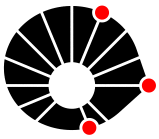

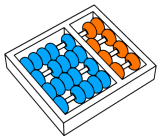

Explainability in Credit Scoring Neural Networks

Leonardo Almeida Reis *Marcos Medeiros Raimundo*

Relatório Técnico - IC-PFG-23-65
Projeto Final de Graduação
2023 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Explainability in Credit Scoring Neural Networks

Leonardo Almeida Reis* Marcos Medeiros Raimundo*

Abstract

Credit scoring is a crucial element in the economic sector, relying on an individual's trustworthiness in honoring financial commitments. Institutions providing credit increasingly leverage Artificial Intelligence (AI) for widespread credit assessments. However, the integration of Machine Learning (ML) models raises ethical concerns, particularly regarding biases inherent in AI models. This document explainabity methods based on feature importance and emphasizes the need for transparency in data utilization. The research aims to enhance model explainability and proposes techniques for a clearer understanding of model operations.

The study employs Feature Importance Techniques, like Permutation Importance, in order to study which features impact the model the most. To address data generation anomalies created by the previous method, Novelty Detection Algorithms, including Isolation Forest and Density Forest, are introduced. Additionally, we also explore Counterfactual Explanations as a method to explain ML model outcomes and how to change an specific data to retrieve a desired prediction.

Keywords— Credit Scoring, Feature Importance, Permutation Importance, Novelty Detection, Isolation Forest, Density Forest, Counterfactual Explanations.

1 Introduction

1.1 Credit Scoring definition

Credit is a fundamental instrument within the economic sector, hinging on an individual's reliability to honor their debts. This reliance on trust is established primarily through the debtor's commitment, necessitating institutions providing credit to employ secure methods for credit distribution. To conduct widespread credit assessments, these institutions are increasingly turning to AI [BCG+21].

Thus, ML algorithms, in particular, have allowed financial institutions to analyze large data sets more efficiently and consider a wider range of variables, which were not taken into account in the old statistical treatments carried out in Excel and even completely manual and subjective methods. Now, AI-based Credit Scoring models can include unconventional data, such as online behavior, spending patterns, social media, and other indicators that provide a more complete view of applicant's financial behavior.

1.2 Ethical questions and research goal

The incorporation of ML models into various applications has prompted a critical examination of ethical considerations. A prominent concern in current discussions revolves around the inherent biases in AI models, often mirroring the prejudices present in training data and the decisions made by developers. This document seeks to assess the impact of features on model behavior, enabling more informed decisions during its development. Additionally, it addresses the imperative of transparency to consumers regarding the utilization of their data by institutions. The objective is to explore and propose techniques that enhance the interpretability of the model, fostering a clearer understanding of its operations.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP

2 Theoretical Background

This section covers background theory we need to understand in order to have a better comprehension of the algorithms used in our project.

2.1 Decision Trees

A decision tree is a supervised ML algorithm that is used for both classification and regression tasks. It is a graphical representation of a decision-making process, where each node in the tree represents a decision based on the value of a specific feature. The tree structure consists of nodes, branches, and leaves. The topmost node in the tree is called the root node. It represents the entire dataset and is split into branches based on the value of a selected feature. Nodes that follow the root node are decision nodes, where a decision is made based on the value of a particular feature. Each decision node has branches corresponding to the possible values of that feature. The branches has two possible values (True or False) and represent the possible outcomes or decisions based on the values of the features. For example, to split data in a node is made a question, like "Is age greater than 40?", then if a sample has a positive answer for that question, it's directed to the left branch, and to the right otherwise. The terminal nodes or leaves of the tree represent the final outcome [KS08].

The final outcome determine the type of a Decision Tree. If the output is a class, the tree is called Classification Tree, if it's a value range instead, the tree is called Regression Tree. Decision trees stand out for their interpretability compared to other classifiers, as they formulate straightforward questions about the data in an easily understandable manner. They also showcase flexibility in handling diverse data types, including a mix of real-valued and categorical features [KS08].

2.2 Impurity Measure

In order to build a Decision Tree, it is important to choose the right question in order to maximize the information gain of a tree. Decision trees are constructed by incrementally adding question nodes based on labeled training examples. The process involves selecting a question that separates the examples as cleanly as possible. A totally pure split would represent a leaf (or terminal node) in a tree, which means an homogeneous subset of data with only one class. Two common measures used to evaluate the degree of impurity in a set of items are Entropy and the Gini Index. The construction of decision trees can be stopped when no question increases the purity of the subsets significantly [KS08].

To calculate the Entropy, we consider a set of training items E that we want to classify into m classes. Let $p_i(i = 1, ..., m)$ be the fraction of items in E that belong to class i. The entropy of the probability distribution (p_i) is given by the equation (1). According to the formula, the lowest entropy occurs when a p_i equals 1 and all others equals 0, meaning that all data belongs to a single class, whereas the highest entropy occurs when all p_i are equal [KS08].

$$-\sum_{i=1}^{m} p_i \log p_i \tag{1}$$

To calculate the Gini Index, we consider a set of items E that we are trying to classify into m classes. Let $p_i(i = 1, ..., m)$ be the fraction of items in E that belong to class i. The Gini Index is computed as 1 minus the sum of the squared probabilities of each class, like in the equation (2). The result is 0 when the set E contains items from only one class [KS08].

$$1 - \sum_{i=1}^{m} p_i^2 \tag{2}$$

While build the tree, the goal is to create split that minimize the entropy. Therefore, the objective is to choose a question that fits our purposes. This is done by calculating the information gain, which is the difference between the entropy of the parent node and the weighted average of the children's entropy. The information gain helps us determine the best question to split the training items and create subsets that are more homogeneous in terms of class labels. Suppose we have a question with k possible answers that divides the training items into subsets $E_1, E_2, ..., E_k$. The impurity of each child node is calculated using the impurity measure. Then, the weighted average of the impurity is computed by taking into account the number of items in each child node. Mathematically, the weighted average

of the impurity is calculated with the formula in equation (2) and we choose a question to minimize it. In the formula, $|E_j|$ represents the number of items in the j-th child node, $|E|$ represents the total number of items in the parent node, and $I(E_j)$ represents the impurity of the j-th child node. [KS08]

$$\sum_{i=1}^{k} \left(\frac{|E_j|}{|E|} \right) \cdot I(E_j) \tag{3}$$

2.3 Ensemble of Decision Trees

While a single decision tree can be effective, combining the results of multiple decision trees can often lead to even better performance. A popular strategies for combining decision trees are random forests and boosting. [KS08]

In random forests, a large number of decision trees are grown using a randomized tree-building algorithm. The data in the training set is randomly selected to create a modified training set of the same size, but with some training items included multiple times. A second modification is that in each node, the chosen question is made based on a random subset of data in that node. This approach creates diverse decision trees that can capture different aspects of the data. The predictions of the ensemble of decision trees are then combined by taking the most common prediction. This approach helps the model decision to be based on a greater diversity of datasets and, therefore, handle complex datasets. [KS08]

3 Feature Importance Techniques

Feature importance is a fundamental concept in ML designed to elucidate the impact of individual features within a model. It quantifies the degree of influence a variable wields in predicted outcomes, offering insights into the model's decision-making process. By comprehending the significance of specific features, we gain an understanding of why particular data points yield specific outputs, empowering us to extract meaningful insights and make decisions based on the model's behavior.

3.1 SHAP Values

SHAP values [LL17] (short for SHapley Additive exPlanations) is an additive feature attribution method, which means that it assigns importance values to features based on their contribution to the prediction. To calculate these importances, the method relies on three desirable properties: local accuracy, missingness and consistency. Local accuracy means that the sum of SHAP values for all features should equal the difference between the model's prediction for a specific instance and the expected prediction (the average prediction for the entire dataset). Missingness refers to the handling of missing values in the dataset, in other words, it should be able to provide insights even when certain features are missing. Consistency implies that similar instances should have similar SHAP values for a given feature.

It calculates the importance of a feature by considering all possible combinations of features and measuring the change in the expected model prediction when conditioning on that feature. The SHAP values represent the average contribution of each feature across all possible orderings of features.

3.2 RFE and EnRFE

The Recursive Feature Elimination (RFE) method is a feature selection method that aims to improve generalization performance by removing the least important features. RFE works by recursively removing features and re-ranking the remaining features. If a feature is deemed weak, RFE will remove it. However, a weak feature may still be important when used in combination with other features. Therefore, simply removing weak features may degrade classification performance. [CJ07]

According to Jong Cheol Jeong [CJ07], To address this limitation, the Enhanced Recursive Feature Elimination (EnRFE) method was proposed. EnRFE redefines the criterion for removing features at each state. It evaluates the importance of a potential weak feature by assessing the classification performance after removing it. If the performance degrades, the feature is retained, even if it has a low

importance score. This process is repeated until a feature is found that does not degrade the classification performance. EnRFE considers the potential usefulness of weak features when combined with other features, leading to improved classification accuracy, especially for a small number of features.

In summary, RFE removes features based on their importance scores without considering the next state, while EnRFE retains potentially useful weak features by evaluating their impact on classification performance. EnRFE outperforms RFE in terms of classification accuracy, particularly for datasets with a small number of features.

3.3 Permutation Importance

To achieve our objectives, we have chosen to employ the method of permutation importance. This approach involves systematically permuting the values of individual features while keeping the others constant. Subsequently, these modified datasets are utilized to retrain a new model. Following this, we compare the predictive accuracy of the new model with that of the original. If the new accuracy proves superior, it indicates that the particular feature under permutation may be impeding our model's performance, signaling a need for its exclusion. Conversely, a decrease in accuracy suggests an increased importance of the feature to the model, highlighting its significance in the overall prediction.

Using real database with payments of credit card clients in Taiwan from 2005 [Tai], the feature importance of each feature was calculated as can be seen in Table 1. The Accuracy Difference is calculate by subtracting the old accuracy by the new one. We can notice that greater values indicate that a feature is more important. In other words, greater values indicates that the accuracy was more impacted by the randomization of data.

Feature	Accuracy Difference
PAY_0	0.0375
PAY_2	-0.015
PAY_3	-0.018

Table 1: Feature Importance in a real database for each feature. PAY_0 meaning repayment status in September, 2005; PAY_2 meaning repayment status in August, 2005; PAY_3 meaning repayment status in July, 2005

According to Table 1, we can notice that Pay_0 has the highest accuracy difference, meaning is the most important feature based on our algorithm. Analyzing the results we can deduct that it makes sense, since that a more recent repayment status have more impact showing the actual financial status of an individual than past repayment status in their records. We can also notice that repayment status in August and July (Pay_2 and Pay_3 respectively) have a negative value of Accuracy difference associated with them (-0.015 and -0.018). This shows us that, after applying the algorithm, the model accuracy has fallen, meaning that this feature is not good for our model.

We can have a better understanding on how this algorithm works based on its pseudo-code presented in Listing 1.

Listing 1: Permutation Importance Pseudo-code

```
def permutation_importance(model, X, num_features):
    # Get the trained model and calculate its accuracy before
    # permutation importance
    current_accuracy = calculate_accuracy(X, model)

    # Initialize an array to store permutation importance scores
    importances = []

    for i in range(num_features):
        # Randomly permutate values of selected feature
        permuted_X = X[feature_index].randomize()

        # Calculate the new accuracy
```

```
new_model_accuracy = calculate_accuracy(permuted_X, model)

# Calculate feature importance subtracting the previous accuracy
# from the new one
feature_importance = current_accuracy - new_model_accuracy

# Append the permutation importance in an array
importances.append(feature_importance)
```

This method were chosen due to its simplicity and versatility. Since it relies solely on how data is structured, it can work with different models we apply to our project. Also, it is extremely simple to implement, providing powerful insight with few code.

3.3.1 Other proposed modifications in Permutation Importance

One variation of the traditional permutation importance algorithm is **AUC (Area Under the Curve) based permutation importance** [JSB13], which focuses on binary classification problems. AUC is a common metric used to evaluate the performance of binary classification models. The main difference from this algorithm to the traditional version is its way of calculating feature importance. It uses the area under the curve (AUC) metric to measure the difference in AUC values before and after permuting the predictor. The AUC is a measure of the model's ability to distinguish between the two response classes, and the larger the difference in AUC, the more important the predictor is considered to be.

A second variation o the traditional permutation importance algorithm is the **Conditional Permutation Importance** [DS20]. It is an extension of permutation importance, which also involves randomly shuffling the values of a single feature and measuring the impact on the model's performance. However, conditional permutation importance takes into account the dependencies between features. It consider the marginal importance (the same calculated by traditional method) and partial importance (importance of a feature relating it with all other features).

3.4 Data generation problems

As this technique involves generating new data, a potential issue that could adversely affect the model may arise. For instance, when permuting the age column in a dataset that includes information such as age and work experience, it's possible to create instances where an individual is 20 years old with 30 years of work experience. It's evident that such data are logically implausible in real-life situations and could be detrimental to the model. Therefore, a systematic approach is required to identify and exclude these anomalies.

4 Novelty Detection Algorithms

Fortunately, the problem stated in the previous section can be address by some existing ML methods. These are knows as Novelty Detection Algorithms and aim to detect data that are differ significantly from the real data. Such data are considered as "anomalies" or "novelties".

4.1 Isolation Forest

The Isolation Forest algorithm uses a collection of trees (also known as iTrees) to detect anomalies and the collection of iTrees is called iForest. Each iTree is constructed by recursively partitioning the data until all instances are isolated. The partitioning process starts by randomly selecting a feature and a split point within the range of that feature. The instances are then divided into two child nodes based on whether their feature value is below or above the split point. The partitioning process continues recursively for each child node until either a predefined maximum tree depth is reached or there is only one instance left in the node. [LTZ08]

Anomalies are easier to isolate. When building an iTree, they are more likely to be nearby the tree root, whereas normal points are more likely to be closer to the leaves of the tree. Analyzing the iForest, in each tree we can score its samples by their path distance from the root. Anomalies will

commonly have shorter paths compared to normal points. The average of these scores will determine if a sample is an anomaly or not. [LTZ08]

Isolation Forest algorithm works well with large datasets with high number of irrelevant attributes. It also has linear time complexity, making it one of the fastest algorithms in anomaly detection. [LTZ08]

For clarification purposes, let's create an example of this algorithm working. Suppose a very simple dataset with 2000 samples and only two features, such as Age and Salary. The first step is to create an iTree. For each tree to be constructed, the algorithm randomly selects one of the two available features (Age or Salary) and an associated cut value. Assume that the first split randomly chooses Age with the value of 45. Therefore, the data is divided into two sets: one where values an age associated of less than 45 and another where values are greater than or equal to it. So now the tree has a root node with all the data split in two child nodes. For example, 1300 goes to the node of people are young than 45 years old and 700 goes fit in other node. Then, a new feature is randomly selected and the splitting process continues recursively until each data point is isolated in a leaf of the tree (or a stop criteria is accomplished). This means that outliers tend to be isolated more quickly, requiring fewer steps in the tree. However, since the splitting decision is random, it is possible that an outlier takes longer to be segregated or a normal data point is segregated really fast. For that reason the process of tree construction and path length calculation is repeated several times. The average paths for each point are then aggregated, providing a more robust view of anomaly. Based on the calculated average paths, a threshold is established. Points with average paths significantly shorter than the threshold are considered outliers, indicating that these points were isolated more quickly in the trees, suggesting an anomaly.

There are two information that can be retrieved from Isolation Forest algorithm. The first is the ease to isolate a data point from the rest of the dataset, as shown in Figure 1. In the figure, darker colors represent data points that are easier to be segregated.

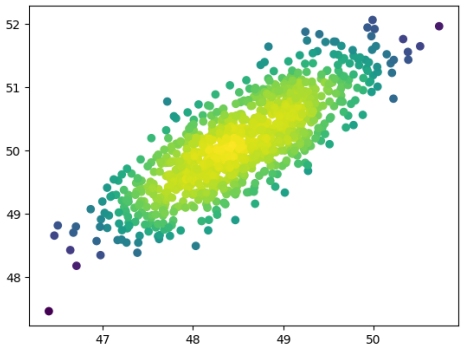

Figure 1: Isolation Forest fit on mock data

The second information is the actual separation between good data and outliers, as shown in Figure 2. This is achieved by establishing a threshold of separation ease for each data. In the figure, yellow points represent data to be used, whereas purple points were classified as outliers.

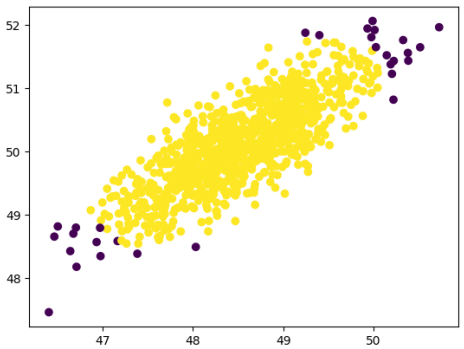

Figure 2: Isolation Forest prediction on mock data

4.2 Density Forest

While Isolation Forest algorithm divides data into two different groups (normal samples and anomalies), Density Forest creates a Gaussian-like spectrum of classifying the data. This algorithm partitions the data based on Gaussianity and uses this information to estimate the confidence of unseen data points. It can effectively detect novelties by identifying data points that deviate significantly from the classes it creates. [WMT19]

First it starts by partitioning the data into random subsets, very similar to the Random Forest algorithm. Then, its Information Gain function calculates the entropy based on a Gaussian distribution for each tree. This function compare the entropy before and after splitting the data with the goal of finding the splits with higher information gain. [WMT19]

To estimate the density, each tree in the forest assigns a weight to each data point based on its position in the tree. The weights are determined by the proportion of training samples that fall into the same leaf node as the data point. The final density estimate is obtained by aggregating the weighted densities from all the trees in the forest. [WMT19]

The idea of this algorithm is that most of the data points will fall closer to the Gaussian mean of its class some of them will be further away, which will be classified as anomalies. The advantage of this algorithm is that we can make a better analysis on how far an specific data point is from is class. With further investigation, we could determine on how much some point has to change in other to become part of a specif class. This technique is also known as counterfactual explanation. In other words, what could a bad payer classified person do to change its classification to a good payer in order to receive credit from the financial institution.

In Figure 3 we can observe that brighter points represent higher certainty of a point belonging to a class, whereas dark color means less assurance. Also, points marked with an "x" symbolize data that do not fit in a class. We can interpret that brighter areas formed by groups of bright points form the classes predicted by the model.

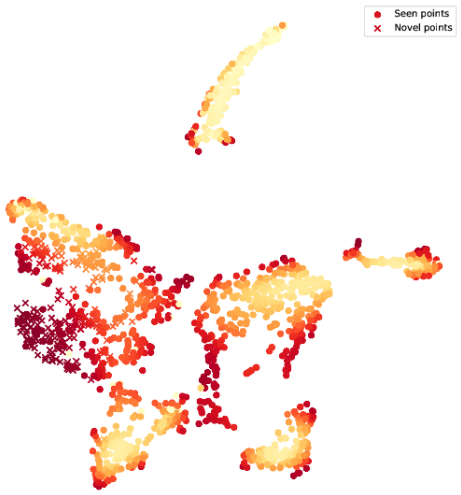

Figure 3: Density Forest result confidence of each data point in a set [Wen18]

5 Experiments

For our experiments, we chose to use the Taiwan database which has information from credit card clients in 2005 [Tai]. This dataset has information related to each individual such as sex, marriage status, education level, repayment status, amount of previous payments, amount in their bill statement and amount of given credit. As feature to be predicted, the database provide the information if the client tend to pay their credit card statement next month or don't. Since the objective of this paper is to investigate explainability, and not the model itself, for our experiments we chose to train a simple Random Forest Classifier from sklearn. Using 70% of the data for training and 30% for testing, their accuracies were 86% and 82% respectively. As said before, to analyze the importance of each feature

we chose to use permutation importance algorithm and ran it on each feature for the training set, resulting in some values of importance for each feature, that are shown in the Table 2. The new accuracy after randomizing each feature was calculated after applying Isolation Forest in the new data to remove outliers. Another point to notice is that, since this algorithm includes randomizing data, a small dataset like ours can suffer from fluctuations in its calculations. Thus, to mitigate this problem, we concatenated the dataset with copies from itself multiple times to create a more stable database.

Feature	Accuracy Difference
PAY_0	0.05424
PAY_2	0.00371
BILL_AMT1	0.00195
PAY_6	0.00146
PAY_5	0.00133
BILL_AMT4	0.00130
LIMIT_BAL	0.00129
PAY_3	0.00128
PAY_AMT5	0.00115
BILL_AMT2	0.00111
BILL_AMT3	0.00071
BILL_AMT5	0.00070
PAY_4	0.00069
EDUCATION	0.00049
AGE	0.00036
PAY_AMT2	0.00008
PAY_AMT1	0.00005
PAY_AMT4	0.00004
PAY_AMT3	-0.00007
SEX	-0.00025
BILL_AMT6	-0.00032
MARRIAGE	-0.00118

Table 2: Feature importance for each feature in Taiwan Database using Isolation Forest as novelty detection algorithm

Observing the Table 2, we notice a list with all Taiwan Database features ranked in descending order by its importance to the model prediction. PAY_0 (repayment status in September 2005) is ranked as the most importance feature, with the highest impact the model accuracy. Since it is a more recent repayment status, it makes sense that it is a more important feature than PAY_2 (repayment status in August 2005) and other repayment status in older months. In other words, it makes sense this feature is more related to one's ability to pay their debts.

We notice that some BILL_AMT (amount of bill statement) features are ranked highly as well. This means that the value one has to pay in a month for their credit card statement, have great impact in their ability to pay for it in the next month. Curious enough, BILL_AMT4 (amount of bill statement in June 2005) is ranked pretty high and has a importance close to more recent statements, like BILL_AMT1 and BILL_AMT2 (related to September and August respectively), with its reasons not clear. It is also interest to notice that some features, like MARRIAGE and SEX, have a negative impact on model's prediction. Meaning that if this feature can be hindering the model, instead of improving its prediction.

After critically looking into these analysis, we have some information about which features to consider or don't. But unfortunately, this method doesn't show how a feature influences the result. For example, having a higher LIMIT_BAL is beneficial or harmful to a customer ability to pay for their credit card statement? That is not an answer to be given solely by the algorithms and methods in this document.

For comparing purposes, we also built a table of features importance before applying the Isolation Forest algorithm, as can be seen in Table 3.

Feature	Accuracy Difference
PAY_0	0.05524
PAY_2	0.00232
BILL_AMT1	0.00231
BILL_AMT4	0.00151
PAY_AMT5	0.00122
BILL_AMT2	0.00114
LIMIT_BAL	0.00106
PAY_4	0.00096
PAY_6	0.00086
PAY_5	0.00082
BILL_AMT5	0.00061
EDUCATION	0.00060
PAY_3	0.00055
SEX	0.00053
BILL_AMT3	0.00045
PAY_AMT2	0.00042
AGE	0.00036
PAY_AMT1	0.00035
PAY_AMT4	0.00031
PAY_AMT3	-0.00001
BILL_AMT6	-0.00009
MARRIAGE	-0.00064

Table 3: Feature importance for each feature in Taiwan Database without using any novelty detection algorithm

According to Table 3, one of the most vital changes our ranking suffered is with SEX feature. Although this feature had a negative value associated in the previous table, is now ranked with a positive importance to the model. It is ethically questionable to determine SEX as an important feature to classify one's ability to pay their debts. Although PAY_2 didn't change its position, its importance had drop significantly in comparison to Table 2. In general, almost all BILL_AMT features had their importance higher before removing outliers, displaying a scenario where the amounts in bill statements throughout the months add up a great importance in predicting whether or not an individual would be classified as a good or bad payer.

6 Conclusion

In conclusion, our exploration into the explainability of neural network prediction models in credit scoring has helped us comprehending and interpreting tools used to accomplish this goal. Through these studies we deepened our knowledge in feature importance and novelty detection algorithms along with all the theoretical background necessary to understand them fully.

The implementation of Permutation Importance was proven as a transparency tool in understanding an existing model and in the decision-making process of modifying it to achieve better results. However, our research also led us to confront challenges associated with the generation of new data by Permutation Importance algorithm, particularly the introduction of outliers.

To address this concern, we employed the Isolation Forest algorithm, demonstrating its effectiveness in mitigating the impact of outliers and enhancing the overall robustness of our analysis. This approach had an significant impact in our feature importance calculation results as we see in our experiments.

The Taiwan Credit Card Clients Dataset from 2005 [Tai] served as a crucial database for our experiments, showing a real-life case where our studies can be applied. The insights gained from our experiments were particularly relevant in the context of credit scoring, giving tools to comprehend predictions more transparently and opening ethical discussions on this matter.

References

[BCG+21] Przemysław Biecek, Marcin Chlebus, Janusz Gajda, Alicja Gosiewska, Anna Kozak, Dominik Ogonowski, Jakub Sztachelski, and Piotr Wojewnik. Enabling machine learning algorithms for credit scoring–explainable artificial intelligence (xai) methods for clear understanding complex predictive models. *arXiv preprint arXiv:2104.06735*, 2021.

[CJ07] Xue-wen Chen and Jong Cheol Jeong. Enhanced recursive feature elimination. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 429–435, 2007.

[DS20] Dries Debeer and Carolin Strobl. Conditional permutation importance revisited. *BMC Bioinformatics*, 2020.

[JSB13] Silke Janitza, Carolin Strobl, and Anne-Laure Boulesteix. An auc-based permutation variable importance measure for random forests. *BMC Bioinformatics*, 2013.

[KS08] Carl Kingsford and Steven L Salzberg. What are decision trees? *Nature Biotechnology*, 2008.

[LL17] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874, 2017.

[LTZ08] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[Tai] Default of credit card clients dataset. https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset. Accessed: 2023-12-06.

[Wen18] Cyril Wendl. Density forest result confidence of each data point in a set. https://github.com/CyrilWendl/SIE-Master/blob/master/Figures/Zurich/GIF/probas.png, 2018.

[WMT19] Cyril Wendl, Diego Marcos, and Devis Tuia. Novelty detection in very high resolution urban scenes with density forests. In *2019 Joint Urban Remote Sensing Event (JURSE)*, pages 1–4, 2019.