

Enhancing Local Explanations with GAN-Based Neighborhood Generation

Luca Corbucci(✉), Francesca Naretto, and Anna Monreale

Department of Computer Science, University of Pisa, Pisa, Italy
luca.corbucci@phd.unipi.it

Abstract. The growing deployment of Machine Learning models has increased the demand for interpretability, leading to the development of various explainability methods. Yet, each method has its strengths and limitations, making it challenging to identify a one-size-fits-all solution. This paper introduces `FIRE360`, a multi-faceted and fast local explanation approach for tabular data. `FIRE360` incorporates various explanation types (feature importances, rules, counterfactuals, exemplars, and counter-exemplars), offering a 360-degree model interpretability. `FIRE360` also includes a dashboard with indicators for prediction reliability and explanation quality, such as fidelity and robustness. To ensure efficiency, `FIRE360` uses GAN-generated synthetic data to approximate the original dataset and selects similar records to the target instance, avoiding costly on-demand neighborhood generation. Multiple local surrogate models are trained to capture different aspects of the data and black-box behavior. `FIRE360` achieves higher fidelity than state-of-the-art methods, with improvements up to 95% on multiclass datasets.

Keywords: Explainable AI · Tabular Data · Machine Learning

1 Introduction

In recent years, the rapid advancement and widespread adoption of Machine Learning (ML) techniques across various fields have raised significant ethical concerns. To regulate ML deployment, several legal frameworks [34] have emerged, addressing fairness [22], privacy [15,24], explainability [4,8], and their interplay [6,7,17,23]. The need for explainability has led to the emergence of Explainable AI (XAI), which seeks to enhance the interpretability of black-box (`BB`) models [4], fostering transparency, trust as well as regulatory compliance.

Several post-hoc explanation methods have been introduced, including `Shap` [21], `Lime` [28], and `Lore` [16]. These differ in mechanisms and outputs: `Shap` and `Lime` provide feature importance, while `Lore` produces rules. Hence, each method offers only one type of explanation at a time.

User studies show no consensus on which explanation type is most interpretable; user preferences often follow patterns depending on expertise and context. For instance, non-expert users tend to favor instance-based explanations,

© The Author(s) 2025
S. Džeroski et al. (Eds.): DS 2025, LNAI 16090, pp. 316–331, 2025.
https://doi.org/10.1007/978-3-032-05461-6_21

while expert users are more likely to prefer global methods such as feature importance [13]. These preferences can also be influenced by the complexity of the model and the decision-making context [29], leading practitioners to combine multiple explanation methods to meet diverse user needs. However, these methods are computationally expensive, as they require generating synthetic data and training surrogate models for each instance to explain. Others lack reliability in key state-of-the-art metrics, such as fidelity and stability, leading to inconsistent outputs when generating explanations for the same instance multiple times. Providing explanations with such limitations can undermine user trust and raise doubts on the reliability of the system. Furthermore, these issues raise concerns regarding compliance with emerging AI regulations, which emphasize the need for transparent, trustworthy, and stable explanations to ensure fairness, accountability and user protection.

To address these limitations, we introduce `FIRE360` (**F**eature **I**mportance, **R**ules, and **E**xemplars with **360**-degree analysis), a novel XAI method that leverages synthetic data generation to train various white-box surrogate models, enabling the extraction of five explanation types: feature importance, rules, counterfactual rules, exemplars, and counter-exemplars.

Unlike traditional methods, which generate synthetic data for each instance to explain, `FIRE360` constructs a *single, large synthetic dataset* by leveraging the `BB` and its training data. The resulting dataset aims to resemble the original while being denser in feature space. In addition, it enables efficient *similarity search* to retrieve the closest instances and train *local white-box surrogate* models, reducing computational overhead while improving explanation coverage and reliability.

Our main contribution is a new multi-faceted XAI method for tabular data, providing five types of explanations per instance within a unified framework. A key strength of our approach lies in our efficient pipeline design: synthetic data is generated once at the beginning of the process, avoiding the computational overhead of per-instance generation. We validate `FIRE360` on six datasets across four metrics, including `Fidelity`, `Stability`, and `Robustness`, showing improved fidelity over popular explanation techniques, such as `Shap` [21] and `Lime` [28]. We release `FIRE360` source code to ensure reproducibility and enable practitioners to integrate our method into their pipelines[1]. Moreover, we publish a web-based user interface[2], designed to enhance the usability of `FIRE360` by making it easier to interact and visualize the explanations compared to existing solutions.

2 Related Work and Background

We review the relevant literature for our methodology, focusing on XAI techniques, particularly local and post-hoc explanations, in Sect. 2.1. Next, in Sect. 2.2, we discuss synthetic data generation approaches, examining state-of-the-art solutions and their application in `FIRE360`.

[1] Github repository of FIRE360.
[2] Dashboard to visualize the explanations obtained with FIRE360.

2.1 Explainable AI

Nowadays, explainability is one of the most important research directions in AI, as it is crucial to achieve Trustworthy Artificial Intelligence. Bodria et al. [4] provide an overview of existing techniques for ML interpretability, identifying two main types of explanation models: *global* and *local*. Local explainers focus on explaining the results of predictions on single instances [16,21,28], while global ones explain the behavior of the whole ML model [11,12,14]. In addition, explanations can be *post-hoc*, meaning they are generated externally after the ML model has been trained, or *by design*, where both the prediction and the explanation are produced during the model's training process. Lastly, XAI methods can be *agnostic*, if they explain any `BB` model, or *ad-hoc*, specifically designed for a particular ML model. This paper proposes `FIRE360`, a method for generating agnostic, local, and post-hoc explanations. In the following, we present the state-of-the-art methods in this context.

Feature Importance Explanations. These methods produce a feature-importance vector, where each value indicates a feature's contribution to the classification. A well-known example is `Lime` [28], a local, model-agnostic method that explains `BB` model predictions by approximating their behavior near a given instance using randomly perturbed samples.

Given a `BB` and an instance x, `Lime` perturbs x by randomly modifying the features to create synthetic instances, $\{z \in \mathbb{R}^d\}$. These synthetic instances are then passed through `BB` to obtain their predictions, $\theta(z)$. The perturbed dataset is used to fit an interpretable model $E(\cdot)$, which serves as a local approximation of θ. The resulting explanation consists of the learned feature importance values, reflecting how much each feature contributes to the prediction. Several `Lime` variants have been proposed to address its limitations [33,37], but they often increase the computational cost of generating explanations. Therefore, in this paper, we use the classic `Lime` implementation for comparison with `FIRE360`.

In this context, there is also `Shap` [21] (SHapley Additive exPlanations), a local and global, post-hoc, model-agnostic approach that computes feature importance using Shapley values, a concept derived from cooperative game theory. `Shap` provides explanations in the form of additive feature attributions, represented by the equation: $E(z') = \rho_0 + \sum_{i=1}^{M} \rho_i z'_i$, where $z' \approx x$ as a real number, $z' \in [0,1]$, $\rho_i \in \mathbb{R}$ represents the contribution assigned to each feature, and M is the number of input features. `Shap` satisfies three key properties: (i) local accuracy, ensuring that $E(x)$ matches $\theta(x)$; (ii) missingness, meaning that features with $x_i = 0$ have no impact on the computed `Shap` values; (iii) stability, ensuring that if a model changes so that the marginal contribution of a feature increases, the corresponding `Shap` value also increases. There are various ways to approximate the computation of `Shap` values using different strategies. Among them, we focus on KernelExplainer, the model-agnostic version of `Shap`, which we employ in this work as a competitor. KernelExplainer approximates Shapley values by sampling perturbations around the instance to explain and estimate feature attributions through a weighted linear regression model.

Rule-Based Explanations. This type of explanation, known as a decision or factual rule, follows the form $p \rightarrow y$, where p is a premise consisting of Boolean conditions on feature values and y is the outcome class. The premise p is a conjunction of conditions of the form $x_i \in [v_i^{(l)}, v_i^{(u)}]$, where x_i is a feature, and $v_i^{(l)}$, $v_i^{(u)}$ are its lower and upper bounds, extended with $\pm\infty$. An instance x satisfies a rule r if all conditions in p hold for x. If the instance to explain satisfies p, the rule $p \rightarrow y$ serves as a candidate explanation for the decision $E(x) = y$.

In this setting, we find `Lore` [16] (LOcal Rule-based Explainer) a local, post-hoc, agnostic explainer that provides explanations in the form of rules and counterfactual rules. `Lore` uses a genetic algorithm to generate the neighborhood of the instance to be explained. Given a `BB` and an instance x, where $\theta(x) = y$, `Lore` creates a synthetic set Z of neighbors exploiting a genetic algorithm. It then trains a decision tree classifier ϕ on this set, labelling it with the `BB` predictions $\theta(Z)$. The explanation extracted from ϕ consists of: (i) a decision rule, i.e. the path in the decision tree followed by x to reach the prediction y, and (ii) a set of counterfactual rules, which lead to a different classification than y.

Prototype-Based Explanations. They are representative records that capture the salient characteristics of a group of objects, usually belonging to the same class. They often act as reference points, allowing users to gain insight into the decision-making process of a `BB` by examining instances similar to them. They are usually considered as local explanations, giving information about similarities and differences w.r.t. the record x to explain. A prototype may be (i) a training sample close to the input x, (ii) the centroid of a cluster, or (iii) a synthetically generated record. Different methods adopt varying definitions and criteria for selecting prototypes [20,32]. As an example, Tree Space Prototype [32] is a local post-hoc explainer for tree ensemble models. The goal is to identify class-wise prototypes using a predefined proximity measure between trees.

Explanation Evaluation. In recent years, increasing attention has been given to evaluating the quality of explanations, leading to the introduction of several validation metrics. Among these, `Fidelity`, `Stability`, and `Robustness` are considered the most important for ensuring reliable explanations [4]. `Fidelity` measures how well the explanation function approximates the behavior of the `BB` θ, with its implementation varying depending on the type of explainer being analyzed. `Stability` and `Robustness`, on the other hand, evaluate the consistency of explanations. `Stability` ensures that the same instance receives the same explanation when analyzed multiple times, meaning that if we explain the same record twice, the results should not change. `Robustness` assesses whether similar instances give similar explanations. It is evaluated by analyzing explanations for points in the vicinity of the instance being explained, ensuring that small variations in input do not lead to significantly different explanations.

Overall, `Lime` is computationally efficient, but its reliance on random sampling makes it unstable, leading to different results across runs. `Shap`, on the other hand, is grounded in strong theoretical foundations, yet its approximation process and the choice of the reference dataset can significantly impact both execution time and explanation quality. For `Lore`, the use of a genetic algorithm

improves stability at the cost of increased computational time, making it less practical for large-scale applications.

To address this, we propose FIRE360, a method that ensures computational efficiency, stability, and fidelity to the BB's predictions. Unlike existing approaches, FIRE360 offers a comprehensive 360-degree explanation by combining feature importance, rules, counterfactuals, exemplars, and counter-exemplars for a complete and interpretable view of the model's decisions.

2.2 Synthetic Data Generation

In recent years, synthetic data generation has gained increasing attention due to the limited availability of real-world data, as well as ethical considerations, such as fairness and privacy. A widely used techniques employs Gaussian Mixture Models (GMMs) [5], which model multivariate distributions based on the original data. This approach is often combined with copula functions to capture dependencies between variables better. Recently, more effective methods than GMMs, have emerged, particularly Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs).

GANs [36] are a generative framework based on adversarial training. This approach involves training two models: a *generator* to learn how to approximate the data distribution and a *discriminator*, which evaluates whether a given sample originates from the real training data or is synthetic. When the discriminator can no longer distinguish between real and synthetic data, it indicates that the generator has successfully learned to produce highly realistic synthetic samples. One of the most widely used GAN-based methods for tabular data is CTGAN[3], which employs a conditional generator to preserve the proportions of different attributes and classes present in the real data. This makes it particularly effective for categorical variables, ensuring that synthetic data maintains the original statistical distributions.

TVAE[4] [18], on the other hand, differs from GAN-based approaches in how it models data distributions. Instead of using adversarial training, TVAE (Tabular Variational Autoencoder) encodes input data into a latent space and reconstructs it from latent representations. In practice, a VAE is a generative model that learns a probabilistic latent representation of data. It consists of an encoder that maps input data to a distribution over latent variables and a decoder that reconstructs data from sampled latent representations. In particular, it enables controlled data generation by modeling the latent space as a continuous distribution. TVAE is suited for handling missing values and sparse data but does not include an explicit mechanism for managing categorical variables.

3 FIRE360 Method

FIRE360 is a local, post-hoc and agnostic explanation method for tabular data which provides users with a *multi-faceted view* on the explanation of a model

[3] CTGAN.
[4] TVAE.

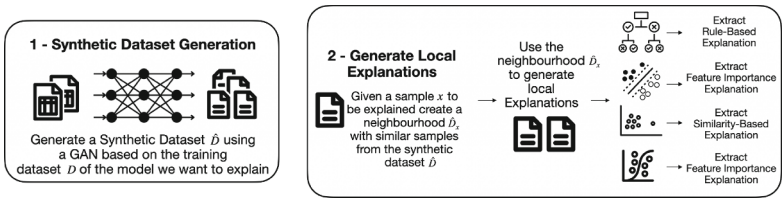


Fig. 1. An overview of our methodology

prediction. It enhances the interpretability by providing explanations in multiple forms: *feature importance*, *decision rules*, *counterfactual rules*, *exemplars* and *counter-exemplars*. By offering diverse perspectives on model predictions, `FIRE360` facilitates a comprehensive understanding of the reasoning behind a given decision. As illustrated in Fig. 1, our approach consists of two main phases: (I) the generation of a synthetic dataset and (II) the use of the synthetic dataset to train multiple white-box surrogate models, which are then used to explain the reasoning behind the prediction of each sample made by the `BB`.

Phase I - Synthetic Data Generation: Given a training dataset D, used to train the `BB` model θ we aim to explain, `FIRE360` generates a synthetic dataset that closely resembles the original data distribution, while ensuring a denser representation of the data space. This allows us to cover a broader range of input variations, enabling the training of white-box models for extracting explanations and analyzing different localities of the feature space more effectively. To achieve our objective, we exploit state-of-the-art synthetic data generation techniques (more information is in Sect. 4). A key aspect of our approach is that the synthetic dataset $\widehat{D}$ serves as the foundation of the explanation pipeline and, differently from the majority of the XAI approaches, is generated only once, ensuring computational efficiency. Once generated, $\widehat{D}$ is labeled using the `BB` θ, obtaining $\widehat{Y} = \theta(\widehat{D})$. This approach ensures that the synthetic data reflects both the characteristics of the original data and the behavior of the `BB`.

Phase II - Explanation via White-Box Models: After generating the synthetic dataset $\widehat{D}$ and labelling it using the `BB` model ($\widehat{Y} = \theta(\widehat{D})$), `FIRE360` proceeds with training the white-box surrogate models, from which explanations will later be extracted. Since our method is local, it does not require the entire dataset $(\widehat{D}, \widehat{Y})$. Instead, to better capture the specific patterns in the vicinity of the instance x to be explained, `FIRE360` focuses only on the closest points to x. In this way, we ensure that the surrogate models learn the most relevant information for local interpretability. The *neighborhood selection* process extracts a small subset of synthetic samples that are close to x. Mathematically, given an instance x to explain, we select $\widehat{D_x} \subseteq \widehat{D}$, where the number of neighbors $|\widehat{D_x}|$ and the notion of proximity, exploited for the selection of the subset, are user-defined. In particular, for the proximity, depending on the specific dataset, it may be a tailored distance metric (e.g., Euclidean, Cosine, or Jaccard distance).

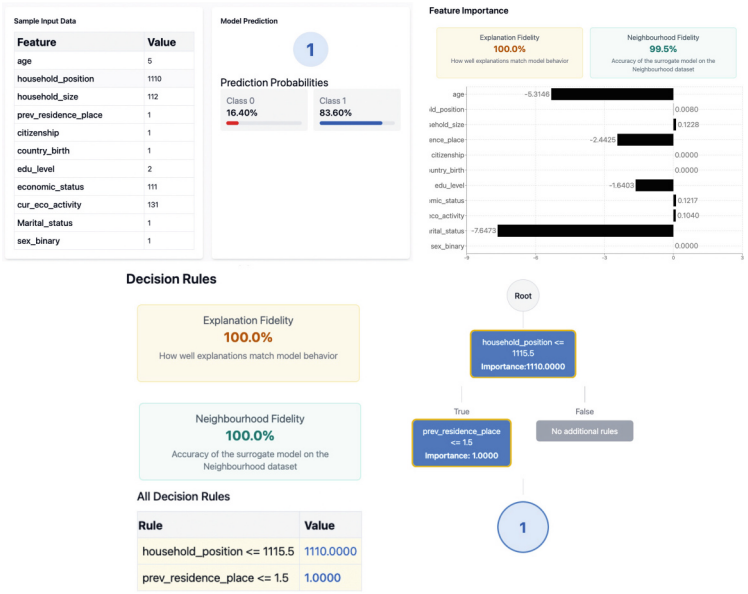


Fig. 2. A screenshot of the dashboard we developed to allow the practitioners to visualize and explore the explanations generated by `FIRE360`.

After the extraction of the local dataset $\widehat{D_x}$, `FIRE360` trains different white-box models for the explanation extraction: (I) a Decision Tree [19] (DT), (II) a Support Vector Machine [9](SVM), (III) a Logistic Regression [10] (LR) and (IV) a K-nearest neighbors [1](KNN). For obtaining robust and reliable explanations, the surrogate models must achieve high predictive performance, particularly when compared to the `BB` model. `FIRE360` first conducts a grid search to identify the optimal set of hyperparameters for each surrogate model. Further details on the hyperparameter tuning process are provided in Sect. 4. Once all surrogate models are trained, we proceed with extracting explanations. Feature importance explanations are derived from the SVM and LR models, while rule-based explanations, either factual or counterfactual, are obtained from the DT. Finally, similarity-based explanations, in the form of exemplars and counter-exemplars, are extracted using the KNN model. A key advantage of `FIRE360` is that it generates the synthetic dataset $\widehat{D}$ once during training, avoiding per-query neighborhood generation and greatly reducing computational overhead. Beyond computational efficiency, `FIRE360` also achieves a higher fidelity compared to existing methods, as its surrogate models more accurately replicate the behavior of the `BB` model, as discussed in the experimental Sect. 2.1. Additionally, it ensures stability, producing consistent explanations across different runs. Finally, by integrating multiple explanation types, `FIRE360` provides a comprehensive and multi-faceted analysis of the `BB` predictions. Our approach simplifies

implementation by enabling multiple explanation types through a single, unified method without the need to manage separate techniques.

Visual Dashboard. A key component of `FIRE360` is its interactive dashboard, an example of which is shown in Fig. 2. The dashboard displays information about the explained record, including its features and the prediction probability assigned by the `BB`. Additionally, it presents multiple types of explanations (e.g., feature importance, rules, exemplars, etc.) with their quality scores, such as `Fidelity` scores. This interface allows end-users to explore the `FIRE360` analysis, offering explanations and supplementary information that enhances transparency and fosters trust in the model's decision-making process.

4 Experimental Setting

To validate and evaluate `FIRE360`, we test it on six diverse tabular datasets[5]: Adult [2], Dutch [35], Covertype [3], House16[6], Letter [30], and Shuttle [31]. These datasets are selected for their distinct characteristics to ensure a comprehensive assessment of our approach. In particular, they include both binary and multi-class classification tasks, numerical and categorical attributes, and vary significantly in both the number of records and of features. This diversity allows us to evaluate the effectiveness and scalability of `FIRE360` across different data distributions and learning scenarios. Table 1 summarizes the key characteristics of the datasets used in our experiments and the metrics computed using the `BB` model on the test dataset. Adult is a binary classification dataset with 48,842 records and 52 features, including both numerical and categorical attributes. Similarly, Dutch is a binary classification dataset with 60,420 samples collected in the Netherlands, where the objective is to predict whether an individual's salary exceeds 50,000$. Covertype is a multiclass dataset with 7 classes and 581,012 records for classifying forest cover types. House16 contains 22,784 samples with a binary target. Letter consists of 20,000 samples and is designed for classifying handwritten English letters into 26 possible categories. Shuttle comprises 43,500 samples with 7 possible targets. For each dataset, we split it into 80% training D_{train} and 20% test sets D_{test}. The training dataset was further divided into training and validation sets again with an 80-20% split for the hyperparameter tuning done before training the models θ. D_{test} is kept separate from the train dataset throughout the `FIRE360` pipeline. In particular, we explain all the samples in D_{test} to evaluate our explanations.

Synthetic Data Generation - Phase 1. To evaluate the effectiveness of the synthetic data generation phase, we test two popular synthesizers: a CTGAN [36] and a TVAE [18] using the implementation provided by the Synthetic Data Vault library (SDV) [25]. Overall, the results are good and in line with our expectations, confirming the ability of both models to generate high-quality synthetic data. For each synthesiser, we test different configurations, varying the

[5] Details about the pre-processing are reported in the FIRE360 Github repository.
[6] House16 Dataset.

Table 1. A description of the datasets used in the experiments. The number of numerical and categorical characteristics listed in the table is given after pre-processing.

Dataset	Samples	Classes	$\lvert D_{test} \rvert$	Num. Feat.	Cat. Feat.	Accuracy	F1 Score
Dutch	60,420	2	12,084	11	0	0.83 ± 0.01	0.83 ± 0.01
Adult	48,842	2	9,769	9	102	0.85 ± 0.01	0.84 ± 0.01
Covertype	581,012	7	20,000	10	44	0.80 ± 0.01	0.79 ± 0.01
House16	22,784	2	4,557	16	0	0.85 ± 0.01	0.85 ± 0.02
Letter	20,000	26	4,000	16	0	0.82 ± 0.02	0.82 ± 0.02
Shuttle	43,500	7	8,700	9	0	0.98 ± 0.01	0.98 ± 0.01

number of generated samples (from 10,000 to 200,000) and training epochs (1000, 2500, and 5000), following the SDV library's hyperparameter tuning guidelines[7]. After generating the synthetic data $\widehat{D}$, we assess its quality using metrics from the Synthcity library [27], before labelling $\widehat{D}$ with the `BB` θ. The results highlight that all of the methods have low values of *Common Row Probability* between real and synthetic datasets, meaning that there is a low risk of leaking real records in the synthetic one. In addition, the high performance of *XGB Performance* metric indicates that training XGB on the synthetic dataset yields strong results in classification tasks. Lastly, we also have good results in terms of privacy metrics: none of the methods has less than $k = 23$ for k-anonymity, and the *Probability of re-identification* is always close to zero. Since the results indicate strong overall performance across all models with no significant differences between settings, we opt for using CTGAN for small datasets, e.g., Shuttle, Letter, and House16, and TVAE for Adult, Dutch, and Covertype to leverage its stable training process, especially when handling tabular data with a huge number of records. We train all datasets for 2,500 epochs, as no further improvements were observed beyond that point. The only exception is Covertype, limited to 1,000 epochs due to its large size and computational cost.

Surrogate Model Training - Phase 2. The second phase of our method involves training four surrogate models, with hyperparameter tuning performed using a grid search for each. As discussed in Sect. 3, `FIRE360` is composed of DT, SVM, LR, and KNN. This choice allows us to extract different types of explanations, ensuring a multi-faceted view by capturing various perspectives on the model's decision-making process. Technically, for the implementation of these models, we exploit the *sklearn* library [26]. Specifically, as DT, we use the *DecisionTreeClassifier* performing hyperparameter tuning with a grid search on criterion, max_depth, min_samples_leaf and class_weight. We also use *LogisticRegression* as LR, with liblinear solver, tuning the penalty, C, and class_weight. For SVM, we use a linear kernel and 1000 as the maximum number of iterations, with a grid search to find the best value for class_weight and C hyperparameters. Lastly, for the KNN, we optimize metric, weights, and n_neighbors.

[7] SDV Hyperparameter tuning.

Formally, we denote with ϕ a surrogate model, and with $\phi(\widehat{D}_x, \theta(\widehat{D}_x))$ its training, where $\widehat{D}_x$ is a subset of the synthetic dataset $\widehat{D}$, created at phase 1 and $\widehat{Y}$ the predictions of the BB θ. In the experiments, $\widehat{D}_x$ is created by finding the $top{-}n$ samples from $\widehat{D}$ similar to the sample x we aim to explain. In our experiments, we used cosine similarity as the similarity metric; however, other metrics, such as Euclidean or Jaccard distance, can also be applied in this phase. The value of n is one of the parameters that can determine the quality of the explanations. To evaluate it, we consider three different values: $n = 1000$, $n = 2500$, and $n = 5000$. In addition, since we want $\widehat{D}_x$ to be as representative as possible of the entire input space, we enforce the presence of at least the 30% of samples with a target value different than the target of the sample x. To validate the quality of the surrogate models generated, we compute the `Fidelity` of the different ϕ to assess how well they replicate the behavior of the BB model. `Fidelity` is defined as the proportion of instances to be explained where the model ϕ's prediction matches that of θ. Mathematically, `Fidelity` is defined as: $\texttt{Fidelity} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[\theta(x_i) = \phi(x_i)]$ where $\mathbb{I}$ is an indicator function, ϕ is the surrogate model and N is the total number of test instances. This means that given a sample x, we get 1 if the prediction of the BB is the same prediction obtained with the surrogate, otherwise, we have 0. In addition, we also evaluate $\texttt{Fidelity}_{neigh}$, which is the fidelity of the surrogate model over the neighborhood of each sample x to be explained, that we define as $\texttt{Fidelity}_{neigh} = \frac{1}{N}\sum_{i=1}^{N} \frac{\sum_{j}^{|\widehat{D}_{x_i}|} \mathbb{I}[\theta(x_j)=\phi(x_j)]}{|\widehat{D}_{x_i}|}$. We also evaluate the quality of the explanations provided by computing two metrics: the `Stability` and the `Robustness` of the explanation. Given a sample x and an explainer E, we generate two explanations e and e' using E, and the `Stability` is computed as the differences between the two explanations in terms of the number of shared features. This metric is dependent on the kind of explanations generated. In the case of a rule-based explanation, we measure the distance between e and e', comparing the premises of the rules. For instance, given $e = [age > 60, house_size > 100]$ and $e' = [age > 30, house_size > 50]$, the `Stability` is 1 while with $e' = [age > 30, education > 3]$ the `Stability` is 0.5, since they share only 1 feature. Our objective is to determine whether the attributes are shared between the two explanations rather than focusing on the exact threshold, which may fluctuate due to minor variations in the composition of the synthetic datasets and is difficult to track. Formally, we define the `Stability` of the rule-based explanations as: $\texttt{Stability}(e, e') = \frac{|e \cap e'|}{|e \cup e'|}$. For the feature importance-based explanations, instead, we focus on the ranking of the explanations and not on the exact values of the feature importance. In particular, we first sort the features based on the absolute value of the feature importance, and then we measure the `Stability` by comparing the order of the features. Mathematically, we define the stability for the feature importance as: $\texttt{Stability}(e, e') = \frac{\sum_{i=1}^{\mathrm{d}} \mathbb{I}(e_i = e_i{}')}{\mathrm{d}}$, where d is the number of features of the dataset. Moreover, we measure the `Robustness` of an explanation. In this case, we evaluate the neighborhood of the record to explain. Hence, given the sample

Table 2. A comparison of `FIRE360` (neighbor size $n = 5000$) with `Lime`, `Lore`, and `Shap` across multiple metrics shows that `FIRE360` consistently achieves the highest fidelity. The "time" column reports the average explanation time (in seconds) over 100 samples.

Dataset	Method	Time (sec) (↓)	Fidelity (↑)	Fidelity_{neigh} (↑)	Stability (↑)	Robustness k=5 (↑)	Robustness k=10 (↑)	Robustness k=20 (↑)
Adult	FIRE360 (DT)	2.14 ± 0.18	**0.90**	0.94 ± 0.03	**0.92 ± 0.15**	**0.61 ± 0.12**	**0.60 ± 0.11**	**0.59 ± 0.09**
Adult	FIRE360 (SVM)	1.11 ± 0.21	0.54	0.51 ± 0.18	0.73 ± 0.34	0.24 ± 0.10	0.24 ± 0.09	0.24 ± 0.09
Adult	FIRE360 (LR)	2.05 ± 0.63	0.90	0.98 ± 0.01	0.47 ± 0.22	0.25 ± 0.09	0.24 ± 0.08	0.24 ± 0.07
Adult	Lime	**0.08 ± 0.01**	**0.90**	0.73 ± 0.02	0.05 ± 0.02	0.05 ± 0.01	0.05 ± 0.01	0.05 ± 0.01
Adult	Shap	2.44 ± 0.06	-	-	0.41 ± 0.19	0.26 ± 0.11	0.25 ± 0.09	0.25 ± 0.09
Adult	Lore (Random)	13.24 ± 0.56	0.65	**1.00 ± 0.00**	0.34 ± 0.22	0.34 ± 0.14	0.34 ± 0.12	0.34 ± 0.11
Adult	Lore (Genetic)	128.50 ± 2.47	0.64	**1.00 ± 0.00**	0.55 ± 0.17	0.25 ± 0.10	0.25 ± 0.08	0.25 ± 0.07
Covertype	FIRE360 (DT)	2.35 ± 0.58	0.84	0.85 ± 0.03	0.94 ± 0.11	**0.62 ± 0.11**	**0.60 ± 0.10**	0.58 ± 0.09
Covertype	FIRE360 (SVM)	1.19 ± 0.60	0.56	0.52 ± 0.09	0.54 ± 0.38	0.16 ± 0.06	0.16 ± 0.05	0.15 ± 0.05
Covertype	FIRE360 (LR)	17.70 ± 9.38	**0.85**	0.86 ± 0.03	**0.94 ± 0.19**	0.24 ± 0.11	0.23 ± 0.09	0.22 ± 0.09
Covertype	Lime	**0.11 ± 0.11**	0.71	0.26 ± 0.11	0.61 ± 0.05	0.60 ± 0.02	0.60 ± 0.02	**0.60 ± 0.01**
Covertype	Shap	18.63 ± 0.44	-	-	0.18 ± 0.06	0.11 ± 0.03	0.10 ± 0.03	0.09 ± 0.02
Covertype	Lore (Random)	5.07 ± 0.40	0.37	**1.00 ± 0.00**	0.42 ± 0.12	0.41 ± 0.07	0.41 ± 0.06	0.41 ± 0.05
Covertype	Lore (Genetic)	35.11 ± 1.13	0.63	**1.00 ± 0.00**	0.74 ± 0.10	0.44 ± 0.06	0.44 ± 0.06	0.44 ± 0.05
Dutch	FIRE360 (DT)	0.93 ± 0.03	**0.99**	0.99 ± 0.01	0.95 ± 0.11	**0.90 ± 0.14**	**0.87 ± 0.14**	**0.84 ± 0.14**
Dutch	FIRE360 (SVM)	0.35 ± 0.03	0.99	0.98 ± 0.02	0.80 ± 0.26	0.73 ± 0.25	0.66 ± 0.23	0.58 ± 0.20
Dutch	FIRE360 (LR)	6.55 ± 6.83	0.99	0.98 ± 0.01	0.82 ± 0.25	0.75 ± 0.23	0.69 ± 0.22	0.61 ± 0.19
Dutch	Lime	**0.09 ± 0.12**	0.89	0.51 ± 0.19	0.47 ± 0.16	0.46 ± 0.10	0.45 ± 0.09	0.44 ± 0.09
Dutch	Shap	1.72 ± 0.07	-	-	**1.00 ± 0.00**	0.833± 0.20	0.78 ± 0.21	0.72 ± 0.21
Dutch	Lore (Random)	2.13 ± 0.13	0.50	**1.00 ± 0.00**	0.71 ± 0.12	0.68 ± 0.07	0.68 ± 0.06	0.68 ± 0.06
Dutch	Lore (Genetic)	34.37 ± 1.57	0.50	**1.00 ± 0.00**	0.81 ± 0.19	0.59 ± 0.12	0.59 ± 0.11	0.59 ± 0.10
House16	FIRE360 (DT)	2.25 ± 0.08	0.91	0.88 ± 0.03	**0.93 ± 0.11**	**0.68 ± 0.08**	**0.68 ± 0.08**	**0.67 ± 0.07**
House16	FIRE360 (SVM)	0.41 ± 0.05	0.67	0.60 ± 0.13	0.52 ± 0.39	0.22 ± 0.11	0.22 ± 0.10	0.21 ± 0.10
House16	FIRE360 (LR)	1.11 ± 0.33	**0.93**	0.91 ± 0.02	0.75 ± 0.20	0.52 ± 0.13	0.51 ± 0.12	0.49 ± 0.11
House16	Lime	**0.06 ± 0.08**	0.87	0.24 ± 0.14	0.58 ± 0.13	0.47 ± 0.06	0.46 ± 0.04	0.46 ± 0.03
House16	Shap	1.64 ± 0.05	-	-	0.75 ± 0.14	0.19 ± 0.06	0.18 ± 0.05	0.17 ± 0.04
House16	Lore (Random)	2.28 ± 0.10	0.57	**1.00 ± 0.00**	0.51 ± 0.14	0.51 ± 0.08	0.51 ± 0.07	0.51 ± 0.06
House16	Lore (Genetic)	36.76 ± 1.01	0.61	**1.00 ± 0.00**	0.77 ± 0.14	0.55 ± 0.10	0.55 ± 0.08	0.55 ± 0.07
Letter	FIRE360 (DT)	1.21 ± 0.08	0.67	0.61 ± 0.08	**0.97 ± 0.07**	0.66 ± 0.10	0.65 ± 0.09	0.64 ± 0.08
Letter	FIRE360 (SVM)	0.87 ± 0.21	**0.82**	0.77 ± 0.06	0.88 ± 0.27	0.14 ± 0.09	0.13 ± 0.06	0.12 ± 0.05
Letter	FIRE360 (LR)	4.40 ± 1.14	0.77	0.71 ± 0.07	0.92 ± 0.18	0.19 ± 0.11	0.17 ± 0.09	0.15 ± 0.07
Letter	Lime	**0.10 ± 0.12**	0.04	0.05 ± 0.02	0.23 ± 0.11	0.18 ± 0.07	0.17 ± 0.06	0.16 ± 0.05
Letter	Shap	2.43 ± 0.12	-	-	0.52 ± 0.15	0.24 ± 0.10	0.22 ± 0.08	0.19 ± 0.07
Letter	Lore (Random)	2.75 ± 0.07	0.04	**1.00 ± 0.00**	0.61 ± 0.10	0.61 ± 0.05	0.60 ± 0.05	0.60 ± 0.04
Letter	Lore (Genetic)	37.56 ± 0.74	0.04	**1.00 ± 0.00**	0.90 ± 0.06	**0.77 ± 0.04**	**0.77 ± 0.04**	**0.77 ± 0.03**
Shuttle	FIRE360 (DT)	1.18 ± 0.40	0.99	0.98 ± 0.01	0.92 ± 0.15	**0.76 ± 0.15**	0.73 ± 0.13	0.72 ± 0.13
Shuttle	FIRE360 (SVM)	0.47 ± 0.40	**0.99**	0.99 ± 0.05	0.80 ± 0.28	0.43 ± 0.15	0.42 ± 0.14	0.40 ± 0.13
Shuttle	FIRE360 (LR)	1.99 ± 1.83	0.99	0.98 ± 0.01	0.84 ± 0.25	0.50 ± 0.18	0.48 ± 0.16	0.46 ± 0.15
Shuttle	Lime	**0.07 ± 0.09**	0.80	0.07 ± 0.04	0.42 ± 0.19	0.33 ± 0.14	0.32 ± 0.13	0.31 ± 0.12
Shuttle	Shap	1.09 ± 0.05	-	-	**1.00 ± 0.01**	0.69 ± 0.17	0.66 ± 0.15	0.63 ± 0.14
Shuttle	Lore (Random)	1.82 ± 0.18	0.66	**1.00 ± 0.00**	0.71 ± 0.11	0.70 ± 0.07	0.70 ± 0.06	0.70 ± 0.06
Shuttle	Lore (Genetic)	34.70 ± 1.28	0.67	**1.00 ± 0.00**	0.87 ± 0.11	0.75 ± 0.08	**0.75 ± 0.06**	**0.75 ± 0.06**

x and the corresponding explanation e, we find its neighborhood composed of the top-k similar instances from the test dataset. Then, we compare the explanation e with the explanations computed for the samples in the neighborhood, using the same formula of the `Stability`. We test $k \in [3, 5, 8, 10, 20]$.

5 Experimental Results

To validate the goodness of our results, we compare our solution with three popular explainers: `Lime`, `Shap`, and `Lore` (using both a Random and a Genetic neighborhood generation)[8].

In Table 2, we report the results obtained with `FIRE360` and the competitors. For a fair comparison, we use 5,000 samples for training the surrogate models used by `FIRE360`, `Lime`, and `Lore`. For `Shap`, which has no surrogate model, we apply k-means to obtain the reference points. As suggested by the authors, we use 100 centroids for all datasets, except Letter (200 centroids). In the table, we

[8] Lime source code, Shap source code and Lore source code.

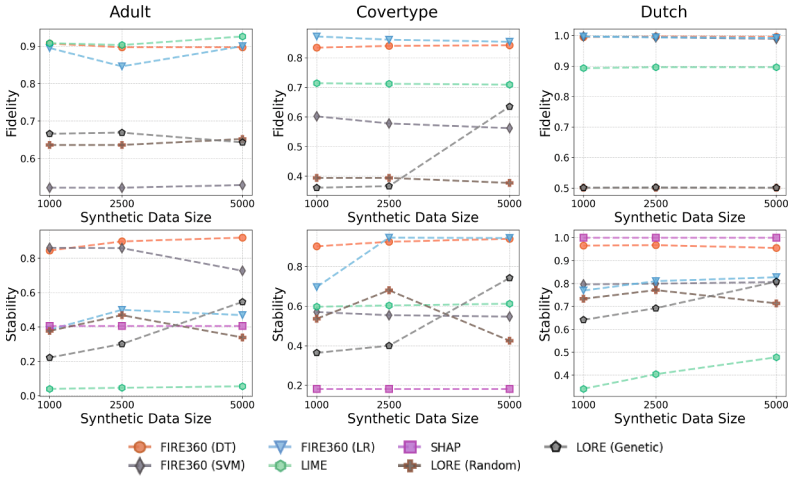


Fig. 3. Ablation study on the number of synthetic records exploited for the explanation generation in the case of Adult, Covertype, and Dutch Datasets, with 2 classes.

report the results on `Fidelity`, $\texttt{Fidelity}_{neigh}$, `Stability` and `Robustness` for `FIRE360` with DT, LR, and SVM, leaving out KNN since, for construction, it does not allow us to measure them. Due to space constraints, we report in the table only the `Robustness` results obtained with $k = 5$, $k = 10$, and $k = 20$; however, we found coherent results with other values of k. From the results reported in Table 2, it is clear that `FIRE360` provides the explanations with the highest `Fidelity`, especially when considering DT and LR models. More specifically, `FIRE360` is stable, particularly when it has to deal with multiclass datasets. Indeed, with Letter `FIRE360` presents a stability that is 95% higher than the competitors, while with Covertype and Shuttle, we observe an improvement of 16% and 19%. Overall, on average, the `Fidelity` obtained with `FIRE360` is 24% higher than the competitors. In addition, even for $\texttt{Fidelity}_{neigh}$, `FIRE360` shows good performance, in line with the ones of `Fidelity`. Regarding the competitors, `Lore` has the lowest fidelity for all the datasets except for $\texttt{Fidelity}_{neigh}$, which is always 1 and hence the highest. `Lime` presents mixed outcomes: it is always better than `Lore`, but it does not reach our `Fidelity` values. High values for `Fidelity` and $\texttt{Fidelity}_{neigh}$ are interesting results since they mean that `FIRE360`'s surrogate models closely mimic the behavior of the `BB`. In terms of `Stability`, i.e., the similarity of two explanations generated for the same record, `FIRE360` usually has the best results, especially using DT. This highlights that our method provides stable explanations and is not affected by randomness. Other competitors, such as `Lore` and `Shap`, present results lower than ours but still acceptable, while `Lime` has the lowest values. The same reasoning applies for `Robustness`, where we evaluate the stability on the k closest records in the test set. The table shows our method yields consistent explanations for sample x as k increases.

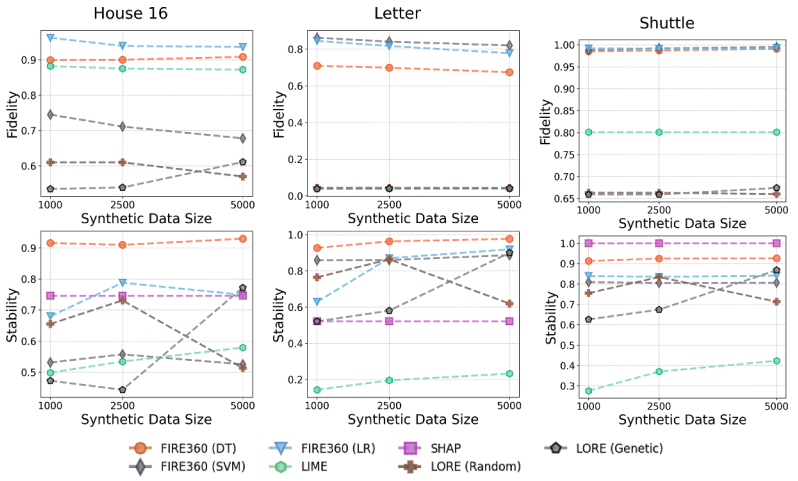


Fig. 4. Ablation study on the number of synthetic records exploited for the explanation generation in the case of House 16, Letter, and Shuttle Datasets, with 2 classes.

We also consider the computational time measured in seconds, reported in Table 2. Depending on the different surrogate models we train, the timing may differ, but overall the results are comparable with those of `Shap`. `Lime`, instead, is always the fastest, even if it is the worst when compared to the other metrics. However, obtaining the same variety of explanations as `FIRE360`, requires running multiple competing methods. Specifically, at least one of `Shap` or `Lime` is needed for the importance of the feature, while `Lore` is necessary to generate rules. This highlights a key advantage of `FIRE360`: it offers multiple explanation types, simplifies implementation, and reduces the need to integrate and manage various separate methods. We also performed an ablation study on the number of synthetic records in $\widehat{D}_x$, evaluating how the quality of the explanations varies as the number of synthetic neighbors for each sample to be explained increases. Results are shown in Figs. 3 and 4. `FIRE360` maintains strong performance regardless of the number of synthetic records. In contrast, other methods perform worse and less consistently, more synthetic data helps some (e.g., `Lore` Genetic) but has little effect on others, especially `Lime`.

6 Conclusion

In this paper, we presented `FIRE360`, a local, post-hoc, and model-agnostic explanation method tailored for tabular data. Its key strength lies in offering diverse explanation forms (*feature importance*, *decision rules*, *counterfactual rules*, *exemplars*, and *counter-exemplars*), to enhance human understanding and streamline explainable AI pipelines without relying on multiple tools.

Experiments show that `FIRE360` outperforms state-of-the-art methods in fidelity, by up to 95% on Letter and 16âĂŞ19% on Covertype and Shuttle.

Beyond fidelity, `FIRE360` offers good performance in terms of stability and robustness while maintaining competitive explanation times, thanks to GAN-based neighborhood generation used to learn white-box surrogate models. Future work includes extending `FIRE360` to image and time series data and conducting user studies to tailor the dashboard and recommend the most suitable explanations.

Acknowledgements. This research was partially supported by the European Union Horizon 2020 program under grant agreement No. 101120763 (TANGO) and PNRR-"SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics" - Prot. IR0000013;. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HaDEA). Neither the European Union nor the granting authority can be held responsible for them.

References

1. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. (1992)
2. Becker, B., et al.: Adult. UCI Machine Learning Repository (1996)
3. Blackard, J.: Covertype. UCI Machine Learning Repository (1998)
4. Bodria, F., et al.: Benchmarking and survey of explanation methods for black box models. Data Min. Knowl. Disc. (2023)
5. Chokwitthaya, C., et al.: Applying the Gaussian mixture model to generate large synthetic data from a small data set. In: Construction Research Congress (2020)
6. Corbucci, L., et al.: Enhancing privacy and utility in federated learning: a hybrid p2p and server-based approach with differential privacy protection. In: SECRYPT (2024)
7. Corbucci, L., et al.: PUFFLE: balancing privacy, utility, and fairness in federated learning. In: 27th ECAI. IOS Press (2024)
8. Corbucci, L., et al.: Semantic enrichment of explanations of AI models for healthcare. In: International Conference on Discovery Science (2023)
9. Cortes, C., et al.: Support-vector networks. Mach. Learn. (1995)
10. Cox, D.R.: The regression analysis of binary sequences. J. Roy. Stat. Soc.: Ser. B (Methodol.) (1958)
11. Craven, M., et al.: Extracting tree-structured representations of trained networks. In: NIPS (1996)
12. Craven, M.W., et al.: Using sampling and queries to extract rules from trained neural networks. In: JMLR. Elsevier (1994)
13. Darias, J.M., et al.: An empirical analysis of user preferences regarding XAI metrics. In: Case-Based Reasoning Research and Development (2024)
14. Deng, H.: Interpreting tree ensembles with inTrees. Int. J. Data Sci. Anal. (2019)
15. Dwork, C., et al.: Fairness through awareness. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (2012)
16. Guidotti, R., et al.: Stable and actionable explanations of black-box models through factual and counterfactual rules. Data Min. Knowl. Disc. (2024)
17. Haffar, R., et al.: GLOR-FLEX: local to global rule-based EXplanations for federated learning. In: 2024 FUZZ-IEEE (2024)

18. Ishfaq, H., et al.: TVAE: triplet-based variational autoencoder using metric learning. arXiv:1802.04403 (2018)
19. Kass, G.V.: An exploratory technique for investigating large quantities of categorical data. J. Roy. Stat. Soc. (1980)
20. Kim, B., et al.: Examples are not enough, learn to criticize! Criticism for interpretability. In: NeurIPS (2016)
21. Lundberg, S.M., et al.: A unified approach to interpreting model predictions. In: NIPS (2017)
22. Mehrabi, N., et al.: A survey on bias and fairness in machine learning. In: ACM Computing Surveys (CSUR) (2021)
23. Naretto, F., et al.: Evaluating the privacy exposure of interpretable global explainers. In: IEEE CogMI (2022). https://doi.org/10.1109/CogMI56440.2022.00012
24. Naretto, F., et al.: Predicting and explaining privacy risk exposure in mobility data. Discov. Sci. (2020)
25. Patki, N., et al.: The Synthetic data vault. In: IEEE International Conference on Data Science and Advanced Analytics (DSAA) (2016)
26. Pedregosa, F., et al.: Scikit-learn: machine Learning in Python. J. Mach. Learn. Res. (2011)
27. Qian, Z., et al.: Synthcity: facilitating innovative use cases of synthetic data in different data modalities (2023). arXiv:2301.07573
28. Ribeiro, M.T., et al.: "Why should i trust you?": explaining the predictions of any classifier. In: ACM SIGKDD (2016)
29. Rong, Y., et al.: Towards human-centered explainable AI: a survey of user studies for model explanations. IEEE Trans. Pattern Anal. Mach. Intell. (2024)
30. Slate, D.: Letter recognition. UCI Machine Learning Repository (1991)
31. Statlog (Shuttle). UCI Machine Learning Repository
32. Tan, S., et al.: Tree space prototypes: another look at making tree ensembles interpretable. In: Foundations of Data Science Conference (2020)
33. Tan, Z., et al.: GLIME: general, stable and local LIME explanation. In: Advances in Neural Information Processing Systems (2023)
34. UE. The Artificial Intelligence Act (2023). https://artificialintelligenceact.eu/the-act/
35. Van der Laan, P.: The 2001 census in the Netherlands: integration of registers and surveys. In: Conference at the Cathie Marsh Centre (2001)
36. Xu, L., et al.: Modeling tabular data using conditional GAN. In: Advances in Neural Information Processing Systems (2019)
37. Zhou, Z., et al.: S-LIME: stabilized-lime for model explanation. In: Proceedings of the 27th ACM SIGKDD (2021)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

