Efficient and effective feature discovery for CART decision tree model

Andreas Benedict Conrad Bien
Supervisor(s): Andra Lonescu, Rihan Hai
EEMCS, Delft University of Technology, The Netherlands
22-6-2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

A common challenge in feature discovery and feature selection is the trade-off between effectiveness and efficiency. The paper proposes a solution that is efficient and effective at ranking features for feature discovery. This paper aims to improve feature discovery techniques, by estimating the overall utility of features, through ranking them by their characteristics, such as the correlation coefficient, gini impurity, information gain, etc. The approach to estimate the overall utility is done by calculating the likelihoods of a feature being selected with a wrapper feature selection technique, given their ranking with respect to their characteristics. The likelihoods of the rankings are recorded and combined to estimate the overall utility of a feature which is used to rank all the features by their utility.

1 Introduction

In machine learning, there is a need to select features from datasets in order to overcome the risk of overfitting vs underfitting; if we select too many features we risk overfitting (where the model finds patterns in randomness) and if we select too few pictures we risk underfitting (where the model has too little information to accurately classify).

The problem that feature selection and discovery methods are facing today, is due to a trade-off between *efficiency* and *effectiveness*. The trade-off can be seen in the different types methods used to select features: filter methods, wrapper methods, embedded methods. Filter methods are very *efficient* at finding *suitable* features, however not very *effective* at finding optimal subsets of features. Wrapper methods on the other hand are robust and effective at finding the optimal features, however they are prone to overfitting and have a high time complexity. Finally there exist embedded methods which are a combination of both, which are more effective than filter methods, but also less efficient, and the inverse is true when compared to wrapper methods.

It is this trade-off between efficiency and effectiveness that the research question is trying to solve: *To train a decision tree model (CART), how to make the feature discovery process efficient and effective?*

This trade-off becomes even more critical, when performing feature augmentation with relational data.
The total set of possible features to consider in a relational data is much larger (scales badly with exhaustive feature selection approaches), and there is a computational cost that must be considered when deciding on joining tables[to join or not to join]. A unique sort of heuristic based on the characteristics of the features, that estimates the utility of joining tables, could be a useful tool to improve feature augmentation algorithms. For example the need to decide whether the computational investment of joining two tables is worth the utility it provides.

The paper consists of 7 sections where the following 6 sections respectively discuss related work, the methodology behind deriving the heuristic data, a section about the heuristic function that goes into depth on how the heuristic function is calculated as well as problems and solutions to those problems, the evaluation which explains the experiment as well as displaying results with analysis, a discussion about responsible research, and finally the concluding thoughts discussing the results and future work.

2 Related work

2.1 Feature discovery

Andra et al focuses on finding join paths to improve machine learning model performance with a two step process: enumerating join paths, and ranking join paths.[8] The article from J. Liu et al Focuses on automatic data acquisition, introducing a system AutoDaTa which automatically searches for training data.[1]

R. C. Fernandez et al tackles the problem of data discovery, introducing a system named AURUM. The system is responsible for building, maintaining and querying an enterprise knowledge graph (EKG), which is a representation of relationships between datasets.[2]

N. Chepurko et al builds off of AURA with their feature discovery tool they developed named ARDA which stands for automatic relational data augmentation. ARDA is a tool that combines heuristics with data discovery tools (like AURA) in order to discover new features relevant to the target variable.[4]

Jiabin Liu et al introduces a reinforcement learning framework AutoFeature which augments features. The framework focuses on an exploration-exploitation strategy which exploits frequently selected tables and explores occasionally selected tables.[5]

2.2 Feature selection

A. Blum and P. Langley go into depth with embedded, filter, and wrapper feature selection techniques.[6]

R. Kohavi and G. H. John discusse wrapper feature selection techniques such as the exhaustive feature selection algorithm (or FOCUS as the article describes it).[7]

3 Methodology

In order to **automatically** and **efficiently** discover features (when performing augmentation in a relational database), a heuristic could be developed, where the inputs are the features' characteristics, and the output is a metric on the utility of each feature. An example of such a function is the gini impurity index, where the inputs are the probabilities of the samples belonging to a class, and the output is the impurity of such a feature.

The focus is to investigate how the characteristics of a feature can be used to predict the likelihood of such a feature being chosen for the final set of features, which are fed to the CART decision tree model.

The likelihoods given by the characteristics of a feature being part of the optimal subset of features will serve as a heuristic to determine the utility of a feature which translates into the utility of joining tables in a relational dataset. The intuition behind using likelihoods is to have an unbiased general metric to judge all the characteristics by objectively, providing a sort of normalization tool to achieve a notionally common scale for all the different utilities. Analogy: Imagine a person who has many different bank accounts with wealth in many different assets (stocks, currencies, bonds, etc), calculating the person's total wealth requires converting all their assets to a single metric (like the USD). The same way a feature's utility can be seen through many different characteristics (correlation coefficient, information gain, gini impurity, etc), those utilities need to be translated into a common metric: the likelihood of being in the optimal subset of features. This serves as a baseline to compute the total utility of a feature.

In order to find these likelihoods, the first step is to select a set of features from the dataset using the most objective and optimal result-yielding feature selection methods. For this I plan on using the exhaustive feature selection method, as it is unbiased, robust, and theoretically should give the optimal set of features. If the feature size is too large to feasibly perform exhaustive feature selection, then I will first reduce the feature size with other feature selection methods.

Once we obtain the theoretically optimal set of features, the next step is to analyse the selected features, by calculating the probability of a feature being selected, given their feature characteristics.

For example, if 3 features from a set of 10 features are selected, where 4 out of the 10 features selected are categorical, and all 3 selected features categorical, then the **prior probability** is 3/10, the **posterior probability** P(feature is selected — feature is categorical) = ¾, and the **likelihood** is 2.5.

The final step is to perform the previous two steps for as many datasets as possible, and record the likelihoods in an array for all considered characteristics. With this array we can calculate the average likelihoods of for each quartile (which will be used for the heuristic function).

3.1 Making characteristics meaningful

A heuristic must be calculated on the characteristics of the data. With some characteristics, can be calculated in a more meaningful way. This was done with the correlation coefficient and gini coefficient, which is discussed below.

Correlation coefficient

: The correlation coefficient might show that 2 features are both highly correlated with a target, this can be misleading. If both features are also very highly correlated with each other, then maybe one feature should be dropped. To combat this, the features are ranked on the correlation of the data that the model incorrectly labels trained on higher ranked features. For example: Feature A is the highest target correlation, so we fit the model with A. We run the model on all the data and filter out all the rows the model incorrectly labeled. Feature D has the highest correlation when only considering the incorrectly labeled features, so we rank D 2nd and fit the model with A and D. This process continues until the model fits all features. If rows are left and only few features are left, then the remaining features are concatenated to the end of ranking.

Gini Impurity

: The same can be said for the gini impurity, two features might have low impurities when considering them alone, but also a low decrease in impurity when considering both. To combat this, the impurity is found by going through the tree structure of the fully trained tree with all features, ranking them by depth first then impurity and then removing any repeats. Since the tree automatically splits the rows into relevant subsets at each branch, with gini impurity as it's splitter.

3.2 Improving heuristic precision

One foreseeable drawback is the precision of the heuristic. The datasets used to calculate the likelihoods only contain max 20 features, as the exhaustive feature selection method is computationally expensive. Out of those features even less are selected. This means that the calculated likelihoods are closer to discrete values rather than continuous values. For example a dataset with 16 features where the possible amount of features selected range from 2-16 there are only 15 possible prior probabilities (2/16,3/16,etc). Furthermore, the posterior probabilities for each quartile only have 5 possible values (0/4,1/4,2/4, 3/4, 4/4), this means that there are only 75 possible values theoretically speaking, realistically speaking there are far fewer possible values. In order combat this problem of imprecision, there is a need to consider as many datasets as possible. This need was fulfilled by connecting to the kaggle api, and automatically downloads 2000 datasets and filtering out all the relevant datasets to consider.

3.3 Creating and Applying heuristic

Once the data is collected it needs to be transformed into a heuristic function that should score different features in different tables. The only requirement for the function is that when the features are ranked by their scores obtained from the function, the ranking should be accurate. Since the heuristic is based on likelihoods, a good way to score them is by estimating the *actual* (or posterior) probability that the feature will be selected. If we were to assume that the likelihoods given by the characteristics of the data are independent (which would be a wrong assumption) the formula would be $p\left(l_1 + l_2 + l_3\right) - p^2\left(l_1 \cdot l_2 + l_1 \cdot l_3 + l_2 \cdot l_3\right) + p^3 \cdot l_1 \cdot l_2 \cdot l_3$ where p is the prior probability that must be estimated and l_i are the different likelihoods a feature gets selected . In this paper we will not assume that the likelihoods are independent, however, due to the difficulty of calculating the independence of the different likelihoods, we will assume that using the formula stated above will still suffice for an accurate ranking. Meaning if a different heuristic function were to score the features with accurate probabilities, the ranking would be the

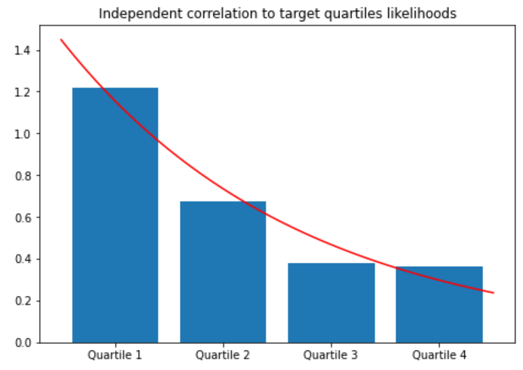
similar for both heuristic functions.
Finally, the heuristic function should continuous to allow for more precice scoring. For example if two features that both score in the top quartile for each metric, but one consistently ranks higher than the other, the discrete function would give both features an equal score, whereas the continouos function would give the first respective feature a more favourable score. To make the heuristic function continous a line of best fit is calculated for each metric.

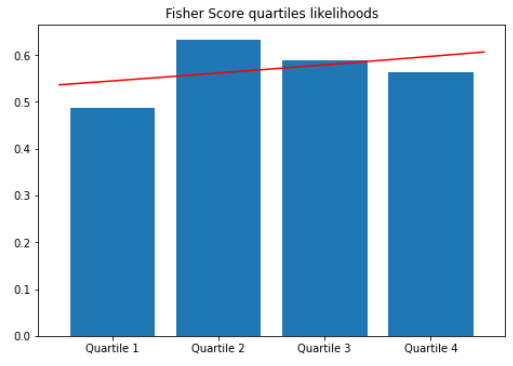
3.4 Evaluating heuristic

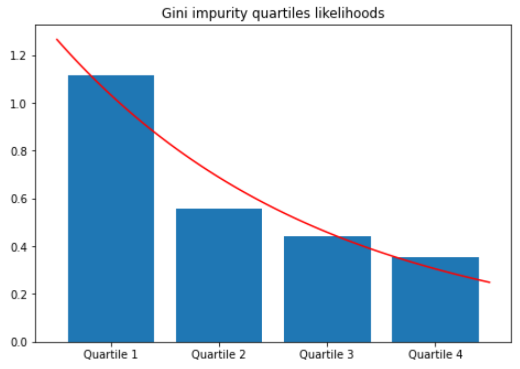
To evaluate the heuristic, experiments on relational datasets are done. The relational datasets contain a base table which contains the target feature. Then 3 experiments will be done using the CART decision tree model. The first is a baseline experiment, where we test the accuracy of the of the model when considering all features in the base table. The second is a dummy experiment where all tables are joined and accuracy with and without feature selection is tested. The final experiment is to test the heuristic, where we analyze the accuracy of features selected using the heuristic.

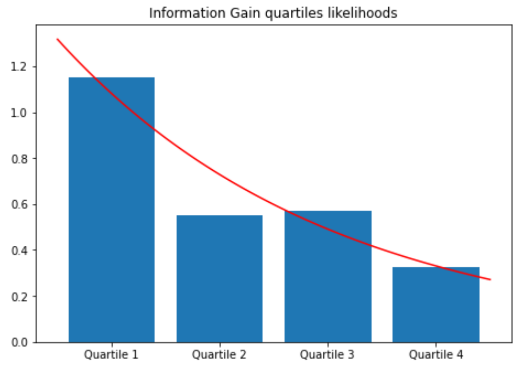
3.5 Heuristic

Data to develop heuristic





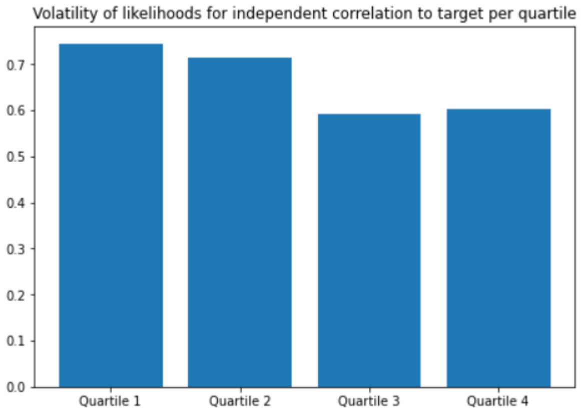


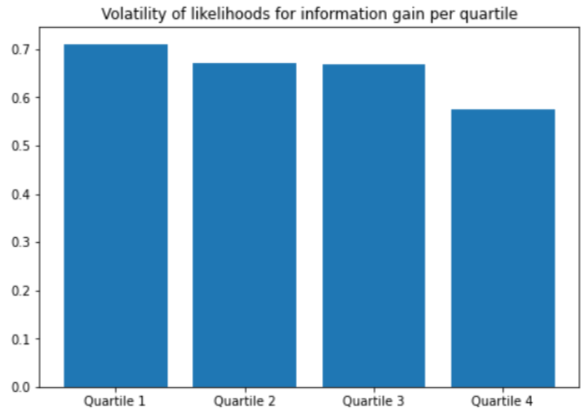


In the 4 figures above, are the likelihoods produced for each quartile (25%) in the ranking of each characteristic. The ranking was ordered in such a way for each characteristic, that theoretically quartile 1 should have the highest likelihoods and quartile 4 the lowest likelihoods. These figures were produced using over 100 datasets. The red line in the graphs indicates the line of best fit for the quartiles, in the form of an exponential function (ae^{bx}, we solve for a and b to find the best fit line, and x is the percentile [0.125,0.375,0.625,0.875] for each quartile). This best fit line will be used to calculate the theoretical likelihoods of more precise percentiles. For example if a feature ranks in the top 19th percentile, the value 0.19 is plugged in for x in $f(x) = ae^{bx}$ where a and b are the coefficients determined by the best line fit.

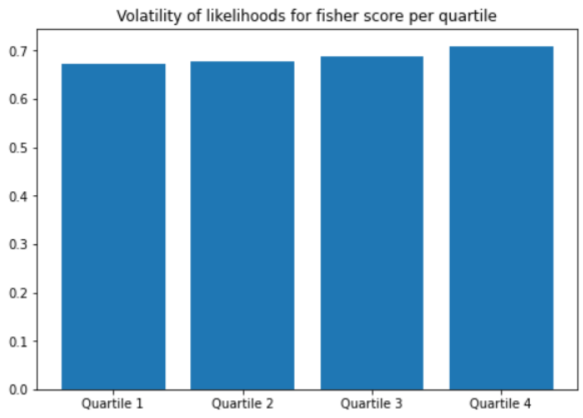
Analysis of heuristic data

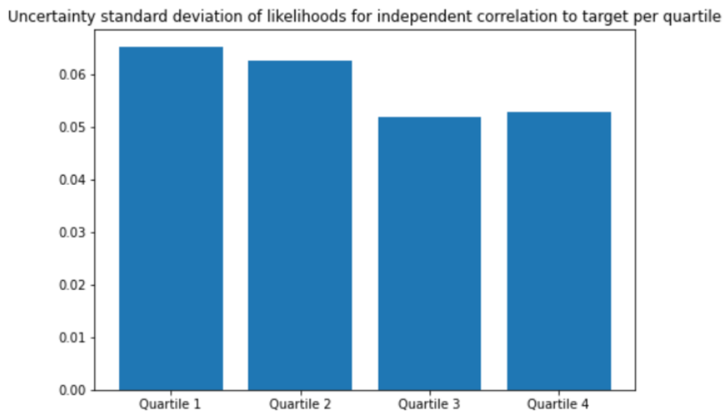
As seen in figure 2, the fisher score quartiles seem to be dominated by noise, which means would be a bad indicator if used in the heuristic. The first quartile which is supposed to perform best has the worst score, and all quartile values do not vary much from each other, the values range from likelihoods of 0.48 to 0.63, which is about a 0.15 range. Using this data for the heuristic will most certainly only introduce noise into the system. As a result, the methods in the code for evaluating the heuristics have an option to not include the fisher score. Furthermore, the fisher score will not be included in the heuristic. The counterproductive results obtained from the fisher score quartiles might be due to the algorithm overfitting and causing imprecision in the obtained data.

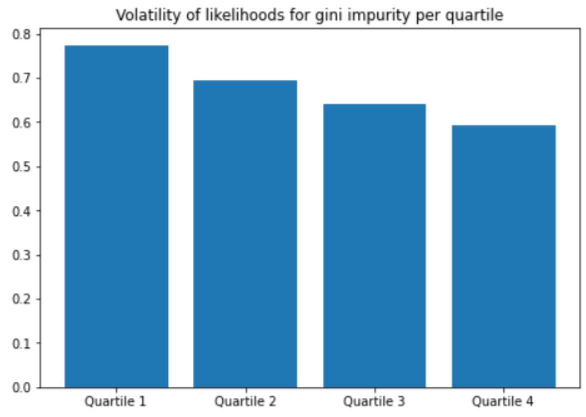


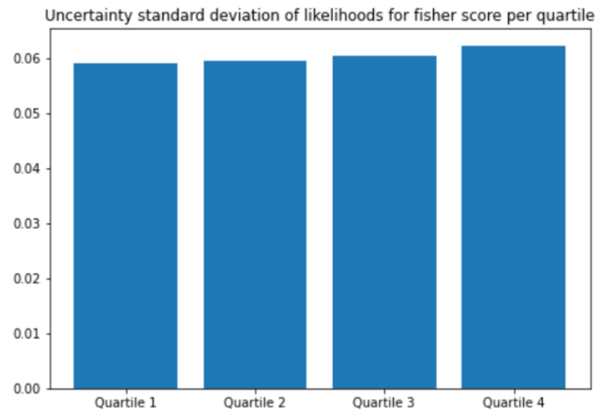


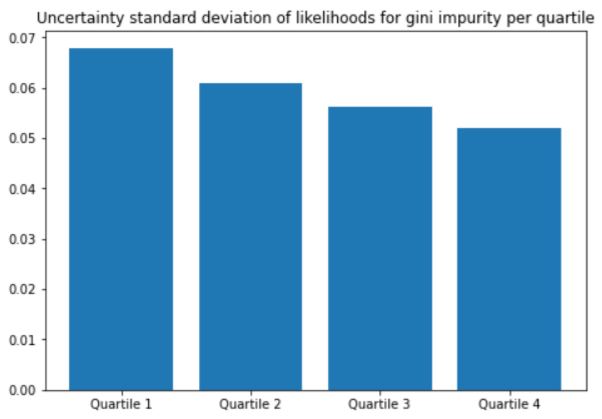
A major drawback of this method is the high imprecision in the results. In order to quantify this imprecision, the volatility of every quartile for each metric (correlation, fisher score, gini impurity, information gain) was calculated and is shown above. The plots below show the uncertainty of each quartile when taking when taking the amount of values into consideration; in other words it shows the volatility of the mean value for each quartile. The equation used is where n is the amount of values and v is the standard deviation of a single value is and $std(\frac{\sum v_i}{n}) = \frac{\sqrt{(vol(v)*n)}}{n}$

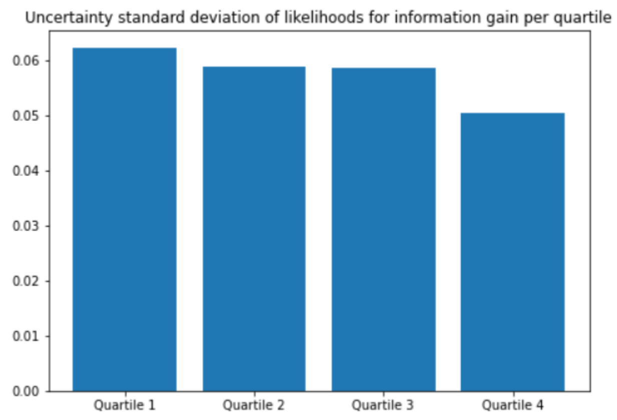












3.6 Experiment results

Below are the experiments with their results and analysis. The recursive feature selection (RFE) algorithm was used in the parts of the experiments where feature selection was used. It was chosen as it is an embedded method, meaning it is less biased than filter and wrapper methods towards certain characteristics, which provides for a more objective comparison between

Experiment baseline

This is where we test the accuracy of the of the model when considering all features in the base table.

Baseline Accuracy		
Dataset name	With feature selection	Without feature selection
football	0.62	0.63
kidney-disease	0.98	0.99
steel-plate-fault	0.74	0.73
titanic	0.52	0.53

Experiment dummy

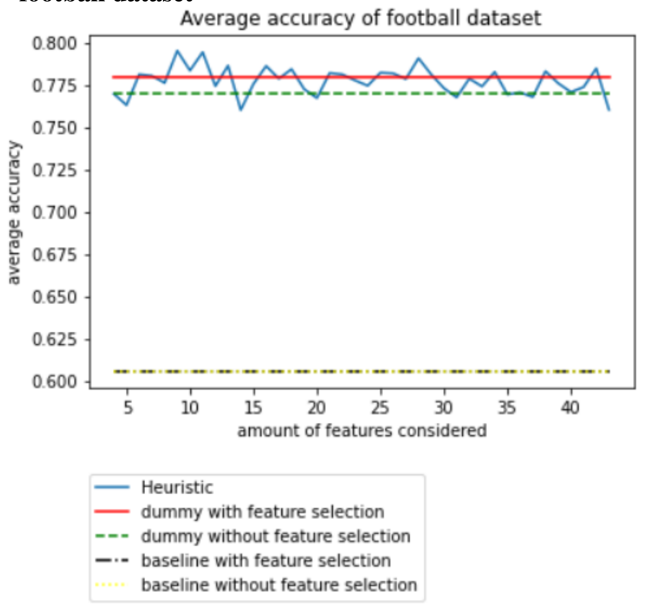
Here all tables are joined and the accuracy with and without feature selection is tested.

Dummy Accuracy		
Dataset name	With feature selection	Without feature selection
football	0.78	0.76
kidney-disease	0.99	0.98
steel-plate-fault	1.0	1.0
titanic	0.80	0.82

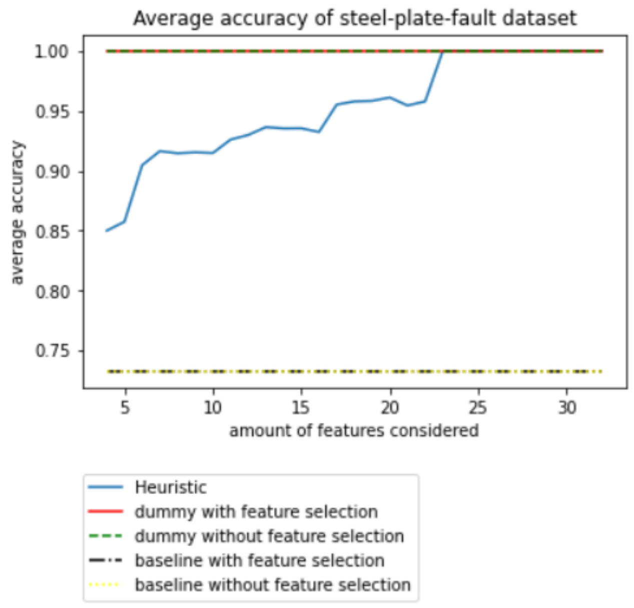
Experiment heuristic

In this experiment the heuristic is put to the test. The heuristic ranks all the tables on utility. The base table joined with the K-best individual features in other tables, then the recursive feature elimination algorithm is used to find the optimal subset of the K-best features. Finally the accuracy is recorded. If the heuristic is good then the results of the dummy experiment with feature selection should be similar or worse when compared to the results of the K-best features selected by the heuristic for small values of K.

football dataset

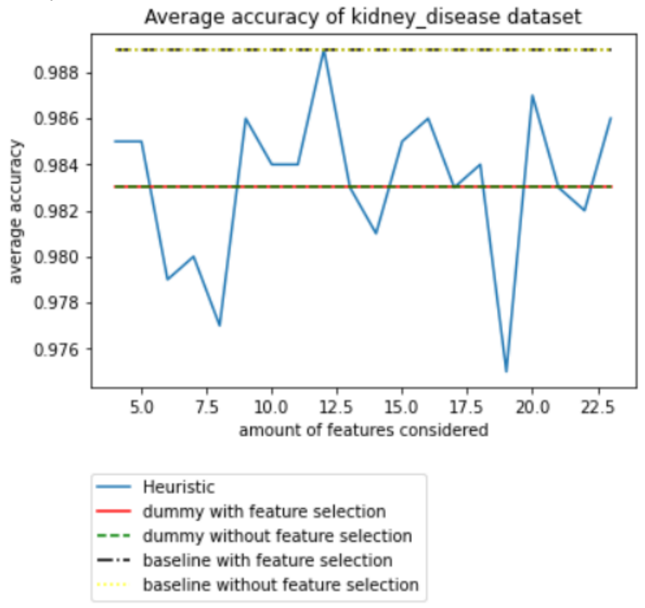


Here the heuristic is able to achieve similar accuracies compared to the dummy experiment, already by only considering 4 features. The average accuracy remains more or less constant. The heuristic, however outperforms the baseline accuracies by a significant amount. This example makes a great case in favor of the heuristic. **steel-plate-fault dataset**

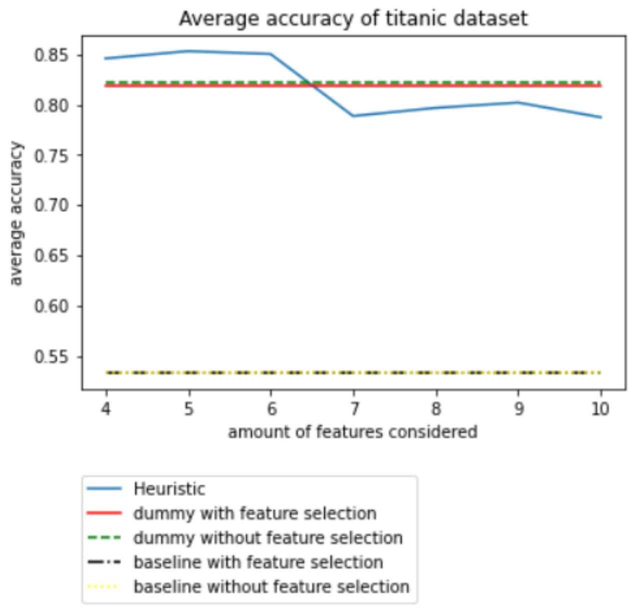


In this dataset the heuristic function performs the worst compared to the other datasets, however it still performs better than the baseline. This might be due there being many features with good characteristics (high correlation, low gini impurity, etc). With many well performing features so it becomes increasingly more difficult to rank them accordingly. This is because the heuristic function is strictly based on the feature's rank in each metric, if all feature's perform well on a metric (say correlation coefficient), some features will still be forced into a low rank. While the heuristic did not perform as well as the dummy experiment, it does show a positive learning curve; in future work the heuristic could also detect learning curves in order to determine whether it should keep joining more tables or stop.

kidney-disease dataset



The results in the kidney dataset are not very informative, as every experiment has accuracies above 0.975. The only thing that can be said is that the heuristic also performed well, so it does not make a case against. the heuristic. **titanic dataset**



For the titanic dataset the heuristic actually outperforms all experiment results, with the accuracy decreasing as more features are considered. This is likely due to the RFE algorithm slightly overfitting with larger values of K.

4 Responsible Research

The research required collecting large amounts of data, by automating the process of acquiring datasets from Kaggle. Investigating any potential biases and whether all the stakeholders of the datasets consented with the dataset being shared was infeasible. As a result there is a non-zero chance some datasets were used without the permission of their stakeholders as well as potential biases existing in the datasets. Moreover, the experiment is completely reproducable, however, there is no guarrantee that the same datasets used will be available in the future and due to some purposeful randomness in the code such as the train-test split the results are likely to have in slight but insignificant differences.

5 Conclusion

In conclusion, the heuristic performs well compared to the other accuracies of the experiment. However, as seen in the steel-plate-fault dataset, the heuristic is not guarranteed to score optimally with when the dataset contains many features with similar characteristics, as it scores based on rankings, which are less meaningful when the utilities of the features do not differ much. Furthermore, the data used to develop the heuristic is very imprecise, however the large amount of datasets considered compensates for the imprecision. That being said, the likelihoods based on the fisher score did not perform well, possibly due to the high imprecision of this method of obtaining a heuristic.

5.1 Future work

Combining with other heuristics

This heuristic is not meant to be a silver bullet for automatic feature discovery, but rather another tool to use in combination with other feature discovery tools. As mentioned in the paragraph above, the heuristic performs less optimally when the features do not differ much in their characteristics. Further research might include combining the heuristic developed in the paper with a heuristic that focuses less on ranking the features through metrics and focuses more on the metrics themselves.

Discovering dependence between likelihoods

Another area of research to consider, could be the dependence between the likelihoods of the metric. For example researching the correlation between the rankings of the different metrics, and possibly collecting data on the conditional likelihoods of the combined metrics as well. For example: what is the average likelihood of being selected of all the features in the first quartile of in the correlation ranking *and* second quartile in the gini-coefficient ranking. Some of this research would require magnitudes of more datasets to be considered, but it might provide valuable insight on how to combine the different utilities shown in characteristics.

Using machine learning to improve the heuristic

Finally, the data collected to create the heuristic could be fed to some machine learning algorithms, that can predict the utility of a feature given it's characteristics. Where the input would be the characteristics and the output would be likelihood of being selected.

A Some further guidelines that go without saying (right?)

- Read the manual for the Research Project. (See e.g. the instructions on the maximum length: less is more!)

A.1 Reference use

References

[1] J. Liu, F. Zhu, C. Chai, Y. Luo, and N. Tang. Automatic data acquisition for deep learning. PVLDB, 14(12):2739–2742, 2021

[2] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker. Aurum: A data discovery system. In ICDE, pages 1001–1012, 2018. [18] R. C. Fernandez, E. M

[3] N. Chep- urko, R. Marcus, E. Zgraggen, R. C. Fernandez, T. Kraska, and D. R. Karger. ARDA: automatic relational data augmen- tation for machine learning. VLDB, 13(9):1373–1387, 2020.

[4] N. Chep- urko, R. Marcus, E. Zgraggen, R. C. Fernandez, T. Kraska, and D. R. Karger. ARDA: automatic relational data augmen- tation for machine learning. VLDB, 13(9):1373–1387, 2020.

[5] Liu, J., Luo, Y., Luo, Y., Feng, J., amp; Tang, N. (2022). (rep.). Feature Augmentation with Reinforcement Learning. Tsinghua: Tsinghua University.

[6] A. Blum and P. Langley. Selection of relevant features and ex- amples in machine learning. Artif. Intell., 97(1-2):245–271, 1997.

[7] R. Kohavi and G. H. John. Wrappers for feature subset selection. Ar- tif. Intell., 97(1-2):273–324, 1997.

[8] Lonescu, A., Hai, R., Fragkoulis, M., amp; Katsifodimos, A. (2022). (tech.). Join Path-Based Data Augmentation for Decision Trees. Delft, South-Holland: Tu Delft.