Automatic feature augmentation ranking: XGBoost

Oliver Neut
Supervisors: Dr. Rihan Hai, Andra Ionescu
EEMCS, Delft University of Technology, The Netherlands

20-6-2022

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

Automatic machine learning is a subfield of machine learning that automates the common procedures faced in predictive tasks. The problem of one such procedure is automatic data augmentation, where one desires to enrich the existing data to increase model performance. In relational data repositories, the data is stored in normal form. This causes problems, since joining all tables and subsequently performing feature selection is highly inefficient. This paper provides AFAR, an approach to efficiently and effectively perform automated feature augmentation by ranking candidate joins in a data repository. Additionally, an experimental evaluation that validates the approach's capabilities, is presented.

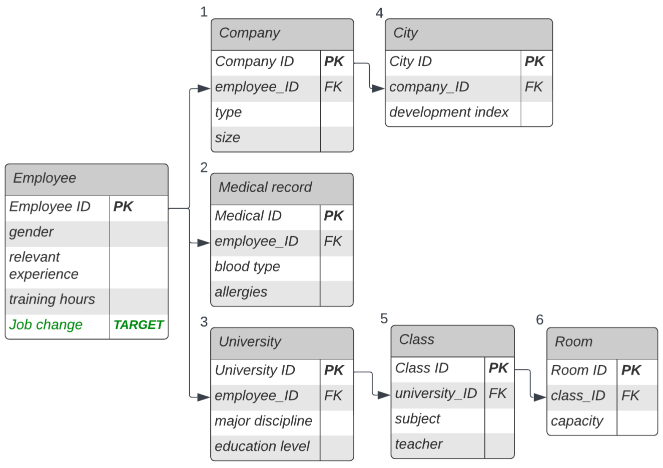


Figure 1: Example of relational data repository

1 Introduction

An increasing amount of machine learning users are non-experts in the field and rely upon automated ML to perform predictive tasks in their professional environment [12]. In most cases, users of these systems are prompted with providing a dataset containing features and their corresponding labels and a goal on which to optimize [11]. Automatic machine learning (AML) is a subfield of machine learning that focuses on automating the process of building predictive models. The goal of AML is to compose a coherent pipeline of ML tasks: data cleaning, feature engineering and hyperparameter tuning.

Problems arise when model performance is not satisfactory, this can be due to the quality of the dataset that is relied upon. A solution to this is to augment the data with new features that can be obtained from a relational data repository. When joining new data to the existing base table by means of key-foreign key joins, it is not assured that the model performance (F1-score, accuracy...) will increase. Exhaustively joining all tables, in order to obtain a better performing model, is a computationally expensive task, that tends to overfit often.

To illustrate the problem in-depth and demonstrate the need for a viable solution, we look at an example. Figure 1 depicts a database diagram of the predictive problem of whether an employee will stay at a company. There are many different ways to augment the base table with new features. Joining the *Employee* with *Company* could benefit the model's performance. Consequently, the transitive join of *Employee* with *City* could entail an even higher accuracy since the city development index likely has an impact on the employee being satisfied. However, the transitive join of *Employee* with *Class* will likely not result in a better performing model. This is not as intuitive for a machine compared to a human to detect. To perform all combinations of joins on the table is an expensive task and not scalable for AML systems. Thus more efficient and effective solutions are needed for feature discovery situations like this.

The challenges in this field consist of whether the automatic feature augmentation approach is effective and efficient. Meaning, does it entail a good accuracy for the predictive task your model tackles. This can be validated by comparing it against the performance of the base table. Secondly, does the data augmentation method add an acceptable amount of time to the AML pipeline? By which a more efficient way of selecting tables is used as opposed to iteratively joining all possible key-foreign-key joins and ranking the tables based on that.

"To join or not to join", by Kumar et al. (2016) solves part of this problem by identifying the joins that can be avoided [8] without significantly affecting the accuracy of the model. Since this leaves potential joins that could be model performance boosters, Chepurko et al. (2020) present ARDA [3], a complementary system. It allows for automatically augmenting a base dataset with new features by selecting beneficial joins from a data repository. Since both only address the problem of joining tables directly associated with the base table with key-foreign-key joins. They neglect the fact that a feature could be found on a join path of more than 1 hop. Tables 4, 5 and 6 in figure 1 illustrate this problem. Besides this, these approaches do not look into other performance metrics such as overfitting.

The paper answers the question: "how can automatic data augmentation be applied efficiently and effectively in an AML pipeline that uses XGBoost". The main motivation for this research is to make the process of feature discovery more effective and to improve the efficiency of data augmentation by narrowing the problem down to 1 specific model. Secondly, its robustness is validated by combining it with other classifiers. The XGBoost decision tree classifier or extreme gradient boosting is an ensemble model that is known for being one of the fastest models in the space and is applicable to a wide range of problems [2]. While extreme gradient boosting is able to build models for regression and multiclass classification problems, the main focus of this paper remains within the boundaries of binary classification.

In summary the following contributions are made:

- A set of heuristics to select joins in the data augmentation process for XGBoost in an efficient and effective manner.
- An approach AFAR (Automatic Feature Augmentation Ranking) that ranks join paths from a relational data repository, using the previously defined heuristics.
- An extensive experimental evaluation using 4 datasets to validate that the approach AFAR:
  - selects candidate joins that have a similar accuracy as compared to selecting all joins
  - selects candidate joins that have a smaller depth as compared to selecting all joins
  - has a significantly faster run time compared to selecting all joins
  - is robust against other decision tree algorithms

Section 2 explains the background of the research in more depth by discussing the related work. Section 3 presents the approach: AFAR (Automatic Feature Augmentation Ranking), the methodology of how it is derived and an overview of the approach. The experiments and their evaluation are described in section 4. In section 5, the responsible research is discussed. Finally, a conclusion is outlined in section 6.

2 Related work

The following section covers the range of problems and its existing work related to automatic feature augmentation.

2.1 AML

Automatic Machine Learning (AML) has recently arisen as a separate field of study aimed at automating the method of establishing the optimal machine learning pipelines for a given learning objective [3]. These systems enable the possibilities of performing machine learning for people unfamiliar with ML, by automating feature selection, model selection and hyperparameter tuning. Alpine Meadow, Google Cloud ML, Auto sklearn and Microsoft Azure Auto ML are examples of existing AML tools [11]. However, these systems do not automate feature augmentation since their input requires a complete dataset.

2.2 Feature selection

Feature selection is aimed at selecting a subset of the available features in a given dataset, that entails the best possible model performance. It consists of 3 categories: filter, wrapper and embedded [8] [6]. In filter feature selection, univariate and multivariate evaluation metrics (Pearson correlation, Spearman correlation, Gini index, mutual information) are computed and the best k features are kept [1]. Wrapper methods, attempt to find the optimal subset of features by iteratively selecting features and training the model to obtain the best possible performance. The main downside of this method is computational cost, since retraining is done exhaustively [5]. Existing wrapper methods struggle in the presence of a high-dimensional feature space due to a lack of scalability [4]. Finally, embedded methods are feature selection methods built into the ML model. For example, decision trees only select features for which the information gain after a split is satisfactory, thus not all features in the input dataset are guaranteed to be selected by the model.

2.3 Feature augmentation

In *"To join or not to join"*, Kumar et al. (2016) present a set of rules that can accurately predict a priori whether a join is safe to omit [8]. These are validated by an empirical analysis using real-world datasets. However, the candidate joins excluded by their method do not necessarily leave the valuable joins.

ARDA attempts to do the discovery of top-n joins that improve model performance [3]. This is done by sampling the base table and iteratively joining different tables and ranking the joins with a feature selection method that compares features against random noise. The sampling of the base table ensures that the tables join faster.

Liu et al. (2022) take a deep learning approach to data augmentation with their AutoFeature approach [9]. It attempts to explore features that are commonly selected and at the same time exploit the ones that are rarely selected. It characterizes features based on variance, Pearson correlation and Mutual information, indicating the importance of a specific feature.

2.4 XGBoost preliminaries

XGBoost or extreme gradient boosting is a highly scalable decision tree ensemble learning algorithm. Ensemble meaning that it combines many small decision trees called weak learners to obtain its classification or regression model. Boosting is a special kind of ensemble model that iteratively builds decision trees that compensate for the residual error of the previous decision tree. The resulting prediction of an XGBoost model is the weighted sum of all weak learners' predictions. Figure 2 gives an overview of how an XGBoost model works on a high level.

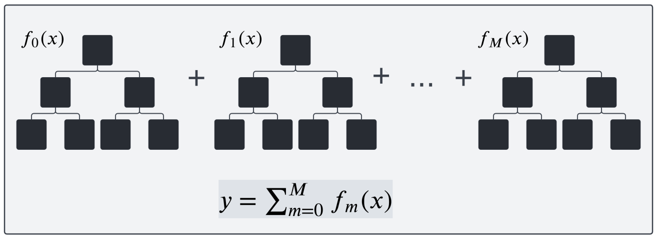


Figure 2: Overview of an XGBoost model with M weak learners

3 AFAR: Automatic Feature Augmentation Ranking

The steps taken to achieve the approach are explained in the following section. First, a clear description of the problem is provided. After this, the methodology used to attain the approach is explained in detail, followed by an overview of the procedure that the approach applies.

3.1 Problem overview

The problem that ought to be tackled in this paper is to find a method to augment a base table in a data repository in an efficient and effective manner. This data repository contains a set of join paths which need to be ranked in decreasing order based on the improvement in model performance. When augmenting the base table with new features, the accuracy of the model is projected to increase. For an AML system, selecting appropriate join paths is not a trivial problem. However, a greedy approach to obtain a significantly better model performance is to join all candidate tables with the base table. By doing this, the model will make use of both informative and uninformative features. The latter will consequently affect the model's ability to generalize and thus cause overfitting.

To summarize the problem goal: we want to derive a feature augmentation system that is able to achieve a model accuracy similar to a greedy approach (selecting all join paths) while significantly lowering the time to augment the dataset and reducing the overfitting of the model. This is done by ranking the join paths on their ability to improve the model accuracy.

3.2 Methodology

In order to obtain a solution that is tailored to the XGBoost decision tree model, a number of approaches can be taken. The aim is to use methods that do not have much computational overhead and are effective in selecting useful features. First, feature characteristics are examined in depth. After this, we look into filter methods by examining the feature-target correlation and feature-feature correlation. We do not look into wrapper methods as these require retraining the model and thus have high computational overhead [5].

Feature characteristics

We select features in candidate tables based on feature characteristics that are commonly selected by the XGBoost algorithm. The motivation for doing this stems from the fact that it requires very little time to identify the features based on heuristics. We narrow down the univariate analysis of the features by analysing based on 4 measurements.

The characteristics of features can be defined by the following 4 descriptive properties:

- frequency distribution of values
- mean: central tendency of values
- variance: dispersion of values
- categorical vs. numerical

To gather insights into the commonly selected features, we look at 10 datasets [1] containing a mix of categorical and numerical features. We identify the top features in the resulting XGBoost decision tree model and look for common characteristics of these features. After training the XGBoost model on the raw data, a majority of categorical features can be seen at the top of the decision tree. For 10 datasets, the top feature was a categorical feature 77% of the time and for the top 5 features, the average was 75%. This heuristic of including more categorical data can be included in the approach using an intuitive formula.

The XGBoost algorithm natively does not support categorical data for binary classification [2]. To use categorical features, the data needs to be encoded. One-hot encoding and label encoding are examples of doing this. If a *color* feature contains string values: "green", "red", "blue", One-hot encoding creates 3 new features *color_green*, *color_red*, *color_blue* that consists of binary values indicating whether the category is present. For categorical data with many values like *country*, this causes a considerable increase in the number of features, which is undesirable. Label encoding is done by converting the different categories of a feature into numerical values. For XGBoost, splitting feature subsets to decrease the residual values is done solely with inequalities of numerical values.

Since categorical data contains a small amount of uniquely encoded values. We can represent the usefulness of a feature by scoring on $\frac{1}{n}$ with $n = \#unique\ values$. That means that a binary feature will have the best score, 0.5 and for the $\lim_{n\to\infty} \frac{1}{n} = 0$, the score approaches 0.

Note that we disregard the use of identifiers and or any other unique values in our approach because these do not have informative properties and only tend to overfit the model.

Filter feature selection

Filter feature selection is a type of feature selection that is model agnostic, meaning its outcome is independent of the machine learning model [10]. It scores features by performing statistical tests which calculate the correlation between a feature and its target. Filter methods include the following: mutual information, ANOVA, Pearson correlation, Spearman correlation, Gini index, ReliefF (and more) [7].

In order to perform feature selection between attributes of the base table and the candidate table, we need to join the candidate table with the base table. Since this is rather inefficient, we attempt to limit this $O(n * m)$ task by making use of sample joins. There are two types of sample joins: uniform and stratified sampling. In uniform sampling, a random sample of a specified size is taken from the base table. While this is a viable method, it can misrepresent the distribution of the target variable. That is exactly what stratified sampling solves. It ensures an equal ratio of values in the target column of the base table and in the sample table.

We aim at finding features with high filter selection scores, as these will have a positive impact on the classification accuracy. However, this does not guarantee that this information is not already provided by the features of the base table. We can counter this problem by examining the correlation values between features from the candidate table and the base table. This ensures that the model performance will be improved by the subject feature. In summary, we follow the heuristics below:

- Use stratified sampling in order to join a smaller portion of the data to perform filter methods:
  - to select features with high correlation with target features.

[1]https://github.com/oliverneut/CSE3000_AFAR/tree/main/datasets

    – to select features with low correlation with base table features.

- Select candidate tables that have a high average feature score for $\frac{1}{n}$ with $n = \#unique\ values$

3.3 AFAR overview

The overview of the approach AFAR is formally described in this subsection.

Formal description

The join path rank algorithm has as input a base table T_B with a target feature F_t and a set T of n candidate tables $T_{C1}, T_{C2}, \dots T_{Cn}$ along with the set K of n KFK join tuples $(pk_1, fk_1), (pk_2, fk_2)...(pk_n, fk_n)$. We ought to rank the join paths based on how well they will improve the model performance in terms of accuracy in decreasing order.

The algorithm only ranks the usefulness of the candidate table and not the possible tables between it and the base table. When the ranking proposes table 6 in figure 1, it will join table 6 but train the model only on the columns table 6 adds to the base table. By doing this, we evaluate the improvement of the candidate table on the model performance in isolation and we can validate the effectiveness of our approach better.

Algorithm 1 Join path rank
```
Input:
KFK join tuples $K = \{(pk_1, fk_1), (pk_2, fk_2)...(pk_n, fk_n)\}$,
base table $T_B$, candidate tables $T_C = \{T_{C1}, T_{C1}...T_{Cn}\}$ with
$n \geq 1$, target $F$
Output: ranked list of join paths $Q$
  $paths \leftarrow \{\}$
  for $T_{Ci}$ in $T_C$ do
      $paths \leftarrow paths \cup left\_samplejoin(T_B, T_{Ci}, K)$
  $Q \leftarrow rank(paths)$                    ▷ Rank_1 or Rank_2

  return $sort\_descending(Q)$
```

Algorithm 2 Rank_1
```
Input: base table $T_B$, target $F$, $paths$, candidate tables $T =$
$\{T_{C1}, T_{C1}...T_{Cn}\}$ with $n \geq 1$
Output: ranked list of join paths $P'$
  $Q \leftarrow \{\}$
  for augmented table $C_i$ in $paths$ do
      $c_t \leftarrow pearson\_corr(T_{Ci}, F)$
      for feature $f_i$ in $T_{Ci}$ do
          $c_f \leftarrow max(pearson\_corr(T_B, f_i))$
          $c_t[i] \leftarrow c_t[i] * (1 - c_f)$
      $Q[i] \leftarrow max(c_t)$
  return $Q$
```

Join path rank (alg. 1), shows the pseudocode for ranking the join paths. For each join path, it performs a left stratified sample join since we want to preserve all the values in the base table. After all the candidate joins have been joined to the base table, we end up with n unique augmented tables

Algorithm 3 Rank_2
```
Input: base table $T_B$, target $F$, $paths$, candidate tables $T =$
$\{T_{C1}, T_{C1}...T_{Cn}\}$ with $n \geq 1$
Output: ranked list of join paths $P'$
  $Q \leftarrow \{\}$
  for augmented table $C_i$ in $paths$ do
      $c_{t1} \leftarrow pearson\_corr(T_{Ci}, F)$
      $c_{t2} \leftarrow information\_gain(T_{Ci}, F)$
      $c_{t3} \leftarrow gini\_index(T_{Ci}, F)$
      $c_{t4} \leftarrow mean(\frac{1}{\#unique\_values(T_{Ci})})$
      for feature $f_i$ in $T_{Ci}$ do
          $c_f \leftarrow max(pearson\_corr(T_B, f_i))$
          $c_{t1}[i] \leftarrow c_{t1}[i] * (1 - c_f)$
      $Q[i] \leftarrow [c_{t1}, c_{t2}, c_{t3}, c_{t4}]$
  $Q\_normalized = normalize\_columns(Q)$
  $Q\_sum = sum\_columns(Q\_normalized)$
  return $Q\_sum$
```

$C_1, C_2, ..., C_n$. For each C_i we can now compute a score which will rank the candidate table for its effectiveness. Two rank functions are proposed, Rank_1 uses Pearson correlation and Rank_2 extends this with more filter methods to score its candidate tables.

Rank_1 (alg. 2) iterates over each augmented table C_i. It then computes for each feature f_i of C_i (except the identifiers) that is an element of the candidate table T_{Ci}, the *Pearson_correlation* score for feature f_i and target F_t and stores it in target correlation list c_t. Then, the *Pearson_correlation* scores for f_i and the features from T_B are computed. The max feature-feature correlation is stored in feature correlation c_f. Then $c_t[i]$ is multiplied by the highest non-correlation c_f, by storing $c_t[i] = c_t[i] * (1 - c_f)$. A high score for $c_t * (1 - c_f)$ means a high feature-target correlation and a high feature-feature non-correlation. This score is stored in Q.

Rank_2 extends Rank_1 with more filter methods and the feature characteristic formula. It iterates over each augmented table C_i. In addition to the *Pearson_correlation*, it computes the *information_gain*, *Gini_index* and the mean of the unique values score of the candidate table T_{Ci}. These scores are stored in $Q[i]$ after which they are normalized, so the scores are treated equally. Then these scores are summed and Q_sum is returned.

4 Experiments evaluation

This section covers the experiments done on the approach proposed in the previous section. Additionally the experiments validate the following:

- AFAR achieves a similar or slightly worse model accuracy by joining the top candidate table compared to joining all tables.
- AFAR achieves a smaller model depth by joining the top candidate table compared to joining all tables.
- AFAR has a significantly faster runtime by joining the top candidate table compared to joining all tables.

- AFAR is robust against other classifiers: Support Vector Machines, Random Forests and CART decision tree algorithm.

4.1 Datasets

For the experiments, we consider 4 datasets which are suited for binary classification: 'kidney-disease', 'football', 'steel-plate-fault' and 'titanic'. Each of these has a base table and a number of candidate tables which can be joined by primary key-foreign key join relationships. The datasets and code for the evaluation are publicly available in the Github repository[2].

4.2 Metrics

To optimally assess our approach, a set of metrics is needed. The goal of this research is to augment data effectively and efficiently. To assess the effectiveness we rely on the accuracy of the model.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

To further assess the effectiveness of the model, we need to look at the ability of the model to generalize. For this, we rely on the depth of the tree. Since decision tree algorithms like XGBoost that overfit, tend to have great depth, this metric is a good indicator of overfitting.

To assess the efficiency of our approach, we compare the time our approach takes from the ranking algorithm to joining the best join path table.

4.3 Scenarios

The experiments consist of 4 different scenarios, which we will compare on accuracy, depth and runtime accordingly.

1. *Baseline:*

   The baseline scenario benchmarks the standard performance of the base table without data augmentation or feature selection.

2. *Dummy:*

   The dummy scenario benchmarks the performance of the base table with all candidate tables joined. We further split the results of this scenario in:

   - with filter feature selection after joining all candidate tables
   - no feature selection

3. *Approach AFAR:*

   The approach AFAR scenario benchmarks the performance of the approach derived in section 3. The approach ranks the candidate tables and joins the best candidate table out of this ranking.

4. *Robustness:*

   The robustness scenario benchmarks the performance of the approach derived in section 3, by combining it with different ML classifiers such as CART, random forest and support vector machine.

[2]https://github.com/oliverneut/CSE3000_AFAR

4.4 Results

In this subsection, the experimental evaluation on the 4 datasets is showcased. By using the 4 different scenarios, the approach is benchmarked in different contexts.

Configuration: AFAR is configured to join the paths to perform the ranking on a 50% stratified sample of the base table. Since the datasets do not contain a large number of rows, using 10% or 5% of the data will decrease the filter methods' ability to rank the different paths. The configuration further applies the *Rank_1* algorithm to rank the candidate tables. The experiments only measure the performance on the columns of the candidate table and not the possible tables between it and the base table. This shows the improvement of the candidate table on the model performance in isolation and the effectiveness of our approach can be validated better.

AFAR: baseline and dummy benchmarks

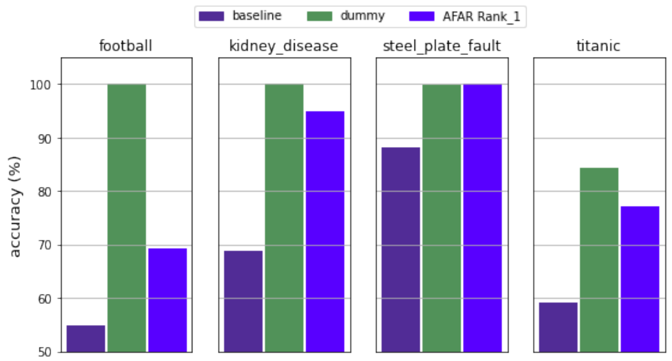


Figure 3: XGBoost model accuracy of 3 different scenarios for the 4 datasets

Figure 3 shows the accuracy of the model using the AFAR *Rank_1* approach benchmarked against the baseline and dummy scenario. The AFAR approach is in all cases an improvement to the baseline scenario, this is a trivial result. The dummy scenario displays similar or better accuracies in comparison to the AFAR approach. Since the dummy scenario encompasses all possible features, it has a higher probability of selecting the best features available. Only in the *football* dataset displays AFAR a significantly smaller improvement in model accuracy compared to the dummy scenario.

To further analyze the approach, the max depth and runtime metrics are benchmarked in the same 3 scenarios. Figure 4 shows the depth benchmarked against baseline and dummy. The baseline scenario tends to overfit the *steel_plate_fault* datasets due to the lack of useful features available. Overall, AFAR entails a low maximum depth and thus does a good job at reducing overfitting.

The runtime metrics in figure 5 showcase impactful improvements of the approach. AFAR runs much faster compared to dummy since the XGBoost model has much fewer features to select from. The runtime-accuracy tradeoff is very visible in these plots because only a small addition in runtime adds significant accuracy improvement. The stratified sample joins also reduce the runtime of the path joining during the approach by a factor of 2 in the current configuration.

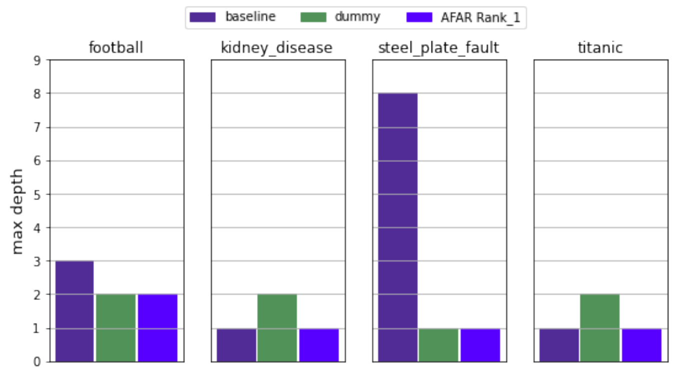


Figure 4: XGBoost model depth of 3 different scenarios for the 4 datasets

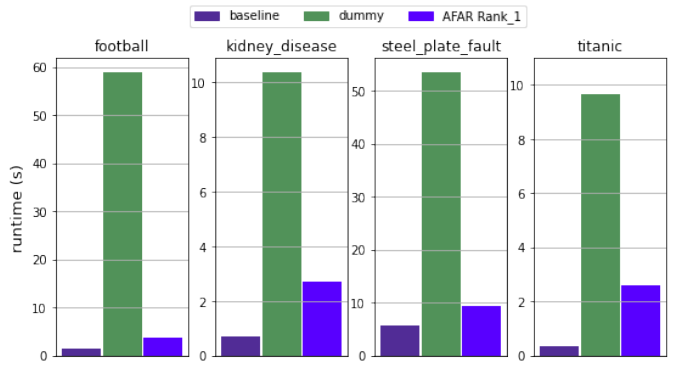


Figure 5: runtime of 3 different scenarios for the 4 datasets

Dummy: filter method benchmarks

To select the best filter feature selection method, the standard dummy is benchmarked against different feature selection methods. The 50% best scoring features of the entire datasets are selected.

Figure 6 shows clearly that Pearson correlation, information gain, Gini index and symmetrical uncertainty (SU) are performing equally well in this scenario. In the *titanic* dataset, they perform slightly better than the dummy approach, since that is the only dataset where the dummy approach does not achieve maximum accuracy. Spearman correlation does not prove to be the best filter method in terms of accuracy when considering the *football* dataset.

The max depth of tree comparison shown in figure 7, indicates that Pearson correlation, information gain, Gini index and symmetrical uncertainty remain good filter methods. The max depth is high for the Spearman correlation for the *football* data, where this filter method also entailed a below-average accuracy.

The runtime performance of filter methods is reduced in most cases by at least a factor of 2 (fig. 8). This is primarily due to the fact that the top 50% of the features are selected for this experiment. Note that Pearson correlation, information gain, Gini index and symmetrical uncertainty remain good performing filter methods.

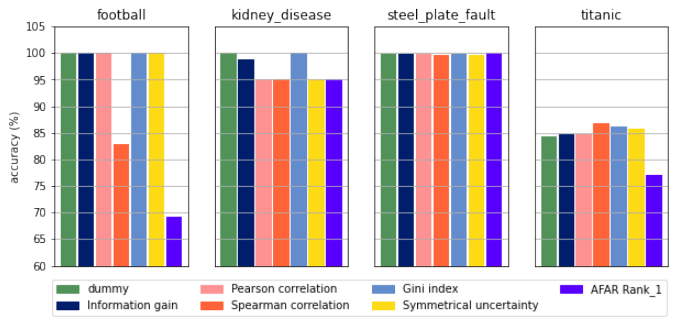


Figure 6: XGBoost model accuracy of the dummy scenario and 5 different filter methods for the 4 datasets

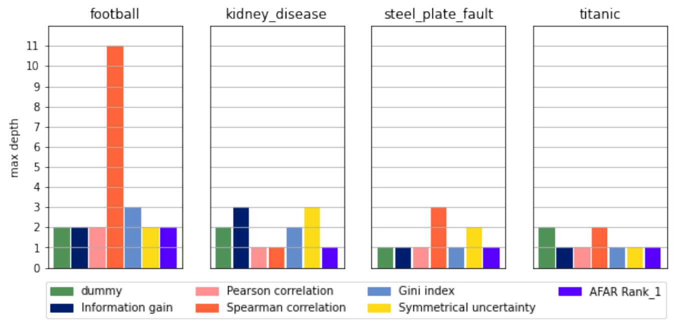


Figure 7: XGBoost model depth of the dummy scenario and 5 different filter methods for the 4 datasets

Robustness: algorithm & filter method comparison

Figure 9, shows the effects of running the AFAR Rank_1 approach in combination with different ML classifiers to test its robustness. To validate the robustness of Rank_1 even further, the Pearson correlation score is substituted by other filter methods. The accuracy results are measured for 4 different filter methods: Pearson correlation, information gain, Gini index and symmetrical uncertainty (SU) which are selected based on their positive results in the Dummy experiment. The variance of the results for each dataset remains low in all cases for the standard Rank_1 algorithm (alg. 2). Other filter methods tell a different story, the random forest and CART classifier have increased accuracy for Gini index in the *football* dataset. Additionally, the random forest classifier benefits from information gain in the *football* dataset.

Overall, Pearson correlation and Information gain seem the best performing on all 4 datasets for the 4 classifiers. In 3 out of the 4 datasets, Pearson correlation and information gain, are able to pick the best candidate table, while Gini index and SU can only predict 2 correctly (fig. 10).

4.5 Approach tuning

To get the optimal approach we combine the best heuristics from the previous experiments. We apply Rank_2 in our algorithm, that ranks candidate tables based on Pearson correlation, information gain, Gini index and the unique values formula. Rank_2 results in an almost optimal ranking (fig. 10).

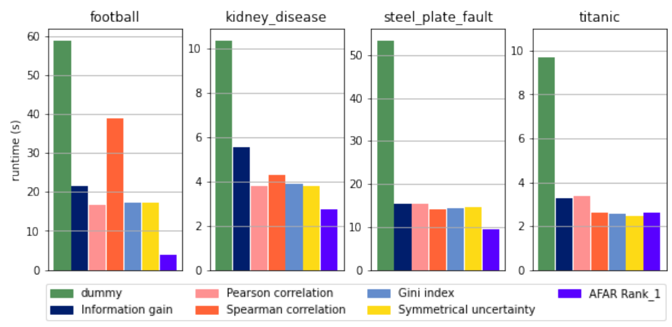


Figure 8: runtime of the dummy scenario and 5 different filter methods for the 4 datasets

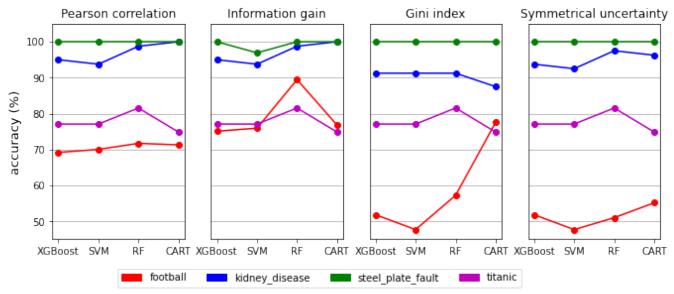


Figure 9: Accuracy of AFAR Rank_1 approach in combination with different ML classifiers and filter methods

The Rank_2 algorithm (alg. 3) has equivalent results because for the top candidate join, it selects the same top tables. The consequences of joining more than 1 table are considered for this algorithm (fig. 11). Overall the Rank_2 algorithm shows positive performance measures in the first joins. The accuracy grows slightly slower for the football dataset, this is because the best candidate table ('strong_team') is only joined when k=4. For an increasing number of joins, the accuracy only improves slightly. While the runtime for increasing joins grows exponentially.

5 Responsible Research

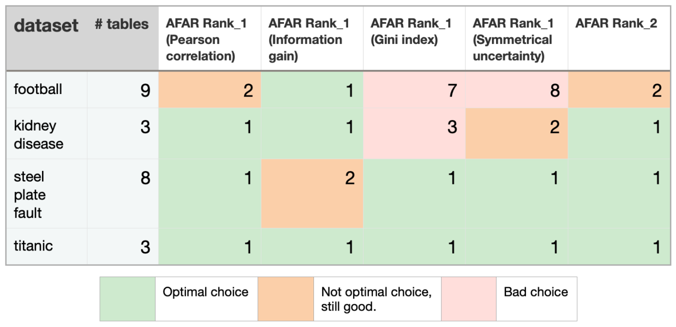
This section reflects on the ethical aspects and the reproducibility of the research done in this paper. Since AML systems are increasingly being used, the research on improving such systems has become of greater importance. It is, therefore, crucial to consider the consequences and implications of the enhancement of these technologies.

5.1 Ethical implications

AML encompasses a wide range of topics and has a multitude of ethical concerns associated with it that are worth discussing. Since this research focuses on data augmentation and discovery, it's important to note that these parts of the AML pipeline also impact the outcome and could have ethical consequences. The AFAR approach derived in this paper lacks a moral compass to determine which features to include and which to exclude. Consequently, biased machine-learning models cannot be ruled out. Training data can contain human biases that can propagate into the results of the model decisions. In some cases, this drives a feedback loop that perpetuates the bias of the model. It is therefore crucial that the data is carefully analyzed and cleaned, before being applied in real-world impactful situations.

Since AML is not error-proof, a user of AML should be aware of the implications and make sure to not rely heavily on the functionality AML provides. It is important to maintain a good synergy between humans and the intelligent system. At the end of the day, the human is responsible for the consequences and should ultimately remain in control of moral decisions.

Because of the rapid growth in ML and the impact it already has on society, it is possible that there are yet-to-be-identified ethical implications.



dataset	# tables	AFAR Rank_1 (Pearson correlation)	AFAR Rank_1 (Information gain)	AFAR Rank_1 (Gini index)	AFAR Rank_1 (Symmetrical uncertainty)	AFAR Rank_2
football	9	🟧 2	🟩 1	🟥 7	🟥 8	🟧 2
kidney disease	3	🟩 1	🟩 1	🟥 3	🟧 2	🟩 1
steel plate fault	8	🟩 1	🟧 2	🟩 1	🟩 1	🟩 1
titanic	3	🟩 1	🟩 1	🟩 1	🟩 1	🟩 1

🟩 Optimal choice	🟧 Not optimal choice, still good.	🟥 Bad choice

Figure 10: For each dataset, the top candidate table is compared to the actual ranking of tables. 1: the approach chose the top table, 2: the approach chose the 2nd best table etc.

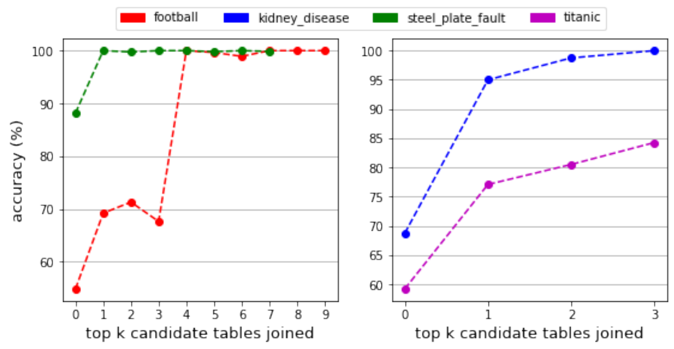


Figure 11: The accuracy for the top k candidate tables joined for AFAR Rank_2.

5.2 Reproducibility

The experiments in this research have been conducted with reproducibility in mind. The results and the visualizations of the experiments provided in this research are publicly available in the dedicated Github repository, along with the 4 datasets used in the experiments. To ensure consistent results, all the algorithms that are based on randomness, have been given a specific random_state that is similar all across the code.

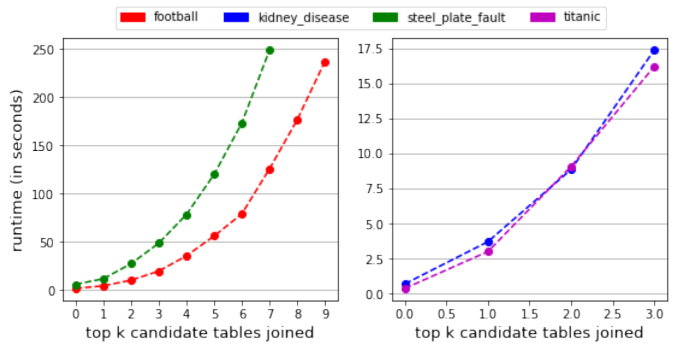


Figure 12: The runtime for the top k candidate tables joined for AFAR Rank_2.

6 Conclusions and Future Work

Through the experimental evaluation of the AFAR approach derived in this paper, it can be confirmed that efficient and effective automatic feature augmentation is possible. By making use of low-cost feature selection algorithms and stratified sample joins, the efficiency can be improved considerably while effectively improving the model performance. Two different variations of ranking join paths have been proposed. Rank_2 extends Rank_1 by utilizing additional filter methods and a unique values formula to score candidate joins. These have been shown to be robust against other classifiers.

The field of automatic feature discovery and augmentation still has room for improvement in both effectiveness and efficiency. Although stratified sample joins can reduce the runtime of joining by an arbitrary factor. For large data lakes with many candidate joins and millions of data points, joining remains an expensive task.

References

[1] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[3] Nadiia Chepurko, Ryan Marcus, Emanuel Zgraggen, Raul Castro Fernandez, Tim Kraska, and David Karger. Arda: automatic relational data augmentation for machine learning. *arXiv preprint arXiv:2003.09758*, 2020.

[4] Naoual El Aboudi and Laila Benhlima. Review on wrapper feature selection approaches. In *2016 International Conference on Engineering & MIS (ICEMIS)*, pages 1–5. IEEE, 2016.

[5] Mark A Hall and Lloyd A Smith. Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. In *FLAIRS conference*, volume 1999, pages 235–239, 1999.

[6] George H John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Machine learning proceedings 1994*, pages 121–129. Elsevier, 1994.

[7] Alan Jović, Karla Brkić, and Nikola Bogunović. A review of feature selection methods with applications. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 1200–1205. Ieee, 2015.

[8] Anne Layne-Farrar and Josh Lerner. To join or not to join: Examining patent pool participation and rent sharing rules. *International Journal of Industrial Organization*, 29(2):294–303, 2011.

[9] J L Liu, C C Chai, Y L Luo, J F Feng, L Yin, and N Tang. Feature Augmentation with Reinforcement Learnin. IEEE, 2021.

[10] Noelia Sánchez-Maroño, Amparo Alonso-Betanzos, and María Tombilla-Sanromán. Filter methods for feature selection–a comparative study. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 178–187. Springer, 2007.

[11] Zeyuan Shang, Emanuel Zgraggen, Benedetto Buratti, Ferdinand Kossmann, Philipp Eichmann, Yeounoh Chung, Carsten Binnig, Eli Upfal, and Tim Kraska. Democratizing data science through interactive curation of ml pipelines. In *Proceedings of the 2019 international conference on management of data*, pages 1171–1188, 2019.

[12] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855, 2013.