they survived its sinking or not. We obtained "Credit" and "Heart Disease" from the UCI machine learning repository (Dua and Graff (2017)), and "Titanic" from Kaggle (2010).

Data sets "Bank, "Credit", "Heart Disease" and "Titanic" contain categorical attributes, meaning the attribute can take on a fixed amount of values. An example of this is the *marital* attribute from "Bank", which can take on values from the set {married, divorced, unknown}. It is difficult to provide interpretable thresholds for decision rules in the tree in this way. Therefore we split each categorical attribute into N binary attributes, where N is the number of categories of the attribute. Such a binary attribute then indicates whether an observation has that category or not. This makes it possible to interpret decision rules in the tree.

Originally, "Heart Disease" has five target classes numbered 0-4, where 0 means no presence of a heart condition and numbers 1-4 indicate the kind of heart condition in case of presence. We simplify this by using a binary classification for which classes 1-4 are merged in a single class which indicates the general presence of a heart condition. Furthermore, "Heart Disease" has 6 observations which contain missing values, and therefore we remove these.

The "Titanic" data set originally consists of 10 attributes. However, we neglect the *name* and *ticket number* attributes as we assume they do not affect survival probability. Furthermore, the *cabin* attribute is always empty, so we omit this attribute as well. There are 179 observations with at least one of the remaining attribute values missing. In the absence of a more suitable option, we choose to delete these observations.

The other data sets do not contain missing values, and therefore we do not remove any of their observations. In Table 1, a summary of the data sets is provided, where for "Bank", "Credit", "Heart Disease" and "Titanic", the total number of attributes after splitting the categorical attributes into binary attributes is provided in parentheses.

Table 1: Characteristics of the data sets

Data set	**# Observations**	**# Attributes (Splitted)**	**# Classes**
Iris	150	4	3
Diabetes	768	8	2
Bank	4521	16 (48)	2
Credit	125	10 (16)	2
Heart Disease	297	13 (22)	2
Titanic	712	7 (11)	2

2.1 Transformation

In decision tree literature the data is often linearly scaled to the interval [0.0, 1.0], see for example Bennett and Blue (1996) and Bertsimas and Shioda (2007). This has numerous advantages. For example, this avoids numerical problems in big-M formulations, which are often used in integer programming. However, Verwer and Zhang (2017) propose a nonlinear data transformation which maps all unique feature values to a unique integer value, while maintaining the correct ordering of these values. They claim to significantly improve their obtained solutions using this transformation, and mention the following advantages of this approach:

- Thresholds in decision nodes can be presented as integers, which causes the MIO solver to be able to branch on these exact values.
- For observations which have successive integers as attribute values and the same class label, the values can be mapped to the same integer.
- The most occurring attribute value can be mapped to the value zero.
- Transformed attribute values can be ranged around zero.

All of these help the MIO solver to run faster. However, Verwer and Zhang (2017) do not emphasize the exact details of their data transformation. We were unable to obtain exactly their transformed data, however, using Algorithm 1 we were able to transform the data in such a way that the properties listed before still hold. Note that class labels are not transformed.

Algorithm 1 Data transformation

```
1: Input: Data set $S$
2: for each non-binary attribute column $i$ of $S$ do
3:     Initialize maps $M_1 = \{(key, val) : key \in K_1, val \in V_1\}$ and
4:     $M_2 = \{(key, val) : key \in K_2, val \in V_2\}$, where $K_1$ and $K_2$ are key sets,
5:     and $V_1$ and $V_2$ are value sets.
6:     Store all unique attribute values, and sort them in ascending order in an array $A_i$.
7:     for each attribute value $v_{r,i}$ in $A_i$ do
8:         Map $v_{r,i}$ to itself in $M_1$. That is, put $(v_{r,i}, v_{r,i})$ in $M_1$.
9:     end for
10:    for all pairs $(v_{r1,i},\ v_{r2,i})$ in $A_i$ where $v_{r1,i}$ and $v_{r2,i}$ are neighbours do
11:        if all observations having $v_{r1,i}$ as attribute value have the same class label, the same
       holds for observations having $v_{r2,i}$ as attribute value, and the class labels of these two sets
       of observations are equal then
12:            Put $(v_{r2,i}, v_{r1,i})$ in $M_1$. This pair now tells us that attribute value $v_{r2,i}$
13:            should be transformed to the same integer as $v_{r1,i}$.
14:        end if
15:    end for
16:    Store all unique values in $V_1$ in ascending order in an array $B_i$.
17:    Map most occurring value $x_i$ in $B_i$ to zero, that is, put $(x_i, 0)$ in $M_2$.
18:    Map all values in $B_i$ smaller than $x_i$ to decreasing negative integers, following the ordering
       of $B_i$. Put these pairs in $M_2$.
19:    Map all values in $B_i$ larger than $x_i$ to increasing positive integers, following the ordering
       of $B_i$. Put these pairs in $M_2$.
20:    Transform the attribute column as follows. Use the column's values as keys in $M_1$,
       then use the corresponding values in $M_1$ as keys in the other map $M_2$. The values in $M_2$
       corresponding to these keys are the transformed attribute values.
21: end for
```

3 Integer optimization for decision trees

In order to find optimal decision trees, we want to use integer optimization and therefore have to formulate the problem as an integer linear program. We use the formulation proposed by

Verwer and Zhang (2017), which is called Decision Trees as Integer Programs (DTIP). We use roughly the same notation as used by Verwer and Zhang (2017), however we change some names to make the formulation a bit more readable. The notation is presented in Table 2.

Table 2: Notation

Symbol	Type	Definition
D	set	set of decision nodes
L	set	set of leaf nodes
R	set	set of observations
T	set	set of target values
n	constant	number of observations in the data set
m	constant	number of attributes in the data set
y	constant	number of unique target values
k	constant	depth of the tree
d_j	constant	depth of node j
$v_{r,i}$	constant	value of attribute i in observation r
t_r	constant	target value (class label) of observation r
$p_{l,t}$	binary decision	equals 1 if classifier prediction of leaf node l is equal to target value t
e_r	binary decision	prediction error of observation r
$f_{i,j}$	binary decision	equals 1 if attribute i is used in decision rule of node j
$x_{h,r}$	binary decision	equals 1 (0) if path of observation r goes left (right) at depth h
c_j	integer decision	threshold in decision rule of node j

Furthermore, we have values LF and UF which denote the minimum and maximum value over all attributes, respectively. That is, $LF = \min\{v_{r,i} : 1 \leq i \leq m, 1 \leq r \leq n\}$ and $UF = \max\{v_{r,i} : 1 \leq i \leq m, 1 \leq r \leq n\}$. Using these values we can define appropriate big-M values which are necessary for some of the constraints. Let $M_r = \max\{v_{r,i} - LF : 1 \leq i \leq m\}$ and $M'_r = \max\{UF - v_{r,i} : 1 \leq i \leq m\}$ be tight big-M values which are used in some of the upcoming constraints.

Finally, to denote the path direction to node j starting from the root node, we have:

$$\operatorname{dir}(h,j,r) = \begin{cases} x_{h,r} & \text{if the path towards node } j \text{ goes left at depth } h \\ 1 - x_{h,r} & \text{if the path towards node } j \text{ goes right at depth } h \end{cases} \tag{1}$$

The intuition is as follows. When at a certain moment observation r comes across node j, $\operatorname{dir}(h,j,r)$ will have value one for every depth h smaller than the depth of node j. Thus, nodes having this smaller depth h will be above node j in the tree. In this way, the path to node j can be easily tracked, which is necessary for determining what the values of the thresholds on this path must be.

Using the notation in Table 2, we can explain DTIP step by step. The objective is minimizing the total prediction error

$$\sum_{r=1}^{n} e_r. \tag{2}$$

For every node $j \in D$ we require that its decision rule consists of exactly one attribute i,

$$\sum_{i=1}^{m} f_{i,j} = 1 \qquad j \in D. \tag{3}$$

Furthermore, every node $j \in D$ needs a threshold c_j in its decision rule. For each observation r, and each decision node j, we have:

$$\sum_{h=0}^{d_j-1} M_r \operatorname{dir}(h,j,r) + M_r x_{d_j,r} + \sum_{i=1}^{m} v_{r,i} f_{i,j} \le M_r(d_j+1) + c_j \qquad j \in D, \quad r \in R \tag{4}$$

and

$$\sum_{h=0}^{d_j-1} M'_r \operatorname{dir}(h,j,r) - M'_r x_{d_j,r} - \sum_{i=1}^{m} v_{r,i} f_{i,j} \le M'_r d_j - c_j - 1 \qquad j \in D, \quad r \in R. \tag{5}$$

To explain how these constraints work, consider (4). When observation r comes across node j in the tree, it holds that $\sum_{h=0}^{d_j-1} \operatorname{dir}(h,j,r) = d_j$, as $\operatorname{dir}(h,j,r)$ equals one for all depths h in the summation. If at node j the path continues to the left sub-tree, the $M_r(d_j+1)$ term cancels out and we require the attribute value $v_{r,i}$ for which i is the attribute represented in the decision rule of node j, to be smaller or equal to the threshold. When the path would continue in the right sub-tree, or does not come across node j at all, the big-M multiplier M_r makes the constraint inactive. Constraint (5) models this same behaviour but for an observation r which needs to continue its path in the right sub-tree.

Note that we have incorporated some minor differences with respect to the constraints Verwer and Zhang (2017) propose. First of all, the first summation starts from depth 0, and ends at the depth of node j's parent node, instead of it starting at depth 1 and continuing up to the depth of node j. As we define the root of the tree to be at depth 0, this should be included in the path to node j, while the depth of node j itself should not be included. In this way, also the right hand sides of (4) and (5) have to be changed accordingly, by adding one M_r and M'_r term respectively. Finally, we subtract 1 from the right hand side of (5), such that it enforces a strict inequality between the left hand side and $M'_r d_j - c_j$. This is necessary because in case some attribute value $v_{r,i}$ would be equal to the threshold c_j, the tree has to decide on taking the left or right sub-tree for that observation r.

Next we require each leaf node to have exactly one classifier prediction:

$$\sum_{t=1}^{y} p_{l,t} = 1 \qquad l \in L. \tag{6}$$

Furthermore, it does not make sense for two leaf nodes coming from the same parent node to have the same classifier prediction,

$$p_{l,t} + p_{l',t} \leq 1 \qquad (l, l') : l \text{ and } l' \text{ share the same parent node}, \quad l, l' \in L, t \in T. \tag{7}$$

Note that in Verwer and Zhang (2017), this constraint is an equality. However, when there are more than two classes this would want to allocate all these classes to only two leaves, and as a leaf can only have one classifier prediction, this is not possible. Therefore we replace it by an inequality.

The prediction errors are computed as follows:

$$\sum_{h=0}^{k-1} \text{dir}(h, l, r) + \sum_{t \in T : t \neq t_r} p_{l,t} \leq e_r + k \qquad l \in L, \quad r \in R. \tag{8}$$

When observation r ends up in leaf l, $\sum_{h=0}^{k-1} \text{dir}(h, l, r) = k$ as at every depth h the value of $\text{dir}(h, l, r)$ equals 1. In this way the k on the right hand side cancels out, and the prediction error is set to one if the classifier prediction of l is different than the target value of observation r. When r does not end up in leaf l, the constraint becomes inactive as the right-hand side is always larger. Again we have changed the first summation's starting and ending index like we did for (4) and (5).

Finally, Verwer and Zhang (2017) strengthen their formulation by including the following bounds on c_j,

$$\sum_{i=1}^{m} LFf_{i,j} \leq c_j \leq \sum_{i=1}^{m} UFf_{i,j} \qquad j \in D. \tag{9}$$

In complete form, DTIP is as follows:

$$\begin{aligned} \min \quad & \sum_{r=1}^{n} e_r \\ \text{s.t} \quad & (3) - (9). \end{aligned}$$

4 Discrimination-aware DTIP

Verwer and Zhang (2017) show the flexibility of their formulation by adjusting the objective function of DTIP to handle different goals like handling fairness, discrimination and imbalanced data. We test discrimination-aware DTIP (DA-DTIP) on data sets "Credit", "Heart Disease" and "Titanic". We focus on gender discrimination for all three data sets, that is, we consider the *gender* attribute to be a sensitive attribute. For "Credit", there might be discrimination against females in terms of being denied credit by the bank. This is because although Japan is a high-income country, there is quite some gender inequality. Support for this choice can be found in the Gender Inequality Index (United Nations Development Programme (2021)), where Japan is ranked 22nd out of 188 countries, which is low given that Japan is a high-income country. Furthermore, in their Global Gender Gap Report, World Economic Forum (2022) ranks Japan even lower at a dismal 116th place out of 146 participating countries. As the data was collected more than 30 years ago, this was even more relevant back then.

Within the context of the "Heart Disease" data set, gender can also be regarded as a discriminatory factor. For instance, there may exist medical biases (Hamberg (2008)) where specific symptoms or diagnostic standards are predominantly based on research conducted on male patients, resulting in various complications. Heart disease symptoms can differ between men and women, and if the decision-making process prioritizes male-oriented symptoms, female patients might be underdiagnosed or misdiagnosed. Consequently, this can result in receiving sub-optimal treatment or experiencing unnecessary delays in obtaining the appropriate care.

Finally, in the context of "Titanic", discrimination might be a bit less obvious. One can think of males being stronger and more athletic than females, and therefore having a higher survival probability as escaping the ship could be easier. The other way around, females might have a greater survival probability due to the 'women and children first' phenomenon, where women and children are evacuated to safety before men are. We want to stress the fact that although a decision tree for this data set is unusable in the future, given that there will not be another Titanic trip, this data set is used as an extra data source to see how well our method works in-sample. Furthermore, the discrimination in this data set might be less relevant in society than in "Credit" and "Heart Disease". However, as there is a large difference in survival probability for men and women in this data set it serves an excellent data set to test our method on.

Including discrimination in DTIP can be done in multiple different ways. Verwer and Zhang (2017) add a simple expression to the objective function which computes the difference in positive class probability for different sets of observations, using the prediction errors. However, we opt for a different approach. Instead of including an extra term in the objective, we integrate discrimination-awareness by adding two new sets of constraints. The reason for this becomes clear in a moment. First we define

$$z_r = \begin{cases} 1 & \text{if observation } r \text{ is predicted to be in a positive class} \\ 0 & \text{otherwise} \end{cases}$$

as a binary variable indicating whether observation r is predicted to be in a positive class or not. We assume that there is only one negative class, and this class is labeled as 0. To compute the z_r variables, we add to (3)-(9) the following two sets of constraints:

$$\sum_{h=0}^{k-1} \mathrm{dir}(h,l,r) + \sum_{t \in T : t \neq 0} p_{l,t} \leq z_r + k \qquad l \in L, \quad r \in R \tag{10}$$

and

$$-\sum_{h=0}^{k-1} \mathrm{dir}(h,l,r) + \sum_{t \in T : t \neq 0} p_{l,t} \geq z_r - k \qquad l \in L, \quad r \in R. \tag{11}$$

These two constraints together force the z_r variables to have the correct values. To explain how (10) and (11) work together, assume observation r ends up in leaf l. In this case, $\sum_{h=0}^{k-1} \mathrm{dir}(h,l,r) = k$, therefore (10) reduces to $\sum_{t \in T : t \neq 0} p_{l,t} \leq z_r$ and (11) reduces to $\sum_{t \in T : t \neq 0} p_{l,t} \geq z_r$. If the classifier prediction of leaf l is (one of) the positive class(es), such that $\sum_{t \in T : t \neq 0} p_{l,t} = 1$, z_r will be forced to equal 1. Using this same argument, z_r is forced to 0 if the classifier prediction of leaf l is the negative class. In case observation r does not end up in leaf l, both (10) and (11) become inactive.

Using the constructed z_r variables we can easily create a constraint which bounds discrimination, by bounding the absolute difference in positive class probability for two sets of observations. We use two set of observations R_m and R_f which contain all male and female observations respectively. Note that although we focus on gender discrimination in our study, the two sets of observations can be chosen to represent any two groups for which discrimination might exist between them. Now we add to (3)-(11) the following:

$$\left| \frac{1}{n_m} \sum_{r \in R_m} z_r - \frac{1}{n_f} \sum_{r \in R_f} z_r \right| \leq \epsilon, \tag{12}$$

where n_m and n_f are the number of male and female observations respectively. The advantage of this approach is that the feasible region becomes smaller because we have added constraints. Furthermore, we can try different values for the discrimination tolerance ϵ to see which one yields the best results in terms of discrimination and accuracy, giving rise to a trade-off between these two objectives. We prefer this method over adding a similar discrimination term to the objective function, as with this method we are able to bound discrimination explicitly.

In complete form, DA-DTIP is as follows:

$$\begin{aligned} \min \quad & \sum_{r=1}^{n} e_r \\ \text{s.t} \quad & (3) - (12). \end{aligned}$$

5 Empirical results

To verify the work of Verwer and Zhang (2017), we run standard DTIP using CPLEX (IBM ILOG CPLEX (2014)) for data sets "Iris", "Diabetes" and "Bank", and like Verwer and Zhang (2017) we use a time limit of 30 minutes. The results are presented in Table 3.

Table 3: Classification accuracies using DTIP for "Iris", "Diabetes" and "Bank", for depths 1-5. Optimal solutions are denoted by (*).

Data set	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
Iris	0.667*	0.960*	0.993*	1*	1*
Diabetes	0.75*	0.777	0.794	0.773	0.733
Bank	0.893*	0.895	0.887	0.115	0.115

We manage to obtain similar results as Verwer and Zhang (2017), despite our different transformed data. The only few minor differences are in the "Diabetes" and "Bank" data sets, where for some depths the accuracy is a bit smaller or larger. This only is the case for solutions obtained after running for the complete time frame of 30 minutes, indicating this might be due to using different computers to run DTIP. In Table 3 we see that DTIP can find optimal decision trees of depths 1-5 for the small data set "Iris", however for larger data set "Diabetes", it is only able to do so for a depth of 1. Using depths 2-5, DTIP always runs until the time limit. For the "Bank" data set, which is by far the largest set, we see that for depths 4 and 5, DTIP builds a tree which is too large for CPLEX to find even a decently accurate solution within the time limit.

Now that we have verified the performance of DTIP, we run both DTIP and DA-DTIP for the "Credit", "Heart Disease" and "Titanic" data sets with a time limit of 30 minutes as well. In DA-DTIP we use a discrimination tolerance of 5 percent as a quite strong upper bound, so we can see the effect on accuracy with ease. Results are reported in Table 4. Note that discrimination is computed as in the left-hand side of (12).

Table 4: Classification accuracies using DTIP and DA-DTIP for "Credit", "Heart Disease", and "Titanic", for depths 1-5. Optimal solutions are denoted by (*), and discrimination is reported in parentheses. A tolerance level of $\epsilon = 0.05$ is used in DA-DTIP.

Data set	**Method**	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
Credit	DTIP	0.768* (0.083)	0.824* (0.082)	0.856 (0.085)	0.912 (0.072)	1* (0.128)
	DA-DTIP	0.720* (0.026)	0.800* (0.046)	0.880 (0.012)	0.896 (0.035)	0.960* (0.042)
Heart Disease	DTIP	0.764* (0.415)	0.798* (0.204)	0.855 (0.363)	0.848 (0.354)	0.842 (0.401)
	DA-DTIP	0.687* (0.016)	0.778* (0.045)	0.771 (0.048)	0.838 (0.050)	0.818 (0.047)
Titanic	DTIP	0.779* (1.000)	0.816* (0.919)	0.826 (0.884)	0.838 (0.696)	0.798 (0.819)
	DA-DTIP	0.622* (0.035)	0.669 (0.047)	0.666 (0.049)	0.645 (0.043)	0.596 (0.000)

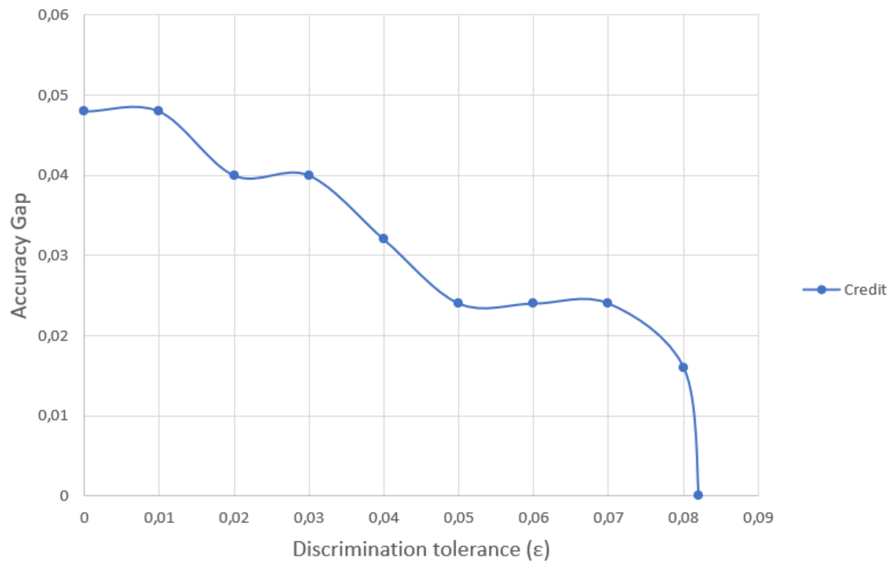
We can clearly see the flexibility of DTIP from these results. When we impose a discrimination bound of no more than 5%, for "Credit" we observe only a slight decrease in accuracy for tree depths 1, 2, 4, and 5. However, for depths 3 the tree even gains accuracy (due to using a time limit) while also having less discrimination against females.

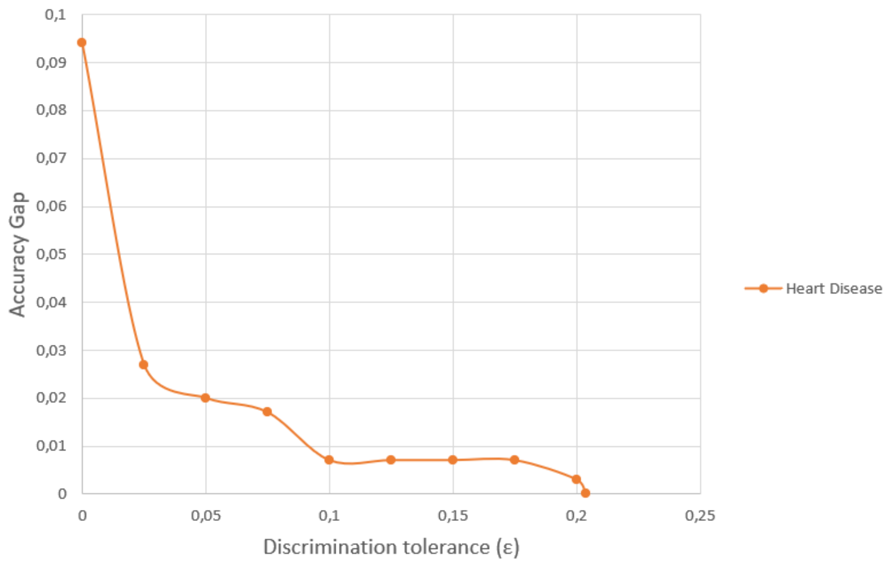
For "Heart Disease" we see that standard DTIP gives solutions with quite high discrimination. The probability for a male person to have a heart disease based on the diagnostic criteria is much larger than for females, which might indicate the presence of medical bias. However, with DA-DTIP discrimination can be drastically decreased, while the decrease in accuracy is much smaller, especially for depths 2, 4 and 5.

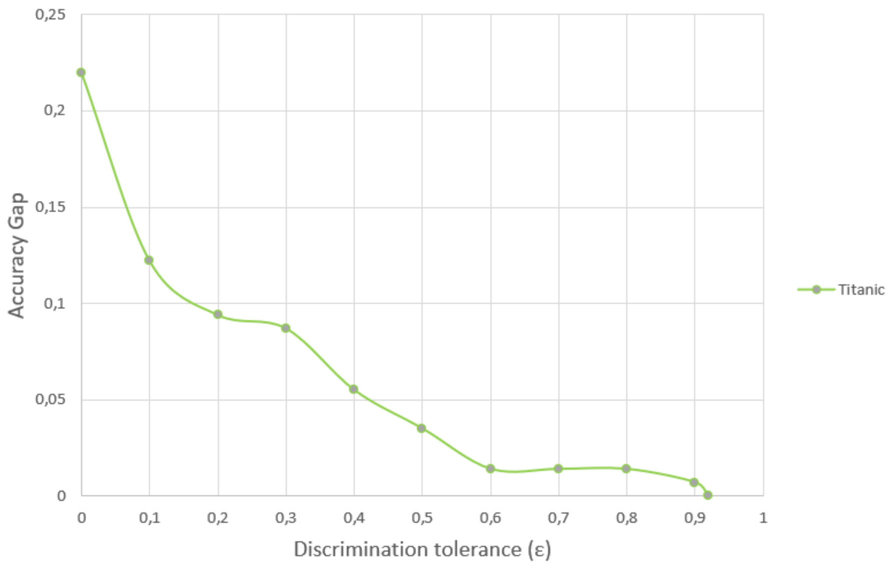
Finally, we see that there is extremely high discrimination against males in the “Titanic” data set, indicating the ‘women and children first’ phenomenon might really be what saved a lot of female lives, and lead to male deaths. Therefore bounding the discrimination at 5 percent causes the accuracy to drop significantly for all depths. For depth 5, DA-DTIP is only able to find a solution within the time limit where all passengers are predicted not to survive.

The point of including discrimination-awareness as a constraint in the formulation is that one can vary the discrimination tolerance ϵ. In Figure 2 we show the accuracy-discrimination trade-off for “Credit”, “Heart Disease”, and “Titanic” using a depth of 2 as an illustrative example. The reason for using this depth is that DTIP is able to find optimal solutions within the time limit for depth 2, which means we can compare DA-DTIP solutions with optimal DTIP solutions. We plot the accuracy gap, being the difference between the optimal DTIP accuracy and the DA-DTIP accuracy, against the discrimination tolerance ϵ.

Figure 2: Accuracy-discrimination trade-offs for $d = 2$







The accuracy gap for “Credit” between the optimal solution of DTIP and the solution of DA-DTIP with zero discrimination is only 0.048. This can be explained that originally, the discrimination level for the “Credit” data set was not that high, and therefore achieving a zero-discrimination solution does not come at a large cost. For “Heart Disease” the accuracy gap equals 0.094, which is in line with the higher discrimination levels of this data set. We see that after increasing the discrimination tolerance to only 0.025, the gap to the optimal DTIP solution is already more than halved. The same pattern can be seen for “Titanic”, but it is more subtle than for “Heart Disease” and needs bigger jumps in the discrimination tolerance. As the original discrimination level for “Titanic” is extremely high, the accuracy gap for $\epsilon = 0$ is quite large. Still, DA-DTIP is able to cut discrimination in half while bringing the accuracy gap below 5 percentage points. This shows that DA-DTIP can quite easily find solutions with significantly less discrimination and close-to-optimal accuracy.

6 Conclusions

We have formulated the problem of discrimination-aware decision tree learning as an extended version of the formulation of Verwer and Zhang (2017), where we add constraints which bound discrimination explicitly. Our empirical testing in the context of gender discrimination shows that we are able to decrease gender bias in the decision tree often by over 50% of the original level, while the accuracy is only a few percentage points lower than the optimal accuracy obtained using the formulation without discrimination-awareness. This can be achieved efficiently for trees up to depth 5 and data sets with up to 1000 observations. In the future it could be interesting to use heuristic solution as a warm start for the MIO solver, as such a solution initially might not satisfy the desired maximum discrimination level. Furthermore, since we performed our tests in-sample, a logical continuation would be to test our method out-of-sample to investigate its generalization.

References

Aghaei, S., Gómez, A., and Vayanos, P. (2021). Strong optimal classification trees. *Journal of Machine Learning Research*, 22(70):1–50.

Bennett, K. and Blue, J. (1996). Optimal decision trees. (214).

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

Bertsimas, D. and Shioda, R. (2007). Classification and regression via integer optimization. *Operations Research*, 55(2):252–271.

Blanquero, R., Carrizosa, E., Molero-Río, C., and Romero Morales, D. (2021). Optimal randomized classification trees. *Computers amp; Operations Research*, 132:105281.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). Classification and regression trees.

Carrizosa, E., Molero-Río, C., and Romero Morales, D. (2021). Mathematical optimization in classification and regression trees. *TOP*, 29(1):5–33.

Dua, D. and Graff, C. (2017). UCI machine learning repository; http://archive.ics.uci.edu/ml.

Esmeir, S. and Markovitch, S. (2007). Anytime learning of decision trees. *The Journal of Machine Learning Research*, 8:891–933.

Hamberg, K. (2008). Gender bias in medicine. *Women's Health*, 4(3):237–243.

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

IBM ILOG CPLEX (2014). IBM ILOG CPLEX: V12.1 user manual.

Kaggle (2010). Kaggle: Your Machine Learning and Data Science Community. `https://www.kaggle.com`. Accessed on 20-06-2023.

Moro, S., Cortez, P., and Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31.

Norton, S. W. (1989). Generating better decision trees. *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 2*, 2:800–805.

Quinlan, J. R. (1986). The use of decision trees for machine learning. *Machine Learning*, 1:81–106.

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

United Nations Development Programme (2021). Human development indicators 2021.

Verwer, S. and Zhang, Y. (2017). Learning decision trees with flexible constraints and objectives using integer optimization. *Integration of AI and OR Techniques in Constraint Programming*, page 94–103.

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

World Economic Forum (2022). The global gender gap report 2022. Geneva, Switzerland: World Economic Forum. Available at: `https://www.weforum.org/reports/global-gender-gap-report-2022`.

A Appendix: Code description

The main programming language we used is Java. With IBM ILOG CPLEX (2014) we were able to run DTIP and DA-DTIP in Java. Our code consists of 9 .java files. For each file, we provide a short description:

1. DTIP.java: class used for solving standard DTIP for data sets "Iris", "Diabetes" and "Bank".

2. DADTIP.java: class used for solving DA-DTIP for data sets "Credit", "Heart Disease" and "Titanic". Note that it is also possible to run standard DTIP using this class by setting the mode parameter to 0. Upon solving, the resulting discrimination level, together with the positive class probabilities for males and females is printed as output next to the objective value. This is different from DTIP.java as in that class only the objective value is printed, because we do not consider discrimination for "Iris", "Diabetes" and "Bank".

3. Iris.java, Diabetes.java, Bank.java, Credit.java, Heart.java, Titanic.java: these classes read the corresponding data set. In particular, they read the classification column and the

attribute values. Using a method from Transform.java, the transformed attribute values can be obtained.

4. Transform.java: class used mainly for the general data transformation method. This method is used by all data set classes. Next to this, contains some minor functionalities like getting a certain column from a matrix.

We performed the following runs to obtain our final results:

1. DTIP results for “Iris”, “Diabetes” and “Bank”: run DTIP.java with the target values and attribute values for the desired data set, for depths 1-5. The input setups for all three data sets are in the main method.

2. DTIP results for “Credit”, “Heart Disease” and “Titanic”: run DADTIP.java with mode 0, using the target values and attribute values for the desired data set, for depths 1-5. The input setups for all three data sets are in the main method, one just has to provide 0 as the mode parameter in the solve method.

3. DA-DTIP results for “Credit”, “Heart Disease” and “Titanic”: run DADTIP.java with mode 1 and epsilon is 0.05, using the target values and attribute values for the desired data set, for depths 1-5. The input setups for all three data sets are in the main method, one just has to provide 1 as the mode parameter and 0.05 as the epsilon parameter in the solve method.

4. Accuracy-discrimination trade-off for “Credit”: run DADTIP.java for all $\epsilon \in \{0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.082\}$, using a depth of 2 and mode 1. This grants 10 classification accuracies. Using the optimal accuracy of standard DTIP of 0.824, one can compute the accuracy gap for each of these 10 discrimination tolerances. These resulting data can be used to create a plot of the accuracy-discrimination trade-off for “Credit”.

5. Accuracy-discrimination trade-off for “Heart Disease”: run DADTIP.java for all $\epsilon \in \{0, 0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.175, 0.2, 0.204\}$, using a depth of 2 and mode 1. This grants 10 classification accuracies. Using the optimal accuracy of standard DTIP of 0.798, one can compute the accuracy gap for each of these 10 discrimination tolerances. These resulting data can be used to create a plot of the accuracy-discrimination trade-off for “Heart Disease”.

6. Accuracy-discrimination trade-off for “Titanic”: run DADTIP.java for all $\epsilon \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.919\}$, using a depth of 2 and mode 1. This grants 11 classification accuracies. Using the optimal accuracy of standard DTIP of 0.812, one can compute the accuracy gap for each of these 11 discrimination tolerances. These resulting data can be used to create a plot of the accuracy-discrimination trade-off for “Titanic”.

Note that all .java files are well-documented, so complete descriptions of the functionalities of each .java file can be found in the Javadoc of these files.