Improving Existing Optimal Decision Trees Algorithms by Redefining Their Binarisation Strategy

Aleksandra Wolska
A.K.Wolska@student.tudelft.nl
TU Delft

26 June 2020

Abstract

Optimal decision trees are not easily improvable in terms of accuracy.owever, improving the pre-processing of underlying dataset can be the answer to creating more accurate decision trees. In this paper, multiple methods of binarising datasets are considered and the resulting decision trees compared. The binarisation is divided into two stages: discretisation and encoding, with various algorithms considered for both of the stages. Additionally, processing the data during the decision tree building, referred to as online processing, instead of beforehand, was considered. It was discovered that for smaller datasets, unsupervised discretisation was preferred, and extending one-hot encoding to also consider multiple categories at once as target gave better accuracy for trees with lower depth. For bigger datasets, online processing has shown to be beneficial.

1 Introduction

Decision trees are a very important type of data representation widely used in machine learning. They assist in solving classification problems, where based on multiple data features, each data instance is classified into one of two or more classes. Decision trees represent the conditions for each classification decision of a data instance in a tree-like structure, providing a concise way of presenting data learned from a dataset in a form that is easy for human users to understand and reason about. There exist heuristic-based decision trees (that use heuristics for the tree construction) and optimal decision trees, such as the MurTree[1], which produce an optimal result based on exhaustive search or its equivalent. Heuristic trees are often used, as they provide a relatively short runtime and high accuracy, but they are not necessarily the best possible representations of the data. Optimal decision trees offer an increase in accuracy; they minimise the amount of misclassified instances by the use of exhaustive search. Exhaustive search considers exhaustively many possible options, while heuristic search only considers limited amount of options that are chosen by the heuristic, thus optimal decision trees have much greater runtime. It is the biggest downside of the optimal decision trees, when compared to heuristic decision trees.Optimal decision trees are the optimal representation based on the dataset provided to the algorithm and therefore not improvable in terms of accuracy. However, even if the decision tree *algorithm* is difficult or impossible to improve, is there a way to improve the *given dataset*? This yields the motivation behind the research question.

In many decision trees, each non-leaf node has exactly two children. For such a tree, the various features in the data should have values in the binary domain. However, for most datasets it is not the case, and they require pre-processing before they can be used in the decision tree algorithm. This pre-processing - binarisation of the data - is used not only for decision trees, but also for other models of classification problems, however this paper will be focusing only on the use for decision trees.

There are two parts of the binarisation problem. A different approach must be taken for data in continuous and discrete, ordered units and data in categorical units. For the purpose of this paper, the binarisation problem has been defined as (i) the discretization element, when data in continuous domain is redefined into few ranges representing categorical values, and (ii) actual binarisation, where data in categorical domain is being encoded in terms of a binary domain. For clarity, the second stage of the binarisation problem will be referred to as *encoding*.

Two different methods of data processing are considered: offline and online data processing. The former is a standard way for processing data for the decision tree building, when the data is wholly processed before the tree building algorithm is run, and during runtime the values of the dataset are not modified. The latter appears when unprocessed data is inputted to the algorithm, and the discretization and binarisation takes place during the runtime of the algorithm, when the results are influenced by the branch of decision tree chosen. The terms "offline" for the standard data processing and "online" are defined for the purpose of this paper.

Now, the research question can be stated:

> **Research Question** *Can we improve existing optimal decision tree algorithms by improving their discretization strategy, wither online, offline, or a combination of the two?*

Three main subqestions can be defined:

1. Can offline binarisation strategy improve the optimal decision tree algorithm?
   - Is there a better offline discretization strategy (converting from continuous domain into categorical domain)?
   - Is there a better offline binarisation strategy (converting from categorical domain into binary domain)?
2. Can an online binarisation strategy improve the optimal decision tree algorithm?
   - How can binarisation be incorporated into the decision tree algorithm?
   - Is the online binarisation strategy better than the offline alternatives?
3. Can a combination of online and offline decision strategies improve the optimal decision tree algorithm?

This paper is constructed as follows: the second section describes the related work an research in the area of discretization and binarisation of data; the third section covers preliminary information, including the methodology used in the research. The fourth section

talks about own contributions to the research subject, describing the algorithms used to obtain the results to be compares. The fifth section outlines the conducted experiments. The sixth section covers Responsible Research. After that, discussion is featured in seventh section. The last section outlines conclusions and future research to be done in this area.

2 Related work

As mentioned above, the main contribution that is the basis of this research is the MurTree algorithm[1]. The data processing approach used when testing the MurTree algorithm assumed to first convert all non-binary features into categorical features based on the minimal description length principle (MLPD)[2], and to binarize them afterwards using one-hot encoding. The MDLP algorithm uses the heuristic of minimizing enthropy in each interval. However, the authors of the MurTree algorithm acknowledge that there might be better ways to prepare the data.

Another discretisation algorithm evaluated in this research is the CAIM discretisation algorithm[3]. The algorithm uses the heuristic of maximising class-attribute interdependence, while also possibly resulting in minimum amount of intervals.

Mayoraz and Moreira[4] give an overview of existing data binarisation methods for the purpose of classification tasks, and propose their own approach. Besides the binarisation algorithms, there is also research strictly focused on the discretization element. Yang Y. and others[5] present an collection of methods of discretizing data. Perner and Trautzsch[6] give another collection of discretization methods.

3 Preliminary information and methodology

3.1 Defining terminology

First, a decision tree will be defined. The non-leaf nodes of the tree are called feature nodes, as they take a feature (also called an attribute), and based on the value of that feature, use either their left or right child for further classification of the given data instance. The leaf nodes are called classification nodes, and independently of the features of given data instance, always classify it to one of two classes. The depth of the tree is defined as amount of levels with a feature node. A tree of depth 0 contains only a classification node, and is defined as the classification node of the given dataset.

As mentioned previously, the main measure to evaluate achieved

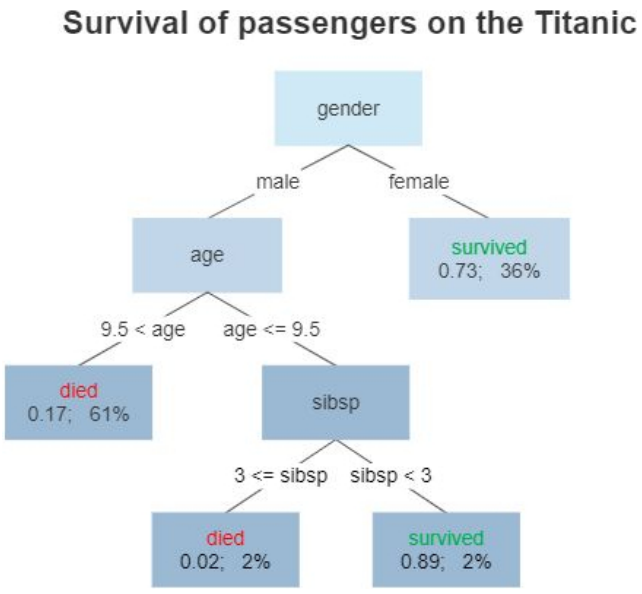

Figure 1: Example of a binary decision tree [7]

trees is accuracy. However, before it can be defined, other terms must be introduced first. A data instance is a concrete instance in the domain of a given dataset that represents a situation or object to be classified with the decision tree. For every data instance, there is a class it belongs to, and the purpose of the decision tree is to correctly decide on that class. If the decision tree decides to assign the given data instance to a class it does not belong to, it is a misclassification. A dataset is a collection of data instances in the same domain, representing a classification problem. A domain of a dataset is defined as a certain amount of attributes, where for each data instance, the given attribute has the same domain.

3.2 Accuracy measurements

Now, accuracy can be defined; it is defined as the percentage of data instances that is classified correctly by the given Decision Tree specific for that dataset. However, it is not proper to measure accuracy based on the exact same data instances as the ones used to create the Decision Tree, as this will not give a correct measure for the whole domain related to what the data represent, but only a result corresponding for the given specific set of instances that does not generalise well for the whole domain; this situation is called overfitting. On the opposite side of the spectrum there is underfitting of the model, where the model is not detailed enough to give accurate results, and by introducing greater details (which in case of decision trees is usually done by generating trees with a higher depth parameter). Therefore, for measuring accuracy, a separate validation set must be used[8], that was not used in the process of building the decision tree. By using the validation set as an independent measure, it is possible to notice underfitting and overfitting of the model.

The MurTree[1] algorithm as its measure of optimality uses the misclassification count, which is the absolute amount of misclassified features in the given training dataset. The MurTree algorithm, instead of doing exhaustive search over the whole search space, takes as one of the parameters the upper bound for misclassification score, under which the tree is assumed to be optimal. With this parameter set too high, an underfitted tree could be found and returned as optimal. However, when this parameter is set too low, either no tree is found, or the found tree is overfitted. As this parameter is very costly in time to control, it was decided to initially set the parameter as the dataset's classification node's misclassification score minus 1. The reason for the subtraction is to avoid premature convergence of the decision tree to an underfitted result. For every depth starting at 2 up until n, the MurTree algorithm is run for the maximum possible number of nodes. If tree at any depth yields a misclassification score lower than the allowed upper bound, the upper bound is set to the value for the following runs of the MurTree algorithms for higher depths. For benchmarking, n equal 4 was used.

3.3 K-fold cross-validation

The validation of accuracy measure is set up as follows. After the data is pre-processed, the data instances are divided into k sets, in such manner that every k-th data instance is placed in the k-th dataset. This allows each of the created sets to accurately represent the domain of the data, in case the data instances were sorted in any way. A structured way of dividing the data instances between sets was used to allow consistency and reproducibility of the results.

After the k datasets are created, the first k-1 datasets are used for a k-validation procedure. In this procedure, Each of the k-1 datasets is used once as the test set for a decision tree built using the rest of the k-1 datasets. The accuracies computed on the test sets are compared, and the tree with the highest accuracy between them is validated using the last kth set, that did not participate in the k-validation procedure. This allows to give a measure how well does the decision tree generalise on unseen data. The described procedure is applied for trees built with given depth parameter separately, So that overfitting can be noticed when for a tree with higher depth, a lower validation accuracy can be observed. For the experiments, the chosen k is 7, giving approximately 14,28% of data instances to the validation dataset, likewise 14,28% of data instances to each of the test sets, and 71,43% to each of the data instances to the training sets. The given k was chosen, as it is a good practice for the test and validation set to contain about 10%-15% of the initial datset.

3.4 Data

The datasets used for benchmarking are provided by the supervisor, which were collected from open source machine learning repositories[1]. For the use of this research, the subset of datasets was chosen, were at least part of the data was in continuous domain, as opposed to datasets already binarised. In case a dataset contained multi-class classification problem, the dataset was featured multiple times with one-vs-all classification, where in each instance of the dataset, a different class is set as target. The full list of datasets is as follows: Balance Scale[9], banknote authentication[9], QSAR biodegradation[9], Pima Indians Diabetes[10], Car Evaluation[9], Default of Credit Card Clients[9], Ionosphere[9], Iris[9], magic04[10], Messidor[11], MONK's problems[9], seismic bumps][9], spambase[9] and Wine[9]. In these datasets, the amount of data instances varies from 123 (3rd MONK dataset) up to 30 000 (Default of Credit Card Clients). The amount of attributes in the datasets varies between 4 and 57 (Spambase).

4 Comparing existing algorithms

There are three stages on which different algorithms will be compared. The first stage is the scope of the tree building algorithm, were various procedures for the latter two stages, discretisation and binarisation, will be compared. In this section, different algorithms for each of the stages are described.

4.1 Online and offline data processing

As mentioned earlier, the MurTree algorithm requires the data to be provided in binary format: every value of every attribute, including the target, must be either true or false. Offline data processing, or data pre-processing, is the standard procedure when building a decision tree. It requires the data to be pre-processed before it is used by the tree-building algorithm. While the MurTree algorithm does not modify the values of the data, during the runtime it divides the provided dataset into two separate subsets.

Online data processing will require the MurTree algorithm to be modified, such that the data will be provided in both continuous and binary format. Each time the dataset must be divided, the new subset will be created based on the continuous format of the data, and the discretization and binarisation will be recomputed.

Combined data processing, where offline and online data processing are combined, can be realised in many ways, but for the scope of this research it was decided to implement a limited version thereof. In this implementation, the MurTree algorithm will be provided data in categorical format (post-discretization), instead of continuous. On each division of the dataset, the binarisation will be recomputed.

4.2 Discretization algorithms

For comparing various discretization algorithms, two specialised algorithms and two additional naive algorithms were chosen. First, the specialized algorithms will be discussed, following with the description of the naive algorithms.

4.2.1 MDLP and CAIM discretization

The two specialised algorithms are MDLP[2] and CAIM[3]. Both algorithms work by dividing the data in continuous domain into few categories that are defined by certain ranges based on values of the continuous domain. The MDLP discretization works by the use of heuristic of minimizing entropy between multiple classes that are defined as intervals of the values of a continuous-valued feature. The CAIM discretization uses the heuristic of maximising class-attribute interdependence, while also possibly resulting in minimum amount of intervals. While both heuristics sound similar, there is a fundamental difference between them. MDLP discretization's entropy measure takes into account how homogenous a range is, while CAIM's interdependence considers a only how much the class labels are correlated with each interval.

4.2.2 Interval discretization methods

The two naive algorithms are defined as follows; mean-based interval discretisation and median-based interval discretisation. Both algorithms take in as parameter the amount of categories it should produce per attribute. Mean-based interval discretisation takes the lowest and highest values in the attribute, and defines the predefined amount of output intervals based of equal ranges of the values of the attribute. Therefore, the ranges may greatly differ in amount of data instances assigned to them. Meanwhile, median-based interval discretisation defines the ranges by sorting the values of the attribute in a ascending order and ensuring every range has equal, or close to equal, amount of data instances asssigned. This property of proportions of data instances between ranges of course only holds for the given dataset, and a different dataset from the same domain was chosen, the values might have been different. Both algorithms are defined as naive, as they both do not take into account the target value, therefore being unsupervised classification; but also they do not really take into account the features of the attribute domain to define the amount of output intervals.

4.3 Binarisation algorithms

For binarisation element, two algorithms were chosen: One-hot encoding and its proposed extension, hereby called Multi-hot encoding.

4.3.1 One-hot encoding

One-hot encoding proceeds as follows: in case of an attribute with 2 categories, it defines one of them as target, and the other one as a non-target category. In case of an attribute with n categories, it creates n new attributes, in each of which one category of the original column is defined as target, and the rest is defined as non-target. The original attribute is then discarded, and the resulting n attributes are added to the set of binarized attributes.

One-hot encoding was chosen, as it is the easiest way to preserve information, after the compressing done by discretisation algorithms. However, the one-hot encoding has the disadvantage that typically, the target value has a much smaller amount of data instances in it, compared to a non-target value. This is particularly inconvenient for the tree building, as the decision boundary created with such attributes has very unbalanced amount of data instances between each left and right child, which influences the granularity of the decisions and might simultaneously lead to underfitting in the left subtree and overfitting in the right subtree. Therefore, the extension multi-hot encoding was proposed.

4.3.2 Multi-hot encoding

The extension Multi-hot encoding is defined as follows: for attributes with m ranges, where m >= 4, additionally to one-hot encoding results, there also are created attributes where more than one of the ranges of the original attribute is defined as the target value. Initially, all possible combinations of ranges in output attributes were considered, however after some sample experiments on datasets, it was discovered that if an attribute created by multi-encoding was used in the tree algorithm over a one-hot encoding attribute, the ranges assigned as target values in the given multi-encoding column were always adjacent to each other.

As the initial setup was very time-costly due to the exponentially rising amount of attributes, which influences the runtime of the MurTree algorithm, the Multi-hot encoding is was redefined. First, the maximum amount of ranges that can be defined as target value in an output attribute, is $m - 2$, as $m - 1$ would be complimentary to one-hot encoding results. Secondly, for each amount of ranges that are set to target value in the output column, where that amount is defined as $2 \geq k \geq m - 2$, there is exactly $m - k$ output columns. For each of this output columns, k adjacent ranges are defined as target values. However, the k ranges that include the m-th column, is not included, as it would also be complimentary to a column were $(m - k)$ ranges that include the 1st column are set as target.

5 Experiments

In this section, the algorithms described in Section 4 will be compared using the methods and measures described in Section 3. The experiments and results for a chosen dataset will be discussed first, to give an overview of capabilities of different algorithms. Later, the results of benchmarking on the whole subset of continuous datasets are provided and discussed in order to determine whether there exist a better offline binarization strategy (Subquestion 1) and if online binarization strategy can improve the decision tree algorithm (Subquestion 2). Finally, possibilities of combining online and offline binarization strategies will be discussed (Subquestion 3).

5.1 Analysis for a chosen dataset

The dataset chosen for this analysis is the banknote authentication dataset[9]. It contains 1372 data instances and 4 attributes (excluding the target class), thus making it a small dataset, useful for building deeper trees, that otherwise would be very time-costly.

First, based on Fig. 2, results of various discretization algorithms combined with one-hot encoding binarisation will be discussed. While the control discretization method, MDLP (Section 4.2.1), gives reasonably high results and converges to a result without overfitting, better results are obtained by other algorithms. Mean-based interval and median-based interval results (Section 4.2.2) were obtained as follows: the results were run for various values of the input parameter (number of intervals per feature) p, where $2 \leq p \leq 8$. The details of this discretisation can be seen in Fig. 4.

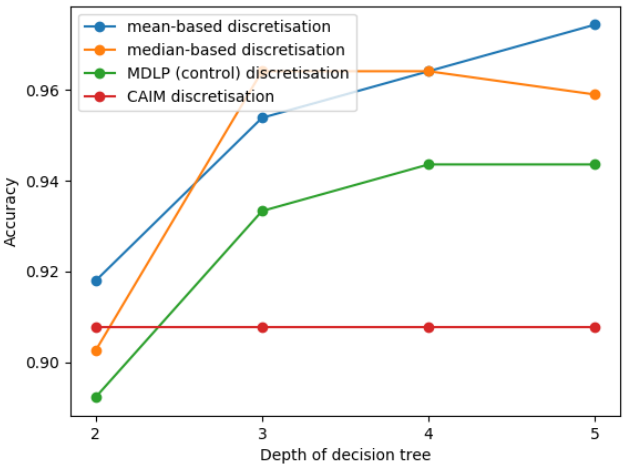

Figure 2: One hot encoding validation accuracies

For each depth, the result with the parameter value giving the highest validation accuracy was chosen. The mean-based discretization does not reach underfitting and with a tree of depth 5, provides the highest accuracy observed. Median-based discretization reaches the best accuracy for depth of 3, and above that depth shows signs of overfitting (while still giving better results than the control discretization). CAIM discretization shows a convergence to of accuracy since the lowest depth, thus providing stable results, which might allow better accuracy for very shallow trees. However, it is a disadvantage when using the MurTree algorithm for decision trees of higher depth, when considering offline data processing.

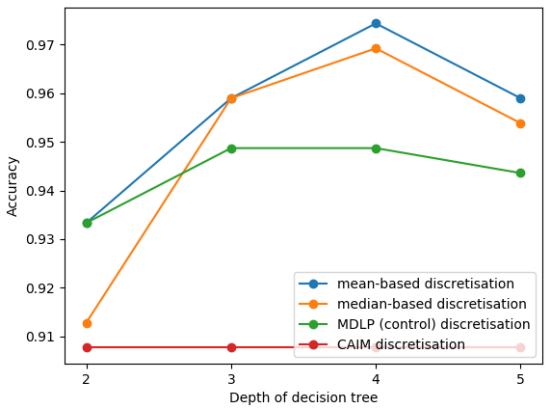

Figure 3: Multi-hot encoding validation accuracies

For Multi-hot encoding, in Fig. 3 the algorithms overfit for depth 5, when the validation accuracy is visibly lower than for depth 4. It is certain that the overfitting occurs, as in Section 3.2 it was explained that for the training set the misclassification for deeper trees can only lower, but not rise. Therefore, the train set accuracy for depth 4 and depth 5 must have been the same, but the validation accuracy is much lower. However, the best results for depth 4 match those for depth 5 of one-hot encoding, even exceeding them in case of MDLP and median-based discretization strategies. It is apparent that even for depth 2, the results of aggregating more ranges in one attribute give a considerable increase in accuracy. This outcome appears due to multi-

hot encoding allowing more balanced divisions of data.

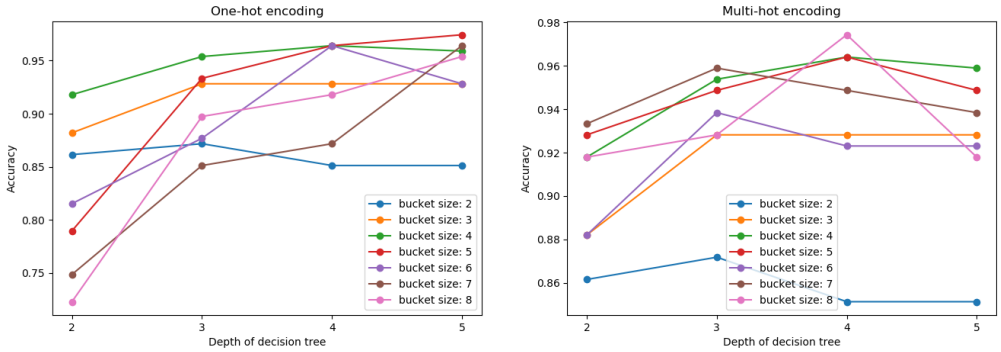

Figure 4: Mean-based discretisation results for different sizes of the input parameter

In Fig. 4, greater details about the differences between one-hot encoding and multi-hot encoding are visible. As explained in section 4.3.2, the results for bucket size 2 and 3 (when each attribute is divided into 2 or 3 ranges, respectively) are exactly the same for both types of bianrization. Again, much better results are observed for depth 2. With multi-hot encoding, there are benefits for using higher amount of intervals (buckets), with lower depths of the algorithm.

It is also visible that accuracy is not strictly increasing with the amount of buckets present: for one-hot encoding, the best results are observed for bucket size 4 and 5, while for multi-hot encoding, also bucket size 8 and 7 yield high accuracy. This can be explained by the fact that the interval-based algorithms, both mean-based and median-based, do not actually intend to find the "best" intervals for decision tree building, as they are unsupervised. However, some of the divisions into ranges manage to be close enough to a range or parameter that seems to be useful for decision tree building. The division into 4 and into 5 ranges per feature both manage to come close to the optimal ranges for at least some of the initial attributes. This also explains the success of division into 8 buckets with multi-hot encoding, when the results for multi-hot encoding with that discretization are less impressive. With one-hot encoding, the granularity of the ranges is too big, making the post-discretization categories too narrow for providing useful information. However, when some of the categories are merged together as with multi-hot encoding, it gives space for improvement.

With this explanation it becomes apparent that there is no single "best" parameter input for every dataset, as for different datasets, different divisions of the attributes into ranges will yield results close to the best ranges of that dataset. For a dataset, it might even yield better results to have a different amount of ranges for each attribute. A way to mitigate this would be to use a higher input parameter for interval-based discretization and use it with multi-hot encoding. To avoid division into sets of drastically different sizes, that could easily result in overfitting, multi-hot encoding could be modified so that per each output attribute, at least a m ranges are defined as target value. With current version of multi-hot encoding, m can be equal to 1.

The results for both naive discretization algorithms in Figures 2-4 must be discussed. After inspecting the decision trees giving highest validation accuracy, it was discovered that on highest levels of the decision trees, features which divide the dataset into two subsets of roughly the same size, with roughly the same percent of target data instances in each, are

promoted by the MurTree algorithm. Mean-based discretisation shows to be superior over median-based discretisation, especially for higher depths.

A reason for this could be that it allows the different intervals to have different amount of instances per each value, thus making features dividng the dataset in equal halves more likely to be present. These kind of features, instead of trying to find a measure to minimize entropy of each interval, provides a division that allows to use two different sets of rules for each of the datasets after divison. This puts into question whether specialised discretisation algorithms are useful for the MurTree algorithm? It must be stated that results obtained on one dataset are not generalisable; therefore, rejecting specialised algorithms at this step is not possible. Further, there is no evidence that any of the naive algorithms gives similar results for other datasets, so they cannot be proposed as a reasonable alternative yet. Another proposed reason of this state of being is that the specialised discretisation methods already give overfitting for such a small dataset. This might suggest that online data processing must stop at some depth of the tree, to avoid overfitting of the algorithms.

5.2 Benchmarking

In this section, results regarding comparison of the datasets listed in Section 3.4 will be discussed. To accurately display differences between different binarisation methods, box plots were used. In each box, the red line represents the median over all the datasets. The box represents two middle quartiles of the data and the whiskers represent the outer quartiles. The single dots represent outliers.

To compare results between various strategies in greater details, additional plots were used; the chance in accuracy was plotted against 3 properties of the dataset: amount of features, amount of data instances, and disproportion between target and non-target classes. The last measure is defined as follows:$|\frac{n_1}{n_1+n_0} - 0.5|$, where n_0 is the amount of data instances in the non-target class and n_1 is equivalent measure for the target class. Therefore, for this measure, for value 0.0, target and non-target class have the same amount of data instances, and for value 0.4, one class is ten times bigger than the other. Each dot represents a different dataset. Of those three values, amount of data instances is plotted in logarithmic scale for greater readability od the plot. Additionally, linear regression was plotted to mark a trend in the findings, if it exists.

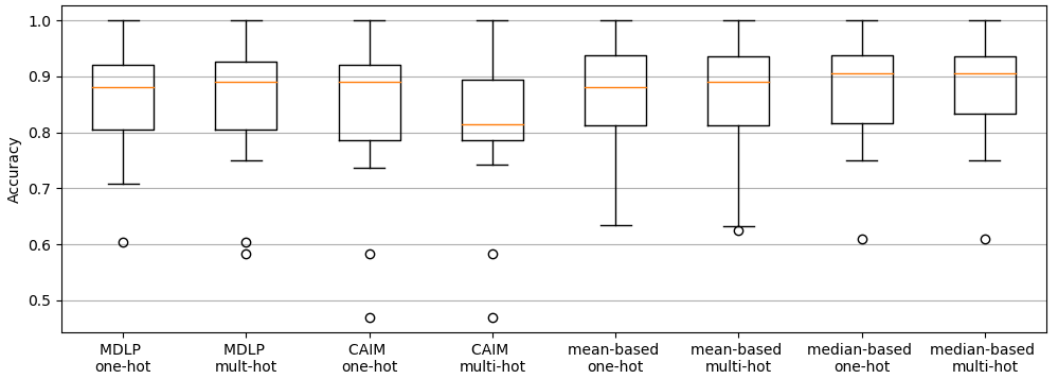

Figure 5: Comparison of best validation accuracies over the datasets listed in section 3.4

The input parameter for interval-based discretisation used was set to values between and

including 2 and 5.In case a decision tree building algorithm could not produce a decision tree with given misclassification score, it was excluded from the results.

5.2.1 Offline data processing

For offline data processing, the results were run up to depth 4 of the decision tree.

In Fig. 5, for each binarisation strategy the highest occurring accuracy was chosen. In case of interval-based discretization methods, the highest occurring accuracy was chosen regardless of input parameter, as the best parameter to use depends on the dataset itself. For each of the discretization strategy, it combined to one-hot encoding is displayed next to the combination with multi-hot encoding.

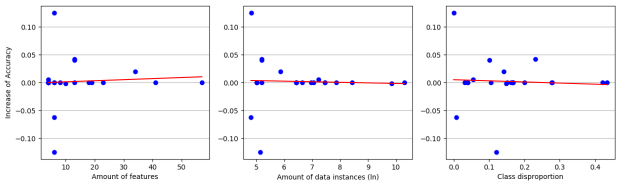

Figure 6: Comparison of control strategy versus MDLP with multi-hot encoding

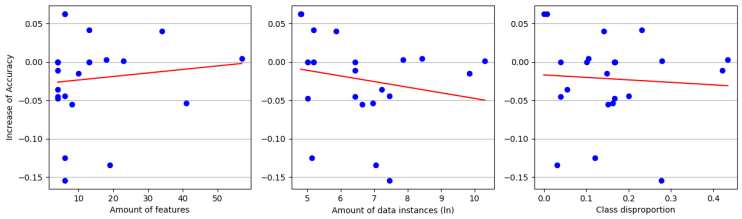

Figure 7: Comparison of control strategy versus CAIM with one-hot encoding

As such, it is visible that while the results for each type of encoding are very similar, they indeed do differ. For every discretization method besides CAIM, the results for multi-hot encoding are at least as good as, or visibly better, than for one-hot encoding. The difference in CAIM discretization can be explained as with multi-hot encoding, the training sets were much more prone to overfitting, so the best results were never reached.

For the specialised discretization algorithms, MDLP with one-hot encoding is preferred over MDLP with multi-hot encoding; as seen in Fig. 6, in most cases one-hot encoding is at least as good as the control strategy. The biggest change is observed for data with low amount of features and smaller datasets.

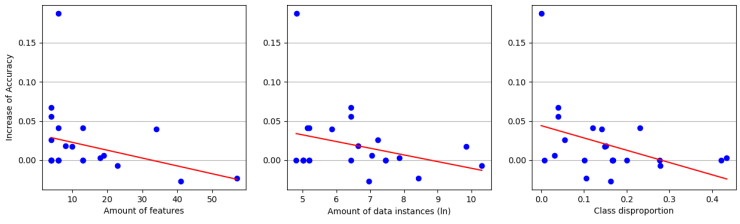

Figure 8: Comparison of control strategy versus median-based with multi-hot encoding

The control strategy also seems comparable to CAIM with one-hot encoding, however

from the comparison with Fig. 7 it is visible that for a lot of datasets the results are significantly worse, and the increase marginal. Therefore, it is not preferred over MDLP discretisation.

When inspecting the naive algorithms, median-based discretization gives higher accuracy (based on median) which much higher stability (looking at the lower half of the box). Overall, out of all offline binarisation-strategy, median-based discretization combined with multi-hot encoding gives the highest average accuracy, based on median and lower boundary of the box, with outliers being comparable with all other datasets. When inspecting Fig. 8, it is visible that only a few results, particulary those with a higher amount of feature, have marginally lower accuracy than the control strategy. This can be explained as for datasets with higher amount of features, the same parameter value for all features is less likely to give valuable results.

In conclusion for offline processing, the most promising strategy was the naive strategy for median-based interval discretization compared with one-hot encoding. However, the success of this strategy depends on the used parameter for amount of intervals. For the same reason, this strategy is less successful for bigger datasets, where the domains of different features might require different values of the parameter for a better use to the decision tree building algorithm.Since CAIM discretization compared with multi-hot encoding gives visibly worse results, it was decided to exclude it from further experiments and comparisons.

5.2.2 Online data processing

For online data processing, naive discretization algorithms were run for depths 2, 3 and 4. For specialized algorithms, it was discovered that running them for depth 4 often resulted in the algorithm not being able to find *any* division into ranges whatsoever in the lower parts of the tree and creating attributes were each value was in range $(-Inf, Inf)$. This resulted in much lower accuracy for depth 4. Therefore, it was decided to examine specialised online algorithms only for depth 2 and 3.

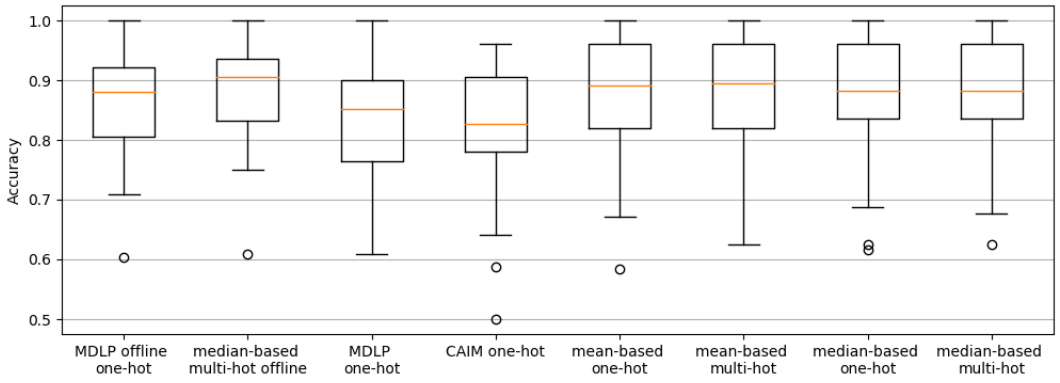

Figure 9: Comparison of best online accuracies

In Fig. 9, the results of online discretization were compared with control ofline discretization (MDLP combined with one-hot encoding) and best offline discretization (median-based combined with multi-hot encoding). For specialized algorithms, the median is visibly lower than control value, and the lower quartiles are much more spread downwards, thus making offline method worse for specialized algorithms.

This is the opposite of what was expected; it was presumed that discretizing using smaller datasets, after the division at the root of the tree, would provide a split that would make it easier to find meaningful splits, as the data not included into that branch of the tree would not introduce any error or division that would be not meaningful in the given branch of the tree. This assumption, however, was incorrect; trying to apply specialized discretization methods to smaller datasets resulted in the algorithms overfitting and not finding meaningful information.

When comparing the naive algorithms executed online with the two offline algorithms, it is visible that they have median accuracy a bit higher than the control strategy, but lower than the best found strategy. The top edge of the boxes is higher, thus it seems that in some cases, online strategies are more successful for naive discretization strategies; however, the lower whiskers is also longer for the online methods, thus making them less stable.

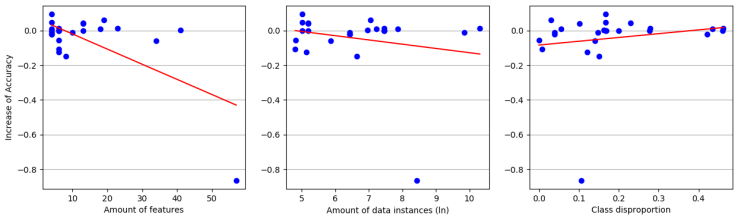

Figure 10: Comparison of median-based offline strategy with multi-hot encoding versus mean-based with multi-hot encoding

In Fig. 10, comparison was made between "best" established offline strategy and online mean-based multi-hot strategy, as it had the highest median from all online strategies. In this figure, it is visible that while a lot of the datasets can achieve a slightly higher accuracy, the increase is minimal, and much lower results can occur.

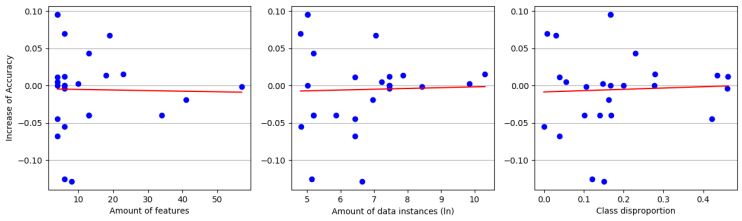

Figure 11: Comparison of median-based offline strategy with multi-hot encoding versus median-based with multi-hot encoding

However, in Fig. 11, it is visible that substantial increase over the offline strategy is still possible. While it gives no certainty and no stability in the results, it is still worth investigating. The most surprising is the increase of accuracy for datasets with bigger amount of data instances, which did not occur for any other strategies. It is possible that dividing bigger datasets into smaller parts does not lead to overfitting so quick, so an increase in accuracy

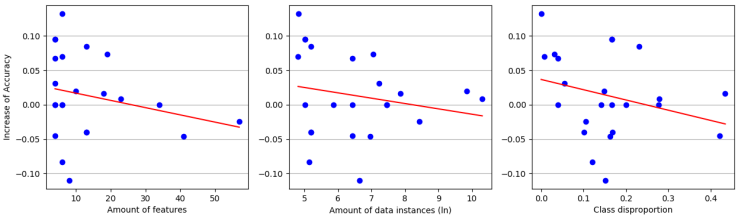

Figure 12: Comparison of control strategy versus median-based with multi-hot encoding

can be observed. In Fig. 12, it is also visible that compared to the control strategy, higher accuracy is more likely to occur than in Fig. 8.

6 Responsible Research

To ensure the reproducability of the research, the setup used has been described in section 3. The MurTree algorithm, that is the base of all implementation, was taken from source. To ensure that the implementation is correct, a test was run on a pre-agreed dataset, and the results were compared with the rest of the research group; as the built tree was identical between the implementations, it gave the assurance that all of the implementations are correct.

In the codebase for this research, all the methods used to obtain the results are documented. The codebase is versioned using git. While it is currently only in the private repository hosted on gitlab, there is possiblity to release the codebase on github, thus giving full access to the details of implementation and rescinding any doubts regarding reproducability.

A general issue that must be talked upon is possible application of the findings. The MurTree algorithm builds the optimal decision tree based on the data provided. If bias was to occur in the data, the decision tree based upon that data will propagate this bias. Therefore, the bias in the data must be treated independently by excluding certain features or ensuring the dataset is representative of the domain space, before applying any of the data processing described in the algorithm.

7 Conclusions and future work

When analysing offline strategies, the most promising strategy was the naive strategy for median-based interval discretization compared with one-hot encoding. However, the success of this strategy depends on the used parameter for amount of intervals. For the same reason, this strategy is less successful for bigger datasets, where the domains of different features might require different values of the parameter for a better use to the decision tree building algorithm. Therefore, for the future work, testing this strategy both for higher values of input parameter, and testing a version where parameter value varies between columns could be taken out. Additionally, more work could be put into analysing what exactly made certain division more successful than measures such as in specialised algorithms MDLP and CAIM.

Regarding online strategies, they proved to be less successful for specialised algorithms, as they quickly reach the point where the amount of data instances is too small and the algorithms cannot find any useful information in them. It did however find an increase of accuracy for both naive algorithms.

In the final conclusions, it is apparent that multi-hot encoding in most cases, bar some datasets with lowest amount of features and lowest amount of data, gives results at least as good as one-hot encoding. The biggest increase in accuracy was observed for datasets with small amount of data instances combined with low amount of features. For these kind of datasets, the most conservative way of increasing accuracy is to collect more data, as training decision trees with higher depth would lead to overfitting instead of increase in accuracy. After the findings of this paper, it is encouraged to try various methods of data processing for those kind of datasets, as the runtime for such datasets is negligible, and some

data processing strategies can yield better results, but never with 100% certainty. Therefore, trying some of those strategies could be a good alternative to the standard methods used.

References

[1] Emir Demirović, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J. Stuckey. Murtree: Optimal classification trees via dynamic programming and search. *CoRR*, abs/2007.12652, 2020.

[2] Usama M Fayyad and Keki B Irani. Multi-interval discretization of continuous-valued attributes for classication learning. In *IJCAI.*, pages 1022–1029, 1993.

[3] Ł.A. Kurgan and K.J. Cios. Caim discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16(2):145–153, 2004.

[4] Eddy Mayoraz and Miguel Moreira. Combinatorial approach for data binarization. In Jan M. Żytkow and Jan Rauch, editors, *Principles of Data Mining and Knowledge Discovery*, pages 442–447, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[5] Ying Yang, Geoffrey I. Webb, and Xindong Wu. *Discretization Methods*, pages 101–116. Springer US, Boston, MA, 2010.

[6] Petra Perner and Sascha Trautzsch. Multi-interval discretization methods for decision tree learning. In Adnan Amin, Dov Dori, Pavel Pudil, and Herbert Freeman, editors, *Advances in Pattern Recognition*, pages 475–482, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[7] Wikipedia user Gilgoldm. Decision tree showing survival of passengers on the titanic, 2020.

[8] Jason Brownlee. What is the difference between test and validation datasets?, 2017.

[9] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[10] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 20135.

[11] Etienne Decencière, Xiwei Zhang, Guy Cazuguel, Bruno Lay, Béatrice Cochener, Caroline Trone, Philippe Gain, Richard Ordonez, Pascale Massin, Ali Erginay, Béatrice Charton, and Jean-Claude Klein. Feedback on a publicly distributed database: the messidor database. *Image Analysis & Stereology*, 33(3):231–234, August 2014.