Towards Extending XAI for Full Data Science Pipelines

Nadja Geisler
nadja.geisler@cs.tu-darmstadt.de
Technical University of Darmstadt
Germany

Carsten Binnig
carsten.binnig@cs.tu-darmstadt.de
Technical University of Darmstadt, DFKI
Germany

ABSTRACT

Data preprocessing and engineering are essential parts of any AI system, as indicated by the current trend of data-centric AI. However, until now, explainability efforts have almost exclusively focused on models. We propose explanations for preprocessing pipelines that express the impact of each step on the resulting model behavior based on existing feature attribution methods. In the process, we introduce two related but distinct measures of impact for preprocessing steps: *Leave-out Impact* (What do we lose/gain by leaving out this step?) and *Immediate Impact* (What do we lose/gain by adding this step at this time?). Both are obtained by constructing variations of the original pipeline and comparing the resulting model behavior represented as feature importance vectors. These measures reflect the intuition of impact but also express the effects of a step and its interactions with the rest of the pipeline on the internal workings of the trained model.

ACM Reference Format:
Nadja Geisler and Carsten Binnig. 2024. Towards Extending XAI for Full Data Science Pipelines. In *Workshop on Human-In-the-Loop Data Analytics (HILDA 24), June 14, 2024, Santiago, AA, Chile.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3665939.3665967

1 INTRODUCTION

Data preprocessing is essential for AI. Machine learning systems used in production today depend on more than model architecture and hyperparameters. Malformed, incomplete, or inconsistent data can make contained information inaccessible to the model in training. Data engineering is, therefore, just as important and requires substantial resources and effort to be done well. We can also see this reflected in the current trend towards data-centric AI.

ACM ISBN 979-8-4007-0693-6/24/06
https://doi.org/10.1145/3665939.3665967

Example. We consider a data engineer working on a preprocessing pipeline for a predictive maintenance classifier. In the latest iteration, they modified the preprocessing pipeline to be faster and produce a smaller model with the available training data. This process has involved removing steps from the pipeline (such as a unit conversion from centimeters to meters, which does not change information content), modifying steps (exchanging the undersampler for class balancing for a different algorithm), and adding outlier removal. The before (A) and after (B) are shown in Figure 1. This has resulted in significant changes to model outputs, with accuracy increasing overall, but some areas suffered significantly. We will expand on how the data engineer might address this, while we present our motivation, approach and contribution in the remainder of this section.

The need to explain data preprocessing. In order to fix the problem, the data engineer needs to identify which addition, removal, or modification of a pipeline step might have caused this undesired behavior. For example, when using a neural network architecture, removing a scaler might have close to none or very drastic effects, based on the attributes' value ranges, as the training process might become so slow the model cannot converge. The data engineer is now tasked with answering the question: *How does the model behavior change due to each employed data preprocessing step?* However, so far, there are very few approaches to providing additional insight into the effects of preprocessing to assist developers in improving pipelines and model performance.

A Primer to Attribution-based XAI. Explainable AI (XAI) research focuses on insights into model behavior, i.e., how outputs are created in a trained model. Feature attribution methods, which explain (local) model behavior by quantifying the contribution of each input feature to model output, exist and work well (popular examples include LIME[11] and SHAP[8]). In cases such as this one, however, this does not provide any information on what caused the model to behave in such a way and does not consider data preprocessing at all. The data engineer would be able to obtain information on input features such as a distance measurement that contributes

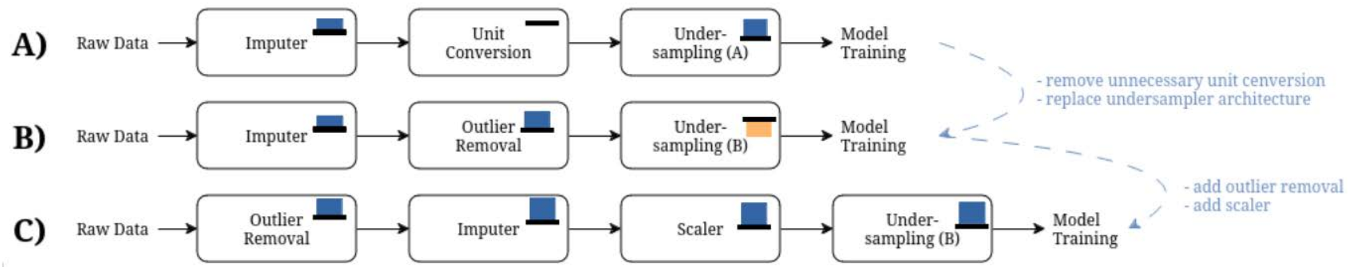

Figure 1: Pipeline before (A), after initial modifications (B), and after leveraging preprocessing step impact (C). Impact indicators for each step, 🟦 signals change in the intended direction, 🟧 in the opposite.

strongly to model output and a temperature-sensitive light sensor that contributes to the correct labels for low ambient temperatures but incorrect labels for high temperatures. What they are interested in, however, is how the impact of the unit converter changed drastically as the undersampler was exchanged (the new one being based on a KNN architecture) or how the contributions of the missing value imputer and outlier removal interacted. From the example explanations in Figure 1, they can gather the following observations: (1) The intended effect of the undersampler suffered a lot from the changes (🟦 turned orange, indicating the opposite direction of effect). (2) The positive effect of the added outlier compensates for this, potentially leading to similar or slightly better overall performance. (3) The effect of both the value imputer and outlier remover increases with the appropriate order (4) The added scaler significantly improves the direction of the undersampler's effect. This allows the data engineer to reach a more effective pipeline than without this insight.

Our Contribution; A first attribution-based method. We want to take inspiration from this idea into an attribution method for preprocessing steps, quantifying the impact of each processing step on the behavior of a model. Previous research has dealt with the impact of preprocessing on overall model performance, e.g., measured through accuracy[12][3][7]. However, these measures do not capture how individual model outputs are affected and, especially, not the model's internal logic or behavior beyond a resulting label. This is insufficient as preprocessing fundamentally affects how the model operates, even if the resulting output remains the same. Instead, we want to provide users with insights into the impact of each step in a pipeline.

The need to extend XAI. Suppose the data engineer knows how big the contribution of each individual step is and whether this contribution brings the model closer to the desired output. In that case, this enables them to judge the importance of each step, detect flaws, and make improvements in a more directed manner. The engineer in the example would see that the change in undersampler increases the impact of the unit conversion, as KNN architectures are sensitive to variations in the value ranges between features. This would enable them to add a scaler for even better model performance. They would observe the expected overall benefit of the added outlier removal but also note that this effect increases further when adding it before the imputer rather than after, as the imputer will infer the missing values where outliers were removed. Explanations for the pipeline modified this way would show a much more homogeneous picture, as illustrated in Figure 1. To provide explanations, we introduce two related but distinct measures of impact for a given preprocessing step: One to explain *What do we lose or gain by leaving out this step?* and one to explain *What do we lose or gain by adding this step at this moment in processing?*

Existing methods enabling this approach. We propose a process of quantifying and measuring these impacts by representing model behavior around a given sample as feature weights obtained through an existing feature attribution explainer. These feature attribution-based explainers reflect model behavior through feature importance, which we can translate into preprocessing steps. If an individual step changes the feature importance significantly compared to what the overall pipeline does, then it is important for the resulting model. This sets us apart from existing approaches to explaining preprocessing, which focus either on changes in the data that disregard the model completely or measure change in the model only via model output. For the scope of this paper, we focus on supervised classification tasks on tabular data, but analogous handling of other tasks and data types is possible. We treat the classifier as an opaque unit and do not require any knowledge about its inner workings.

Outline of the paper. The remainder of this paper is structured as follows: We give an overview of previous work on explaining preprocessing in 2. Then, we elaborate on our proposed measures in Section 3 and give an overview of the framework we implemented to compute them in Section 4. This is followed by preliminary evaluation results in Section 5. We finish with an overview of open challenges in Section 6 and a brief conclusion in Section 7.

2 RELATED WORK

Dasu and Loh [4] have proposed an approach for evaluating data cleaning, examining similar types of processing steps. However, they aim to measure the impact of cleaning strategies on the statistical properties of underlying data, i.e., quantifying how much the data itself is skewed through processing in relation to glitch improvement (handling unwanted artifacts). While this affects model training, their work does not look at the effects on model behavior. It would be interesting to study where their metric, Earth Mover's Distance, aligns with our measures and where it doesn't. Through the nature of statistical distortion, their work only looks at global changes in data, whereas we have the ability to inspect local impact, too.

The work of Gonzalez Zelaya [6] examines preprocessing steps through changes in model output resulting from preprocessing steps. They measure volatility, i.e., the likelihood of a data point changing label, and propose a learned model to approximate this measure instead of calculating it directly. This method works with less information for each explanation (label vs. feature weights), and volatility does not consider how close to the decision surface a point was before the label change. Volatility also does not reflect in any way changes in model behavior that do not result in a different label ("same result for a different reason").

3 MEASURING IMPACT

From our motivation and the limitations of previous work, we derive a new path to measuring impact. This measure should be evaluated using the individual steps of a pipeline, reflect the changes in the internal behavior of the resulting model (as opposed to changes in the data or changes in overall model performance), work without assumptions about model architecture, and incorporate the complex interactions of steps in a pipeline. We propose two distinct but complementary measures to achieve these characteristics.

Immediate Impact. The first, *immediate impact*, aims to answer the question *What do we lose or gain in adding this specific step at this point in the pipeline?* To answer this question, we compare two pipelines: One includes all steps prior to the one being evaluated, and the second additionally includes the step itself. Including previous steps (and implicitly the order of the pipeline) is important since they influence behavior. For example, consider a pipeline that includes outlier detection and removal and scaling to a set value range. The result of scaling will change drastically with/without extreme outliers. Immediate impact captures the difference in the behavior of models trained on the two pipelines. Therefore, the difference the step makes right as it is applied, disregarding interactions with later steps. To show this is quantifiable, we consider the pipeline's effect on model behavior as traversing a vector space. As such, it has a direction and a distance. Quantifying immediate impact now becomes a measure of how much of the way of the entire pipeline this step takes the data. As illustrated for a two-dimensional space in Figure 2, adding the immediate impact of each step results in the vector for the entire pipeline. To make the relationship between pipeline and step interpretable, Immediate impact must be normalized appropriately.

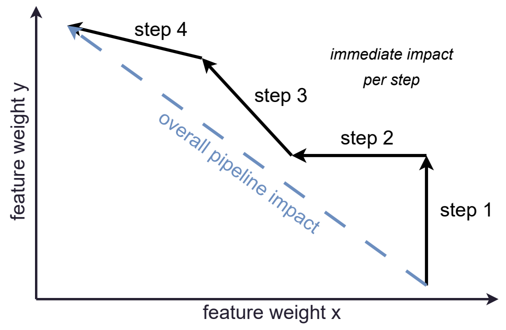

Figure 2: Step-wise immediate impact in two-dimensional vector space.

Leave-out Impact. The second measure, *Leave-out Impact*, aims to answer the question *What do we lose or gain when removing this step from the full pipeline?* This is based on the understanding that the following steps might have different effects based on previous ones. As an example, we can again look at the same pipeline as before, but this time, consider the Leave-out Impact on the Outlier handling, which changes the path of the scaler that comes after it. To get the full picture, we need both measures for both steps. In terms of the same vector space, it quantifies how close to/far away from the result of the full pipeline a pipeline without this step takes the data, as visualized in Figure 3. Again, we create two pipelines: One corresponds to the full pipeline, and the other includes all steps but the one being examined. Leave-out Impact, therefore, captures changes in the data originating directly from the step in question or its interaction with other steps, regardless of their position in the pipeline.

Interplay of the two. Both measures originate from similar albeit complementary intuition; still they are both needed to give a thorough impression. Immediate impact demonstrates the formation of the total impact of the pipeline. This provides users with an intuitive understanding of what the

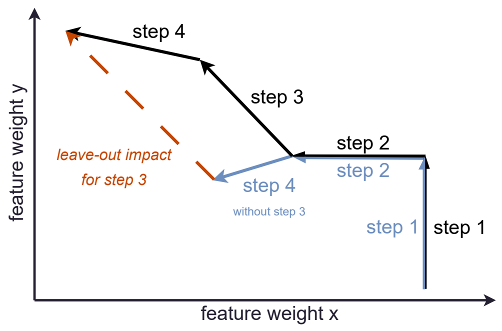

Figure 3: Leave-out Impact for step 3 (🟥) in 2D vector space. Complete pipeline (■) vs. pipeline without step 3 (🟦). Note difference in effect of step 4.

single value means but also, upon further inspection of the vectors themselves, the divergence (difference of direction in the vector space) from the overall impact and intensity (relative length of the vectors). This is not skewed by later processing steps that might counterbalance the impact, either causally or incidentally. Leave-out impact gives a more complete picture of the step and its interactions, but vectors are less intuitive. More importantly, to judge interaction effects, e.g., whether a scaler and a standardized in the same pipeline causally negate each other's effect in a pipeline, we need both impacts for both steps: Immediate impact shows that both diverge significantly from the overall pipeline, although in the opposite direction. Leave-out impacts show us whether their effects are conditional on one another or coincidental; if the scaler puts values between 1 and 100 and the standardizer creates a normal distribution around 0 with a standard deviation of 1, this will largely negate the effect of the previous scaler and in any case either provides pointless or counterproductive computations by the scaler. In another pipeline, scaler and class balancing might also show counterbalancing behaviors in Immediate Impact but not show interaction in Leave-out Impact. Note, that under certain circumstances, both measures might be the same for a step e.g., the last step in each pipeline.

4 PROTOTYPE IMPLEMENTATION

From these measures and their intuition, we derive a concrete computation process and a prototype implementation.

Leveraging XAI methodology. The key point that remains open in the approach concerns the numeric representation of model behavior. So far, we have assumed that it is possible to calculate a difference between the behavior of two models that goes beyond the model output itself. Once again, taking inspiration from existing XAI research, we propose using feature attribution explainers to open a vector space with one dimension per feature. An explanation consisting of feature weights provides one point in this space, and the difference between explanations represents the vector between them. These vectors become interpretable statements about features, i.e., a step vector in a two-dimensional space from bottom right to top left means a decrease in the importance of the feature on the x-axis and an increase in the importance of the feature on the y-axis. A user with domain knowledge, i.e., having an assumption of relevance for the features, might even be able to judge that direction is desirable. The two vectors (of a step and the full pipeline) give us an impression of the intensity of a step's impact (length ratio) and a degree of accordance/divergence, represented through the angle between the two.

Scope. The overall approach itself is flexible. It allows for any type of data as long as there is a stable (i.e., using the same features independent of preprocessing stage, e.g., superpixels (images), tokens (text), or columns (tables)) feature attribution method to generate numerical weights. Depending on the feature attribution method, it can generate local (one explanation for one model output) and global (entire model) explanations. As difference is measured through model behavior, the model output format/task (e.g., classification, regression) is not an inherent requirement. Finally, any preprocessing step that may be omitted and does not change the number of features, can be measured.

However, for our prototype implementation, we chose a more restricted setting. We work with classification tasks on tabular data (due to the restriction on non-optional steps only numerical data, see 6 for info on categorical features) and generate local explanations. Our task involves explaining the impact of preprocessing on individual predictions for three reasons: In some applications, this is of particular interest, justifying the existence of both local and global approaches. It is even less covered through model performance measures such as accuracy, often used exclusively. Lastly, it still provides information on the model as a whole when applied to a broader selection of features suitable to the use case. Our task can, therefore, be phrased as follows: For a given tabular, numerical dataset, classifier architecture, data sample, and preprocessing pipeline, provide the user with two numerical measures of impact on model output and a representation of divergence from the overall pipeline impact for each step of the pipeline in a way that enables pipeline and model improvement.

Implementation. In this prototype, we use LIME[11] as an established feature attribution method providing weights between -1 and 1 for each input feature. Using LIME explanations to represent the local model behavior, we can compute a vector per impact measure for a pipeline step by subtracting the vector obtained from the step in question from the vector obtained using that step. The resulting vector represents the difference made and can be provided to the user as such, as a length and angle of divergence from the pipeline vector (numerically or visually). However, we propose one scalar value per step as a core output, which is computed by projecting the difference vector onto the pipeline vector. This is visualized in Figure 4.

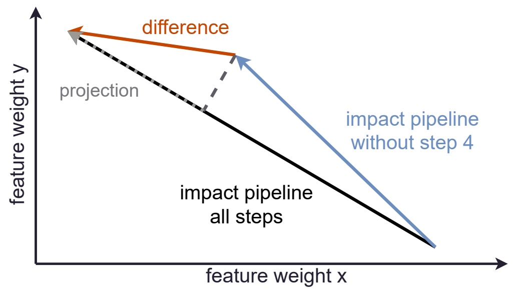

Figure 4: Example of vector calculation. ■ represents the vector of a full pipeline, 🟦 excludes the last step; 🟥 is the difference between the two, projected onto the full pipeline gives the dotted vector, the length and direction of which is the resulting impact.

This most closely corresponds to the question *How much of the way there does this step take us?* (Immediate Impact) or *How close to/far away from the result of the full pipeline does a pipeline without this step take us?* (Leave-out Impact). This normalization means that the absolute value is expressed in percent of the length of the pipeline vector, and the sign indicates the direction, e.g., *1* would mean this covers precisely the path of the pipeline, and any negative value would signify counteracting the effect of the pipeline.

5 PRELIMINARY RESULTS

For the prototype, we list some preliminary results.

Parameters. So far, we have deployed the system on three tabular datasets from the UCI machine learning repository: the Adult Income dataset [1] (dropping categorical columns and binary encoding the target variable beforehand), the MetroPT-3 dataset [5] (encoding date numerically, added target label) and the Phishing Websites dataset [9]. Those datasets were combined with four different, common classifier architectures: Multilayer Perceptron/Neural NetworkfootnoteStandard implementation by scikit-learn[10], Decision Tree[5], Support Vector Machine[5] and Gradient-boosted Trees[1] These cover several common but very different architectures. Samples were chosen randomly from the test portion of each dataset, and at least ten different samples were used for each setting. Our basic pipeline included outlier detection/removal[5], scaling[5], missing value imputation[5], discretization[5] for a subset of features and class balancing through undersampling[5].

Consistency with Intuition. Consistency with the expected behavior is important for trust when providing users with an interpretable metric. With a small series of experiments, we attempted to capture this behavior. Across architectures and datasets, we observed:

(1) When omitting a step with a very low (<0.001) impact, the ratios between impacts of the remaining steps stayed the same.
(2) Impact for missing value imputation was 0 for both datasets without missing values.
(3) When increasing missing values, both impacts first increase and then begin to drop. The percentage of missing values at this turning point varies between data/architecture combinations, but always correlates with a reduction in the importance of this feature, indicating loss of necessary information content.
(4) Scaler had a much smaller impact across the experiments for Decision Trees and gradient-boosted trees than Neural Networks and Support Vector Machines.
(5) When omitting the highest-impact step from a pipeline, overall model performance dropped anywhere from 10 to 40 percentage points.
(6) Class balancing had the greatest impact on the adult income data. Upon closer inspection, models for this data set mainly became majority class voting when the data is unbalanced and restricted to numerical features, while class balancing forced the classifiers to consider feature values.

Runtime. As a tool in data engineering, there are certain soft restrictions on feasible runtime. Our non-optimized prototype implementation is able to consistently perform a full run on a consumer machine solely using CPU in 3 seconds or less for Decision Tree classifiers and gradient-boosted trees. Depending on the dataset, the SVC and the neural network took 1 to 5 minutes. This indicates that even on larger datasets and longer pipelines, usage should be feasible with some code optimization and running on GPUs remotely. A limit is expected, especially in very large datasets; we comment on this in Section 6.

General Trends. In addition, we observed some general trends across different datasets/model architectures that are hard to verify and quantify at this point but were interesting nonetheless. The MetroGPT dataset is least suited to standard classification, being much more directed towards event detection and including sensor data over time. This resulted in worse classifier performance, which correlated

[1]Implementation by xgboost[2]

with the largest variation in measured impact across the same experiment's runs.

6 LIMITATIONS AND OPEN CHALLENGES

In Section 4, we have highlighted the difference between the underlying measures and the scope set for our prototypical implementation. The overall approach is mainly limited to a stable feature attribution method generating numerical weights and preprocessing steps that may be omitted and do not change the number of features. The prototype's scope was set to classification tasks on tabular, numerical data and local explanations using LIME. Yet, for both, open challenges remain that are limitations to the current work described in this paper:

Essential Steps. The current approach handles pipelines consisting only of steps that can be omitted without making preprocessing or model training impossible. On tabular data, this limits us to numeric features as categorical features need to be expressed in numbers for computation, no matter which type of encoding is used. Other restrictions on which numeric values may be provided can also apply, e.g., architectures only compatible with positive values. There are two approaches to handling this: A baseline would involve retaining all essential steps at their relative position in the pipeline and executing them in any variation. This would quickly extend the prototype to categorical features and provide more variations in numeric values, but it would not provide an impact measure for those features. An alternative would involve specifying a baseline variation for the type of processing occurring in each essential step so that these would not be left out entirely but instead replaced with a naive approach to solving the same issue, measuring a relative impact.

Step & Sequence Variations. This comparison process could also be applied to provide an alternate view of the impact of all steps. Comparison between different approaches to one type of processing or different hyperparameters to the processing step are generally of interest to data engineers. Our framework could be extended to automatically generate pipeline variations, replacing a step with all valid, specified alternatives and aggregate resulting impact values to allow for comparison. However, alternatives could also be used in the underlying process for a modified impact measure, as described above. However, this would require additional work to make the resulting values easy to interpret. Another wrapper for the framework could generate pipeline variations with regard to the order of the steps, enabling data engineers to derive conclusions as to optimal sequencing.

Application. Some more insights would be beneficial to make this approach usable for data scientists. A user study would give insight into the interpretability of provided information as well as the best way to present it. Our impact measures provide information to the user in order to enable them to conduct experiments in a more purposeful order. They do not by themselves constitute recommendations. However, it would be interesting to search for patterns made by data engineers after extensive tuning and measuring impact in a larger quantitative evaluation. A study on the effectiveness of different feature attribution explainers in the same setting would also be very interesting.

Approximation and strategic training. Finally, as our current approach for computing the impact measures involves several preprocessing pipelines and training a number of different models scaling with the number of steps in the pipeline, there is an expectation that this will not scale well with specific architectures and data set sizes. With the gold standard provided by these impact measures, an optimized system might try several approaches for reducing overhead: 1) approximating Immediate impact through heuristics such as Earth Movers Distance [4] or 2) learning a model to combine heuristics for approximation based on dataset stats and model architecture to reduce the number of models that need to be trained, 3) implementing a warm start for model training based on previously trained models.

7 CONCLUSION

We presented a local, model-agnostic explainer for preprocessing pipelines in the classification on tabular data. To provide explanations, we introduced two related but distinct measures of impact for a given preprocessing step: *Leave-out Impact* derived from the question *What do we lose or gain by leaving out this step?* and results from comparing model behavior when trained on the full pipeline or the pipeline without that step. *Immediate Impact*, which as a counterbalance relates to the question *What do we lose or gain by adding this step at this moment in processing?* and results from comparing models behavior obtained through a pipeline up to that step and a pipeline including that step. We put both measures in relation to the effect obtained by the pipeline as a whole. As a result, they not only match human intuition, but the combination also covers both the effects of a step's positioning and its interaction with other steps anywhere in the pipeline. We then proposed a process of quantifying and measuring these impacts by representing model behavior around a given sample as feature weights obtained through LIME. However, any feature attribution explainer could be used. Finally, we demonstrated possible observations and their causes on example pipelines.

ACKNOWLEDGMENTS

This research and development project is partially funded by the German Federal Ministry of Education and Research (BMBF) within the "The Future of Value Creation – Research on Production, Services and Work" program (funding number 02L19C150). The author is responsible for the content of this publication.

REFERENCES

[1] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

[2] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[3] Sven F. Crone, Stefan Lessmann, and Robert Stahlbock. 2006. The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. *European Journal of Operational Research* 173, 3 (2006), 781–800. https://doi.org/10.1016/j.ejor.2005.07.023

[4] Tamraparni Dasu and Ji Meng Loh. 2012. Statistical distortion: consequences of data cleaning. *Proc. VLDB Endow.* 5, 11 (jul 2012), 1674–1683. https://doi.org/10.14778/2350229.2350279

[5] Narjes Davari, Bruno Veloso, Rita Ribeiro, and Joao Gama. 2023. MetroPT-3 Dataset. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5VW3R.

[6] Carlos Vladimiro Gonzalez Zelaya. 2019. Towards Explaining the Effects of Data Preprocessing on Machine Learning. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 2086–2090. https://doi.org/10.1109/ICDE.2019.00245

[7] Carlos Adriano Gonçalves, Célia Talma Gonçalves, Rui Camacho, and Eugénio Oliveira. 2010. The Impact of Pre-processing on the Classification of MEDLINE Documents. In *Proceedings of the 10th International Workshop on Pattern Recognition in Information Systems - Volume 1: PRIS, (ICEIS 2010)*. INSTICC, SciTePress, 53–61. https://doi.org/10.5220/0003028700530061

[8] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 4768–4777.

[9] Rami Mohammad and Lee McCluskey. 2015. Phishing Websites. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C51W2X.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[11] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2939672.2939778

[12] Alper Kursat Uysal and Serkan Gunal. 2014. The impact of preprocessing on text classification. *Information Processing & Management* 50, 1 (2014), 104–112. https://doi.org/10.1016/j.ipm.2013.08.006