Join Path-Based Data Augmentation for Decision Trees

Andra Ionescu[δ] Rihan Hai[δ], Marios Fragkoulis[η*], Asterios Katsifodimos[δ]
[δ]Delft University of Technology, [η]Delivery Hero SE
{A.Ionescu-3, R.Hai, A.Katsifodimos}@tudelft.nl, marios.fragkoulis@deliveryhero.com

***Abstract*—Machine Learning (ML) applications require high-quality datasets. Automated data augmentation techniques can help increase the richness of training data, thus increasing the ML model accuracy. Existing solutions focus on efficiency and ML model accuracy but do not exploit the richness of dataset relationships. With relational data, the challenge lies in identifying join paths that best augment a feature table to increase the performance of a model.**

In this paper we propose a two-step, automated data augmentation approach for relational data that involves: (i) enumerating join paths of various lengths given a base table and (ii) ranking the join paths using filter methods for feature selection. We show that our approach can improve prediction accuracy and reduce runtime compared to the baseline approach.

I. Introduction

Automated Machine Learning (AutoML) reduces the time required in order to perform model selection, hyper-parameter tuning, and feature engineering. As part of AutoML, automatic data augmentation has emerged as a technique to improve the amount and diversity of training data through image transformation [1], [2]. In this work, we study the problem of automatic augmentation of relational data for machine learning models. We examine the typical scenario: in a relational database the tables are connected by primary key-foreign key (PK-FK) relationships. Given these relations, a base table and a machine learning model, our goal is to find new tables with *useful* features that will improve the current ML model. Following, a user can join the base table with the newly discovered tables, and obtain an augmented table with more features for training the ML model.

State-of-the-art solutions. The above problem has been addressed in [3], [4], which use feature-target correlation methods to discover highly relevant features for classification. For an automatic data augmentation process, the key challenges always lay on: 1) for the automated computation, what factors are relevant? 2) how to make the process efficient? Kumar et al [3] focus on whether the new tables bring additional information and their impact on the performance of the ML models. ARDA [4] samples the tables and judges the importance of a new feature by comparing it to random noise. However, in [3], [4] they assume that the useful tables are directly joinable with the base table; and relevant factor of choosing the features is mainly the model accuracy. With the following example we show the insufficiency of such assumptions.

*Work done while at TU Delft

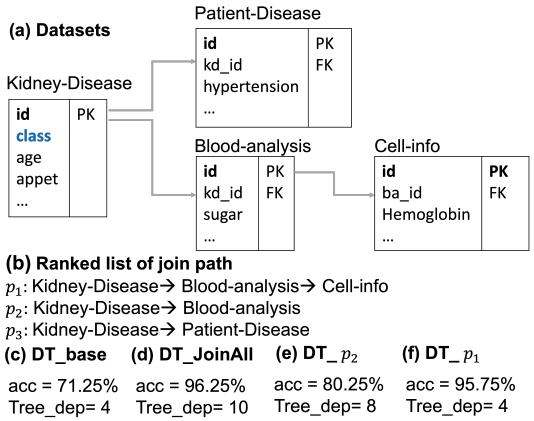

Fig. 1. Running example

Example. In fig. 1a, we illustrate a data augmentation scenario of four related tables connected with PK-FK relationships. *Kidney-Disease* is the base table with the target column $class$. Figure 1c describes the accuracy and the tree depth of a decision tree model trained on the base table. Figure 1e shows another decision tree model using the base table and a directly joinable table *Patient-Disease*. We can observe in Figure 1f that by adding a third table *Cell-info*, we obtain a new model with higher accuracy. Notably, when a decision tree is deep, it is more likely to overfit. In this sense, we can also say that compared to fig. 1e, the model in fig. 1f is more desirable.

Challenges. Therefore, given a base table, a ML model and a set of PK-FK related tables, we aim to find the tables with better predictive features, which reduce the likelihood of overfitting and improve model accuracy. Our intuition is that maybe such tables are not directly joinable with the base table, but on a join path. Moreover, we also consider the efficiency of downstream model training after data augmentation. A naive approach would join all the possible tables, then rely on a feature selection algorithm to select the most appropriate ones for training the decision trees, which could be computationally intensive. In addition, by comparing fig. 1d and f, we can see that such an approach does not necessarily provide the best model. We aim to find the join paths that indicate a *selective* set of tables for augmentation. Such a subset of tables should provide similar ML model performance compared to joining all tables, but lead to much less time for decision tree training.

Why decision trees. We try to find the tables with good features that are likely to improve the model accuracy before training the models with the augmented dataset. However, not all the feature selection methods allow comparing features without executing the ML models; and sometimes they are black boxes [5]. Decision trees use feature importance measures such as information gains, Gini index. It becomes possible to calculate such performance measures without the completion of model building. Our intuition is to integrate such performance measures into the automatic dataset augmentation process. This is one of the main reasons why in this work we focus on decision trees, besides its interpretability and popularity as a classification model.

Contributions. Our solution is a two-step approach that consists of (i) iterating through the space of possible join paths and (ii) ranking the paths according to their correlation with the target. We investigate if indirect foreign tables (tables discovered over a join path longer than one hop) can benefit to the augmentation by increasing the performance of the decision trees. We rank the join paths using filter-based feature selection methods, which capture the feature-target correlation. This approach helps discover the relevant features and also prune the feature space, leading to smaller yet improved datasets, which speed up the model training and increase the accuracy.

Outline. Following, we present some background information in section II and describe our solution in section III. Then, in section IV we assess the performance of our solution. Finally, we discuss our future work and conclude in section V.

II. Related work

A. Dataset augmentation

Augmenting relational data with relevant features for machine learning has been studied from different angles. Kumar et al [3] optimise data augmentation through join space reduction by identifying and eliminating the spurious ones. Our work is complementary, as we detect the most relevant joins to increase the performance of ML models. ARDA [4] describes an augmentation approach based on feature selection and preliminary join materialisation. COCOA [6] is a system for data augmentation based on correlation which uses virtual joins to optimize efficiency. In our work, we materialise the joins at every step during the traversal phase and reuse them to save computation time.

The effectiveness of indirect joins has been proved by JUNEAU [7] [8] and InfoGather [9]. JUNEAU extends computational notebooks and uses notebook workflows to find indirect candidates for augmentation via provenance tracking. InfoGather requires not only a base table to find indirect tables for augmentation, but also the attribute name or data value examples for that attribute. Our approach requires the base table and the PK-FK relations and outputs a ranked list of all the possible combinations for augmentation.

B. Join path discovery

Aurum's Enterprise Knowledge Graph (EKG) [10] uses graphs to capture the syntactic relations between columns, as well as to detect the PK-FK relations. Using this graph, Aurum outputs all the transitively connected join paths of $hop = k$ given a source and a target node. Niffler [11] uses the same EKG to find all possible combinations of columns, generates only 2-hop join paths between these columns and ranks the paths according tot the score from EKG. Our ranking function is customised for ML classification. Thus we use feature-target correlation to rank the paths.

Aurum and Niffler both use the traditional path enumeration approach, which takes two nodes and outputs all the possible or shortest paths between the two nodes [12]. Instead, we focus on traversing the graph given only one node, the source, and enumerating join paths of various lengths.

Another approach which differs from the traditional path enumeration is D3L [13]. They enumerate join paths of various length starting from a source attribute to discover other related attributes whose values fit within the target table. Our approach has a different focus, as we augment a table with more attributes, instead of instances.

III. Join Path-based Data Augmentation

A. Setup and approach overview

Problem setting. The inputs are a base table with features and a target column, and a set of candidate tables that maintain PK-FK relationships between them. The output is a ranked list of possible join paths, starting from the base table, which includes optimal features for training a decision tree model.

Preliminaries. In previous work [14], we have proposed a system that can find PK-FK relationships among the tables based on either schema information, data discovery systems [15] or data profiling [16]. We represent the discovered PK-FK relationships as graphs that we term *Dataset Relation Graphs* (DRGs). A DRG is a direct acyclic graph $G = (V, E)$, where each node V corresponds to a column from a table in a database. The nodes are connected by edges E which denote three types of relations between the columns: 1) *sibling* edges indicate columns belonging to the same source table, 2) *pk_fk* edges represent inclusion dependencies such as the primary-key foreign-key dependency, and 3) *match* edges denote any other syntactic or semantic relations captured by data discovery or schema matching methods. In this work, we assume that the input tables have already been processed and the DRGs generated with the *sibling* and *pk_fk* edges. Composite keys and self-joins fall outside the scope of this paper and will be addressed in future work.

Approach overview. The goal of this work is to recommend top-ranked paths (e.g., p_1 in Fig. 1b) to users. The approach consists of two steps. The first step is the enumeration of all the possible join paths (Alg. 1) to discover features that are not directly joinable with the base table and that could improve model accuracy while reducing the chances of overfitting. The second step is the ranking of join paths (Alg. 2), using a

ranking function integrated with feature importance measures, in order to reduce the set of joined tables returned to the user.

B. Enumerating Join Paths

Given a user-provided base table t_0 we traverse the DRG in a Breadth-First Search manner. By following the $sibling$ edges (5) we can reach all the column nodes of the current table. Among them we filter and obtain nodes with pk_fk edges. For those nodes, we retrieve their joinable nodes/columns $foreign\_nodes$, by following those pk_fk edges. In lines 7-12, for each foreign node, we find its table t_f, create a new path p based on existing paths P. Consider the datasets in Fig. 1a, where the corresponding DRG has three pk_fk edges: with $Kidney\text{-}Disease.id \rightharpoonup Blood\text{-}analysis.kd\_id$ and $Kidney\text{-}Disease.id \rightharpoonup Patient\text{-}Disease.kd\_id$ we find paths p_2 and p_3; with the pk_fk edge $Blood\text{-}analysis.id \rightharpoonup Cell\text{-}info.ba\_id$, we construct the path p_1 based on the existing path p_2. We also save the IDs of the foreign nodes (e.g.,$Blood\text{-}analysis.id$) for further usage in the ranking algorithm. We use the discovered neighbours to retrieve the corresponding table t_f to perform the join between the tables on path p. We save the newly augmented result in D. For example, with the path p_1 of Fig. 1b, we join the three tables on p_1 and store the result. Finally, we append t_f to T, such that we continue with the iterations until no further nodes can be discovered. The output of Alg. 1 including the paths P, ids I, and join result D, will be the partial input of Alg. 2, which we introduce next.

Algorithm 1: Join Path Enumeration

Input : base table t_0, dataset relation graph G
Output: dictionary of paths P and the corresponding IDs I, path-specific join result D

1 **Initialization:** $P, I, D, T \leftarrow []$
2 $T \leftarrow push(t_0)$
3 **while** T **do**
4 $\quad t \leftarrow pop(T)$ // current table t
5 $\quad S \leftarrow filterNodesWithPkFk(t, G)$
6 $\quad foreign\_nodes \leftarrow getFkNodes(S, G)$
$\quad$ // Retrieve the tables of foreign nodes
7 $\quad$ **foreach** *node* $n_f \in foreign\_nodes$ **do**
8 $\quad\quad t_f \leftarrow getForeignTable(n_f, G)$
9 $\quad\quad p \leftarrow createPath(P, t_f)$, *add* p *to* P
10 $\quad\quad i \leftarrow getTablePK(t_f)$, *add* i *to* I
11 $\quad\quad d \leftarrow JoinTablesOnPath(p)$, *add* d *to* D
12 $\quad\quad T \leftarrow push(t_f)$
13 $\quad$ **end**
14 **end**
15 **return** P, I, D

C. Ranking Join Paths

Alg. 2 ranks the join paths according to their contribution to the decision tree model building. In Alg. 2 we iterate through the enumerated join paths P and read the corresponding augmented table from D (line 3). We drop the ID columns of tables on the path p. This is an important pre-processing step, as using the IDs for training an ML model leads to extreme under-fitting. Next, we generate the score for each path. The scores indicate the correlation between each column in d and the target column l. The score is a value in the $[0, 1]$ interval; the closer to one, the higher the correlation. For each path, we save the highest score, which indicates that the dataset contains at least one correlated feature with the target. In Sec. IV-A we elaborate on the five filter-based feature selection methods, which are used for score generation. Finally, we sort the join paths in descending order based on the scores. The list of ranked paths contains the table names, the columns used for join and the join result. Given this list of ranked join paths, a user can choose top-k join paths to augment the base table.

Algorithm 2: Join path ranking

Input : paths P and ids I, join result D, target column l of base table t_0
Output: ranked list of paths R

1 **Initialization:** $S \leftarrow \{\}$
2 **for** $p \in P$ **do**
3 $\quad d \leftarrow get(D, p)$ // join result of current path
4 $\quad i \leftarrow get(I, p)$ // IDs corresponding to current path
5 $\quad d.drop(i)$
6 $\quad score \leftarrow pathScoreGen(l, d)$
7 $\quad S[p] \leftarrow score$
8 **end**
9 $R \leftarrow rankPathMaxScores(S)$ // sort the ranked paths in descending order
10 **return** R

TABLE I
DATASETS STATISTICS. $Base$ IS THE NON-AUGMENTED TABLE, $feat$ IS THE NUMBER OF FEATURES AND t_f IS THE NUMBER OF FOREIGN TABLES.

Dataset	Base #(rows, feat)	#t_f	#feat for each t_f
Titanic	(891, 3)	3	2, 6, 5
Kidney disease	(400, 4)	3	11, 9, 7
Steel plate fault	(1941, 9)	7	7, 6, 5, 6, 5, 5, 8
Football	(1182, 5)	9	13, 5, 5, 6, 6, 6, 6, 6

IV. EVALUATION

In this section, we describe our experimental evaluation, which supports four goals. It shows that (i) by augmenting tables, the accuracy of decision trees increases, (ii) by joining all the related tables, the user faces with the trade-off between efficiency and accuracy, (iii) our approach is robust against other feature selection methods and (iv) our first goal holds irrespective of the decision tree model.

A. Experimental setup

Datasets. We collected four datasets, which are used for binary classification with decision trees, from Kaggle. We created a synthetic dataset collection, summarised in table I, by manually splitting the tables and creating the PK-FK relations. By iteratively extracting the feature that appears on top of decision trees, which is the most correlated feature, we split the tables into smaller logical parts, each containing one of the correlated features. We made the datasets, the algorithms and the experiments publicly available*.

Baseline approach. It involves joining all the split tables connected by a pk_fk edge and using the resulted augmented table to train the decision trees without any feature selection. We call this approach the *JoinAll* approach.

*https://github.com/delftdata/auto-data-augmentation

Evaluation. We measure the performance of our approach in relation with the accuracy of the decision trees trained using the augmented dataset. We create the following evaluation scenarios.

- First, we compare the performance of the non-augmented dataset against the dataset resulting from the best ranked join path, called *BestRanked*.
- Second, we compare the *JoinAll* approach against our *BestRanked* approach. Besides the accuracy of the decision tree model, we also measure the runtime[†] of each approach and the depth of the resulting tree.
- Third, we compare five filter-based feature selection methods [5]: information gain with entropy, information gain with Gini index, Spearman's correlation, symmetrical uncertainty (SU) and ReliefF. This scenario supports our third goal: showing that our approach is robust against any feature selection method.
- Finally, we assess the robustness of our approach against three decision tree models. We implemented the ID3 algorithm and used another two out-of-the-box implementations of decision trees: CART and XGBoost. We used grid search to tune two relevant hyper-parameters before training each dataset (augmented or not). The relevant hyper-parameters to our study are: (i) the maximum tree depth, which is an indication of over or under-fitting and (ii) the splitting criterion (Gini or entropy), which determines the most correlated features that become the nodes of the trees.

B. Results

BestRank against baseline and JoinAll. In fig. 2, we note that our *BestRank* approach results in a non-trivial increase in accuracy against the baseline approach across all algorithms and datasets. However, when comparing with the *JoinAll* approach, our solution performs equally well or slightly worse. This behaviour is unsurprising, since the *JoinAll* approach encapsulates *all* features, both relevant and less relevant. That said, *BestRank* presents an advantage over *JoinAll* in terms of runtime execution time and tree depth, as shown in fig. 3. Moreover, we also observe an interesting behaviour between the decision tree algorithms. While choosing CART results in shorter execution time, the depth of the tree skyrockets for the *JoinAll* approach. With this behaviour the algorithm sacrifices the potential to generalise in order to achieve high efficiency. Contrary, XGBoost prioritises fitting the data properly over runtime performance. Nevertheless, our approach shows improvement in both runtime and tree depth in many cases.

Robustness against feature selection methods. Figure 4 shows the comparison in terms of accuracy between the *BestRank* approach and the non-augmented datasets for each decision tree algorithm and feature selection method. The results show that our approach is robust against both the feature selection method and the decision tree model. However, for the ID3 decision tree algorithm, the path ranked the highest by the information gain feature selection method, results in an augmented table which decreases the accuracy of the ID3 algorithm. Furthermore, fig. 4 illustrates that the feature selection methods impact the length of the join path. For example, all the highest ranked join paths using ReliefF have $length = 1$, while the rest of the feature selection algorithms return paths of various lengths. This illustrates a dependency between the datasets and the feature selection methods. However, despite this dependency, our approach shows an increase in accuracy regardless the feature selection method.

Summary. These experiments reveal three key findings. First, augmenting a base table with more features increases the accuracy of decision trees. Second, using the *JoinAll* approach results in a trade-off between efficiency and the ability of a model to generalise. Lastly, our approach is robust against any filter-based feature selection method.

[†]The experiments were run on a 2.4GHz Quad-Core i5 with 16GB RAM and 500GB storage. All datasets fit in memory.

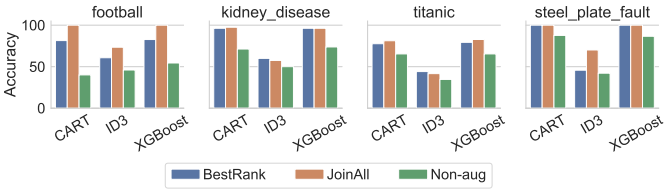

Fig. 2. Accuracy for *BestRank, JoinAll* and the non-augmented base table.

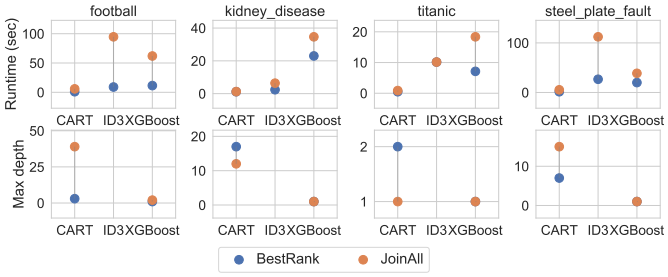

Fig. 3. Runtime and depth for *BestRank, JoinAll*.

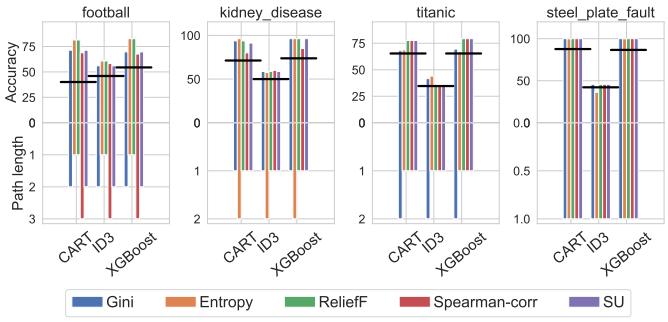

Fig. 4. Accuracy for different feature selection methods for *BestRank*. The horizontal line represents the non-augmented base table.

V. Future work and conclusion

We presented a greedy approach for automatic data augmentation on relational datasets aimed at increasing the accuracy of decision trees by discovering indirect join paths from a given base table. While results are promising, we are currently working on improving the approach to better suit data lake scenarios, where the join space increases significantly.

To this end, we plan to avoid join materialisation through alternative methods, such as virtual joins, join approximation, and data sketches. We will extend the approach to include *match* edges, composite keys, and self-joins. sFinally, we plan to generalise our approach over a wider variety of ML algorithms.

Acknowledgment

This work has been partially funded by the H2020 project OpertusMundi No. 870228.

References

[1] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[2] A. J. Ratner, H. R. Ehrenberg, Z. Hussain, J. Dunnmon, and C. Ré, "Learning to compose domain-specific transformations for data augmentation," *NIPS*, vol. 30, p. 3239, 2017.

[3] A. Kumar, J. Naughton, J. M. Patel, and X. Zhu, "To join or not to join? thinking twice about joins before feature selection," in *SIGMOD*, 2016, pp. 19–34.

[4] N. Chepurko, R. Marcus, E. Zgraggen, R. C. Fernandez, T. Kraska, and D. Karger, "Arda: automatic relational data augmentation for machine learning," *PVLDB*, pp. 1373–1387, 2020.

[5] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *MIPRO*. Ieee, 2015, pp. 1200–1205.

[6] M. Esmailoghli, J.-A. Quiané-Ruiz, and Z. Abedjan, "Cocoa: Correlation coefficient-aware data augmentation." in *EDBT*, 2021, pp. 331–336.

[7] Y. Zhang and Z. G. Ives, "Juneau: data lake management for jupyter," *Proc. VLDB Endow.*, vol. 12, no. 12, 2019.

[8] Y. Zhang and Z. Ives, "Finding related tables in data lakes for interactive data science," in *SIGMOD*, 2020, pp. 1951–1966.

[9] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri, "Infogather: entity augmentation and attribute discovery by holistic matching with web tables," in *SIGMOD*, 2012, pp. 97–108.

[10] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker, "Aurum: A data discovery system," in *ICDE*, 2018, pp. 1001–1012.

[11] Y. Gong, Z. Zhu, S. Galhotra, and R. C. Fernandez, "Niffler: A reference architecture and system implementation for view discovery over pathless table collections by example," *arXiv preprint arXiv:2106.01543*, 2021.

[12] L. Chang, X. Lin, L. Qin, J. Yu, and J. Pei, "Efficiently computing top-k shortest path join," in *EDBT*, 2015.

[13] A. Bogatu, A. A. Fernandes, N. W. Paton, and N. Konstantinou, "Dataset discovery in data lakes," in *ICDE*. IEEE, 2020, pp. 709–720.

[14] A. Ionescu, A. Katsifodimos, and G. Houben, "Interactive data discovery in data lakes," in *VLDB PhD Workshop*, vol. 2971. CEUR-WS, 2021.

[15] C. Koutras, G. Siachamis, A. Ionescu, K. Psarakis, J. Brons, M. Fragkoulis, C. Lofi, A. Bonifati, and A. Katsifodimos, "Valentine: Evaluating matching techniques for dataset discovery," in *ICDE*. IEEE, 2021, pp. 468–479.

[16] T. Papenbrock, T. Bergmann, M. Finke, J. Zwiener, and F. Naumann, "Data profiling with metanome," *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1860–1863, 2015.