Analysis of Explainability ML and DL Techinques against Adversarial Models

Palli Nicola - nicola.palli@studio.unibo.it
Terzi Angelo - angelo.terzi2@studio.unibo.it

Abstract

In the field of machine learning, the rising prevalence of complex models, often referred to as "black boxes", underscores the critical need for explainability. As these intricate models become increasingly central to decision-making across several sectors, understanding the reasons behind their predictions is essential. In this study, we conducted a comprehensive exploration of various explainability approaches, evaluating traditional methods such as LIME, SHAP, and DICE. Additionally, we examined DeepLIFT, a method that provides insight into neural network predictions by decomposing the output in terms of input contributions, offering a more nuanced understanding of the underlying mechanisms driving model behavior. By comparing these techniques, we aim to highlight their strengths and limitations, to assess their applicability in real-world scenarios.

1 Introduction

Artificial Intelligence has markedly improved the accuracy of deriving conclusions from complex data. However, these algorithms often present a challenge for individuals outside the AI domain, as their operations can be intricate and hard to grasp. This becomes particularly important in practical terms because models driven by observational data can be difficult to correct for biases and other inherent artifacts within the dataset. In numerous scenarios, there is a necessity for dependable and transparent models, allowing their credibility to be validated against sector-specific expertise. This goes beyond local explanations, such as feature attributions, requiring a broader interpretation. It involves elucidating how the weight of individual input variables influences the model's response consistently across the entire range of inputs. The rise of powerful AI methods requires users to comprehend the inner workings of such models, emphasizing the importance of model transparency. This transparency refers to a clear and readily understandable flow of information from input to response. In the era of significant AI development, gaining access to methods that enhance the transparency of black-box models becomes crucial. Understanding these complex models is not only beneficial for extracting meaningful insights but also plays a pivotal role in identifying potential problems within the model pipeline. Transparent models facilitate a more effective debugging process, allowing users to pinpoint errors with greater clarity. This could involve the identification of features playing unintended roles or the detection of any anomalies that might affect the model's performance. Overall, the push for transparency aligns with the broader goal of making AI systems more interpretable and user-friendly, enabling users to have a deeper understanding of the decision-making processes within these advanced models.

2 LIME Theory

LIME (Local Interpretable Model-agnostic Explanations) is a popular post-hoc explainability technique designed to provide insight into complex machine learning models by approximating them with simpler, interpretable models locally around a particular prediction. Its fundamental principle lies in understanding the behavior of "black-box" models by perturbing the input data and observing the changes in output, which can then be used to construct a more interpretable, linear model in the local neighborhood of the instance being explained. LIME is model-agnostic, meaning it can be applied to any type of machine learning model—whether it is a decision tree, neural network, support vector machine, or any other. It treats the original model as a black-box and only requires access to the input-output pairs. This agnosticism ensures broad applicability across various models and domains [1]. It focuses on generating local explanations by selecting a specific instance to explain and creating synthetic data by perturbing this instance. The perturbed samples are then fed into the black-box model, and their predictions are recorded. Using this data, LIME fits a simpler interpretable model that approximates the decision boundary of the complex model in the vicinity of the original instance.

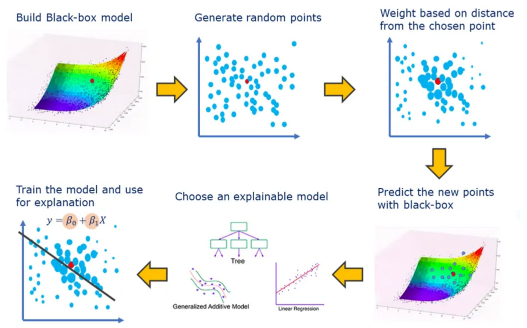

Build Black-box model
Generate random points
Weight based on distance from the chosen point
Predict the new points with black-box
Choose an explainable model
Tree
Generalized Additive Model
Linear Regression
Train the model and use for explanation
$y = \beta_0 + \beta_1 X$

Figure 1: LIME Steps

3 SHAP Theory

SHAP (SHapley Additive exPlanations) is a post-hoc explainability technique based on cooperative game theory to explain the predictions of complex models. SHAP's goal is to assign each feature a contribution value that explains its impact on the model's output. This method provides both local and global explanations, grounded in a solid theoretical framework that ensures the consistency and fairness of the explanations. SHAP is built upon the concept of Shapley values, originating from cooperative game theory. In the context of machine learning, Shapley values are used to determine the individual contribution of each feature to the overall prediction. Simply put, Shapley values calculate the importance of a feature by considering all possible combinations of other features and averaging its marginal contribution to the prediction in each combination. The Shapley value for a feature i is computed as the weighted average of its marginal contribution across all possible subsets of the other features. This ensures that each feature's importance is fairly distributed, accounting for interactions between features. SHAP combines Shapley values with an additive framework to explain predictions of complex models. In an additive model, the prediction f(x) can be expressed as the sum of the contributions from individual features:

$$f(x) = \Phi_0 + \sum_{i=1}^{N} \Phi_i \tag{1}$$

This additive decomposition allows the explanations provided by SHAP to be easily interpretable, as it clearly shows how each feature contributes positively or negatively to the overall prediction.

3.1 SHAP Advantages

SHAP provides a detailed breakdown of the contribution of each feature for a specific prediction. This is particularly useful for understanding why a model made a particular decision. By aggregating SHAP values across multiple instances, it's possible to gain insight into the overall behavior of the model and the relative importance of features. This helps in understanding which features most influence the model's predictions in general. Consistency: if a feature's contribution to a prediction increases, with all else held equal, its SHAP value will increase accordingly. Local Accuracy: SHAP ensures that the sum of the feature contributions exactly matches the model's prediction for a given instance. Symmetry: Features with identical effects on the prediction will receive the same SHAP value. Several implementations and variants of SHAP have been developed to improve efficiency for different types of models:

- **Kernel SHAP**: A model-agnostic variant that can be applied to any machine learning model.
- **Tree SHAP**: Optimized for tree-based models like random forests and gradient boosting, significantly reducing the computational complexity.
- **Deep SHAP**: A variant specifically designed for deep neural networks.

3.2 SHAP Limitations

High Computational Cost: Calculating Shapley values requires considering all possible feature combinations, which can be computationally expensive, especially for models with many features. To address this, SHAP often uses approximation methods. Complexity of Interpretation: While SHAP produces interpretable explanations, understanding them can be challenging for models with many features and nonlinear interactions.

4 DiCE Theory

DiCE (Diverse Counterfactual Explanations) is an explainability technique that generates diverse counterfactual explanations for machine learning models. Counterfactual explanations are hypothetical scenarios that show how minimal changes to the input features could lead to a different prediction. DiCE focuses not only on generating such explanations but also ensuring that the counterfactuals are diverse, giving users multiple possible pathways to achieve a different outcome. Counterfactual explanations answer the question, "What needs to change in the input to get a different result?" This concept is rooted in the idea of causality, where slight changes to an individual's input can cause a different outcome. For example, if a model rejects a loan application, a counterfactual explanation might reveal that increasing the applicant's income by a certain amount would result in

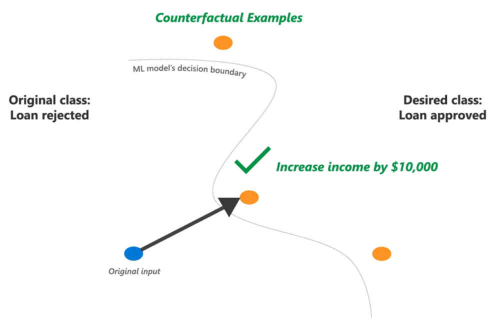


Figure 2: A DiCE example of a 2-classes counterfactual generations

loan approval. Formally, given a model $f(x)$ that makes a prediction for an instance x, the counterfactual explanation x' is an alternative input where the prediction $f(x')$ is different from the original. A good counterfactual explanation must satisfy the following conditions:

- **Validity**: The counterfactual x' must result in a different prediction.
- **Proximity**: The changes made to x to generate x' should be as minimal as possible, ensuring the counterfactual remains realistic and close to the original instance.
- **Sparsity**: Only a few features should change between x and x', making the explanation easier to interpret.
- **Actionability**: The changes suggested in the counterfactual should be feasible for the user to act upon (e.g., changing income might be actionable, while changing age is not).

DiCE extends traditional counterfactual approaches by emphasizing diversity. Instead of providing a single counterfactual explanation, DiCE generates multiple diverse counterfactuals. This is particularly useful because there are often several ways to achieve a different outcome. For example, in a credit decision scenario, one counterfactual might suggest increasing income, while another might propose paying off existing debt.

DiCE uses a generative approach that aims to cover a wide range of feasible counterfactuals while ensuring that the generated instances are valid and actionable. The diversity aspect allows the user to explore different options and select the one that best fits their context or constraints.

5 DeepLIFT Theory

DeepLIFT (Deep Learning Important Features) is a technique designed to provide interpretable explanations for the predictions of neural networks [2]. It offers a way to understand how individual input features contribute to the final output of a deep learning model by decomposing the output into feature-specific contributions. DeepLIFT builds on the idea of backpropagation but addresses some of the limitations of gradient-based methods, such as vanishing gradients. At its core, DeepLIFT operates by comparing the activation of neurons to a reference activation, which represents the network's response when given a neutral input, often called a "baseline." The goal is to compute the contribution of each input feature to the model's output, relative to this baseline. This comparison ensures that DeepLIFT accounts for both the magnitude and direction of changes in activations, providing a more nuanced view of feature importance. DeepLIFT computes contributions using the following key components:

- **Reference Activation**: This is the output of each neuron when the model is given a reference input (e.g., an all-zero or all-average input). The reference serves as a neutral point of comparison for understanding the importance of real inputs.
- **Actual Activation**: This is the output of each neuron for the actual input being analyzed.
- **Contribution Score**: For each neuron, DeepLIFT calculates the difference between its actual activation and its reference activation. This difference is propagated backward through the network, with each neuron's contribution being distributed among its inputs according to their respective influences.

The contribution C(x) of input feature x_i to the model output f(x) is defined as:

$$C(x_i) = \sum_j \frac{\Delta f}{\Delta a_j} * \frac{\Delta a_j}{x_i} \tag{2}$$

Where Δf is the change in the model's output relative to the baseline and Δa_j is the change in the activation of the intermediate neuron j relative to its reference.

6 Adversarial Methods

Explainable artificial intelligence (XAI) methods, including post-hoc techniques like Partial Dependence Plots (PDP), Shapley Additive Explanations (SHAP), Local Interpretable Model-agnostic Explanations (LIME), Integrated Gradients (IG), and others, offer diverse mechanisms for interpreting the predictions of machine learning models. While XAI has found success in various applications such as autonomous driving and drug discovery, it faces criticism for its inability to faithfully explain complex black-box predictive functions. Nevertheless, recent studies on adversarial machine learn-

ing (AdvML) have shed light on the vulnerabilities of explanation methods, raising concerns about their trustworthiness and security. Adversarial attacks, such as data poisoning, model manipulation, and backdoors, have emerged as prominent failure modes of XAI methods, prompting the development of defense mechanisms like focused data sampling and model regularization. The primary aim of attackers is to manipulate model explanations by altering data or models to deceive recipients. This poses a significant threat in sensitive applications, such as explaining decision-making systems used in legal contexts. Notably, attacks on explanations vary depending on the machine learning task, model architecture, and class of XAI methods, highlighting the need for systematic understanding and mitigation strategies.
The fooling consists in training a model by manually influencing the importance of input features, while maintaining an accuracy comparable to the original one. In this way, it becomes possible to both manipulate the fairness of a model and demonstrate the real behavior of a model that might initially appear fair.

- **Passive Fooling**: We define passive fooling as the act of inducing interpretation methods to produce uninformative explanations. To achieve this, *Location Fooling* is implemented. Here, we can directly choose which features should be more relevant for the model, by using the following penalty loss function:

$$L_F^I(D, w, w_0) = \frac{1}{n}\sum_{i=1}^{n} ||h_i^I(w) - m||^2 \qquad (3)$$

  where n is the number of features, h^I is the heatmap generated by a interpretation method I for w and $m \in R^n$ is a vector in which $m_i = 1$ if the i_{th} feature must be relevant, and $m_i = 0$ otherwise

- **Active Fooling**: it's a concept where the goal is deliberately inducing interpretation methods to produce incorrect or misleading explanations by swapping the importance level between couples of features:

$$L_F^I(D, w, w_0) = \frac{1}{n}\sum_{i,j} ||h_i^I(w_0) - h_j^I(w)||^2 \qquad (4)$$

  where n is the number of couples of features to swap and h^I is the heatmap generated by a interpretation method I for w.

The methods are exposed in [3], used in the image processing field.

7 Data

The datasets used were downloaded from Kaggle. Two different datasets were used: the "Medical Insurance Payout" dataset [4], used for regressions, and the "Heart Attack Analysis & Prediction Dataset" [5].

7.1 Datasets

ACME Insurance Inc. offers affordable health insurance to thousands of customers throughout the United States. The dataset contains data from 1,338 individuals, characterized by 3 numerical features (age, number of children, and BMI), 3 categorical features (gender, smoker, and region), and a numerical value referring to the insurance amount.

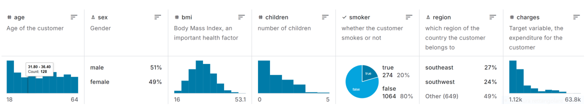
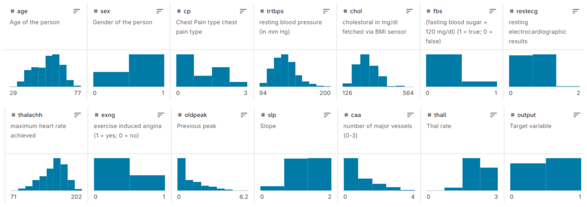


Figure 3: Medical Insurance Payout Dataset Stats

The "Heart Attack Analysis & Prediction Dataset" contains information on 303 patients, including the main clinical factors to consider in correlation with heart attacks, such as age, sex, maximum heart rate, blood sugar, and others. A total of 13 numerical features can be used to predict whether a patient is at risk of a heart attack or not.



# age
Age of the person
29
77
# sex
Gender of the person
0
1
# cp
Chest Pain type chest pain type
0
3
# trtbps
resting blood pressure (in mm Hg)
94
200
# chol
cholestoral in mg/dl fetched via BMI sensor
126
564
# fbs
(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
0
1
# restecg
resting electrocardiographic results
0
2
# thalachh
maximum heart rate achieved
71
202
# exng
exercise induced angina (1 = yes; 0 = no)
0
1
# oldpeak
Previous peak
0
6.2
# slp
Slope
0
2
# caa
number of major vessels (0-3)
0
4
# thall
Thal rate
0
3
# output
Target variable
0
1

Figure 4: Heart Attack Analysis & Prediction Dataset

7.2 Preprocessing

The preprocessing applied is the same for all datasets. The input columns were pre-processed differently depending on the type of values: for numerical features, the columns were normalized using *StandardScaler*, subtracting the mean of their values from each column and scaling their variance to 1. Missing values were filled with the median. The values of the categorical columns were replaced with integers using *OrdinalEncoder*, which assigns a unique identifier to each different categorical value. Missing values were filled with the tag "missing," and were thus assigned a different numerical identifier.

8 System Description

To study the implementation and usage methods of the various explainability techniques, the libraries *shap*, *lime*, and *dice* of Python 3.10.15 were used. DeepLIFT required the use of Python 3.7.17. The versions of the packages used are listed below:

Python 3.10.15:

- numpy: 1.26.4
- pandas: 1.5.3
- keras: 3.7.0
- scikit-learn: 1.5.2
- matplotlib: 3.9.3
- lime: 0.2.0.1
- shap: 0.46.0
- tensorflow: 2.18.0
- dice-ml: 0.11
- protobuf: 5.29.0
- scipy: 1.14.1

Python 3.7.17:

- deeplift: 0.6.13.0
- keras: 2.3.1
- matplotlib: 3.5.3
- numpy: 1.19.5
- pandas: 1.3.5
- protobuf: 3.20.0
- scikit-learn: 1.0.2
- seaborn: 0.12.2
- tensorflow: 1.15.2

The various explainability methods were tested on different classifier and regressor models.

Regressors	Classifiers
Linear Reg.	Naive Bayes
Decision Tree Reg.	Decision Tree Clas.
Random Forest Reg.	Random Forest Clas.
Gradient Boost Reg.	Gradient Boost Clas.

Table 1: ML models for regression and classification

In addition to the ML models presented in Table 1, a neural network was implemented for regression and a neural network for classification. The two networks are presented in Table 2.

Regression	Classification
Dense Layer (2048)	Dense Layer (2048)
Activation ReLU	Activation ReLU
Dense Layer (256)	Dense Layer (256)
Activation ReLU	Activation ReLU
Dense Layer (1)	Dense Layer (1)
Activation Linear	Activation Linear
	Activation Sigmoid

Table 2: NN layers for regression and classification

Once the models are chosen and the datasets prepared, they are trained and tested using a train-test split of 0.75. Neural networks are trained for 1500 epochs. After training is complete, the models are evaluated by calculating the metrics *Mean Absolute Error (MAE)*, *Mean Squared Error (MSE)*, *Root Mean Squared Error (RMSE)*, *Symmetric Mean Absolute Percentage Error (SMAPE)* for the regressors, and, for classifiers, the *micro f1-score* and *accuracy.*

At the end of the training, the explainability methods mentioned in previous chapters are tested.

8.1 Gold Standard

In Machine Learning models, an initial approach to evaluate the importance of a feature is to observe the model parameters. Although coefficients are useful for understanding the effect of changing the value of an input feature, they alone are not an ideal way to measure the overall importance of a feature. This is because the value of each coefficient depends on the scale of the input features. For example, if we were to measure the age in minutes instead of years, the coefficient for this feature would become 365*24*60 times smaller. In this project, the model coefficients are initially considered to obtain an estimate of each feature's importance, which will then be compared with the results from classical methods.

For a linear model, it is sufficient to consider the magnitude of each feature's coefficient to determine its importance. The higher the absolute value of a coefficient, the more it contributes to the model's output. The sign of each coefficient indicates how it affects the output.

For the dt model, feature importance is calculated by observing the reduction in impurity (entropy or Gini Index) that each feature provides when used within a node. The same method is used for the Random Forest, where the average feature importance across all trees is calculated in the end.

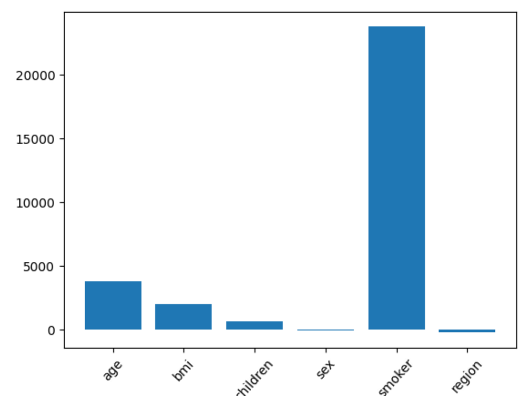


Figure 5: The coefficient measures of an lr trained on "Medical Insurance Payout Dataset".

Reg	lr	dt	rf	gbr	NN
MAE	4107	2737	2593	2733	2218
RMSE	6035	4982	4923	4919	4102
MAPE	19.6%	12.9%	12.6%	14.4%	10.6%
Class	nb	dt	rf	gbc	NN
f1-score	0.84	0.76	0.87	0.88	0.84

Table 3: Models performances

For Gradient Boosting, the calculation is similar to the previous method, although each tree in the ensemble is built sequentially to reduce the residual error of the preceding tree. The average feature importance across trees is thus a weighted average based on the total impurity reduction.

Although the model's performance is not the primary focus of this project, it is important to ensure that it performs well enough to make the calculated feature importances reliable. A poorly performing model could lead to unreliable results when evaluating feature importance, so achieving a minimum level of performance is essential.

8.2 LIME

For LIME, 5 examples of local explainability are generated for each model. Once the model is trained, 5 points are randomly selected from the test set. These points will be used to generate the local explanation. On these points, the explainability function will linearize the model in the form $E(Y) = \beta_0 + \sum_j \beta_j X_j$. The coefficients β will serve as an explanation for the features at the selected point.

8.3 SHAP

The SHAP explainer is initialized using the model's prediction function. This function defines what the SHAP values will explain (e.g., raw predictions, probabilities, etc.). A masker is used to handle feature masking during the computation of SHAP values. In this case, shap.maskers.Independent assumes that the features are independent of each other. The explainer is also provided with a reference dataset, which SHAP uses to approximate the background distribution of the features.
The explainer is applied to the preprocessed training data, producing SHAP values for each instance in the dataset. These SHAP values indicate how much each feature contributes to the model's prediction for every training sample.
The explainer calculates SHAP values for the preprocessed test data. These values reveal how the features influence the predictions in new unseen data.
The expected value (it's mean prediction) of the model for the test data is computed. This provides a baseline value, which can help contextualize the SHAP values. For example, a feature's SHAP value shows how much it shifts the prediction from this baseline.

8.4 DiCE

DiCE is used for the generation of counterfactuals. To assess the model's sensitivity to variations in input within a local area, slightly modified input data are used to verify whether such changes result in an output variation. Counterfactual generation involves making small adjustments to the inputs and modifying only a few features at a time to facilitate identifying the cause of any output change. Additionally, to avoid generating unfeasible inputs, only inputs with numerical feature values within the minimum and maximum values observed in the training data were generated, and no new categorical feature values were introduced.
For each input in the test set, a selected number of counterfactuals are generated. Global explainability evaluation using DiCE is missing. The method used to study the impact of each feature on the output involved calculating the *Counterfactual Absolute Count* (CAC) and *Counterfactual Relative Count* (CRC) metrics for each feature. The first metric assesses how often each feature was modified, considering only those counterfactuals that resulted in output changes. The second metric divides this count by the number of times the feature was altered and a factor proportioned to its output variation. Features with a lower CRC may be those capable of modifying the output more significantly, and thus the most important for the model.

8.5 DeepLIFT

DeepLIFT (Deep Learning Important FeaTures) is a method for interpreting deep neural networks by assigning feature attributions based on a reference input. The implementation follows these theoretical steps:

- **Model Conversion**: The pre-trained neural network model is converted into a DeepLIFT-compatible format. A specific nonlinear transformation mode, such as `DeepLIFT_GenomicsDefault`, ensures that contributions are backpropagated consistently with the model's architecture.

- **Selection of Layers**: A scoring function is created, specifying the input layer for feature contributions and the target layer representing the model's predictions.

- **Computation of Contributions**: Contributions are computed for the test data by applying the DeepLIFT scoring function. The process is performed in batches for efficiency, with contributions calculated for a specified target output (e.g., the first neuron in a multi-output model).

- **Normalization and Visualization**: The absolute contributions are normalized to a $[0, 1]$ range, and median values are used to rank feature importance. The distribution of contributions for each feature is visualized using a boxplot to highlight their significance in the model's predictions.

This process provides a transparent way to interpret the influence of input features on the neural network's decisions, enabling better understanding of complex models in tasks such as genomics or other high-dimensional domains.

8.6 Fooling methods

The employed models are a Polynomial Regressor and a Polynomial classificator, with a focus on the results derived from the Linear Moldes, which are the Polynomials of degree 1.

The proposed model manipulation involves fine-tuning a pre-trained model with an objective function that integrates the standard classification loss with a penalty term based on the interpretation results. Accordingly, the overarching objective function for a model w, aimed at minimizing it for training data D utilizing the interpretation method I, is formulated as follows:

$$L(D, w, w_0) = L_C(D, w) + \lambda L_F^I(D, w, w_0) \quad (5)$$

where L_C is the original MSE loss function, w_0 are the original parameters of the model, D is the dataset and λ is a trade-off parameter.

- **Passive Fooling**: We define passive fooling as the act of inducing interpretation methods to produce uninformative explanations. To achieve this, *Location Fooling* is implemented. Here, we can directly choose which features should be more relevant for the model, by using the following penalty loss function:

$$L_F^I(D, w, w_0) = \frac{1}{n}\sum_{i=1}^{n} ||h_i^I(w) - m||^2 \quad (6)$$

  where n is the number of features, h^I is the heatmap generated by a interpretation method I for w and $m \in R^n$ is a vector in which $m_i = 1$ if the i_{th} feature must be relevant, and $m_i = 0$ otherwise

- **Active Fooling**: it's a concept where the goal is deliberately inducing interpretation methods to produce incorrect or misleading explanations by swapping the importance level between couples of features:

$$L_F^I(D, w, w_0) = \frac{1}{n}\sum_{i,j} ||h_i^I(w_0) - h_j^I(w)||^2 \quad (7)$$

  where n is the number of couples of features to swap and h^I is the heatmap generated by a interpretation method I for w.

There methods are exposed on paper [3], used in image processing field.

A second adversarial techinque is tested, in order to verify its effectiveness on fooling the classical and most used explainability techniques, such as LIME and SHAP [6].In particular, throught an adversarial model which exploits the individual predictions of a given black box model by constructing local interpretable approximations.

To gain insights into the synthetic data points generated through perturbations, we conducted an experiment as follows: Initially, we perturbed input instances using the method utilized by LIME, as described in the previous section. The synthetic data points resulting from input perturbations exhibit a significantly different distribution compared to the instances in the original input data. This observation suggests that discerning whether a data point is a product of perturbation or not is not a challenging task. Consequently, methodologies heavily reliant on such perturbations, such as LIME, can be manipulated.

This insight forms the basis of our proposed approach. By effectively distinguishing between data points originating from the input distribution and instances generated through perturbation, an adversary can construct an adversarial classifier (referred to as scaffolding) that mimics the behavior of the original classifier, potentially displaying extreme discriminatory tendencies on the input data points. However, this adversarial classifier behaves arbitrarily differently, appearing unbiased and fair, on the perturbed instances. Consequently, it effectively deceives techniques like LIME or SHAP into producing innocuous explanations. Subsequently, we formalize this intuition and elucidate our framework for constructing adversarial classifiers capable of misleading explanation techniques.

Let's denote the biased classifier described earlier as f. The adversary aims to develop a framework wherein post hoc techniques can be deceived into perceiving f as innocuous when generating explanations for this black box model.

Recall that the real-world data, upon which this classifier is likely to be applied, follows a distribution X_{dist}, and X represents a set of N data points sampled from this distribution, which the adversary has access to. To deceive the post hoc techniques, the adversary could devise an adversarial classifier that demonstrates biased behavior on instances sampled from X_{dist} while maintaining an unbiased stance on instances that do not originate from X_{dist}.

Since the feature importances provided by LIME and SHAP heavily rely on perturbed instances (which may typically be Out-Of-Distribution (OOD) samples, the resulting explanations will portray the classifier designed by the adversary as innocuous.

Assuming ψ represents an unbiased classifier (for instance, making predictions based on innocuous features that are uncorrelated with sensitive attributes), the adversarial classifier e takes the following form:

$$e(x) = \begin{cases} f(x), & \text{if } x \in X_{\text{dist}} \\ \psi(x), & \text{otherwise} \end{cases}$$

To construct a classifier, is_OOD, which detects whether a given data point is an out-of-distribution (OOD) sample (i.e., is_OOD(x) = True) or not, we proceed as follows:

1. We create a new dataset from the instances in X by perturbing each instance x to generate a new instance x_p. The details of perturbation are provided in the Experimental Evaluation section.

2. Denote the set of all instances generated via perturbation as X_p.

3. The instances in X are then assigned the class label `False`, indicating that they are not OOD samples. Conversely, the instances in X_p are assigned the class label `True` unless they are already in X.

4. We proceed to train an off-the-shelf classifier on the combined dataset $X \cup X_p$ and their corresponding class labels.

This process enables the creation of a classifier capable of distinguishing between in-distribution and out-of-distribution samples based on the characteristics of the perturbed instances.

9 Experimental Results

Our evaluation process diverges from traditional train/test assessments of model performance. Rather than focusing on conventional metrics, our emphasis is on assessing how effectively the models assign importance values to relevant features. In order to reason on the validity of the obtained results, as mentioned before, to establish benchmarks, we defined "golden standards" based on the importance coefficients directly assigned by the scikit-learn models.
For the sake of simplicity, in this report we will only focus on the explainability results obtained on the *Decision Tree* models and *Neural Networks*, however this can be repeated for any model. The complete list of results is attached on the results folder [7].

9.1 Regression

In Figures 6 and 7 two of the SHAP results are presented. SHAP is particularly well suited for global analysis of examples because it applies an analysis on all the input data.

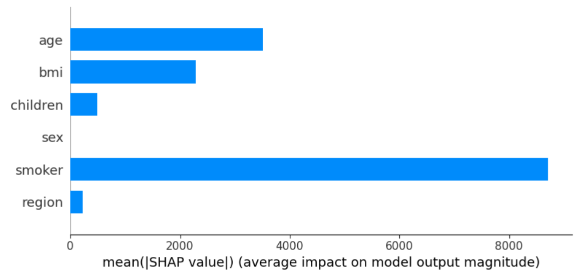


Figure 6: Bar Plot of of global SHAP Results

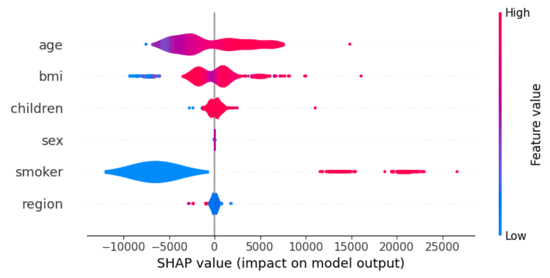


Figure 7: Violin Plot of global SHAP Results

In Figure 6, the mean absolute impact of each feature across all examples is portrayed, offering a comprehensive perspective on the average contribution of each feature to the model's predictions across the entire dataset. This representation allows us to discern the overall significance of individual features, providing valuable insights into the

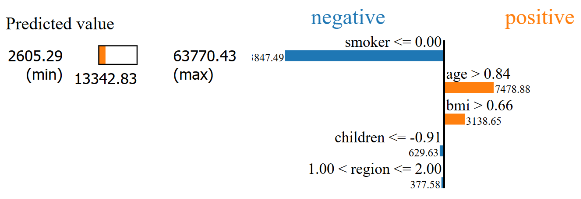


Figure 8: Lime explanation for instance 1

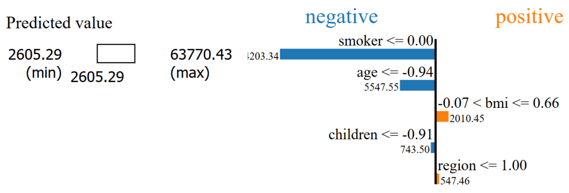


Figure 9: Lime explanation for instance 2

consistent patterns of influence exhibited by different input variables. The sorting of features based on their absolute magnitude of impact facilitates a clear identification of the most influential factors driving the model's outputs. Figure 7 provides a deeper exploration through the distribution of each feature's influence on the dependent variable. This visualization captures the diversity in the impact of individual features across the dataset, revealing how their contributions vary. The sorting mechanism based on absolute magnitude not only highlights the most impactful features but also facilitates a dynamic understanding of their collective effect on the model's predictions.

As already mentioned SHAP is a powerful tool to obtain a comprehensive and lucid overview of the model's global behavior. It excels at highlighting features that, on average, exert the highest impact on the target variable, providing also insightful details on how each variable contributes to the model's predictions.

LIME and DICE, on the other hand, are designed to provide local, instance-specific explanations for model predictions, so they are more suited for the understanding of the decision-making process for a single data point rather than the overall behavior of the model. For this reason, our analysis for what concerns DICE and LIME focuses on the results provided for the single examples.

In Figures 8 and 9, two singular LIME explanations are presented, offering insights into model predictions by highlighting the top 5 most important features. Alongside this, LIME provides a threshold or range for each of these features. This threshold indicates whether a feature's value, when surpassing or falling within the specified limits, has a positive or negative impact on the predicted output.
As mentioned earlier, feature importance with DICE is measured here through the absolute and relative variation of the feature that leads to a change in the output. Specifically, the absolute variation allows us to observe the frequency with which the feature has been chosen, while the relative variation deals with the magnitude of the change: the lower it is, the more relevant the feature might be within the model. The DICE results are collected in 4. The results have been generated by using 3 counterfactuals per sample.

Feature	Relative var	Absolute var
age	0.05	3403
bmi	0.10	5137
children	0.10	4818
sex	0.31	630
smoker	0.07	639
region	0.34	1487

Table 4: DICE Counterfactual relative and absolute variations for each feature.

9.2 Classification

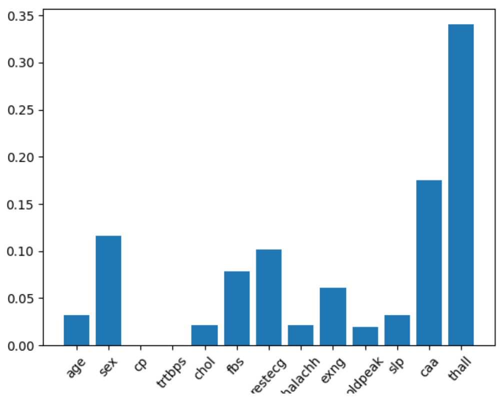


Figure 10: Gold standard Decision Tree

Observing the model coefficients, we can assume that the *thall* feature is the most influential in the model. The LIME results reported in Figures 11 and 12 confirm this hypothesis.

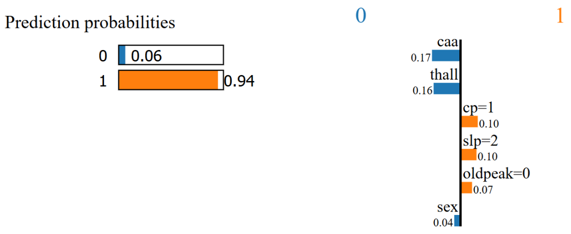


Figure 11: Lime explanation for instance 1

Figures 13 and 14 show the SHAP results. Comparing the results obtained with those of LIME, we can notice some differences. SHAP identifies as

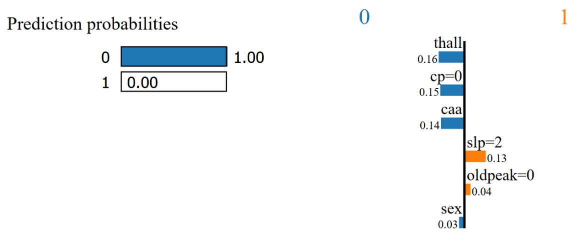


Figure 12: Lime explanation for instance 2

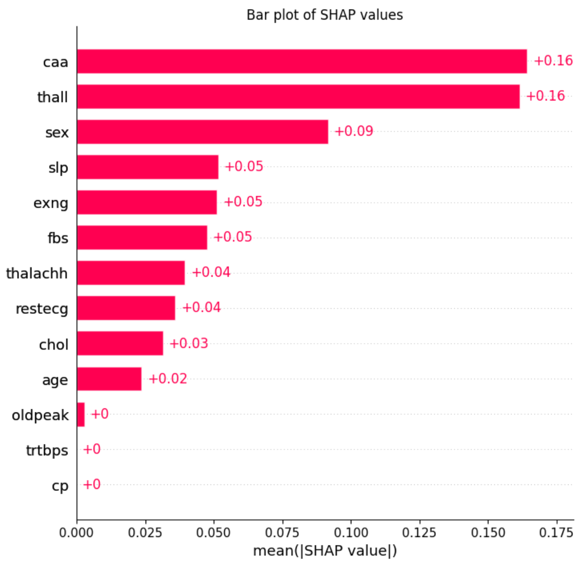


Figure 13: Bar Plot of global SHAP Results

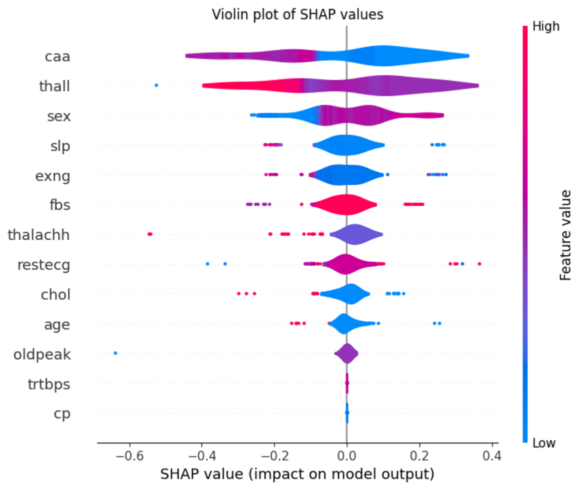


Figure 14: Violin Plot of global SHAP Results

Feature	Relative var	Absolute var
age	1.78	14.22
sex	1.00	4.00
cp	2.00	10.000
trtbps	2.85	39.86
chol	2.03	10.14
fbs	-	0.00
restecg	1.00	1.00
thalachh	2.14	6.41
exng	1.00	3.00
oldpeak	1.34	9.40
slp	1.60	8.00
caa	2.08	29.17
thall	2.23	51.39

Table 5: DICE Counterfactual relative and absolute variations for each feature.

the most important features, as for LIME, '*thall*' and '*caa*'. Shap is better aligned with the golden standard results (for instance, '*ca*' seems to be irrelevant, in opposition to the LIME results).

The results of DiCE in Table [**tab:dice˙cla**] remain the most difficult to interpret. They appear to be misaligned with the results of previous approaches, primarily due to the presence of categorical features, whose contribution is challenging to compare with that of numerical features when using the implemented metrics.

9.3 DeepLIFT

For DeepLIFT we see results dissimilar in classification from what we have seen before, although differences may be caused by the different model used, due to versioning constraints). However, it seems that it works effectively for classification tasks, seeing that the most relevant features in Figure 15 and Figure 16 are all medically or statistically relevant to heart issues, thus more likely to be picked up by a Neural Network.

With regards to regression, we see instead similar results to those of the other tasks, with some exceptions (the bmi and sex features), but these too can be attributed to model differences.

9.4 Adversarial Results

Our evaluation process involves comparing the feature importance assigned by unaffected models with that of manipulated ones. Rather than relying solely on traditional metrics, our focus is on gauging how effectively models attribute importance values to pertinent features. Specifically, the Linear Regressor coefficients, LIME, SHAP, and DiCE are used as indicative of fair feature importance. These established benchmarks serve as a baseline for evaluating the efficacy of the adversarial methodologies employed.

For the regression, our experiments include Location Fooling, where we attempted to emphasize the 'region' feature and to reduce 'smoker' and 'age', which are respectively the most and least influents, and Active Fooling, where we exchanged the relevance of 'age' and 'region', one being the second most significant feature and the other among the least relevant.
For classification, the goal is to emphasize the

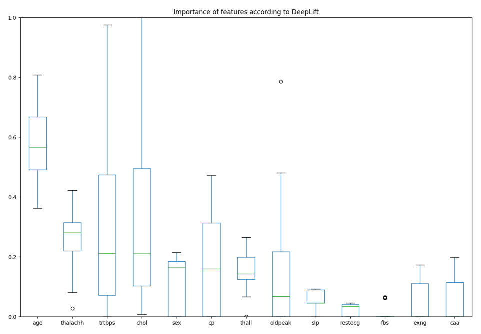


Figure 15: Box Plot of DeepLIFT Results

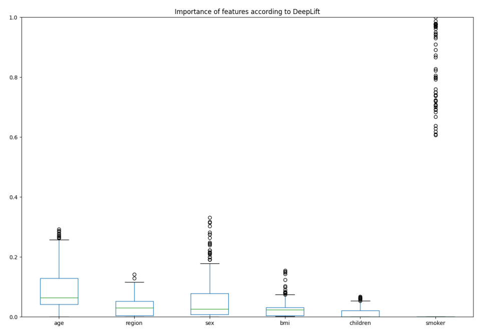


Figure 17: Box Plot of DeepLIFT Results

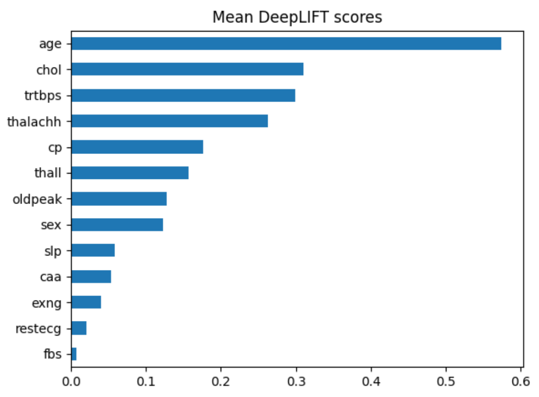


Figure 16: Bar Plot of DeepLIFT Results

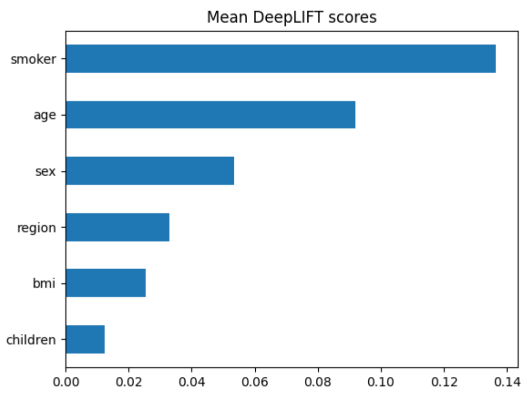


Figure 18: Bar Plot of DeepLIFT Results

'trtbps' feature and reduce the 'cp' feature using Location Fooling, while reversing the importance values of 'cp' and 'sex'.

Regarding Location Fooling, we observe that the goal of changing the model's importance has been achieved: '*region*' now has a coefficient of 913 (compared to the initial 392), while '*smoker*' and '*age*' have decreased from 14476 and 3645 to 11560 and 605, respectively. Although changes in importance were observed, it was not possible to excessively modify the parameters to maintain the model's performance similar to the original one. The Mean Absolute Percentage Error (MAPE) between the fair model and the lf-model remains similar, with an increase in error contained to 4%. The Active Fooling altered the coefficients of all the features. In particular, '*age*' was reduced from 3645 to 14, with an increase in the percentage error of about 4%.

The fooling for classification met expectations without even losing precision in the model. In fact, the f1-score increased by 0.03. It is noticeable that, especially when using Active Fooling, many coefficients that were not explicitly modified changed significantly. This could be a consequence of the model's adaptation to the new constraints on certain features.

Finally, to verify that these methods can be used to deceive traditional explainability methods, the ensembled model previously illustrated is trained:

$$e(x) = \begin{cases} f(x), & \text{if } x \in X_{\text{dist}} \\ \psi(x), & \text{otherwise} \end{cases}$$

To this end, DiCE is used to generate counterfactuals of the original dataset, applying Data Augmentation to the initial data. The new samples will be labeled as ***Out Of Distribution***. A Random Forest Classifier is trained to verify whether a sample belongs to the original data distribution. In this case, the output will be generated by the original model, the linear model previously illustrated. Otherwise, the model used will be the one trained through a custom active loss that attempts to swap the importance values of the features. The ensembled model is then evaluated through LIME. As shown, LIME operates by generating points in the neighborhood of a local point where explainability is to be performed.

Finally, we tested the adversarial model $e(x)$. For each sample in the dataset, one new counterfactuals is generated using DiCE. These newly generated samples are labeled as $OOD = True$, while

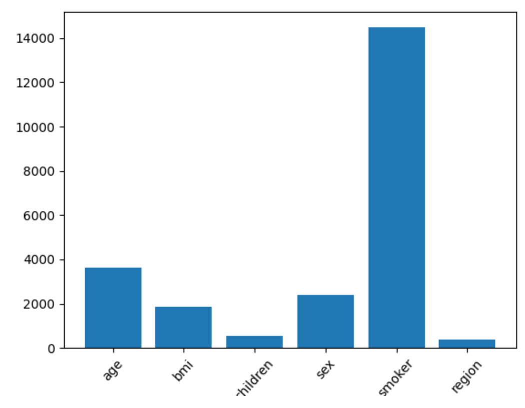


Figure 19: Feature importance of the linear regressor fair model.

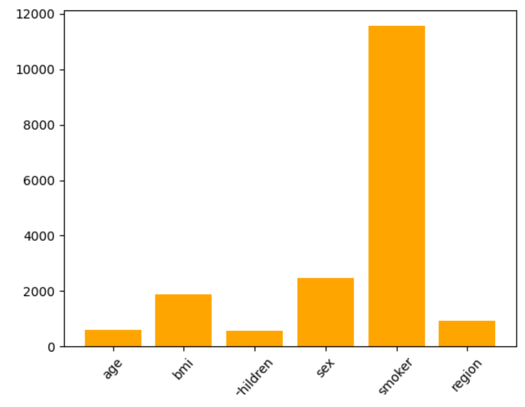


Figure 20: Feature importance after Location Fooling

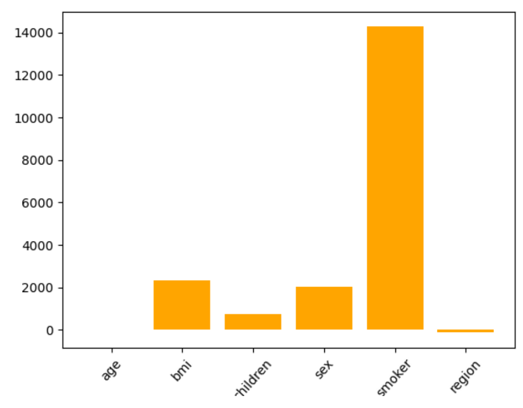


Figure 21: Feature importance after Active Fooling

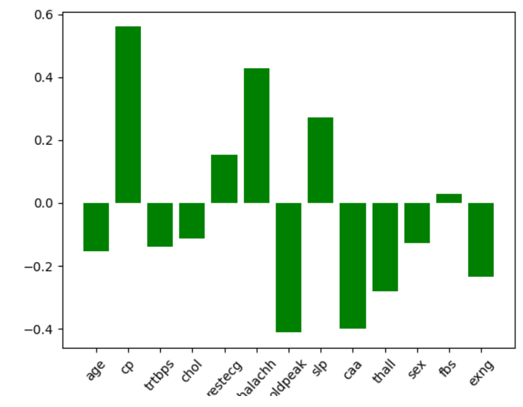


Figure 22: Feature importance of the linear classifier fair model.

the original samples are labeled as $OOD = False$. Subsequently, a Random Forest classifier consisting of 100 Decision Trees is trained to distinguish between Out-of-Distribution (OOD) samples and in-distribution samples. Additionally, two linear regressors, $f(x)$ and $\psi(x)$, are trained on original and OOD samples, respectively.

Specifically, $f(x)$ is trained by incorporating the penalty loss function, presented in the previous part:

$$L_F^I(D, w) = \frac{1}{n_c} \sum_{i \in C} ||h_i^I(w) - 1||^2 \qquad (8)$$

The set C represents the features that are mandated to be relevant.

The LIME results in Figure 27 indicate that the feature importance computed by this algorithm aligns better with the feature importance of $\psi(x)$, even when the original data are evaluated by $f(x)$. Notably, the most relevant features order identified by LIME (*Smoker*, *Sex*, *Age*, *Region* and *BMI*) corresponds to the order of $\psi(x)$. In contrast, $f(x)$ yields markedly different results, as *Smoker*, *Sex*, *BMI* and *Region*.

The same procedure was applied to the classification task using a custom Linear Classifier. The results are stored in the GitHub repository [7], along with those of all the other models.

10 Conclusion

The project focuses on the implementation of different explainability models on various regression and classification models. As already observed, the implemented explainability methods are capable of providing both local and global explanations regarding the importance of input features, consistent with each other and with the degree of importance offered by the model itself. In particular, we can analyze the advantages and limitations of LIME, DiCE, SHAP, and DeepLIFT:

- All the implemented methods are limited to providing a **local** explanation of the model. However, SHAP and DeepLIFT can offer a global perspective by aggregating individual results.

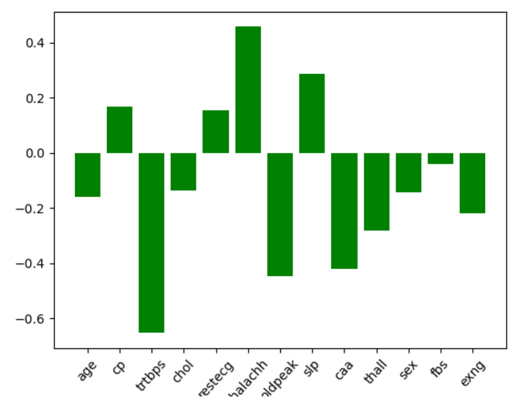


Figure 23: Feature importance after Location Fooling

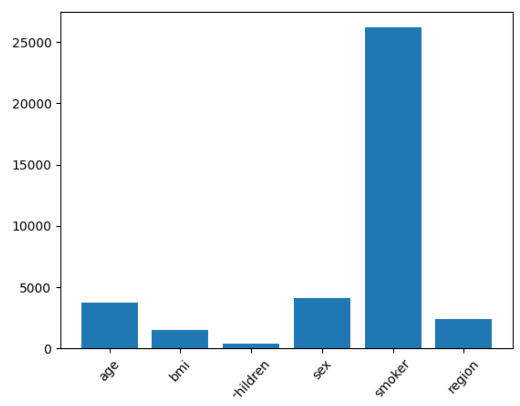


Figure 25: Feature importance of $\psi(x)$

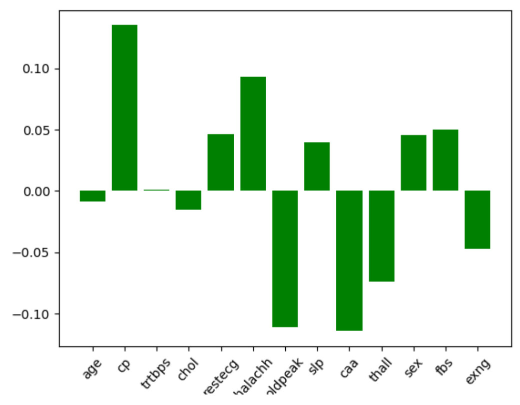


Figure 24: Feature importance after Active Fooling

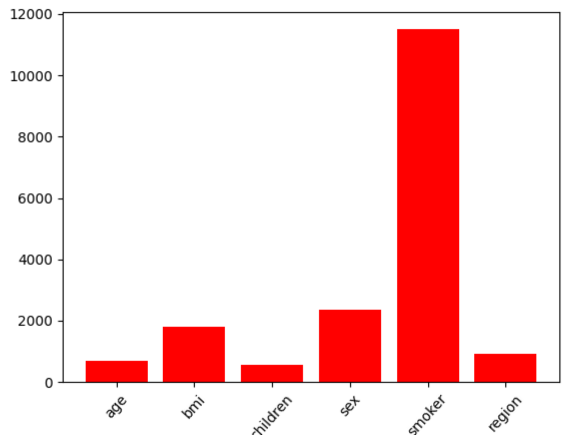


Figure 26: Feature importance of f(x)

- Although the performance and type of model are secondary to the purpose of the project, it is important to note that they influence the results on feature importance. For instance, as shown, the Decision Tree model and the Linear Classifier achieve different performances due to their distinct mechanisms for classification. This impacts the Explainability results, as features relevant for one model (e.g., *cp* for the Linear model) may be negligible for another.

- The evaluation of the model's intrinsic feature importance (defined in previous sections as the *Golden Standard*) offers native interpretability but may not be powerful enough to capture complex patterns. Conversely, the Post-Hoc methods analyzed provide interpretive flexibility even for complex models (e.g., neural networks, random forests) but sometimes sacrifice fidelity to the model.

11 Issues

During the implementation of the code hereby described, we encountered some issues. Here, we wanted to share them to avoid anyone else stumbling upon them in the future. The first and most important issue we faced was that while we were able to implement each and every explainability algorithm we planned to use on our models, we weren't able to implement any of the fooling techniques on the Neural Network model. This is due to Keras and DeepLIFT: Being DeepLIFT part of our evaluation techniques, we needed Keras models to be able to use the technique's library. However, being Keras restricted in how you can manage the training and/or the model, we could not access its entire feature set during training. Thus, we could not generate the heatmaps required by both passive and active fooling losses. Such a thing could be easy to work around using another library that allows for lower-level control on the model or the train loop (such as Tensorflow, PyTorch, and similar). The second issue we faced was with DiCE. Although the algorithm technically works

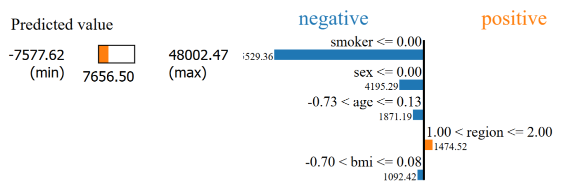


Figure 27: Feature importance detected by LIME

with Categorical results, it is unsuitable when we have both categorical features and a binary classification task. That is because the importance of features is evaluated by a measure we call the Relative count, which measures the ratio between the changes in output and a feature's value (only when both that feature and the output have changed). That measure however will always be 1.0 for categorical features in a single-class task, making it unreliable.

References

[1] Pavan Rajkumar Magesh, Richard Delwin Myloth, and Rijo Jackson Tom. “An explainable machine learning model for early detection of Parkinson’s disease using LIME on DaTSCAN imagery”. In: *Computers in Biology and Medicine* 126 (2020), p. 104041.

[2] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. “Learning important features through propagating activation differences”. In: *International conference on machine learning*. PMlR. 2017, pp. 3145–3153.

[3] Juyeon Heo, Sunghwan Joo, and Taesup Moon. “Fooling neural network interpretations via adversarial model manipulation”. In: *Advances in neural information processing systems* 32 (2019).

[4] Harsh Singh. *Medical Insurance Payout*. 2022. URL: https://www.kaggle.com/datasets/harshsingh2209/medical-insurance-payout.

[5] Rashik Rahman. *Heart Attack Analysis and Prediction Dataset*. 2020. URL: https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset.

[6] Dylan Slack et al. “Fooling lime and shap: Adversarial attacks on post hoc explanation methods”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, pp. 180–186.

[7] Angelo Terzi Palli Nicola. *Analysis of Explainability ML and DL Techinques against Adversarial Models*. 2024. URL: https://github.com/NicolaP00/Ethics.