Engineering Applications of Artificial Intelligence 155 (2025) 110664

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Towards Explainable Artificial Intelligence in Machine Learning: A study on efficient Perturbation-Based Explanations

Ismael Gómez-Talal [a,b,*], Mana Azizsoltani [d,e,f], Luis Bote-Curiel [a], José Luis Rojo-Álvarez [a,c], Ashok Singh [d]

[a] Department of Signal Theory and Communications and Telematic Systems and Computation, Rey Juan Carlos University, Camino del Molino, s/n, Fuenlabrada, 28943, Madrid, Spain
[b] Department of Business and Management, Rey Juan Carlos University, Camino del Molino, s/n, Fuenlabrada, 28943, Madrid, Spain
[c] D!lemmaLab Ldt., Camino del Molino, s/n, Fuenlabrada, 28943, Madrid, Spain
[d] William F. Harrah College of Hospitality, University of Nevada, Las Vegas, 4505 S. Maryland Pkwy, Las Vegas, 89154, NV, USA
[e] International Gaming Institute, University of Nevada, Las Vegas, 4505 S. Maryland Pkwy, Las Vegas, 89154, NV, USA
[f] Lee Business School, University of Nevada, Las Vegas, 4505 S. Maryland Pkwy, Las Vegas, 89154, NV, USA

ARTICLE INFO

Keywords:
Explainable Artificial Intelligence
Machine Learning
Interpretable machine learning
Model explainability
Perturbation-Based Explanations (peBEx)
Interpretability metrics

ABSTRACT

Explainable Artificial Intelligence (XAI) is critical for validating and trusting the decisions made by Machine Learning (ML) models, especially in high-stakes domains such as healthcare and finance. However, existing XAI methods often face significant computational challenges. To address this gap, this paper introduces a novel Perturbation-Based Explanation (PeBEx) method and comprehensively evaluates it versus local interpretable model-agnostic explanation approach (LIME) and SHapley Additive exPlanations (SHAP) across multiple datasets and ML models to assess explanation quality and computational efficiency. PeBEx leverages perturbation-based strategies to systematically alter input features and observe changes in model predictions to determine feature importance. This method not only offers superior computational efficiency, leading to scalability and efficiency for complex models on large datasets. Through testing on both synthetic and public datasets using eight ML models, we uncover the relative strengths and limitations of each XAI method in terms of explanation accuracy, fidelity, and computational demands. Our results show that while SHAP and LIME provide detailed explanations, they often suffer from high computational costs, particularly with complex models like Multi-Layer Perceptron (MLP). Conversely, PeBEx demonstrates superior efficiency and scalability, making it particularly suitable for applications that require rapid response times without compromising explanation quality. We conclude by proposing potential enhancements for PeBEx, including its adoption in a wider array of large-scale models. This study not only advances our understanding of the computational aspects of XAI but also proposes PeBEx as a viable solution for improving the efficiency, scalability, and applicability of explainability in ML.

1. Introduction

Artificial Intelligence (AI) and its branches, including Machine Learning (ML), have permeated virtually all aspects of our everyday lives, from internet product recommendations to loan applications. AI is also at the heart of countless recent advancements in science and technology across a wide range of academic disciplines and industries. Despite its widespread adoption, AI has been criticized for its lack of interpretability and the so-called "black box" nature of many models (Shrikumar et al., 2016). This opacity has raised concerns about the potential for AI systems to perpetuate bias, make unfounded decisions, or operate in ways that are not transparent to users or stakeholders (Rudin, 2019). In fact, there has even been concern that our society is turning into a "black box society", where algorithms are shrouded in secrecy and legalities that hide discriminatory or unethical practices (Pasquale, 2015). Therefore, as individuals and organizations become increasingly reliant on complex ML models, providing explainability is of paramount importance, as understanding the underlying drivers of predictions is critical for fairness, accountability, trust, and transparency (Messalas et al., 2019).

Explainability refers to explaining the operation of a model in a way that is transparent and understandable by a human (Doshi-Velez and Kim, 2017). Explainable AI (XAI) is concerned with developing

* Corresponding author at: Department of Business and Management, Rey Juan Carlos University, Camino del Molino, s/n, Fuenlabrada, 28943, Madrid, Spain.
E-mail addresses: ismael.gomez.talal@urjc.es (I. Gómez-Talal), luis.bote@urjc.es (L. Bote-Curiel), joseluis.rojo@urjc.es (J.L. Rojo-Álvarez).

https://doi.org/10.1016/j.engappai.2025.110664
Received 29 June 2024; Received in revised form 8 October 2024; Accepted 20 March 2025
Available online 9 May 2025
0952-1976/© 2025 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

model interpretability techniques that enable humans to understand and trust AI-based systems, ensure fair decision-making, detect underlying biases, and produce explainable models that maintain high performance accuracies (Vieira and Digiampietri, 2022). The push to achieve these goals has stimulated an exponential increase in research on XAI over the past decade (Minh et al., 2022), which has led to significant scientific advancement (Confalonieri et al., 2021) or regulatory interest (Ebers, 2020).

Interpretability methods can be categorized as either model-specific or model-agnostic. Model-specific approaches are typically built into a certain algorithm and only work for that particular algorithm. Model-agnostic approaches do not require any understanding of the internal mechanics of the black-box model and can be used for any predictive model, making them more versatile. Consequently, the number of model-agnostic post-hoc techniques for model explanation has increased significantly (Ribeiro et al., 2016b; Lundberg and Lee, 2017; Ribeiro et al., 2018; Lakkaraju et al., 2019). These techniques include using linear functions or rules (e.g. MAPLE or Anchor) to locally explain the complex decision-making process of black box models in humanly understandable ways (Lundberg and Lee, 2017; Plumb et al., 2018; Ribeiro et al., 2018). The idea is that although black box models have high non-linear decision boundaries at the global scale, they are less complex at the local scale and can be explained more simply (Slack et al., 2021).

Despite the popularity and utility of such models, recent research has uncovered computational inefficiencies related to local interpretable model-agnostic explanation (LIME; Ribeiro et al. (2016a)) and SHapley Additive exPlanations (SHAP; Lundberg and Lee (2017)). This is because the models need many model queries to produce their local interpretability estimations, requiring an exponential number of model evaluations to achieve an exact calculation (Van den Broeck et al., 2022). Thus, as models are scaled to big, high-dimensional datasets, speed becomes a critical issue. This is particularly challenging in areas such as natural language processing and computer vision where complex and computationally costly Deep Learning (DL) architectures are heavily used (Jethani et al., 2022).

In response to the significant computational inefficiencies identified in existing methods such as LIME and SHAP, particularly in handling DL models and large datasets, this paper introduces a novel Perturbation-Based Explainability (PeBEx) approach. LIME requires extensive model queries to generate local explanations, leading to substantial computation time (Ribeiro et al., 2016b), and SHAP can be computationally prohibitive, particularly when applied to DL models, due to its reliance on complex calculations for exact Shapley values (Lundberg and Lee, 2017; Lundberg et al., 2020). This is particularly critical not only in two-class problems but also in multi-class problems, where the complexity increases due to the sharp combinatorial increase of potential feature coalitions (Molnar et al., 2020b). PeBEx addresses these shortcomings by employing advanced techniques such as efficient array manipulation (Harris et al., 2020), parallel processing (Joblib Development Team, 2024), vectorized operations (Van Der Walt et al., 2011), and uniform perturbation sampling. This approach not only enhances computational efficiency but also retains a straightforward probabilistic interpretation, similar to the interpretation of a beta coefficient in linear regression. PeBEx provides feature importance by quantifying how model prediction probabilities shift in response to perturbations of specific features, while holding other features constant. This contribution is critical for developing more computationally efficient and interpretable AI models, particularly in contexts involving large-scale data and complex models.

In the remainder of this paper, Section 2 provides a Literature Review, covering the key concepts of explainability and interpretability in ML, distinguishing between global vs. local, and intrinsic vs. post-hoc approaches. Section 3 describes the Material and Methods, including the datasets, models, and the evaluation metrics utilized. Section 4 introduces the Perturbation-Based Explanations (PeBEx) model, detailing its novel approach to generating interpretable explanations through systematic feature perturbations. Section 5 presents the Experiments and Results, comparing PeBEx with LIME and SHAP in terms of computational efficiency and interpretability. Section 6 offers a Discussion on our findings, and concludes with potential areas of future research aimed at improving and expanding the PeBEx model. Finally, Section 7 explains the conclusions of this work and the usefulness of PeBEx model.

2. Literature review

In the context of ML, explainability and interpretability are often used interchangeably despite there being a clear definition of interpretability (Lipton, 2018; Carvalho et al., 2019). While these terminologies are conceptually intertwined, Adadi and Berrada (2018) point out that interpretable systems are explainable if humans can understand their operations. Yet, Doshi-Velez and Kim (2017) defined interpretability as the ability to explain the meaning to a human in understandable terms. Due to the terminological disconnect between the terms in the literature, in this study will use these words interchangeably to refer to the human ability to understand model operations.

Explainable ML is a specific case of the problem associated with the explanation of black box algorithms, which can be broke down conceptually into four issues: model explanation, outcome explanation, model inspection, and transparent box design (Guidotti et al., 2018). Model explanation involves shedding light into the black-box model through an interpretable model or predictor. Outcome explanation focuses on understanding the model prediction for a particular input value. Model inspection entails providing textual or visual depictions of specific properties of the black-box model or its outputs. Finally, the transparent box design challenge entails creating an algorithm that is inherently transparent from the outset.

The interpretability of a ML model can then be broken down into three dimensions: scope of interpretability, time limitation, and nature of user expertise (Guidotti et al., 2018). A ML model can be interpretable on either a global or local scale, which refers to how much of a model logic is understandable (Du et al., 2019). The model is globally interpretable if the logic of the model is traceable and understandable for the set of all possible outputs from start to finish. Otherwise, if only a single instance or a group of instances can be explained, the model is deemed locally interpretable. Time limitation refers to how much time is needed for the human user to understand the model. Time for understanding is typically constrained by the gravity or urgency of the prediction task. Finally, the nature of user expertise considers the user's background and familiarity with the problem at hand. User expertise raises the question of subjectivity in interpretability, since the perception of interpretability may vary between a research scientist and external stakeholders.

Explainable ML methods and techniques have thus been conceived to solve the black box explanation problem in various ways. Carvalho et al. (2019) suggest a taxonomy of interpretability that classifies explanation methods and techniques, which considers: (1) the scope of the interpretability of the model, (2) whether the interpretability technique is built into the model, and (3) the model agnosticity of the technique.

2.1. Global vs. Local explainability

Confalonieri et al. (2021) differentiate two classes of explanation methods: (1) global methods and (2) local methods. To solve the model explanation problem, global methods attempt to give overall approximation of the black box behavior by extracting an explainable counterpart from the model that mimics the behavior of the black box model. Local methods have been developed to solve the outcome explanation and model inspection problems. Local methods seek to provide

insight to specific outcomes or instances of a dataset by generating interpretable surrogate models.

The objective of global explanation methods is to explain the expected behavior of a black box model on a particular dataset (Molnar et al., 2020a). Global explanations provide a holistic view of the model general patterns, tendencies, and feature importance across the entire dataset and algorithmic process. The two main approaches for producing global explanations are: (1) to create representations of the decisions made by the black-box model and present it in an interpretable way, and (2) to quantify the effect that different features have on the outcome of a ML model (Confalonieri et al., 2021). Examples of the former approach are the use of decision rules from a neural network (Setiono and Liu, 1995), feature/permutation importance measures (Altmann et al., 2010), or the extraction of decision trees (Krishnan et al., 1999). The latter approach can be exhibited by partial dependence plots (PDPs; Friedman (2001)) for a smaller number of uncorrelated features or Accumulated Local Effects (ALE) plots (Apley and Zhu, 2020) for generalized cases.

Local methods seek to explain the predictions of a ML model for individual instances or a small subset of the dataset by manipulating the input data and analyzing their corresponding model predictions (Molnar et al., 2020a). Among the most common methods of explaining individual predictions are: counterfactual explanations, Shapley values, and surrogate models. Counterfactual explanations supply the user with a series of "what-if" information in regard to how different manipulations of input features would change the prediction outputs of an algorithm (Wachter et al., 2017). The most common approach to generating counterfactuals is through Diverse Counterfactual Explanations (DiCE; Mothilal et al. (2020)). Shapley values stem from game theory, representing the fair payout to each player of a collaborative game computed using a weighted average of each player's marginal contributions to all possible player combinations (Shapley, 1997; Kumar et al., 2020). But in the context of ML, the idea of a Shapley value is applied to measure feature importance where the players are the input variables and the payouts are the predictions. The use of surrogate models is another popular local explanation method. Surrogate models are interpretable models that mimic the behavior of the black box model (Azodi et al., 2020). The most popular surrogate model method for local explanation is a family of models known as LIME, which explain individual predictions by using proximal data to the data point being explained to train the interpretable model.

2.2. Types of interpretability

Interpretability in ML can be broadly categorized into two primary dimensions: *Intrinsic vs. Post-Hoc* and *Agnostic vs. Specific*. These dimensions help in understanding the different approaches to making ML models comprehensible to human users.

Intrinsic vs. Post-Hoc. Intrinsic and post-hoc interpretability are concerned with whether the interpretability of the ML model comes from a mechanism that is built into the model or an analysis after the model has already been trained. Intrinsic interpretability results from constraints imposed on the complexity of the ML model (Carvalho et al., 2019). Essentially, an intrinsically interpretable model is interpretable by itself through sparsity, causality or monotonicity constraints imposed on the model (Molnar et al., 2020a). Examples of intrinsic interpretability are the beta coefficients from a linear model or the nodes in a decision tree. Both of these intrinsic interpretability features are built into the model and allow the model to yield interpretable outcomes. Post-hoc, or post model, interpretability techniques are applied to the model once the training is complete, and can be applied to intrinsically interpretable models as well. Post-hoc techniques have become increasingly prevalent with the proliferation of DL techniques (Chakraborty et al., 2017). Some widely used examples of post-hoc interpretability methods are SHAP, LIME, and Generalized Additive Model (GAM).

Agnostic vs Specific. Interpretability methods can also be classified as either model-agnostic or model-specific. Model-agnostic methods are explainability methods that can be applied to any ML model, independent of whether the model is a black box. Model-agnostic techniques are applied after model training, which allow them to score yield interpretability scores without sacrificing any predictive power of the model (Lipton, 2018). Additionally, since they do not have access to the inner workings of the black box, they rely on following input/output feature pairings to interpret the model (Molnar et al., 2020a). Among the two most popular model-agnostic interpretability techniques used in ML are SHAP and LIME (Ribeiro et al., 2016b; Messalas et al., 2019). Model-specific interpretability methods, on the other hand, are limited to a specific model. This is because model-specific methods are tied to the internal mechanisms of a particular model (Molnar et al., 2020a). For example, in the case of a linear regression model, the beta coefficients are a model-specific interpretability technique, since they are based on the least square error estimates derived from the mathematical formula of the model. An example in DL would be saliency maps, which are specifically used for convolutional neural networks to explain individual predictions using network gradients (Adebayo et al., 2018).

2.3. Computational improvements in XAI

Several studies on explainable ML have outlined the need for clear evaluation goals and metrics for interpretability methods (Guidotti et al., 2018; Arrieta et al., 2020; Burkart and Huber, 2021; Belaid et al., 2022), allowing scientists to examine the strengths and weaknesses of the myriad XAI methods. One of the most important evaluation metrics for scientific researchers is run time, especially in engineering applications that demand fast computational times such as those in the fields of finance (Lyuu, 2002; Alexandrov et al., 2011; Alam et al., 2020), optimization (Chelouah and Siarry, 2022), cybersecurity (Chalé et al., 2020), and autonomous driving (Jean-Quartier et al., 2023). This is particularly important in the era of big data, where runtimes must be optimized to effectively handle the increasing amount of large, complex, and high-dimensional datasets (Xing et al., 2016; Zhou et al., 2017).

Despite the field of XAI still being quite nascent, three interesting veins of research have emerged with respect to computational runtime: (1) the improvement of previous algorithms, (2) the introduction of more efficient XAI techniques, and (3) the comparison of different algorithms. The stream of research associated with the improvement of existing algorithms seeks to analyze and improve upon the inefficiencies of a particular XAI technique, primarily SHAP. Other research has developed and introduced novel, more computationally efficient XAI methodologies. And finally, there is research concerned with comparing computational efficiencies across various interpretability methods.

Because of the computational inefficiencies of XAI methodologies, researchers have been interested in developing faster, more efficient versions of certain algorithms. The focus has been almost exclusively on improving SHAP algorithms, where researchers have proposed either stochastic estimators that sample permutations or subsets of features (Lundberg et al., 2019; Covert et al., 2020) or model-specific approximations for DL models (Ancona et al., 2019; Wang et al., 2021) or tree-based models (Lundberg et al., 2020). These contributions improve computation time through efficient approximation of the Shapley values themselves (Chen et al., 2018; Covert and Lee, 2021), the removal of features (Frye et al., 2020; Covert et al., 2021), and the amortization of SHAP values (Schwab and Karlen, 2019; Jethani et al., 2022).

In addition to improving existing methods, there has been research on the creation of new XAI techniques that yield better results. Two of the most notable are SAGE (Covert et al., 2020) and MAPLE (Plumb et al., 2018). MAPLE is a supervised neighborhood approach that uses concepts from local linear models and decision tree ensembles to

improve upon the computational efficiency of previous methods. SAGE is an additive importance measure that calculates feature importance by applying Shapley values to a function that represents the predictive power of feature subsets. It has been found to outperform SHAP in computational efficiency (Covert et al., 2020; Belaid et al., 2022).

The research on the comparison of different XAI algorithms is much sparser. Nguyen et al. (2021) compared the run time of LIME, SHAP, and Class Activation Mapping (CAM) for image classification. In their first experiment they found that LIME and SHAP took exponentially longer than CAM due to inefficiencies and had to lower the number of LIME and SHAP examples for subsequent experiments to reduce algorithmic run time. More recently, Belaid et al. (2022) conducted a more comprehensive comparative study of post-hoc XAI algorithms, where they considered run time as one of the benchmarks. They found that SHAP and LIME (and their variations) were the least computationally efficient models and were outperformed by MAPLE (Plumb et al., 2018).

We aim to contribute to the literature by introducing PeBEx, a novel and straightforward technique for explaining black box models. Using perturbations, PeBEx yields significantly better model explanations in terms of computational efficiency, which is critical in a variety of disciplines.

3. Material and methods

In this section, we present a comprehensive evaluation of various XAI models, we utilize a combination of synthetic and public datasets to thoroughly test the model performance and applicability. The synthetic datasets, designed with specific characteristics, allow us to control and test various aspects of model behavior under different conditions. In contrast, the public datasets provide real-world scenarios to validate the model effectiveness in practical applications. While this work primarily employs binary classification models for simplicity, the proposed approach is generalizable to multi-class problems, allowing for broader applicability in more complex scenarios. We detail the methodologies employed for dataset generation, model training, hyperparameter tuning, and evaluation metrics. Additionally, we describe the implementation of XAI methods, including LIME, SHAP, to compare their interpretability and computational efficiency across different datasets and ML models. The systematic evaluation and comparison aim to highlight the strengths and weaknesses of each XAI method, particularly in terms of their computational demands and the quality of their explanations, thereby providing a robust framework for selecting the most suitable explainability model for various applications.

3.1. Description of synthetic and public datasets

In our study, we use four different datasets to evaluate the effectiveness of our proposed perturbation-based XAI model, which is posed to offer computational efficiency in generating explanation of black box ML models. The datasets include both synthetic datasets and well-known public datasets, each serving a distinct purpose in demonstrating the applicability and advantages of our model.

Synthetic Datasets. To provide controlled environments for evaluating our model performance, we use four types of synthetic datasets, each with specific characteristics designed to test different aspects of the models' behavior:

- **Gaussian Clusters (Dataset 1).** This dataset is generated to simulate classification problems with varying levels of class overlap and feature noise. Each data point is drawn from multiple Gaussian distributions, where each cluster corresponds to a different class. We define the dataset by the feature matrix $\mathbf{X}_1 \in \mathbb{R}^{n\times 20}$ and the corresponding labels $\mathbf{y}_1 \in \{0,1\}^n$, where n is the number of samples. Out of the 20 features, 5 are informative, meaning they directly contribute to the class separability, while the remaining 15 features are noise. This dataset is useful for testing the model ability to distinguish between well-separated and overlapping classes (Guyon et al., 2008).
- **Sine Waves (Dataset 2).** This dataset is designed to assess model performance on non-linear and periodic patterns. It is generated using a set of sine wave functions with varying frequencies and amplitudes, combined with Gaussian noise to simulate realistic measurement errors. The feature matrix $\mathbf{X}_2 \in \mathbb{R}^{n\times 20}$ represents 20 input variables, and the target vector $\mathbf{y}_2$ is binarized based on a threshold applied to a non-linear transformation of the inputs. This dataset is particularly suitable for evaluating the model handling of non-linear relationships (Friedman, 1991; Breiman, 1996).
- **Mixed Shapes (Dataset 3).** This dataset challenges the model by containig geometric shapes distributed across the feature space. The dataset consists of 20 features, where 5 are informative and contribute directly to the class labels, and 15 are redundant, introducing multicollinearity. The feature matrix $\mathbf{X}_3 \in \mathbb{R}^{n\times 20}$ and the labels $\mathbf{y}_3$ are constructed to evaluate the model ability to manage complex interactions and highly correlated features (Guyon et al., 2008).
- **Random Noise (Dataset 4).** This dataset introduces significant variability and noise into the data, making it a robust test for the model. The feature matrix $\mathbf{X}_4 \in \mathbb{R}^{n\times 20}$ is generated by adding Gaussian noise to a base signal, with the data points grouped around two centers. The standard deviation of the clusters is set high (e.g., $\sigma = 10$) to introduce considerable overlap, challenging the model to extract meaningful patterns. The target labels $\mathbf{y}_4$ are binary, corresponding to the two clusters (Gaddam et al., 2022).

Public Datasets. To validate our perturbation-based XAI model in real-world scenarios, we also utilize five public datasets from reputable repositories (as summarized in Table 1). We selected these datasets for their relevance and the critical importance of explainability in their respective domains:

- **Heart Disease Dataset.** This dataset includes medical attributes such as age, sex, chest pain type, resting blood pressure, cholesterol, and more, with the target variable indicating the presence or absence of heart disease. Timely and accurate diagnosis of heart disease can save lives, and interpretable models can help healthcare professionals understand the respective risk factors (Wolberg et al., 1992).
- **Statlog (German Credit Data) Dataset.** Also from the UCI ML Repository, this dataset evaluates the creditworthiness of individuals based on various attributes such as credit history, purpose, amount, savings, employment, and personal status. The target variable classifies individuals as good or bad credit risks. Financial institutions require interpretable models to justify credit decisions to regulators and customers (Hofmann, 1994).
- **Breast Cancer Wisconsin (Diagnostic) Dataset.** Sourced from the UCI ML Repository, this dataset contains features computed from breast cancer biopsy images, including characteristics like radius, texture, perimeter, area, and smoothness. The target variable indicates whether the tumor is malignant or benign. Early and accurate detection of cancer is crucial, and understanding the model decision-making process can significantly impact clinical outcomes (Azar and El-Metwally, 2013; Wolberg et al., 1995).
- **Car Evaluation Dataset.** Evaluating cars based on attributes such as buying price, maintenance cost, number of doors, person capacity, luggage boot size, and safety, this dataset classifies cars as acceptable or unacceptable. Understanding the factors that influence customer preferences can guide design and marketing strategies in the automotive industry (Bohanec, 1997).
- **Santander Customer Satisfaction Dataset.** This dataset is provided by Santander Bank via a Kaggle competition. It includes hundreds of anonymized features, aiming to predict customer satisfaction or dissatisfaction with their banking experience. The goal is to identify dissatisfied customers early to allow the bank to take proactive steps to improve customer satisfaction before churn (Jimenez and Cukierski, 2016).

Table 1
Summary of various datasets used for classification tasks in different fields. Each dataset is described by its characteristics, the subject area it pertains to, the types of features it includes, the size of the sample, and the total number of features.

Dataset	Subject	Feature Type	Sample Size	Features
Heart Disease (Wolberg et al., 1992)	Health	Categorical, Integer, Real	303	13
German Credit Data (Hofmann, 1994)	Social Science	Categorical, Integer	1000	20
Breast Cancer Wisconsin (Wolberg et al., 1995)	Health	Real	569	30
Car Evaluation (Bohanec, 1997)	Other	Categorical	1728	6
Santander Customer Satisfaction (Jimenez and Cukierski, 2016)	Banking	Real, Anonymized	76 020	370

3.2. Overview of binary classification machine learning models

In our experiment, we evaluated eight distinct binary classification models, applying them across various scenarios. Each model underwent training using stratified k-fold cross-validation, after the data was undersampled to ensure an equal distribution of observations of target classes. Subsequent optimization phases involved hyperparameter tuning using an exhaustive grid search to identify the top-performing model among the candidates. Below, we provide a summary of the models examined.

Logistic Regression (LR) is a fundamental model for binary classification, predicting the probability that a given input belongs to a particular category. The model estimates probabilities using a logistic function, which is expressed as:

$$p(\boldsymbol{y} = 1|\boldsymbol{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \boldsymbol{x})}}, \tag{1}$$

where $p(\boldsymbol{y} = 1|\boldsymbol{x})$ is the probability of the input $\boldsymbol{x}$ being classified as class 1, β_0 is the intercept, and β_1 is the coefficient for the input feature x. This model is favored for its simplicity and interpretability (Pedregosa et al., 2011; Hosmer et al., 2013).

Support Vector Machines (SVM) construct a hyperplane in high-dimensional spaces to separate different classes. The optimal hyperplane is the one that maximizes the margin between the closest points of the classes, which are called support vectors. For non-linearly separable data, SVM uses kernel functions to map inputs into high-dimensional feature spaces. The decision function is defined as:

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i \boldsymbol{y}_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right), \tag{2}$$

where $K(\boldsymbol{x}_i, \boldsymbol{x})$ is the kernel function, $\boldsymbol{y}_i$ are the labels, α_i are the Lagrange multipliers, and b is the bias (Cortes and Vapnik, 1995; Pedregosa et al., 2011; Buitinck et al., 2013).

Random Forest (RF) improves upon decision trees by creating an ensemble of trees and aggregating their predictions. Each tree is trained on a random subset of the data, and the final classification is typically determined by majority vote: $\hat{\boldsymbol{y}} = \text{mode}\{y_1, y_2, \ldots, y_n\}$, where $\hat{\boldsymbol{y}}$ is the predicted class and $\boldsymbol{y}_i$ are the predictions of the ith tree in the forest (Breiman, 2001).

Gradient Boosting (GB) builds an ensemble of weak prediction models, typically decision trees, in a sequential manner where each subsequent model corrects errors made by previous models. The update rule for each model can be expressed as:

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \rho_m h_m(\boldsymbol{x}), \tag{3}$$

where $F_m(\boldsymbol{x})$ is the model at iteration m, $h_m(\boldsymbol{x})$ is the weak learner, and ρ_m is the weight of $h_m(\boldsymbol{x})$ (Friedman, 2001).

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements ML algorithms under the Gradient Boosting framework, with enhancements for performance and speed (Chen and Guestrin, 2016).

LightGBM is a gradient-boosting framework that uses tree-based learning algorithms. It is designed for speed and efficiency, with a focus on large datasets. LightGBM improves on traditional gradient boosting methods by using a novel tree growth algorithm called Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), which significantly reduces the computational cost of handling large data, where GOSS keeps the instances with large gradients and randomly samples the instances with small gradients, while EFB bundles mutually exclusive features to reduce the feature dimension (Ke et al., 2017).

Multi-Layer Perceptron (MLP) is a class of feedforward artificial neural networks that consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. The output of each neuron is a nonlinear function of a weighted sum of its inputs:

$$\boldsymbol{y} = \sigma\left(\sum_{i=1}^{n} \boldsymbol{w}_i \boldsymbol{x}_i + b\right), \tag{4}$$

where σ is the activation function, $\boldsymbol{w}_i$ are the weights, $\boldsymbol{x}_i$ are the inputs, and b is the bias. MLPs can model complex relationships between inputs and outputs, making them suitable for a wide range of applications, including classification and regression tasks (Pedregosa et al., 2011; Rumelhart et al., 1986).

PyTorch Neural Network. In addition to the traditional ML models, we also implemented a DL model using PyTorch, focusing on a simple feedforward neural network architecture (Paszke et al., 2017). This model was designed to explore the potential of neural networks in binary classification tasks within the scope of our experiment. The PyTorch neural network, named *SimpleNet*, consists of three fully connected (dense) layers (Paszke et al., 2017). The architecture can be summarized as follows:

- **Input layer.** The input dimension corresponds to the number of features in the dataset.
- **First hidden layer.** This layer consists of 64 neurons with ReLU activation functions.
- **Second hidden layer.** This layer reduces the dimensionality with 32 neurons, also using ReLU activation functions.
- **Output layer.** A single neuron with a sigmoid activation function outputs the probability of the input belonging to the positive class (class 1).

The forward propagation through the network is defined as:

$$\boldsymbol{y} = \sigma(\text{ReLU}(\boldsymbol{W}_2 \cdot \text{ReLU}(\boldsymbol{W}_1 \cdot \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2) + \boldsymbol{b}_3), \tag{5}$$

where $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ are the weight matrices, $\boldsymbol{b}_1$, $\boldsymbol{b}_2$, $\boldsymbol{b}_3$ are the biases, and σ is the sigmoid activation function.

The model was trained and optimized using the *NeuralNetClassifier* from the Skorch library, which integrates PyTorch models with scikit-learn functionalities (Thomas et al., 2018). The optimization involved tuning several key hyperparameters through grid search:

- **Learning Rate (lr):** The learning rate determines the step size at each iteration while moving toward a minimum of the loss function. We tested values of 0.01 and 0.1.
- **Max epochs:** This parameter sets the maximum number of epochs for training. We experimented with values of 10, 20, and 50 epochs to evaluate the model performance with different levels of training.

The model was trained using the Binary Cross-Entropy Loss (BCELoss) function, which is suitable for binary classification tasks (Paszke et al., 2019). The training data was shuffled at each epoch to improve generalization. This PyTorch model allowed us to explore the flexibility and performance of DL techniques in comparison to traditional methods (Paszke et al., 2019).

3.3. Stratified K-fold cross-validation for binary model training

Stratified K-Fold cross-validation is a crucial technique, especially beneficial for datasets with imbalanced class distributions. This method ensures that classes are evenly distributed across all folds, providing a fair and consistent evaluation of model performance (Wong and Yang, 2017; Prusty et al., 2022).

In this approach, the training dataset (which makes up 80% of the total dataset) is organized by the target class and divided into five stratified folds (reflecting the choice of k=5 for this scenario). Each fold maintains the proportion of target classes present in the full dataset. During each iteration of validation, one fold is used as the validation set, while the remaining folds are combined to form the training set. The model is trained on this combined training set and then evaluated on the validation set (Zeng and Martinez, 2000).

To ensure reproducibility, the dataset split is performed randomly using a fixed seed. The choice of the seed value 42 is conventional in many ML studies because it serves as a playful reference to The Hitchhiker's Guide to the Galaxy, where "42" is humorously referred to as the "Answer to the Ultimate Question of Life, the Universe, and Everything" (Adams, 1995). By fixing the seed, we ensure that the data splitting procedure can be exactly replicated in future experiments.

This process is repeated five times, with each fold serving as the validation set exactly once. It is important to note that models are recalibrated in each iteration to avoid any potential bias from previously trained models. The performance metrics from each fold are then averaged, resulting in a comprehensive and accurate assessment of model performance (Purushotham and Tripathy, 2011).

3.4. Evaluation metrics

In this study, six fundamental metrics were used to evaluate each model: accuracy, recall, F1-Score, precision, specificity, and area under the curve (AUC) (Bishop and Nasrabadi, 2006). The following sections describe each of these metrics and their respective calculations.

Accuracy provides an overall measure of model performance by indicating the proportion of correct predictions relative to the total number of predictions made. It is calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{6}$$

where TP represents true positives, TN represents true negatives, FP represents false positives, and FN represents false negatives.

Recall measures the model ability to correctly identify all positive cases, which is especially useful in situations where false negatives are more problematic than false positives. It is defined as:

$$Recall = \frac{TP}{TP + FN}. \tag{7}$$

Precision measures the proportion of correctly classified positive samples (true positives) among all samples classified as positive (TP + FP). It is calculated as:

$$Precision = \frac{TP}{TP + FP}. \tag{8}$$

The F1-Score is a balanced measure between precision and recall, and it is computed as follows:

$$F1 = \frac{2 \times (Precision \times Recall)}{Precision + Recall}. \tag{9}$$

Specificity measures the proportion of correctly classified negative samples (true negatives) among all samples classified as negative (true negatives + false positives). It is defined as:

$$Specificity = \frac{TN}{TN + FP}. \tag{10}$$

Finally, the AUC represents the area under the receiver operating characteristic (ROC) curve. It evaluates the overall performance of the model by considering the true positive rate and the false positive rate. The AUC is defined as the integral calculating the area as follows:

$$AUC = \int_0^1 \mathrm{ROC}(t), \mathrm{d}t, \tag{11}$$

where an AUC of 1 denotes a perfect classification model, while a value of 0.5 suggests performance no better than random classification (Fawcett, 2006; Powers, 2020).

3.5. Explainable AI

XAI has become a crucial area of research, aiming to make the decisions of ML models more transparent and interpretable. This section provides an overview of three prominent methods in XAI: LIME, SHAP, and the novel PeBEx model based on perturbations.

Local interpretable model-agnostic explanations is a technique designed to explain individual predictions of any ML model. LIME operates by approximating the complex model with an interpretable one locally around the prediction. This is achieved by generating perturbed samples around the instance of interest and observing the changes in the model predictions. The locally weighted linear model is then trained on these samples to provide an explanation. Mathematically, the explanation model g is trained to minimize the loss function:

$$L(f, g, \pi_{\boldsymbol{x}}) = \sum_{z \in Z} \pi_{\boldsymbol{x}}(\boldsymbol{z})(f(\boldsymbol{z}) - g(\boldsymbol{z}))^2, \tag{12}$$

where f is the original model, $\boldsymbol{Z}$ is the set of perturbed samples, and $\pi_{\boldsymbol{x}}(\boldsymbol{z})$ is a proximity measure between the original instance $\boldsymbol{x}$ and the perturbed sample $\boldsymbol{z}$ (Ribeiro et al., 2016b). LIME is widely used for its simplicity and model-agnostic nature, providing intuitive explanations for individual predictions.

SHapley Additive exPlanations originates from game theory, where Shapley values quantify the average marginal contribution of a participant within a cooperative setting (Roth, 1988). When applied to ML, these participants are analogous to features or attributes, transforming the cooperative game into the predictive task executed by the model. Thus, Shapley values elucidate the influence of each attribute on the prediction outcome of a ML model (Merrick and Taly, 2020). Specifically, for a given query point, the Shapley value of a feature elucidates its contribution to the model prediction, whether it be the predicted value in regression or the class scores in classification. The contribution of a feature is gauged by its impact on the prediction deviation from the model average prediction at that query point. Mathematically, for a feature i at query point $\boldsymbol{x}$, its Shapley value is defined by the value function $v_{\boldsymbol{x}}$ as

$$\varphi_i(v_{\boldsymbol{x}}) = \frac{1}{M} \sum_{S} \subseteq \boldsymbol{M}_s \backslash \{i\} \frac{v_{\boldsymbol{x}}(\boldsymbol{S} \cup \{i\}) - v_{\boldsymbol{x}}(\boldsymbol{S})}{\binom{M-1}{|S|}}, \tag{13}$$

where M denotes the total number of features, $\boldsymbol{M}_s$ the set of all features, $|\boldsymbol{S}|$ the cardinality of set $\boldsymbol{S}$, and $v_{\boldsymbol{x}}(\boldsymbol{S})$ the expected contribution of set $\boldsymbol{S}$ features to the prediction at $\boldsymbol{x}$ (Lundberg and Lee, 2017; Lundberg et al., 2020).

The SHAP Python package offers several algorithms, known as *interventional algorithms*, for computing these values. These algorithms define the value function for a set of attributes at a query point as the expected prediction under the intervention distribution, representing the joint distribution of the attributes not in the set:

$$v_{\boldsymbol{x}}(\boldsymbol{S}) = E_D[f(\boldsymbol{x}_S, \boldsymbol{X}_{S^c})], \tag{14}$$

where $\boldsymbol{x}_S$ represents the values of the features in set $\boldsymbol{S}$ at the query point, and $\boldsymbol{X}_{S^c}$ those not in $\boldsymbol{S}$. Assuming features are independent, the value function $v_{\boldsymbol{x}}(\boldsymbol{S})$ is approximated using data $\boldsymbol{X}$ as samples from the intervention distribution D for $\boldsymbol{S}^c$:

$$v_{\boldsymbol{x}}(\boldsymbol{S}) = E_D[f(\boldsymbol{x}_S, \boldsymbol{X}_{S^c})] \approx \frac{1}{N} \sum_{j=1}^{N} f(\boldsymbol{x}_S, (\boldsymbol{X}_{S^c})_j), \tag{15}$$

with N representing the sample size, and $(\boldsymbol{X}_{\boldsymbol{S}^c})_j$ the values for $\boldsymbol{S}^c$ in the jth observation (Lundberg and Lee, 2017).

Interventional algorithms, while computationally more efficient, presume feature independence and rely on samples from outside the distribution, potentially leading to unrealistic observations (Kumar et al., 2020). Key SHAP interventional algorithms include: *Kernel SHAP*, employing a kernel approximation for Shapley value estimation, suits high-dimensional spaces albeit with higher computational demands (Lundberg and Lee, 2017); *Linear SHAP* offers an analytical computation of Shapley values for linear models, enhancing computational efficiency compared to Kernel SHAP (Lundberg and Lee, 2017); and *Tree SHAP*, tailored for tree-based models like decision trees, RF, and gradient boosting algorithms, excels in computational efficiency and explicitly accommodates feature interactions (Lundberg et al., 2020).

3.6. Evaluation metrics for XAI models

Evaluating the effectiveness of XAI models involves several key metrics that ensure the provided explanations are accurate, comprehensible, consistent, stable, locally accurate, similar, and robust. Below, we describe each metric, its significance, and how it is generally calculated across SHAP, LIME, and PeBEx.

Fidelity. This metric measures how well the explanations approximate the actual model predictions, with higher values indicating more precise explanations (Velmurugan et al., 2021). It is defined as:

$$\text{Fidelity} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y^{(i)}_{\text{pred\_m}} = y^{(i)}_{\text{pred\_e}}), \tag{16}$$

where $\mathbb{1}$ is the indicator function, $y^{(i)}_{\text{pred\_m}}$ is the prediction of the model, and $y^{(i)}_{\text{pred\_e}}$ is the prediction based on the explainer. In SHAP, $y_{\text{pred\_e}}$ is derived from Shapley values; in LIME, it is obtained from local predictions; in PeBEx, it is approximated using perturbation importance values.

Comprehensibility. This metric refers to the number of features used in the explanation; fewer features indicate a simpler and more understandable explanation (Doran et al., 2017; Došilović et al., 2018). In SHAP and PeBEx, it is the number of significant features; in LIME, it is the number of features in the explanation list. It is defined as:

Consistency. This metric assesses the similarity between explanations of similar instances, where higher cosine similarity values between feature importance vectors suggest better consistency (Lin et al., 2021b). It is defined as:

$$\text{Consistency} = \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i} \text{cosine\_similarity}(\text{explanation}^{(i)}, \text{explanation}^{(j)}), \tag{17}$$

where cosine_similarity is the cosine similarity between the explanations of instances i and j (Wojke and Bewley, 2018). SHAP uses SHAP values, LIME uses local feature importance, and PeBEx uses perturbation-based explanations.

Fidelity, Comprehensibility, and Consistency Score (FCC Score). In an effort to provide a comprehensive evaluation of XAI models, we introduce the FCC Score, which integrates Fidelity, Comprehensibility, and Consistency into a single metric. This score offers a holistic measure of the effectiveness of XAI models by combining these three key aspects. For each metric M_i (where M_1 represents Fidelity, M_2 represents Comprehensibility, and M_3 represents Consistency), we consider the mean (μ_i) and standard deviation (σ_i). The normalized mean ($\tilde{\mu}_i$) for each metric is calculated within the range [0, 1], ensuring that higher values are always better, including for metrics such as Comprehensibility and Consistency where lower raw values might be more desirable. For these, appropriate transformations are applied to reflect the "higher is better" principle.

The variance (V_i) for each metric is calculated as the square of the standard deviation:

$$V_i = \sigma_i^2. \tag{18}$$

Each metric is assigned an equal weight ($w_i = \frac{1}{3}$) since all metrics are considered equally important. The weighted mean for each metric is computed, and the combined mean (μ_{FCC}) is obtained by summing the weighted means:

$$\mu_{\text{FCC}} = \sum_{i=1}^{3} w_i \cdot \tilde{\mu}_i. \tag{19}$$

For Comprehensibility, which is ideally lower, normalization involves a transformation to fit the range [0, 1], so that higher values indicate better comprehensibility. This is done by reversing the scale, where the normalized score is given by:

$$\tilde{\mu}_2 = 1 - \frac{\mu_2 - \min}{\max - \min}, \tag{20}$$

with min and max being the range of observed values for this metric. Similarly, Consistency is normalized such that values closer to 0 are better, using the formula:

$$\tilde{\mu}_3 = 1 - |\mu_3|. \tag{21}$$

Similarly, the combined variance (V_{FCC}) is calculated by summing the weighted variances:

$$V_{\text{FCC}} = \sum_{i=1}^{3} w_i^2 \cdot V_i. \tag{22}$$

The combined standard deviation (σ_{FCC}) is then derived as the square root of the combined variance:

$$\sigma_{\text{FCC}} = \sqrt{V_{\text{FCC}}}. \tag{23}$$

The final FCC Score is represented by the combined mean (μ_{FCC}) and the combined standard deviation (σ_{FCC}). This method ensures that each metric is normalized to fit within the [0, 1] range, where higher values represent better performance, regardless of the raw metric characteristics. This ensures a consistent interpretation across Fidelity, Comprehensibility, and Consistency. This approach allows for a balanced evaluation of XAI models, taking into account the accuracy of explanations (Fidelity), their understandability (Comprehensibility), and their consistency across similar instances (Consistency).

Stability. This metric evaluates the variation in explanations when small perturbations are applied to the input data, with lower L2 norm differences indicating more stable explanations, where stability is defined as:

$$\text{Stability} = \frac{1}{n}\sum_{i=1}^{n} \|\mathrm{e}_{\mathrm{o}}^{(i)} - \mathrm{e}_{\mathrm{p}}^{(i)}\|, \tag{24}$$

where $\|\cdot\|$ denotes the L2 norm. SHAP, LIME, and PeBEx all measure stability by comparing the original (defined as e_{o}) and perturbed ((defined as e_{p})) explanations (Alvarez Melis and Jaakkola, 2018; Eykholt et al., 2018; Rojat et al., 2021; Barbado and Corcho, 2022).

Local Accuracy. This metric measures how well the local explanations predict the model decision in the neighborhood of the instance of interest, with higher accuracy scores being better (Hoffman et al., 2018). It is defined as:

$$\text{Local Accuracy} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y_{\mathrm{a}}^{(i)} = y_{\mathrm{l\_p}}^{(i)}), \tag{25}$$

where y_{actual} is defined as actual prediction, and $y_{\mathrm{l\_p}}$ is defined as local prediction. In SHAP, the local prediction is derived from SHAP values; in LIME, it is obtained from the local surrogate model; in PeBEx, it is based on the mean perturbation importance values.

Similarity. This metric measures the resemblance between explanations of different instances, with higher cosine similarity values

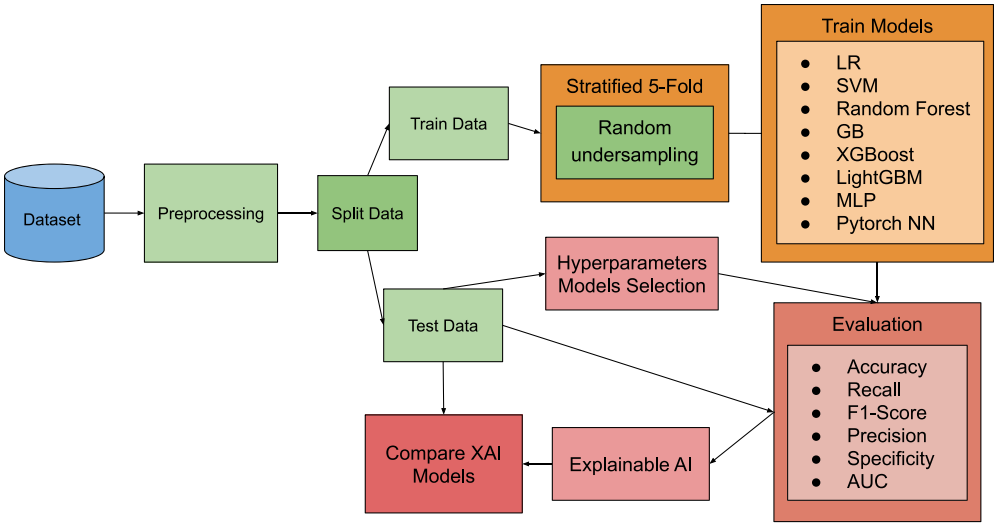

Fig. 1. Comprehensive workflow for evaluating post-hoc explainability models. The process includes (1) preprocessing datasets, (2) splitting data into training and test sets, (3) addressing class imbalance, (4) training multiple models with grid search optimization, (5) evaluating model performance, and (6) applying XAI techniques (LIME, SHAP, and PeBEx) to the best models to compare their interpretability and computational efficiency.

indicating greater similarity (Hu et al., 2022). It is defined as:

$$\text{Similarity} = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i} \text{cosine\_similarity}(e^{(i)}, e^{(j)}), \tag{26}$$

where e is defined as explanation model. SHAP, LIME, and PeBEx all use cosine similarity to measure the similarity between explanations. Cosine similarity is defined as:

$$\text{cosine\_similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}, \tag{27}$$

where $\mathbf{a}$ and $\mathbf{b}$ are vectors, $\cdot$ denotes the dot product, and $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ are the magnitudes of the vectors. This metric calculates the average cosine similarity between all pairs of explanations, excluding self-pairs (i.e., $i \neq j$). A higher average cosine similarity value indicates that the explanations are more similar to each other, implying consistency in how the model explains its decisions across different instances (Hanawa et al., 2020).

Robustness. This metric assesses the variation in explanations when large perturbations are applied, with lower L2 norm differences indicating more robust explanations (Alvarez-Melis and Jaakkola, 2018; Agarwal et al., 2018), where robustness is defined as:

$$\text{Robustness} = \frac{1}{n} \sum_{i=1}^{n} \|e_o^{(i)} - e_{l\_p}^{(i)}\|, \tag{28}$$

where $\|e_o^{(i)} - e_{l\_p}^{(i)}\|$ is the L2 norm difference between the original and large perturbed explanations for instance i, and n is the number of instances.

3.7. Workflow for evaluating Post-Hoc explainability models

To systematically evaluate and compare different post-hoc explainability models, we adopt a structured workflow diagram (as shown in Fig. 1). The first stage of the workflow involves dataset preprocessing and splitting. Each dataset is cleaned and preprocessed to ensure consistency and quality, which includes handling missing values, normalizing features, and encoding categorical variables. Following preprocessing, the dataset is divided into training and test sets while maintaining the original class distribution.

In the model training and evaluation stage, stratified 5-fold cross-validation is applied to the training data to preserve the class distribution in each fold. Random undersampling is then performed to balance the class distribution within each fold, ensuring equitable model training. The selected models (LR, SVM, RF, GB, XGBoost, LightGBM, MLP and PyTorch NN) are then trained using the training data. Hyperparameter tuning is conducted using grid search techniques to optimize each model performance.

Model performance is evaluated based on several metrics, including accuracy, recall, F1-Score, precision, specificity, and AUC. The best-performing model for each dataset is identified based on these evaluation metrics.

In the final stage, the XAI model comparison is conducted. For the best-performing model from each dataset, three different XAI methods (LIME, SHAP, and PeBEx) are applied and compared to evaluate their interpretability and computational efficiency. Particular attention is given to the computational efficiency of PeBEx compared to SHAP when applied to DL models like MLPs, as PeBEx often requires less computational time, providing an advantage in applications where computational efficiency is paramount.

Hardware Specifications. All experiments were conducted on a computer with the following specifications: (1) 2.3 GHz Intel Core i9, 8 cores, (2) Radeon Pro 560X with 4 GB VRAM, Intel UHD Graphics 630 with 1536 MB VRAM and (3) 32 GB of 2400 MHz DDR4 RAM. Since this paper involves comparisons of computational times to evaluate different XAI models, it is necessary to provide the context of the hardware specifications used for the experiments.

4. Perturbation-based explanations

PeBEx is a novel method designed to offer interpretable explanations by perturbing the input features and observing the resulting changes in the model predictions. This method involves systematically altering the values of input features and measuring the effect on the output to determine the importance and influence of each feature. The perturbation process is defined by a series of theorems and notations, as outlined below.

In this context, let $\boldsymbol{X}$ denote the feature matrix and $\boldsymbol{X}_{test}$ a subset of $\boldsymbol{X}$ used for model evaluation. For each feature $\boldsymbol{x}_i$ in $\boldsymbol{X}$, we define $\tilde{\boldsymbol{x}}_i$ as the perturbation range for the i-th feature. The function f represents

the classification ML model, and $P(c|\boldsymbol{X})$ indicates the probability that the sample $\boldsymbol{X}$ belongs to class c as predicted by the model f. The range of values generated for the i-th feature during perturbation is denoted by range($\tilde{\boldsymbol{x}}_i$).

Theorem 1 (*Generation of Perturbation Range*)**.** *For each feature $\boldsymbol{x}_i$ in $\boldsymbol{X}_{test}$, the perturbation range $\tilde{\boldsymbol{x}}_i$ is defined by the set:*

$$\tilde{\boldsymbol{x}}_i = \left\{ \boldsymbol{x} \mid \boldsymbol{x} = \min(\boldsymbol{X}_{test,i}) + k \cdot \delta,\ 0 \leq k < n \right\}, \tag{29}$$

where $\delta = \frac{\max(\boldsymbol{X}_{test,i}) - \min(\boldsymbol{X}_{test,i})}{n-1}$ and n is the number of points in the perturbation range.

This theorem establishes the method of constructing a linearly spaced set of perturbation values $\tilde{\boldsymbol{x}}_i$ for each feature. It calculates equally spaced points between the minimum and maximum observed values of the feature in the testing set.

Theorem 2 (*Effect of Perturbation on Class Probability*)**.** *Given a value V within $\tilde{\boldsymbol{x}}_i$, the i-th feature of all samples in $\boldsymbol{X}_{test}$ is modified to V. The probability for each class is then recalculated as:*

$$P(c|\boldsymbol{X}_{test}, i = V) = \frac{1}{|\boldsymbol{X}_{test}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_{test}} f_c(\boldsymbol{x}_1, \ldots, V, \ldots, \boldsymbol{x}_n), \tag{30}$$

where f_c is the model function that returns the probability of class c for the modified sample $\boldsymbol{X}$.

Theorem 3 (*Calculation of Feature Importance Based on Probabilities*)**.** *The importance of feature $\boldsymbol{x}_i$ is determined by calculating the variation in the predicted probability for a particular class c as:*

$$I(\boldsymbol{x}_i) = \max_{V \in \tilde{\boldsymbol{x}}_i} \left(P(c|\boldsymbol{X}_{test}, i = V) \right) - \min_{V \in \tilde{\boldsymbol{x}}_i} \left(P(c|.\boldsymbol{X}_{test}, i = V) \right) \tag{31}$$

This measure shows how much the decision of the model (in terms of class probability) can change due to variations in feature $\boldsymbol{x}_i$ across its perturbation range.

PeBEx provides a straightforward and intuitive approach to understanding feature importance by directly measuring the impact of feature changes on model predictions. One of the main advantages of PeBEx, particularly in the context of DL models such as MLP, is its computational efficiency. Unlike SHAP, which can be computationally intensive and time-consuming, especially for DL models, PeBEx offers a faster alternative. This is particularly advantageous in critical applications where computational time is a significant concern.

It is important to note that the granularity of PeBEx explanations can be controlled through the number of points in the perturbation range n. The perturbation range defines how many values within the variation interval of a feature are tested when evaluating the importance of that feature. Increasing the number of points n in the perturbation range provides more granularity, allowing PeBEx to capture subtler changes in the model predictions. This can enhance the precision of the explanations by observing smaller variations in feature values.

Conversely, reducing the number of points can speed up the computation process but may result in a loss of detail in how a feature impacts the model predictions. If the perturbation points are too sparse, the relationship between the feature and the model might not be fully captured, potentially lowering the quality of the explanation.

Therefore, adjusting the number of points in the perturbation range allows for a trade-off between computational efficiency and the level of interpretability. This flexibility ensures that PeBEx can be tailored to different applications, where varying levels of feature granularity may be required depending on the computational resources available and the need for detailed explanations.

PeBEx Algorithm. The PeBEx algorithm operates by generating perturbations for each feature in the dataset and evaluating the impact on the model predictions. The notation used in the algorithm is detailed in Table 2, to ensure clarity and consistency throughout the explanation.

Table 2
Notation Table. This table defines the symbols used in our algorithm.

Symbol	Description
f	Trained ML model
$\boldsymbol{X}$	Dataset
$\boldsymbol{X}_{test}$	Subset of $\boldsymbol{X}$ for model evaluation
$\boldsymbol{x}_i$	ith feature in $\boldsymbol{X}$
$\tilde{\boldsymbol{x}}_i$	Perturbation range for ith feature
N_p	Number of perturbations
I_i	Importance score of feature i
$\boldsymbol{X}_{temp}$	Temporary dataset with perturbed values
$pred\_probs$	Predicted probabilities for perturbed dataset
$probabilities$	Average predicted probabilities for perturbed dataset
n	Number of points in the perturbation range
δ	Step size in the perturbation range

Algorithm 1 Perturbation-Based Explanation Algorithm

Input: Trained model f, dataset $\boldsymbol{X}$, number of perturbations N_p
Output: Feature importance scores I
Initialize an empty dictionary I to store importance scores.
for each feature $\boldsymbol{x}_i$ in $\boldsymbol{X}$ **do**
  Generate N_p perturbation values for $\boldsymbol{x}_i$ using a uniform distribution:
  $\tilde{\boldsymbol{x}}_i = \text{uniform}(\min(\boldsymbol{X}_i), \max(\boldsymbol{X}_i), N_p)$
  Create a temporary dataset $\boldsymbol{X}_{temp}$ by tiling $\boldsymbol{X}$:
  $\boldsymbol{X}_{temp} = \text{tile}(\boldsymbol{X}.values, (N_p, 1))$
  Replace the values of $\boldsymbol{x}_i$ in $\boldsymbol{X}_{temp}$ with $\tilde{\boldsymbol{x}}_i$:
  $\boldsymbol{X}_{temp}[:, \text{get\_loc}(\boldsymbol{x}_i)] = \text{repeat}(\tilde{\boldsymbol{x}}_i, \boldsymbol{X}.shape[0])$
  Predict probabilities using the model f on the perturbed dataset $\boldsymbol{X}_{temp}$:
  $pred\_probs = f.\text{predict\_proba}(\boldsymbol{X}_{temp})$
  Compute the average predicted probabilities:
  $probabilities = \text{mean}(pred\_probs[:, 1].\text{reshape}(N_p, \boldsymbol{X}.shape[0]), \text{axis} = 1)$
  Calculate the variance of the predicted probabilities for feature $\boldsymbol{x}_i$:
  $I_i = \text{var}(probabilities)$
  Store the importance score in the dictionary I:
  $I[x_i] = I_i$
end for
Return the importance scores I.

Initially, for each feature $\boldsymbol{x}_i$ in the dataset $\boldsymbol{X}$, a set of perturbation values $\tilde{\boldsymbol{x}}_i$ is generated using a uniform distribution. These perturbation values span the full observed range of the feature. A temporary dataset $\boldsymbol{X}_{temp}$ is then created by tiling the original dataset $\boldsymbol{X}$ to accommodate all perturbation instances.

The values of the feature $\boldsymbol{x}_i$ in $\boldsymbol{X}_{temp}$ are replaced with the perturbation values $\tilde{\boldsymbol{x}}_i$. The trained model f is used to predict probabilities for this perturbed dataset, and the average predicted probabilities are calculated. The variance of these probabilities is computed for each feature $\boldsymbol{x}_i$, which quantifies the feature importance. The feature importance scores are stored in a dictionary $\boldsymbol{I}$ and returned upon completion.

4.1. Mathematical proofs

We provide mathematical proofs to demonstrate the effectiveness of the PeBEx method in determining feature importance. These proofs are divided into an inductive proof and a probabilistic proof by construction.

Inductive Proof: To demonstrate the effectiveness of the PeBEx method in determining feature importance, we provide a proof by induction.

Base Case. Consider a simple model with a single feature $\boldsymbol{x}$ and a dataset $\boldsymbol{X}$ containing a single data point $\boldsymbol{x}_1$.

1. For the point $\boldsymbol{x}_1$, we generate a perturbation range $\tilde{\boldsymbol{x}}_1$ using the given formula:

$$\tilde{\boldsymbol{x}}_1 = \left\{ \boldsymbol{x} \mid \boldsymbol{x} = \min(\boldsymbol{X}_1) + k \cdot \delta,\ 0 \leq k < n \right\} \tag{32}$$

where $\delta = \frac{\max(\boldsymbol{X}_1) - \min(\boldsymbol{X}_1)}{n-1}$.

2. We evaluate the model f for each value in $\tilde{\boldsymbol{x}}_1$ and compute the probabilities:

$$P(c|\boldsymbol{X}, i = V) = f_c(\boldsymbol{x}_1) \tag{33}$$

where f_c is the model function that returns the probability of class c for the modified sample.

3. The importance of feature $\boldsymbol{x}$ is calculated as:

$$I(\boldsymbol{x}) = \max_{V \in \tilde{\boldsymbol{x}}_i} (P(c|\boldsymbol{X}, i = V)) - \min_{V \in \tilde{\boldsymbol{x}}_i} (P(c|\boldsymbol{X}, i = V)) \tag{34}$$

In this base case, if the feature $\boldsymbol{x}$ significantly impacts the model predictions, the variation in probabilities will be large, which will be reflected in a high $I(\boldsymbol{x})$ value. If $\boldsymbol{x}$ has no impact, $I(\boldsymbol{x})$ will be close to zero.

Inductive Step. Now, assume that the perturbation method works for any dataset $\boldsymbol{X}$ with k features. That is, we can determine the importance of each of the k features using the described perturbation procedure.

We want to demonstrate that the method remains valid for a dataset $\boldsymbol{X}$ with $k+1$ features.

1. Consider the dataset $\boldsymbol{X}$ with $k+1$ features $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k, \boldsymbol{x}_{k+1}$.

2. For each feature $\boldsymbol{x}_i$ in $\boldsymbol{X}$, we generate the perturbation range $\tilde{\boldsymbol{x}}_i$ as before:

$$\tilde{\boldsymbol{x}}_i = \left\{ \boldsymbol{x} \mid \boldsymbol{x} = \min(\boldsymbol{X}_i) + k \cdot \delta,\ 0 \leq k < n \right\} \tag{35}$$

where $\delta = \frac{\max(\boldsymbol{X}_i) - \min(\boldsymbol{X}_i)}{n-1}$.

3. We modify each feature $\boldsymbol{x}_i$ in $\boldsymbol{X}$ to the value V within $\tilde{\boldsymbol{x}}_i$ and evaluate the model f to compute the probabilities:

$$P(c|\boldsymbol{X}, i = V) = \frac{1}{|\boldsymbol{X}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}} f_c(\boldsymbol{x}_1, \ldots, V, \ldots, \boldsymbol{x}_{k+1}) \tag{36}$$

4. We calculate the importance of each feature $\boldsymbol{x}_i$ as:

$$I(\boldsymbol{x}_i) = \max_{V \in \tilde{\boldsymbol{x}}_i} (P(c|\boldsymbol{X}, i = V)) - \min_{V \in \tilde{\boldsymbol{x}}_i} (P(c|\boldsymbol{X}, i = V)) \tag{37}$$

5. By the inductive hypothesis, we know that the perturbation method works for the first k features.

6. For the $k+1$-th feature $\boldsymbol{x}_{k+1}$, we generate its perturbation range $\tilde{\boldsymbol{x}}_{k+1}$ and evaluate its importance $I(\boldsymbol{x}_{k+1})$ using the same procedure.

Since we are following the same systematic procedure of perturbations and class probability evaluation, the value of $I(\boldsymbol{x}_{k+1})$ will correctly reflect the importance of $\boldsymbol{x}_{k+1}$ in the model, just as it does for the features $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$.

Therefore, we have proven by induction that the PeBEx method is effective in determining feature importance in any dataset with n features. The consistency of the method and the systematic way it measures the impact of perturbations ensure that it accurately captures the influences of each feature on the model predictions.

Probabilistic Proof by Construction:

We define $\boldsymbol{X}$ as a dataset with n features and m data points, and f as the predictive model trained on $\boldsymbol{X}$. We assume that $P(\boldsymbol{Y}|\boldsymbol{X})$ represents the true conditional probability distribution of the output $\boldsymbol{Y}$ given the input features $\boldsymbol{X}$.

Definition 4 (*Expected Model Output*)**.** For each feature $\boldsymbol{x}_i$, we generate a perturbed dataset $\boldsymbol{X}_i^V$ and calculate the expected output of the model as:

$$E[f(\boldsymbol{Y}|\boldsymbol{X}_i^V)] = \frac{1}{|\boldsymbol{X}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}} f(\boldsymbol{Y}|\boldsymbol{x}_1, \ldots, V, \ldots, \boldsymbol{x}_n) \tag{38}$$

Definition 5 (*Feature Importance*)**.** The importance of feature $\boldsymbol{x}_i$ is defined as the variance of the expected model output:

$$I(\boldsymbol{x}_i) = \sigma^2(\boldsymbol{x}_i) = \frac{1}{n} \sum_{V \in \tilde{\boldsymbol{x}}_i} \left(E[f(\boldsymbol{Y}|\boldsymbol{X}_i^V)] - \mu(\boldsymbol{x}_i) \right)^2 \tag{39}$$

where $\mu(\boldsymbol{x}_i)$ is the mean of the expected outputs.

We compare the feature importance values calculated using the PeBEx method and the constructed method, and conduct a statistical correlation analysis.

Definition 6 (*Pearson Correlation*)**.** The Pearson correlation coefficient, denoted as r, is calculated by:

$$r = \frac{\sum_{i=1}^{n} (I_{\text{PeBEx}}(\boldsymbol{x}_i) - \overline{I_{\text{PeBEx}}})(I(\boldsymbol{x}_i) - \overline{I})}{\sqrt{\sum_{i=1}^{n} (I_{\text{PeBEx}}(\boldsymbol{x}_i) - \overline{I_{\text{PeBEx}}})^2 \sum_{i=1}^{n} (I(\boldsymbol{x}_i) - \overline{I})^2}} \tag{40}$$

Definition 7. Under the assumption that the variations in feature importance measures between the two methods are linearly proportional without a constant term, i.e., $I(\boldsymbol{x}_i) = a \cdot I_{\text{PeBEx}}(\boldsymbol{x}_i)$ with $a > 0$, we can rewrite the calculation of r as follows:

$$r = \frac{a \cdot \sum_{i=1}^{n} (I_{\text{PeBEx}}(\boldsymbol{x}_i) - \overline{I_{\text{PeBEx}}})^2}{\sqrt{\sum_{i=1}^{n} (I_{\text{PeBEx}}(\boldsymbol{x}_i) - \overline{I_{\text{PeBEx}}})^2 \cdot a^2 \cdot \sum_{i=1}^{n} (I_{\text{PeBEx}}(\boldsymbol{x}_i) - \overline{I_{\text{PeBEx}}})^2}} = 1 \tag{41}$$

This equation demonstrates that if the variations are exactly proportional and linear, the Pearson correlation coefficient will be exactly 1, indicating perfect correlation.

4.2. Strategies to enhance computational speed

To efficiently compute the perturbation-based explanations, several strategies are implemented to optimize the computational speed. These strategies leverage advanced numerical and parallel processing techniques to reduce the overall runtime, especially for complex models such as DL models (e.g., MLPs).

Efficient array manipulation. The use of NumPy, a powerful library for numerical computations in Python, enables efficient creation and manipulation of large arrays. NumPy operations are highly optimized and executed at a low level, significantly reducing the computational overhead compared to standard Python loops (Harris et al., 2020).

Parallel processing. The joblib library is employed to parallelize the computation of feature importance scores. By utilizing the `Parallel` and `delayed` functions, the algorithm can distribute the perturbation and evaluation tasks across multiple CPU cores. This parallelization harnesses the power of multi-core processors, effectively decreasing the computation time (Joblib Development Team, 2024).

Vectorized operations. Vectorized operations within NumPy are utilized to minimize explicit loops. Vectorization allows batch processing of data, where operations are applied simultaneously to entire arrays rather than iterating over individual elements. This approach not only speeds up computations but also leads to more concise and readable code (Van Der Walt et al., 2011).

Uniform perturbation sampling. The perturbation values are generated using a uniform distribution, ensuring that the entire range of observed values for each feature is evenly sampled. This method reduces the risk of bias and ensures a comprehensive evaluation of feature importance.

Tiling and repeating arrays. The use of `np.tile` and `np.repeat` functions allows for efficient replication of the original dataset and perturbation values. These functions enable the creation of large, modified datasets necessary for evaluating the impact of feature perturbations on model predictions.

By integrating these strategies, the PeBEx algorithm cuts down on computational complexity. These optimizations are particularly beneficial in scenarios where real-time or near-real-time explanations are

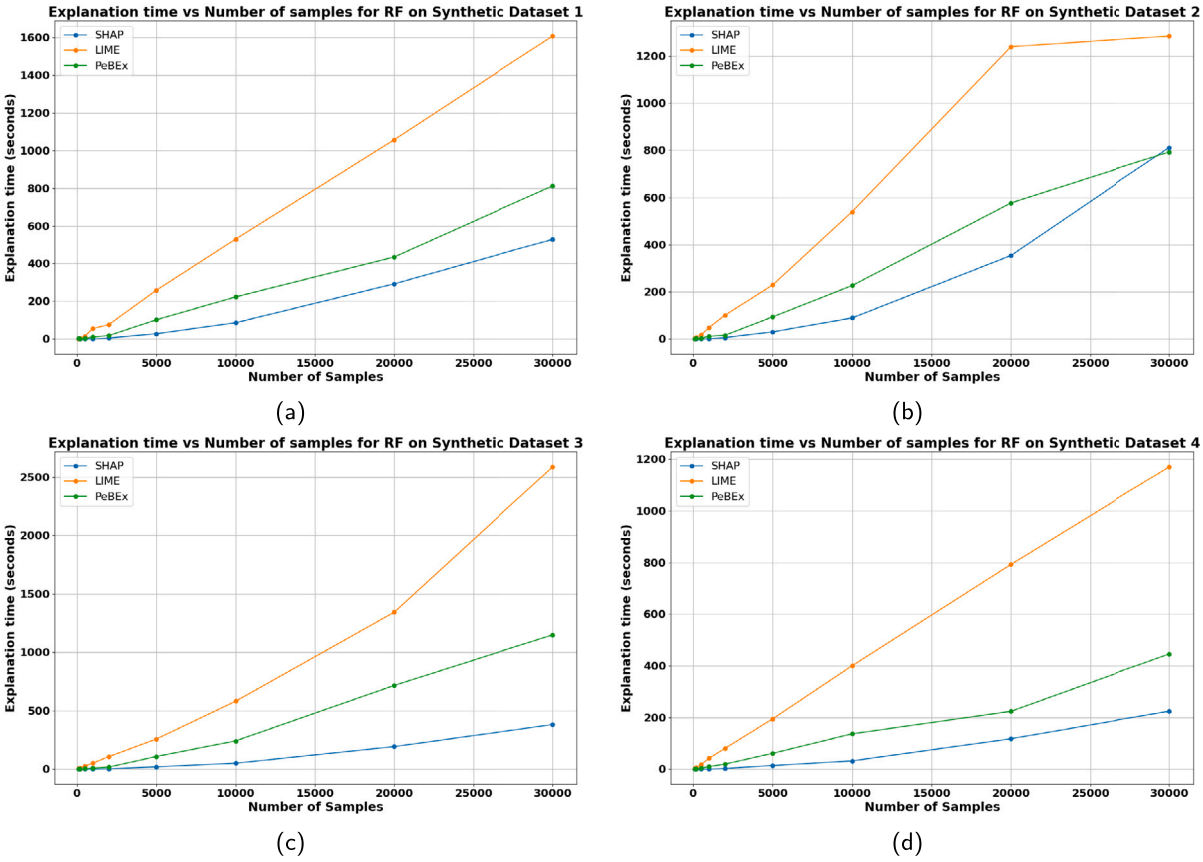

Fig. 2. Comparative analysis of explanation times for RF on Synthetic Data 1 using SHAP with TreeExplainer (blue), LIME with LimeTabularExplainer (orange), and PeBEx (green). The x-axis represents the number of samples and the y-axis represents the explanation time in seconds (both training and testing phases).

required, such as in critical applications in medicine, finance, or autonomous driving that involve DL models. The low computational run time allows the PeBEx method to quickly provide meaningful model explanations.

5. Experiments and results

In this section, we present a comprehensive evaluation of various XAI models using both synthetic and public datasets. The goal is to assess the performance and computational efficiency of LIME, SHAP, and PeBEx when applied to different ML models. The experiments are divided into two main subsections: experiments with synthetic data and comprehensive explainable ML model evaluation.

5.1. Experiments with synthetic data

In this subsection, we present a comprehensive evaluation of various XAI models using synthetic datasets. The goal is to assess the performance and computational efficiency of LIME, SHAP, and PeBEx when applied to RF and MLP models across multiple synthetic datasets.

5.1.1. Dataset overview and feature analysis

The comparative analysis of feature importances for RF on four synthetic datasets. These figures summarize the performance of three XAI models (LIME, SHAP, and PeBEx) in extracting feature importances and compare them to the intrinsic feature importances of the RF model. Specifically, Fig. 4 shows the feature importances for Dataset 1, Fig. 5 for Dataset 2, Fig. 6 for Dataset 3, and Fig. 7 for Dataset 4. Across all datasets, the feature importances identified by LIME, SHAP, and PeBEx exhibit a high degree of similarity to the feature importances of the RF model itself. This alignment suggests that the interpretability of these XAI models is consistent with the model intrinsic explainability, thereby validating their effectiveness in accurately identifying significant features in synthetic datasets.

5.1.2. Model training and execution time analysis

Analysis with RF. In the comparative analysis of explanation times for RF on four synthetic datasets, we present in Fig. 2 that showcase the performance of different XAI methods: SHAP (blue), LIME (orange), and PeBEx (green). The x-axis represents the number of samples and the y-axis represents the explanation time in seconds. Fig. 2(a) illustrates the explanation times for Synthetic Dataset 1. As the number of samples increases, LIME consistently shows the highest explanation time, followed by PeBEx and SHAP. PeBEx demonstrates a more efficient performance compared to LIME, particularly as the number of samples increases. Fig. 2(b) shows the explanation times for Synthetic Dataset 2. Similar to Dataset 1, LIME exhibits the highest explanation time, with PeBEx again outperforming LIME. SHAP remains the most efficient across all sample sizes, maintaining the lowest explanation time. Notably, between 20,000 and 25,000 samples, the lines for SHAP and PeBEx intersect, suggesting that PeBEx may offer better computational performance in this range. Fig. 2(c) presents the explanation times for Synthetic Dataset 3. The pattern remains consistent, with LIME showing the highest explanation time, followed by PeBEx and SHAP. PeBEx continues to provide a more computationally efficient alternative to LIME, especially as the sample size increases. Fig. 2(d) depicts the explanation times for Synthetic Dataset 4. LIME once again has the highest explanation time, while PeBEx demonstrates better efficiency compared to LIME. SHAP continues to be the most computationally efficient method, maintaining the lowest explanation time across all sample sizes.

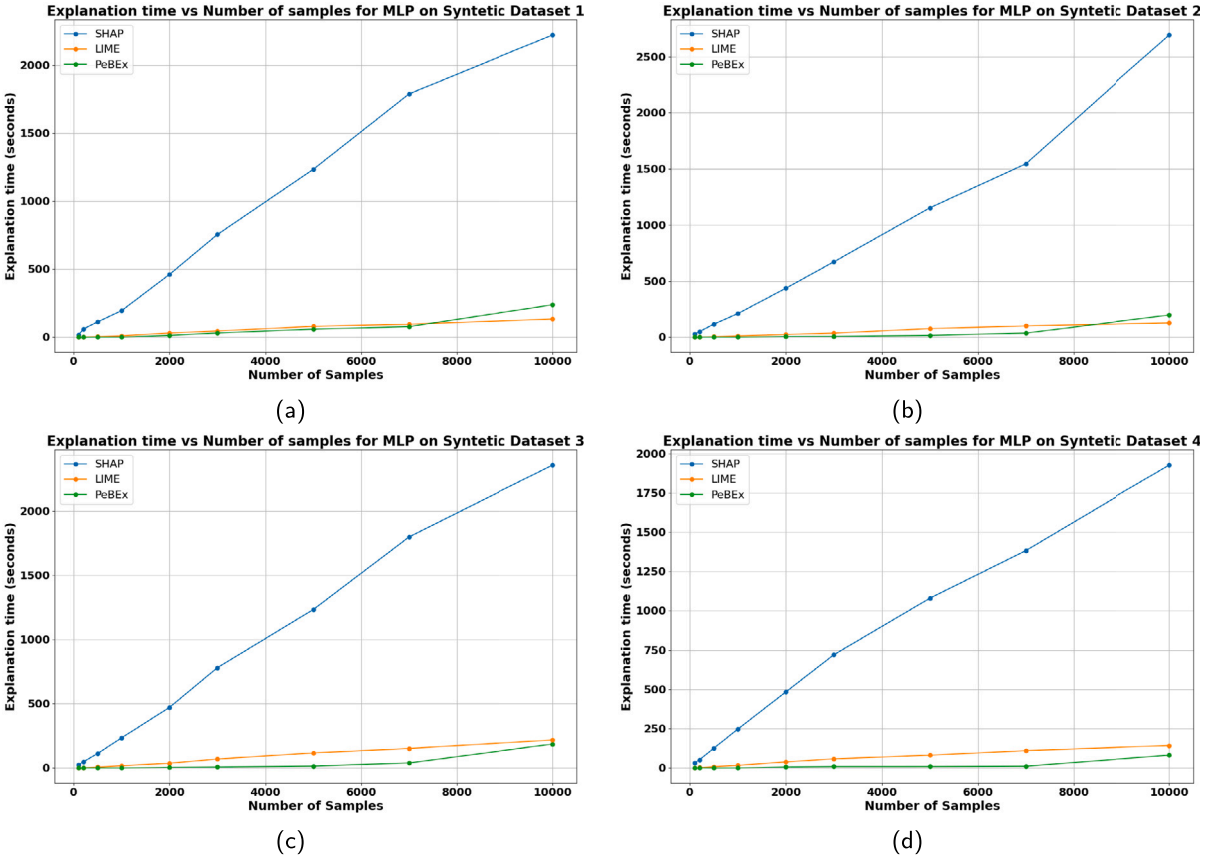

Fig. 3. Comparative analysis of explanation times for MLP on synthetic data using SHAP with TreeExplainer (blue), LIME with LimeTabularExplainer (orange), and PeBEx (green). The x-axis represents the number of samples and the y-axis represents the explanation time in seconds (both training and testing phases).

Analysis with MLP. In the comparative analysis of explanation times for MLP on four synthetic datasets, we present in Fig. 3 the performance of different XAI methods: SHAP (blue), LIME (orange), and PeBEx (green). The x-axis represents the number of samples and the y-axis represents the explanation time in seconds. Fig. 3(a) illustrates the explanation times for Synthetic Dataset 1. As the number of samples increases, SHAP consistently shows the highest explanation time, followed by LIME and PeBEx. In the case of the 10,000 samples, the lines cross and the performance of PeBEx and LIME is fairly even. Fig. 3(b) shows the explanation times for Synthetic Dataset 2. Similar to Dataset 1, SHAP exhibits the highest explanation time, with PeBEx again outperforming LIME. PeBEx remains the most efficient across all sample sizes, maintaining the lowest explanation time. Fig. 3(c) presents the explanation times for Synthetic Dataset 3. The pattern remains consistent, with SHAP showing the highest explanation time, followed by LIME and PeBEx. In the case of LIME, at around 7000 samples the performance of LIME becomes even in time concerning SHAP. PeBEx continues to provide a more computationally efficient alternative to LIME, especially as the sample size increases. Fig. 3(d) depicts the explanation times for Synthetic Dataset 4. SHAP once again has the highest explanation time, while PeBEx demonstrates better efficiency compared to LIME. PeBEx continues to be the most computationally efficient method, maintaining the lowest explanation time across all sample sizes.

The experiments on these synthetic datasets with known solutions have provided evidence that the SHAP model, particularly when using the recommended `KernelExplainer` method, is significantly slower. In this case, the novel model proposed in this paper shows better computational performance in terms of run time. In the remaining experiments with the synthesized data, the SHAP model shows the best computational performance with the exception of the DL model case. This series of experiments were then carried out on public, widely-applicable datasets with the objective of evaluating the behavior of the new PeBEx model, especially in the case where DL models (such as MLP) demonstrate better performances.

5.2. Comprehensive explainable ML model evaluation

To comprehensively evaluate the selected ML models across different datasets, we perform a detailed comparison of various performance metrics. This section aims to identify the best-performing models for each dataset and subsequently assess the interpretability and computational efficiency of different XAI models.

5.2.1. Model selection and hyperparameter tuning

The evaluation of various ML models across four different datasets is summarized in Table 3. For the Heart Disease dataset, the RF model achieves the best performance with an accuracy of 0.8202 ± 0.0634, recall of 0.8254 ± 0.1288, F1-Score of 0.6958 ± 0.0939, precision of 0.6157 ± 0.1176, specificity of 0.8185 ± 0.0831, and AUC of 0.9154 ± 0.0557. The hyperparameters for this RF model are 'n_estimators': 100, 'max_depth': 10, and 'min_samples_split': 2, as shown in Table 5.

In the case of the German Credit dataset, the RF model again outperforms others with an accuracy of 0.8301 ± 0.0282, recall of 0.8516 ± 0.0490, F1-Score of 0.7505 ± 0.0387, precision of 0.6737 ± 0.0529, specificity of 0.8211 ± 0.0400, and AUC of 0.9248 ± 0.0213. The optimal hyperparameters for this model are 'n_estimators': 100, 'max_depth': None, and 'min_samples_split': 2, as detailed in Table 6.

Table 3
This table presents the evaluation of several ML models (LR, SVM, RF, Gradient Boosting, XGBoost, LightGBM, MLP and PyTorch NN) across four datasets: Heart Disease, German Credit, Breast Cancer Wisconsin, and Car Evaluation. The metrics used for comparison include Accuracy, Recall, F1-Score, Precision, Specificity, and AUC. The reported values are the means and standard deviations obtained using the bootstrap method with 100 samples.

	Heart Disease					
Model	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
LR	0.7441 ± 0.0792	0.7172 ± 0.1543	0.5811 ± 0.1183	0.4984 ± 0.1194	0.7528 ± 0.0891	0.8240 ± 0.0762
SVM	0.8091 ± 0.0731	**0.8306 ± 0.1309**	0.6837 ± 0.1122	0.5936 ± 0.1294	0.8021 ± 0.0853	0.8966 ± 0.0634
Random Forest	**0.8202 ± 0.0634**	0.8254 ± 0.1288	**0.6958 ± 0.0939**	**0.6157 ± 0.1176**	**0.8185 ± 0.0831**	**0.9154 ± 0.0557**
Gradient Boosting	0.7959 ± 0.0761	0.8076 ± 0.1368	0.6626 ± 0.1194	0.5740 ± 0.1343	0.7921 ± 0.0914	0.8800 ± 0.0675
XGBoost	0.7925 ± 0.0694	0.8146 ± 0.1423	0.6605 ± 0.1049	0.5673 ± 0.1136	0.7835 ± 0.0892	0.8793 ± 0.0601
LightGBM	0.7968 ± 0.0706	0.8296 ± 0.1274	0.6705 ± 0.1053	0.5731 ± 0.1207	0.7858 ± 0.0858	0.8597 ± 0.0796
MLP	0.7975 ± 0.0682	0.8195 ± 0.1303	0.6672 ± 0.1058	0.5738 ± 0.1238	0.7895 ± 0.0810	0.8603 ± 0.0722
PyTorch NN	0.7268 ± 0.0742	0.7101 ± 0.1585	0.5607 ± 0.1065	0.4754 ± 0.1054	0.7322 ± 0.0957	0.8074 ± 0.0853
	German Credit					
Model	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
LR	0.7258 ± 0.0357	0.7305 ± 0.0576	0.6154 ± 0.0426	0.5338 ± 0.0472	0.7238 ± 0.0480	0.7955 ± 0.0304
SVM	0.8027 ± 0.0302	0.8260 ± 0.0545	0.7152 ± 0.0420	0.6325 ± 0.0481	0.7928 ± 0.0372	0.8807 ± 0.0304
Random Forest	**0.8301 ± 0.0282**	**0.8516 ± 0.0490**	**0.7505 ± 0.0387**	**0.6737 ± 0.0529**	**0.8211 ± 0.0400**	**0.9248 ± 0.0213**
Gradient Boosting	0.8218 ± 0.0295	0.8468 ± 0.0535	0.7403 ± 0.0409	0.6606 ± 0.0541	0.8112 ± 0.0415	0.9060 ± 0.0240
XGBoost	0.8160 ± 0.0280	0.8500 ± 0.0469	0.7350 ± 0.0366	0.6498 ± 0.0494	0.8014 ± 0.0398	0.8918 ± 0.0252
LightGBM	0.8172 ± 0.0280	0.8396 ± 0.0479	0.7338 ± 0.0386	0.6540 ± 0.0507	0.8075 ± 0.0384	0.8882 ± 0.0250
MLP	0.8142 ± 0.0319	0.8335 ± 0.0502	0.7294 ± 0.0429	0.6504 ± 0.0524	0.8060 ± 0.0397	0.8893 ± 0.0279
PyTorch NN	0.7265 ± 0.0314	0.7477 ± 0.0663	0.6209 ± 0.0392	0.5333 ± 0.0399	0.7175 ± 0.0449	0.8051 ± 0.0348
	Breast Cancer Wisconsin					
Model	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
LR	0.9795 ± 0.0138	**0.9871 ± 0.0133**	0.9836 ± 0.0110	0.9804 ± 0.0164	0.9667 ± 0.0284	0.9956 ± 0.0052
SVM	0.9778 ± 0.0154	0.9844 ± 0.0167	0.9823 ± 0.0122	0.9805 ± 0.0162	0.9669 ± 0.0281	0.9960 ± 0.0049
Random Forest	0.9754 ± 0.0172	0.9751 ± 0.0234	0.9802 ± 0.0141	0.9857 ± 0.0151	0.9757 ± 0.0267	0.9962 ± 0.0059
Gradient Boosting	0.9713 ± 0.0185	0.9699 ± 0.0243	0.9768 ± 0.0150	0.9843 ± 0.0171	0.9738 ± 0.0295	0.9954 ± 0.0063
XGBoost	0.8160 ± 0.0280	0.8500 ± 0.0469	0.7350 ± 0.0366	0.6498 ± 0.0494	0.8014 ± 0.0398	0.8918 ± 0.0252
LightGBM	0.9778 ± 0.0161	0.9810 ± 0.0189	0.9822 ± 0.0129	0.9837 ± 0.0161	0.9727 ± 0.0272	0.9962 ± 0.0052
MLP	**0.9825 ± 0.0121**	0.9837 ± 0.0174	**0.9860 ± 0.0098**	**0.9886 ± 0.0134**	**0.9804 ± 0.0241**	**0.9975 ± 0.0036**
PyTorch NN	0.9768 ± 0.0154	0.9772 ± 0.0197	0.9813 ± 0.0125	0.9857 ± 0.0137	0.9761 ± 0.0232	0.9964 ± 0.0046
	Car Evaluation					
Model	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
LR	0.6747 ± 0.0297	0.6811 ± 0.0483	0.5015 ± 0.0342	0.3977 ± 0.0318	0.6727 ± 0.0359	0.7656 ± 0.0277
SVM	0.9725 ± 0.0141	0.9949 ± 0.0095	0.9460 ± 0.0265	0.9028 ± 0.0461	0.9655 ± 0.0181	0.9969 ± 0.0033
Random Forest	0.9605 ± 0.0163	0.9954 ± 0.0111	0.9241 ± 0.0299	0.8638 ± 0.0505	0.9495 ± 0.0210	0.9979 ± 0.0019
Gradient Boosting	0.9826 ± 0.0102	0.9950 ± 0.0091	0.9651 ± 0.0200	0.9374 ± 0.0340	0.9786 ± 0.0123	0.9984 ± 0.0027
XGBoost	0.9790 ± 0.0110	0.9955 ± 0.0103	0.9581 ± 0.0214	0.9243 ± 0.0394	0.9737 ± 0.0145	0.9983 ± 0.0017
LightGBM	**0.9842 ± 0.0098**	**0.9977 ± 0.0059**	**0.9683 ± 0.0189**	**0.9412 ± 0.0353**	**0.9799 ± 0.0129**	**0.9989 ± 0.0017**
MLP	0.9659 ± 0.0180	0.9867 ± 0.0138	0.9334 ± 0.0322	0.8875 ± 0.0553	0.9593 ± 0.0236	0.9950 ± 0.0051
PyTorch NN	0.7485 ± 0.0397	0.8756 ± 0.0587	0.6267 ± 0.0485	0.4903 ± 0.0535	0.7084 ± 0.0513	0.8669 ± 0.0346
	Santander Customer Satisfaction					
Model	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
LR	0.6888 ± 0.0102	0.7615 ± 0.0195	0.1616 ± 0.0051	0.0904 ± 0.0031	0.6858 ± 0.0108	0.7975 ± 0.0087
SVM	0.6877 ± 0.0106	0.7856 ± 0.0187	0.1654 ± 0.0058	0.0924 ± 0.0036	0.6837 ± 0.0113	0.8177 ± 0.0091
Random Forest	0.7555 ± 0.0069	0.7826 ± 0.0179	0.2013 ± 0.0057	0.1155 ± 0.0037	0.7544 ± 0.0073	0.8370 ± 0.0086
Gradient Boosting	0.7791 ± 0.0059	0.8409 ± 0.0150	0.2307 ± 0.0069	0.1337 ± 0.0045	0.7766 ± 0.0063	0.8773 ± 0.0073
XGBoost	0.7774 ± 0.0070	0.8242 ± 0.0222	0.2257 ± 0.0056	0.1308 ± 0.0037	0.7755 ± 0.0079	0.8704 ± 0.0059
LightGBM	0.7806 ± 0.0054	0.8424 ± 0.0193	0.2321 ± 0.0067	0.1346 ± 0.0043	0.7781 ± 0.0058	0.8788 ± 0.0094
MLP	**0.8085 ± 0.0068**	**0.8568 ± 0.0172**	**0.2606 ± 0.0094**	**0.1537 ± 0.0064**	**0.8065 ± 0.0071**	**0.9034 ± 0.0066**
PyTorch NN	0.7770 ± 0.0120	0.8114 ± 0.0235	0.2228 ± 0.0054	0.1292 ± 0.0032	0.7756 ± 0.0130	0.8388 ± 0.0114

For the Breast Cancer Wisconsin dataset, the best-performing model is the MLP with an accuracy of 0.9825 ± 0.0121, recall of 0.9837 ± 0.0174, F1-Score of 0.9860 ± 0.0098, precision of 0.9886 ± 0.0134, specificity of 0.9804 ± 0.0241, and AUC of 0.9975 ± 0.0036. The hyperparameters for this MLP model are 'hidden_layer_sizes': (50, 50), 'activation': 'tanh', and 'solver': 'adam', as shown in Table 7.

For the Car Evaluation dataset, the LightGBM model achieves the highest performance with an accuracy of 0.9842 ± 0.0098, recall of 0.9977 ± 0.0059, F1-Score of 0.9683 ± 0.0189, precision of 0.9412 ± 0.0353, specificity of 0.9799 ± 0.0129, and AUC of 0.9989 ± 0.0017. The hyperparameters for this LightGBM model are 'n_estimators': 200, 'max_depth': 7, and 'learning_rate': 0.1, as specified in Table 8.

For the Santander Customer Satisfaction dataset, the MLP model delivers the best performance, achieving an accuracy of 0.8085 ± 0.0068, recall of 0.8568 ± 0.0172, F1-Score of 0.2606 ± 0.0094, precision of 0.1537 ± 0.0064, specificity of 0.8065 ± 0.0071, and AUC of 0.9034 ± 0.0066. These results underscore the capacity of the MLP model to handle the dataset imbalance effectively, with the hyperparameters 'hidden_layer_sizes': (100,), 'activation': 'relu', and 'solver': 'adam', as detailed in Table 9. Additionally, it is important to note that the Santander dataset contains a significantly larger number of features and rows compared to the other datasets, which makes it better suited for DL models like MLP. These models are more effective in handling large and complex datasets, as their architecture allows them to capture intricate patterns in high-dimensional data.

The analysis of these results reveals several insights. For the Heart Disease and German Credit datasets, RF consistently outperforms other models in most evaluation metrics. This suggests that tree ensemble methods are highly effective for these datasets. In contrast, for the Breast Cancer Wisconsin dataset and Santander Customer Satisfaction

Table 4
This table summarizes the performance of three XAI models (LIME, SHAP, and PeBEx) across four different datasets. The metrics used for evaluation include Fidelity, Comprehensibility, Consistency, Stability, Local Accuracy, Similarity, Robustness, and Computation Time. The results are averaged using the bootstrap method with 100 samples to obtain the mean and standard deviation.

		Heart Disease								
XAI Model	Explainer Type	Fidelity (↑)	Comprehensibility (↓)	Consistency	FCC Score (↑)	Stability (↓)	Local Accuracy (↑)	Similarity	Robustness (↓)	Time (↓)
LIME	LimeTabularExplainer	0.36 ± 0.00	10.00 ± 0.00	−0.89 ± 0.05	0.39 ± 0.02	0.00 ± 0.00	0.06 ± 0.01	−0.90 ± 0.04	0.07 ± 0.02	13.27 ± 0.63
SHAP	TreeExplainer	**1.00 ± 0.00**	**2.00 ± 0.00**	−0.97 ± 0.00	**0.68 ± 0.00**	0.01 ± 0.00	**1.00 ± 0.00**	−0.96 ± 0.00	0.02 ± 0.01	**0.04 ± 0.00**
PeBEx	N/A[a]	0.36 ± 0.00	30.00 ± 0.00	**0.00 ± 0.00**	0.45 ± 0.00	**0.00 ± 0.00**	0.00 ± 0.00	**0.00 ± 0.00**	**0.00 ± 0.00**	4.42 ± 0.54
		German Credit								
XAI Model	Explainer Type	Fidelity (↑)	Comprehensibility (↓)	Consistency	FCC Score (↑)	Stability (↓)	Local Accuracy (↑)	Similarity	Robustness (↓)	Time (↓)
LIME	LimeTabularExplainer	0.81 ± 0.00	10.00 ± 0.00	0.68 ± 0.19	0.56 ± 0.06	0.17 ± 0.04	**1.00 ± 0.00**	0.04 ± 0.14	0.17 ± 0.04	20.71 ± 0.72
SHAP	TreeExplainer	**0.99 ± 0.00**	**2.00 ± 0.00**	0.50 ± 0.00	**0.83 ± 0.06**	0.01 ± 0.00	**1.00 ± 0.00**	0.43 ± 0.00	0.04 ± 0.02	**1.48 ± 0.04**
PeBEx	N/A[a]	0.81 ± 0.00	20.00 ± 0.00	**0.00 ± 0.00**	0.60 ± 0.00	**0.00 ± 0.00**	**1.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	5.59 ± 0.58
		Breast Cancer Wisconsin								
XAI Model	Explainer Type	Fidelity (↑)	Comprehensibility (↓)	Consistency	FCC Score (↑)	Stability (↓)	Local Accuracy (↑)	Similarity	Robustness (↓)	Time (↓)
LIME	LimeTabularExplainer	0.38 ± 0.00	**10.00 ± 0.00**	−0.11 ± 0.20	**0.76 ± 0.07**	0.22 ± 0.07	0.00 ± 0.00	−0.13 ± 0.13	0.21 ± 0.05	22.32 ± 1.23
SHAP	KernelExplainer	**1.00 ± 0.00**	114.00 ± 0.00	−0.69 ± 0.03	0.44 ± 0.01	0.03 ± 0.00	1.00 ± 0.00	−0.68 ± 0.02	0.05 ± 0.01	820.76 ± 10.00
PeBEx	N/A[a]	0.38 ± 0.00	30.00 ± 0.00	**0.00 ± 0.00**	0.73 ± 0.00	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**3.86 ± 0.80**
		Car Evaluation								
XAI Model	Explainer Type	Fidelity (↑)	Comprehensibility (↓)	Consistency	FCC Score (↑)	Stability (↓)	Local Accuracy (↑)	Similarity	Robustness (↓)	Time (↓)
LIME	LimeTabularExplainer	0.80 ± 0.00	6.00 ± 0.00	0.67 ± 0.02	0.38 ± 0.01	0.74 ± 0.26	0.00 ± 0.00	0.99 ± 0.00	0.66 ± 0.34	17.14 ± 1.01
SHAP	TreeExplainer	**1.00 ± 0.00**	**2.00 ± 0.00**	−0.41 ± 0.86	0.01 ± 0.00	**0.00 ± 0.00**	**1.00 ± 0.00**	0.03 ± 0.00	0.00 ± 0.00	**0.04 ± 0.00**
PeBEx	N/A[a]	0.80 ± 0.00	6.00 ± 0.00	**0.00 ± 0.00**	**0.60 ± 0.00**	**0.00 ± 0.00**	0.00 ± 0.00	**0.00 ± 0.00**	**0.00 ± 0.00**	6.76 ± 0.62
		Santander Customer Satisfaction[b]								
XAI Model	Explainer Type	Fidelity (↑)	Comprehensibility (↓)	Consistency	FCC Score (↑)	Stability (↓)	Local Accuracy (↑)	Similarity	Robustness (↓)	Time (↓)
LIME	LimeTabularExplainer	**1.00 ± 0.00**	10.00 ± 0.00	0.98 ± 0.03	0.60 ± 0.01	1.16 ± 0.16	**1.00 ± 0.00**	0.99 ± 0.00	1.22 ± 0.17	**4.35 ± 0.01**
SHAP	TreeExplainer	**1.00 ± 0.00**	**3.00 ± 0.00**	0.60 ± 0.04	**0.80 ± 0.01**	0.02 ± 0.00	**1.00 ± 0.00**	−0.25 ± 0.03	0.02 ± 0.00	2557.86 ± 3.29
PeBEx	N/A[a]	**1.00 ± 0.00**	37.00 ± 0.00	**0.00 ± 0.00**	0.67 ± 0.00	**0.00 ± 0.00**	**1.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	5.99 ± 3.58

[a] PeBEx does not have a defined explainer type.

[b] For the Santander Customer Satisfaction dataset, the evaluation of XAI models was conducted using local interpretability methods due to computational limitations. This approach allows for a more feasible interpretation of individual predictions without requiring the extensive computational resources needed for global interpretability.

dataset, the MLP model demonstrates superior performance, indicating the potential of DL models in handling complex and high-dimensional data. Similarly, for the Car Evaluation dataset, LightGBM shows superior performance, which can be attributed to its ability to handle large datasets efficiently and its gradient boosting framework that improves predictive accuracy.

This comprehensive comparison across multiple datasets justifies the selection of the best model for each specific case. Notably, in cases where the best model is based on DL, such as the MLP for the Breast Cancer dataset, interpretability becomes a critical factor due to the application in healthcare. In such scenarios, the proposed PeBEx model can be a valuable alternative to consider for quickly yielding interpretability without significantly compromising performance. After selecting the best model for each dataset, the next step would be to compare different XAI models to evaluate the quality of their explanations and the computational time required, thus ensuring a balanced trade-off between accuracy and interpretability.

5.2.2. Integration with XAI methods and performance comparison

Having selected the best model for each dataset, the next step is to compare the different XAI models to evaluate their interpretability and computational performance. Table 4 summarizes the performance of three XAI models (LIME, SHAP, and PeBEx) across four real datasets. The metrics used for evaluation include Fidelity, Comprehensibility, Consistency, Stability, Local Accuracy, Similarity, Robustness, and Computation Time. The results are averaged using the bootstrap method with 100 samples to obtain the mean and standard deviation.

Table 4 provides a detailed comparison of LIME, SHAP, and PeBEx. SHAP generally shows better performance in applications where DL models are not used, offering good computational efficiency and interpretability. However, for DL models like MLP where SHAP employs the `KernelExplainer`, the computational time increases substantially. In these cases, PeBEx demonstrates its strength by providing faster computational times and robust interpretability, making it a suitable alternative.

Limitations. In this study, we encountered several computational challenges, particularly when applying SHAP to DL models such as MLP in the Santander Customer Satisfaction dataset. Due to the large number of features and samples in this dataset, memory limitations significantly impacted the feasibility of performing global interpretability with SHAP `KernelExplainer`. The Santander dataset has a much larger number of features and rows compared to the other datasets, exacerbating these computational issues. Despite attempts to mitigate this issue by reducing the number of background samples through methods like `shap.sample()` or `shap.kmeans()`, the computational burden remained high. Given these constraints, we decided to focus on local interpretability methods to evaluate the XAI models. Local interpretability allows for an analysis of individual predictions, thus significantly reducing the memory requirements without compromising the explanatory power for specific instances. While this decision enabled us to complete the experiments under the available computational resources, it introduced a limitation in our study: global interpretability could not be fully evaluated for DL models like MLP due to memory constraints. The use of local interpretability restricted the analysis to specific predictions rather than providing a holistic view of the model behavior across all data points.

6. Discussion, limitations and future works

The experiments presented in this study evaluated the performance and computational efficiency of three XAI models: LIME, SHAP, and the newly proposed PeBEx, using both synthetic and public datasets. Our

findings highlight the strengths and weaknesses of these methods in various contexts, particularly where computational efficiency is critical.

6.1. Comparative analysis of XAI models

Our experiments with public data demonstrated that PeBEx consistently outperformed LIME in terms of computational efficiency while maintaining comparable interpretability. This finding aligns with the literature, which underscores the computational inefficiencies associated with LIME and SHAP (Belaid et al., 2022; Nguyen et al., 2021). Specifically, PeBEx showed a more efficient and consistent performance as the number of samples increased, positioning it as a faster and more stable alternative XAI model in the modern era of big data and high-dimensionality.

For the RF and LightGBM model, SHAP generally exhibited the lowest explanation times across all public datasets, validating its efficiency in non-DL models as reported in prior studies (Lundberg et al., 2020; Covert et al., 2020). However, SHAP with `KernelExplainer` greatly underperformed when applied to the MLP model, corroborating findings that SHAP can be computationally intensive, particularly for DL models (Ancona et al., 2019; Wang et al., 2021). In contrast, PeBEx demonstrated considerable potential in maintaining both interpretability and computational efficiency in such scenarios. Given its superior performance with DL models in our experiments, PeBEx may indeed serve as a robust alternative for DL applications, where existing methods like SHAP struggle with computational overhead. This robustness is particularly vital in environments where quick, interpretable results are necessary, such as in real-time decision-making systems or in contexts involving large-scale, high-dimensional datasets. The need for further empirical validation in diverse DL architectures remains, but these initial results position PeBEx as a promising tool for enhancing explainability in ML.

SHAP, on the other hand, offers a more consistent theoretical foundation through the use of Shapley values, which provides a fair distribution of feature importance. However, SHAP is also computationally demanding, especially when employing the `KernelExplainer`, which is often necessary for DL models. This leads to significant delays and high computational costs, making SHAP less suitable for applications involving DL models (Ancona et al., 2019; Wang et al., 2021).

6.2. Challenges in current XAI methods and the advantages of PeBEx

PeBEx addresses the computational challenges surrounding XAI methods by introducing efficient array manipulation, parallel processing, vectorized operations, and uniform perturbation sampling which serve to reduce the computational overhead associated with generating good explanations, something that SHAP and LIME struggle with at scale. This is reflected in the experiments, where PeBEx consistently provides interpretations that were as good or better than SHAP and LIME, while consistently reducing computation times across experiments. Moreover, PeBEx maintains a straightforward probabilistic interpretation similar to the beta coefficients in linear regression, which aids in its accessibility and practical utility. These improvements align with recent literature emphasizing the need for computationally efficient and practical XAI methods in fields like finance, cybersecurity, education, and autonomous systems (Lyuu, 2002; Alexandrov et al., 2011; Ryman-Tubb et al., 2018; Chalé et al., 2020; Jean-Quartier et al., 2023; Gómez-Talal et al., 2024).

PeBEx yielded results similar to those of LIME and SHAP for highly important features, but for the features with low true feature importance, PeBEx significantly understated the feature importance in comparison to the other models. Unlike LIME and SHAP, PeBEx downplays the importance of non-informative features that do not significantly contribute to model interpretability. As this behavior was observed consistently across our experimental evaluations, it may suggest that PeBEx has the capacity to be leveraged for feature selection.

In this study, we encountered several computational challenges. Due to the large number of features and samples in the Santander Customer Satisfaction dataset, memory limitations significantly impacted the feasibility of performing global interpretability with `KernelExplainer` (SHAP model). Moving forward, overcoming this limitation would require either adopting more efficient methods for global interpretability or accessing larger computational resources. A possible approach could involve further optimizing the background dataset selection or exploring alternative XAI methods that are less computationally expensive for high-dimensional, large-scale datasets like Santander. Finally, while the focus on local interpretability provided valuable insights into individual model predictions, it is important to acknowledge that the computational constraints prevented a thorough global interpretability analysis. This limitation highlights the trade-off between model complexity, dataset size, and the computational feasibility of interpretability methods when applied to DL models.

Furthermore, there has been a call for the development of falsifiable interpretability research that focuses on incorporating means of testing clear, specific, and falsifiable hypotheses (Leavitt and Morcos, 2020). Despite XAI models' ability to yield quantifiable metrics for feature importance, there is no way to perform statistical tests on the importance metrics.

One of the main concerns with XAI models is the lack of stability (Ng et al., 2022; Pawlicki, 2023; Amparore et al., 2021), meaning that the explanation results vary across repeated applications to the same instance, x. As Lipton (2018) suggested, XAI methods may not converge to a unique solution. Future research could therefore investigate the stability of PeBEx explanations. It could also quantify the degree or the extent to which explanations given by PeBEx correlate with those of other XAI methods.

6.3. Limitations and future works

While this study demonstrates the computational efficiency and potential advantages of PeBEx in comparison to existing XAI methods, it is important to acknowledge certain limitations. First, the applicability of PeBEx has primarily been tested on a limited range of ML models and datasets, which may not fully represent the diversity of real-world scenarios. Additionally, the method performance in highly complex and dynamic environments, such as those involving online learning or real-time data streams, has yet to be evaluated.

The decision to focus on binary classification models in this study was driven by the need for a controlled experimental environment, which allowed for a straightforward comparison of the computational efficiency and interpretability of XAI models. Binary classification has been used to test and validate XAI methodologies as it provides a clear and manageable framework for comparison and evaluation (Amiri et al., 2020; Lin et al., 2021a; Bommer et al., 2024). However, we recognize that real-world applications often involve more complex tasks, including multiclass classification, regression, and the analysis of various data types such as time series, text, and images. Consequently, future research will explore the extension of PeBEx to these domains, aiming to evaluate its performance and adaptability across a broader range of ML problems. This will ensure that PeBEx can be effectively applied in diverse and complex scenarios, thereby enhancing its utility and model agnosticism.

In this study, we observed that PeBEx tended to understate the importance of less informative features compared to LIME and SHAP. While this behavior can lead to more concise and streamlined explanations, it also presents the risk of omitting potentially relevant features. This observation raises an important question about whether a feature is relevant within the context of the problem. For instance, in scenarios where concise explanations are preferred, the tendency of PeBEx to downplay less informative features could be advantageous. However, in domains where retaining subtle but potentially critical information is essential, such as healthcare or finance, this behavior

might require further investigation to ensure that important insights are not overlooked. Future research should aim to explore the impact of this characteristic of PeBEx across various use cases to assess the true nature of this behavior.

Despite the myriad of XAI methods in existence, there has yet to be a standardized approach for development and implementation of model interpretability methodologies. To address this gap, future research could create a standard framework for XAI development and establish generally accepted practices. This could address issues faced by industry in relation to auditability, degree of interpretability, end-user safety, and controllability.

Similarly, there has yet to be a consensus on the proper means of XAI evaluation and comparison. While we used various metrics to evaluate the quality of the interpretability methods, future research could also develop a universal framework or set of stable, tested metrics for evaluating and comparing XAI methods. Another challenge with respect to the evaluation metrics lies in the scale-dependency of current evaluation metrics. Without scale-independent metrics, comparisons across methodologies or datasets may not be meaningful.

Conducting user studies to evaluate the interpretability and usability of PeBEx among different types of end-users, such as data scientists, domain experts, and non-technical stakeholders would provide valuable feedback for further refinement of the model. By addressing these future directions, the development and application of XAI can contribute to the broader goal of making ML models more transparent, interpretable, auditable, and trustworthy.

Building on the findings of this study, several avenues for future research are identified. Future work could explore the application of PeBEx to a broader range of ML models, including other DL architectures and ensemble methods. This would provide a more comprehensive evaluation of its versatility and efficiency. Further optimization of the PeBEx algorithm could be investigated to enhance its scalability for extremely large datasets. Techniques such as parallel computing and hardware acceleration could be employed to reduce computation times further.

Additionally, applying PeBEx in domain-specific contexts, such as healthcare, finance, and cybersecurity, could provide insights into its practical utility and effectiveness in real-world scenarios. These studies could involve collaboration with domain experts to fine-tune the algorithm for specific applications. As the field of XAI continues to evolve, future research could compare PeBEx with newly developed XAI techniques to ensure its competitiveness and relevance. This includes benchmarking and testing PeBEx against the latest advancements in explainability methods.

Interesting avenues of future work should also consider the explainability of PeBEx at a local level. Future research could use influence functions (Koh and Liang, 2017) or Cook's distance metrics (Chatterjee and Hadi, 1986) to understand how individual points influence the overall model prediction. Doing so would allow for local feature selection with PeBEx by considering the impact of perturbations along different model paths for a given test point. Additionally, Eq. (31), which calculates feature importance based on probabilities, could enhance the precision of PeBEx, leading to more accurate and efficient interpretations.

7. Conclusion

This study introduced PeBEx, a novel, straightforward XAI methodology based on perturbations. After a comprehensive evaluation of model performance and computational efficiency of three XAI models (LIME, SHAP, and PeBEx) using both synthetic and public datasets, we found that PeBEx demonstrated similar or superior computational efficiency compared to other XAI models, particularly with increasing dataset sizes. This is particularly valuable in real-time applications that require the efficient computation of model interpretations.

PeBEx differentiates itself from other perturbation-based methods such as LIME and SHAP in several key aspects. LIME focuses on local explanations by generating perturbed samples around a specific instance and fitting a local surrogate model and SHAP uses a theoretically grounded approach to calculate Shapley values by considering all possible feature combinations. PeBEx, on the other hand, leverages a more uniform and global perturbation strategy. This strategy involves systematically altering the input features across the entire dataset to observe the impact on predicted probabilities, making PeBEx not only computationally efficient but also capable of providing global interpretations. This is particularly advantageous in scenarios where understanding the overall behavior of the model is crucial, as opposed to explanations limited to specific instances. Additionally, PeBEx avoids the computational overhead associated with fitting additional models, as seen in LIME, and reduces the complexity of the explanation process compared to SHAP. These distinctions position PeBEx as a highly effective tool for scalable and interpretable model explanations, especially in the context of large datasets and complex ML models.

CRediT authorship contribution statement

Ismael Gómez-Talal: Writing – original draft, Visualization, Software, Methodology, Investigation, Conceptualization. **Mana Azizsoltani:** Writing – original draft, Validation, Investigation, Conceptualization. **Luis Bote-Curiel:** Writing – review & editing, Validation. **José Luis Rojo-Álvarez:** Writing – review & editing, Validation, Supervision. **Ashok Singh:** Writing – review & editing, Supervision, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Partially funded by the Autonomous Community of Madrid (ELLIS Madrid Node). Also partially supported by project PID2022-140786NB-C32/AEI/10.13039/501100011033 (LATENTIA) from the Spanish Ministry of Science and Innovation, Spain. This work was supported by the CyberFold project, funded by the European Union through the NextGenerationEU instrument (Recovery, Transformation, and Resilience Plan), and managed by Instituto Nacional de Ciberseguridad de España (INCIBE), under reference number ETD202300129.

Appendix

A.1. Comparative analysis of feature importance methods

Analysis of Syntetic Data 1. In the comparative analysis of feature importances for RF on Synthetic Data 1, we present four subfigures showcasing the feature importances as derived from different interpretability methods. Fig. 4(a) illustrates the intrinsic feature importances obtained directly from the RF model. Feature 12 is the most significant, followed by Features 3, 9, 19, and 2. The remaining features exhibit relatively low importance, reflecting the model reliance on a few key features. Fig. 4(b) depicts feature importances using the PeBEx method, where we notice a reduction in the importance scores for less significant features compared to the intrinsic RF importances. PeBEx allocates more precise importance to features, highlighting the truly influential ones while minimizing the scores for less impactful features. Fig. 4(c) shows the results of LIME, which identifies a similar set of important features but assigns different importance scores, with a broader distribution of importance across the features compared to

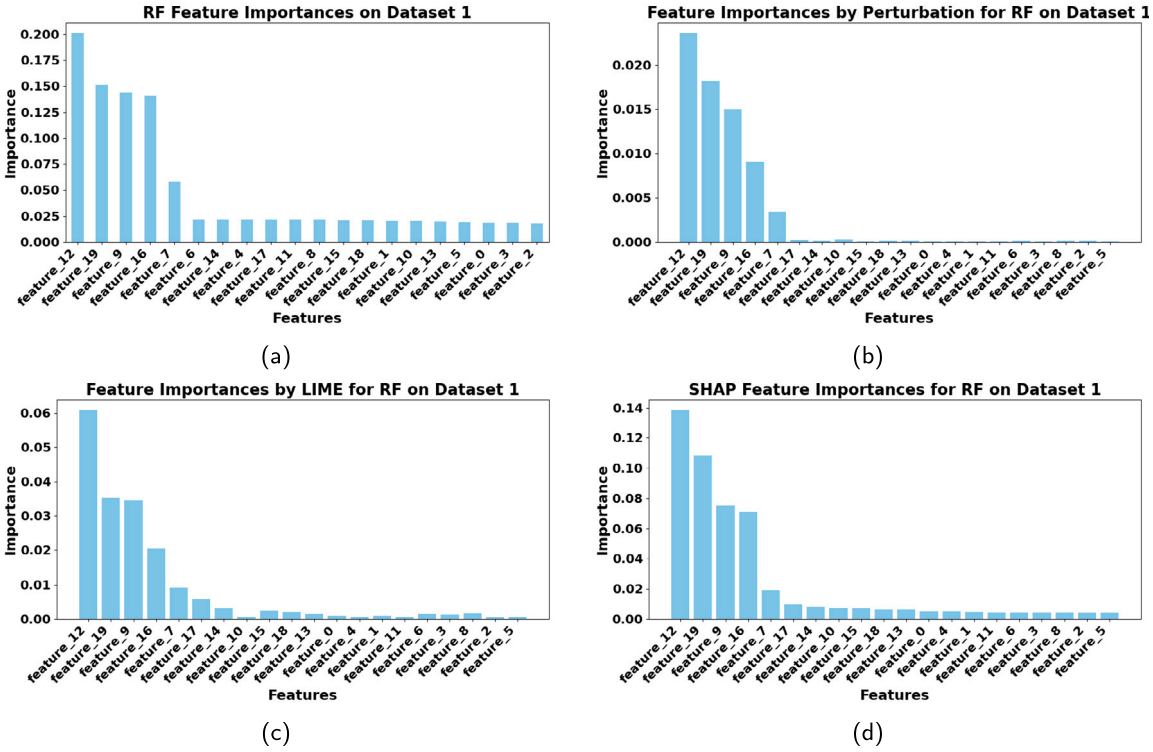

Fig. 4. Comparative analysis of feature importances for RF on Synthetic Data 1 using RF feature importances, PeBEx, LIME, and SHAP.

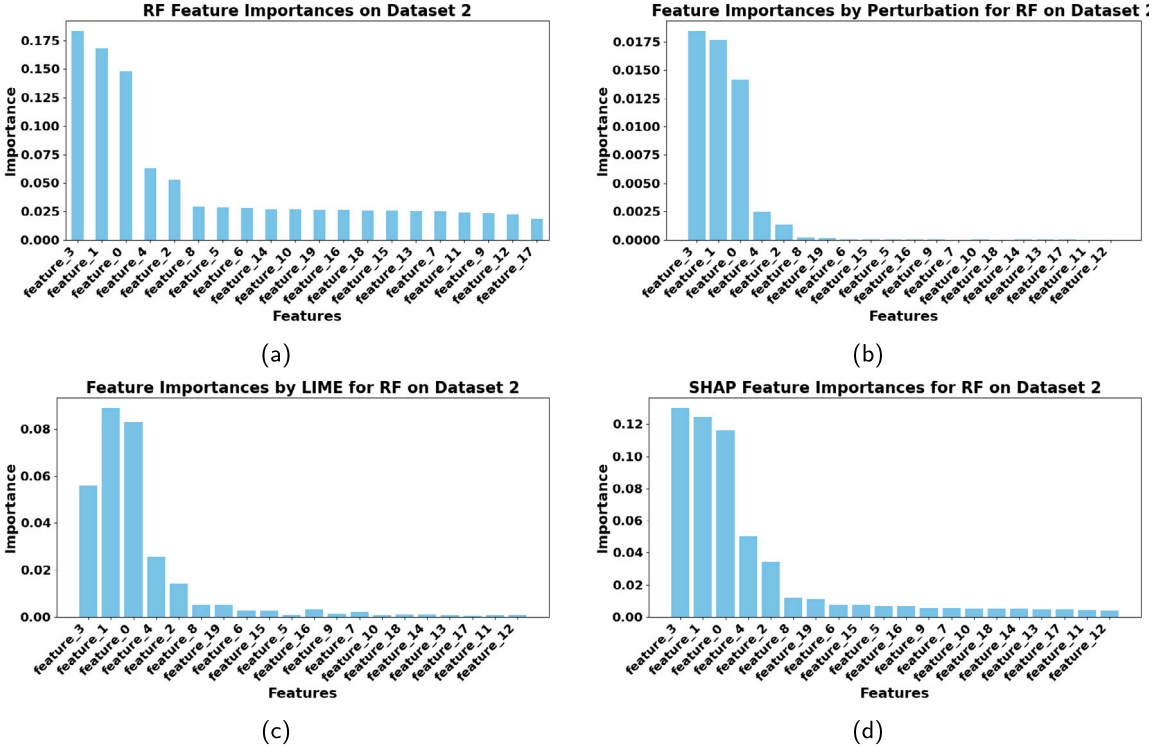

Fig. 5. Comparative analysis of feature importances for RF on Synthetic Data 2 using RF feature importances, PeBEx, LIME, and SHAP.

RF and PeBEx methods. Fig. 4(d) presents feature importances derived from SHAP, whose values align closely with those of the intrinsic RF importances but provide a more balanced view across the features and highlight interactions between features.

Analysis of Syntetic Data 2. In the comparative analysis of feature importances for RF on Synthetic Data 2, we present four subfigures showcasing the feature importances as derived from different interpretability methods. Fig. 5(a) illustrates the intrinsic feature

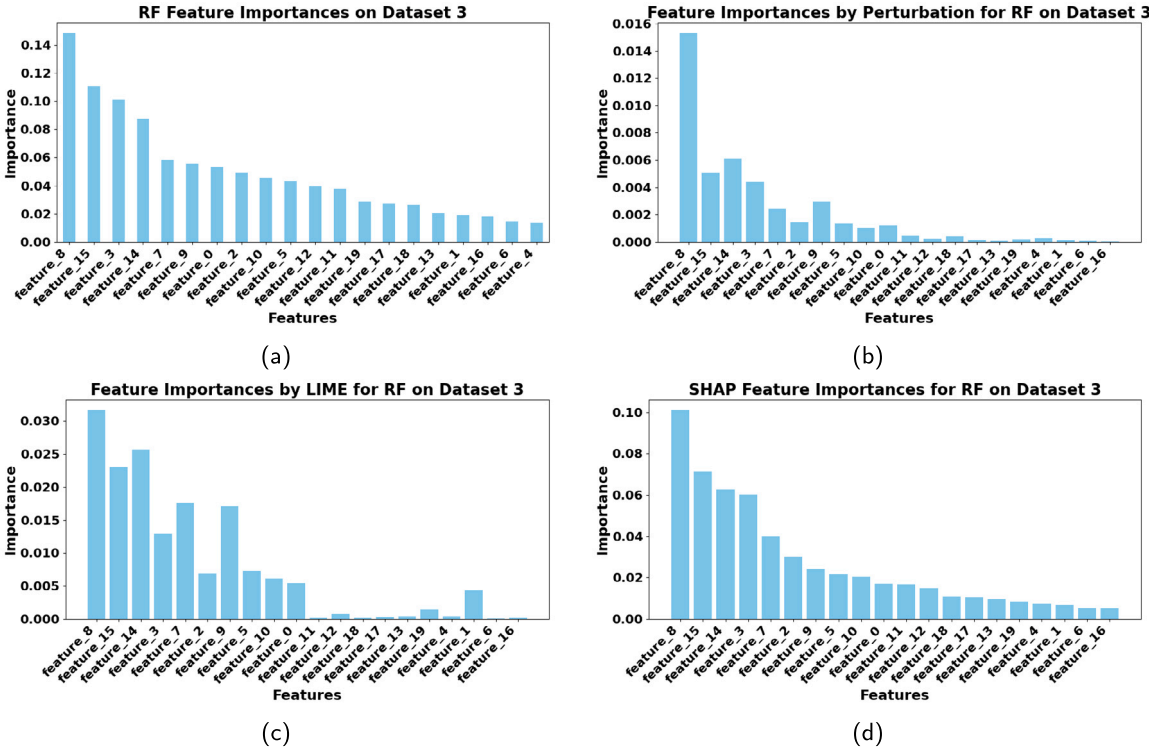

Fig. 6. Comparative analysis of feature importances for RF on Synthetic Data 3 using RF feature importances, PeBEx, LIME, and SHAP.

importances obtained directly from the RF model. Feature 3 is the most significant, followed by Features 7, 19, 2, and 8. The remaining features exhibit relatively low importance, reflecting the model reliance on a few key features. Fig. 5(b) depicts feature importances using the PeBEx method, where we notice a reduction in the importance scores for less significant features compared to the intrinsic RF importances. PeBEx allocates more precise importance to features, highlighting the truly influential ones while minimizing the scores for less impactful features. Fig. 5(c) shows the results of LIME, which identifies a similar set of important features but assigns different importance scores, with a broader distribution of importance across the features compared to RF and PeBEx methods. Fig. 5(d) presents feature importances derived from SHAP, whose values align closely with those of the intrinsic RF importances but provide a more balanced view across the features and highlight interactions between features.

Analysis of Syntetic Data 3. In the comparative analysis of feature importances for RF on Synthetic Data 3, we present four subfigures showcasing the feature importances as derived from different interpretability methods. Fig. 6(a) illustrates the intrinsic feature importances obtained directly from the RF model, with Feature 5 being the most significant, followed by Features 3, 9, 14, and 1. The remaining features exhibit relatively low importance, reflecting the model reliance on a few key features. Fig. 6(b) depicts feature importances using the PeBEx method, which shows a reduction in the importance scores for less significant features compared to the intrinsic RF importances. PeBEx allocates more precise importance to the influential features while minimizing the scores for less impactful ones. Fig. 6(c) shows the results of LIME, which identifies a similar set of important features but assigns different importance scores, resulting in a broader distribution of importance across the features compared to RF and PeBEx methods. Fig. 6(d) presents feature importances derived from SHAP, which align closely with the intrinsic RF importances but provide a more balanced view across the features and highlight interactions between them.

Analysis of Synthetic Data 4. In our comparative study of feature importances for the RF on Synthetic Data 4, we present four distinct subfigures, each reflecting different methods of interpretability. Fig. 7(a) illustrates the intrinsic feature importances as calculated directly from the RF model, with Feature 14 emerging as the most crucial, followed by Features 13, 11, 3, and 12. The lower significance of the other features indicates the model reliance on a core set of features. Fig. 7(b) shows the feature importances as determined by the PeBEx method, which tends to lower the importance scores for the less significant features compared to the intrinsic values from RF. PeBEx assigns more precise weights to the more influential features while diminishing the impact of the less significant ones. Fig. 7(c) displays the outcomes using LIME, pinpointing a similar group of important features but with varying scores, thus spreading the importance more evenly across features compared to the RF and PeBEx approaches. Fig. 7(d) presents the feature importances as derived from SHAP, which generally aligns with the intrinsic RF importances but offers a more distributed view across all features, highlighting the interplay among them.

A.2. Figures and tables

See Tables 5–7.

Data availability

The data used in this paper are publicly recognized and accessible data. They have been properly referenced in the paper.

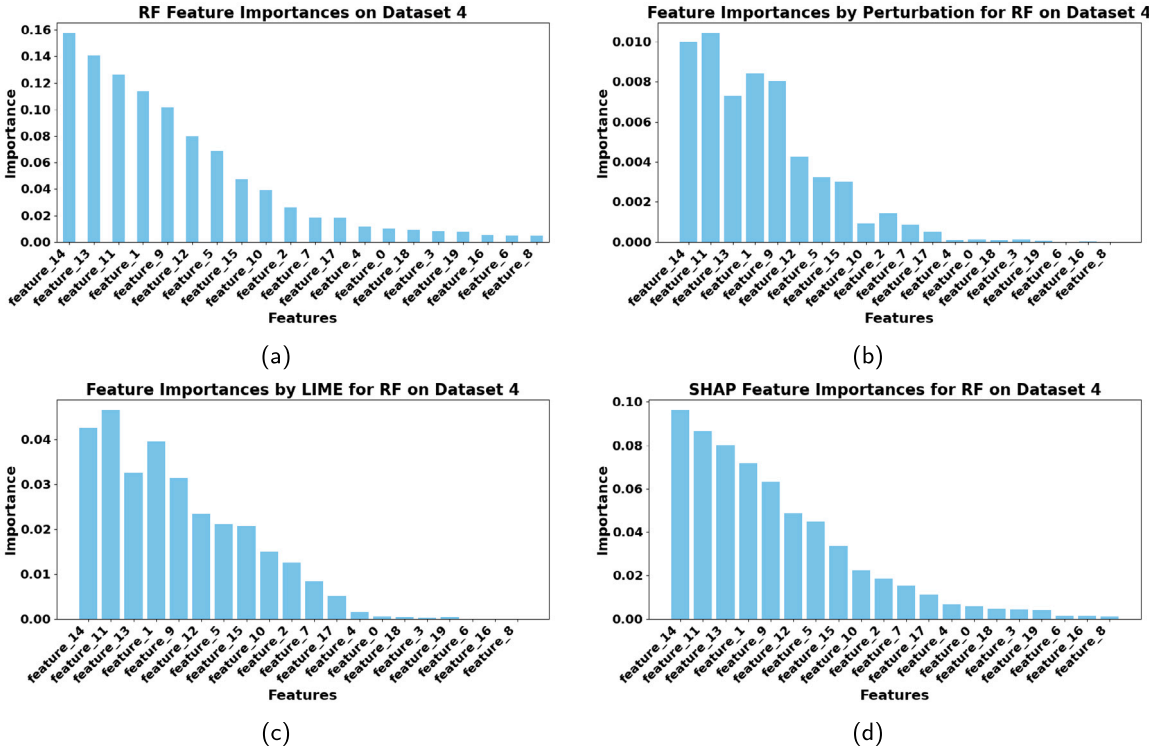

Fig. 7. Comparative analysis of feature importances for RF on Synthetic Data 4 using RF feature importances, PeBEx, LIME, and SHAP.

Table 5
This table presents a comparative analysis of ML models for predicting heart disease outcomes. The evaluation includes metrics such as Accuracy, Recall, F1-Score, Precision, Specificity, and AUC, providing a comprehensive view of each model performance. Bootstrap methods were employed to estimate the ranges.

Model	Hyperparameters	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
LogisticRegression	**'C': 0.1, 'solver': 'lbfgs'**	**0.7441 ± 0.0792**	0.7172 ± 0.1543	0.5811 ± 0.1183	**0.4984 ± 0.1194**	**0.7528 ± 0.0891**	**0.8240 ± 0.0762**
	'C': 0.1, 'solver': 'liblinear'	0.7230 ± 0.0694	0.7127 ± 0.1493	0.5583 ± 0.1040	0.4689 ± 0.1051	0.7273 ± 0.0811	0.8084 ± 0.0817
	'C': 1, 'solver': 'lbfgs'	0.7409 ± 0.0729	0.7370 ± 0.1277	**0.5854 ± 0.0963**	0.4950 ± 0.1026	0.7430 ± 0.0871	0.8216 ± 0.0732
	'C': 1, 'solver': 'liblinear'	0.7273 ± 0.0686	0.7419 ± 0.1502	0.5719 ± 0.1005	0.4735 ± 0.0939	0.7225 ± 0.0817	0.8080 ± 0.0761
	'C': 10, 'solver': 'lbfgs'	0.7307 ± 0.0749	0.7150 ± 0.1525	0.5667 ± 0.1085	0.4807 ± 0.1085	0.7360 ± 0.0936	0.8045 ± 0.0853
	'C': 10, 'solver': 'liblinear'	0.7327 ± 0.0577	0.7275 ± 0.1481	0.5700 ± 0.0973	0.4769 ± 0.0920	0.7351 ± 0.0708	0.8027 ± 0.0711
	'C': 100, 'solver': 'lbfgs'	0.7275 ± 0.0724	**0.7427 ± 0.1332**	0.5742 ± 0.0927	0.4771 ± 0.0946	0.7230 ± 0.0905	0.8036 ± 0.0707
	'C': 100, 'solver': 'liblinear'	0.7348 ± 0.0722	0.7265 ± 0.1332	0.5752 ± 0.1093	0.4832 ± 0.1103	0.7368 ± 0.0817	0.8041 ± 0.0854
SVM	'C': 1, 'kernel': 'linear', 'gamma': 'auto'	0.7270 ± 0.0736	0.7237 ± 0.1535	0.5665 ± 0.1030	0.4768 ± 0.1064	0.7279 ± 0.0941	0.7852 ± 0.0905
	'C': 1, 'kernel': 'linear', 'gamma': 'scale'	0.7325 ± 0.0722	0.7340 ± 0.1460	0.5752 ± 0.1045	0.4839 ± 0.1097	0.7318 ± 0.0902	0.7939 ± 0.0872
	'C': 1, 'kernel': 'rbf', 'gamma': 'auto'	0.7916 ± 0.0630	0.7841 ± 0.1344	0.6502 ± 0.0955	0.5683 ± 0.1092	0.7943 ± 0.0810	0.8763 ± 0.0689
	'C': 1, 'kernel': 'rbf', 'gamma': 'scale'	0.7864 ± 0.0670	0.7823 ± 0.1450	0.6436 ± 0.0998	0.5601 ± 0.1072	0.7885 ± 0.0856	0.8803 ± 0.0698
	'C': 10, 'kernel': 'linear', 'gamma': 'auto'	0.7427 ± 0.0595	0.7248 ± 0.1579	0.5785 ± 0.0957	0.4911 ± 0.0911	0.7487 ± 0.0722	0.7906 ± 0.0889
	'C': 10, 'kernel': 'linear', 'gamma': 'scale'	0.7266 ± 0.0778	0.7272 ± 0.1431	0.5697 ± 0.1016	0.4806 ± 0.1151	0.7255 ± 0.0999	0.7816 ± 0.0795
	'C': 10, 'kernel': 'rbf', 'gamma': 'auto'	**0.8091 ± 0.0731**	**0.8306 ± 0.1309**	**0.6837 ± 0.1122**	**0.5936 ± 0.1294**	**0.8021 ± 0.0853**	**0.8966 ± 0.0634**
	'C': 10, 'kernel': 'rbf', 'gamma': 'scale'	0.7905 ± 0.0684	0.8181 ± 0.1245	0.6603 ± 0.0939	0.5658 ± 0.1110	0.7818 ± 0.0892	0.8883 ± 0.0566
RandomForest	'n_estimators': 100, 'max_depth': 10, 'min_samples_split': 10	0.7920 ± 0.0668	0.8115 ± 0.1338	0.6594 ± 0.0965	0.5716 ± 0.1186	0.7865 ± 0.0898	0.8876 ± 0.0605
	'n_estimators': 100, 'max_depth': 10, 'min_samples_split': 2	**0.8202 ± 0.0634**	**0.8254 ± 0.1288**	**0.6958 ± 0.0939**	**0.6157 ± 0.1176**	**0.8185 ± 0.0831**	**0.9154 ± 0.0557**
	'n_estimators': 100, 'max_depth': None, 'min_samples_split': 10	0.7884 ± 0.0785	0.8155 ± 0.1200	0.6591 ± 0.1055	0.5661 ± 0.1287	0.7810 ± 0.0964	0.8832 ± 0.0653
	'n_estimators': 100, 'max_depth': None, 'min_samples_split': 2	0.8075 ± 0.0635	0.8208 ± 0.1148	0.6802 ± 0.0945	0.5934 ± 0.1218	0.8030 ± 0.0835	0.9063 ± 0.0503
	'n_estimators': 200, 'max_depth': 10, 'min_samples_split': 10	0.7998 ± 0.0622	0.7995 ± 0.1286	0.6633 ± 0.1003	0.5793 ± 0.1223	0.8000 ± 0.0771	0.8841 ± 0.0610
	'n_estimators': 200, 'max_depth': 10, 'min_samples_split': 2	0.8057 ± 0.0732	0.8130 ± 0.1325	0.6765 ± 0.1097	0.5926 ± 0.1308	0.8031 ± 0.0884	0.9051 ± 0.0569
	'n_estimators': 200, 'max_depth': None, 'min_samples_split': 10	0.7902 ± 0.0710	0.7945 ± 0.1291	0.6535 ± 0.1020	0.5678 ± 0.1184	0.7886 ± 0.0886	0.8813 ± 0.0672
	'n_estimators': 200, 'max_depth': None, 'min_samples_split': 2	0.8089 ± 0.0677	0.8156 ± 0.1266	0.6797 ± 0.1033	0.5946 ± 0.1193	0.8066 ± 0.0815	0.9132 ± 0.0563
GradientBoosting	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.01	0.7580 ± 0.0699	0.7628 ± 0.1506	0.6087 ± 0.1016	0.5193 ± 0.1118	0.7562 ± 0.0892	0.8338 ± 0.0834
	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1	**0.7959 ± 0.0761**	0.8076 ± 0.1368	**0.6626 ± 0.1194**	**0.5740 ± 0.1343**	**0.7921 ± 0.0914**	0.8800 ± 0.0675
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.7541 ± 0.0840	0.7947 ± 0.1378	0.6171 ± 0.1193	0.5148 ± 0.1298	0.7398 ± 0.0989	0.8242 ± 0.0900
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	0.7823 ± 0.0754	**0.8347 ± 0.1198**	0.6568 ± 0.1054	0.5530 ± 0.1252	0.7651 ± 0.0935	0.8656 ± 0.0797
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.01	0.7659 ± 0.0742	0.8173 ± 0.1311	0.6345 ± 0.1031	0.5297 ± 0.1166	0.7491 ± 0.0950	0.8561 ± 0.0703
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1	0.7836 ± 0.0718	0.8283 ± 0.1253	0.6556 ± 0.1053	0.5526 ± 0.1208	0.7686 ± 0.0859	**0.8815 ± 0.0743**
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.7664 ± 0.0738	0.7979 ± 0.1361	0.6285 ± 0.1053	0.5299 ± 0.1173	0.7558 ± 0.0937	0.8311 ± 0.0879
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	0.7723 ± 0.0797	0.8041 ± 0.1430	0.6372 ± 0.1206	0.5399 ± 0.1331	0.7614 ± 0.0954	0.8766 ± 0.0814

(*continued on next page*)

Table 5 (*continued*).

Model	Hyperparameters	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
XGBoost	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.01	0.7586 ± 0.0738	0.7630 ± 0.1484	0.6096 ± 0.1019	0.5208 ± 0.1065	0.7564 ± 0.0989	0.8347 ± 0.0719
	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1	0.7859 ± 0.0726	0.7989 ± 0.1330	0.6490 ± 0.1149	0.5557 ± 0.1248	0.7815 ± 0.0809	0.8666 ± 0.0751
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.7495 ± 0.0772	0.7878 ± 0.1250	0.6112 ± 0.0982	0.5119 ± 0.1155	0.7373 ± 0.1011	0.8409 ± 0.0781
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	0.7857 ± 0.0666	0.8015 ± 0.1336	0.6492 ± 0.1024	0.5555 ± 0.1119	0.7795 ± 0.0835	0.8664 ± 0.0734
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.01	0.7682 ± 0.0686	0.7679 ± 0.1418	0.6200 ± 0.1075	0.5288 ± 0.1098	0.7682 ± 0.0796	0.8490 ± 0.0792
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1	0.7893 ± 0.0667	**0.8305 ± 0.1208**	**0.6621 ± 0.0988**	0.5582 ± 0.1060	0.7755 ± 0.0787	0.8743 ± 0.0678
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.7759 ± 0.0756	0.7991 ± 0.1377	0.6386 ± 0.1103	0.5444 ± 0.1208	0.7689 ± 0.0962	0.8535 ± 0.0728
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	**0.7925 ± 0.0694**	0.8146 ± 0.1423	0.6605 ± 0.1049	**0.5673 ± 0.1136**	**0.7835 ± 0.0892**	**0.8793 ± 0.0601**
LightGBM	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.7161 ± 0.0852	0.7263 ± 0.1559	0.5596 ± 0.1094	0.4682 ± 0.1133	0.7116 ± 0.1154	0.8065 ± 0.0862
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	0.7727 ± 0.0798	0.7807 ± 0.1354	0.6313 ± 0.1149	0.5431 ± 0.1288	0.7701 ± 0.0973	0.8394 ± 0.0776
	'n_estimators': 100, 'max_depth': 7, 'learning_rate': 0.01	0.7230 ± 0.0761	0.7065 ± 0.1516	0.5565 ± 0.1139	0.4699 ± 0.1176	0.7290 ± 0.0927	0.8075 ± 0.0723
	'n_estimators': 100, 'max_depth': 7, 'learning_rate': 0.1	0.7784 ± 0.0794	0.7872 ± 0.1456	0.6388 ± 0.1168	0.5505 ± 0.1293	0.7749 ± 0.0951	0.8520 ± 0.0825
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.7386 ± 0.0832	0.7310 ± 0.1385	0.5835 ± 0.1183	0.4957 ± 0.1285	0.7403 ± 0.0982	0.8092 ± 0.0770
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	**0.7968 ± 0.0706**	**0.8296 ± 0.1274**	**0.6705 ± 0.1053**	**0.5731 ± 0.1207**	**0.7858 ± 0.0858**	0.8597 ± 0.0796
	'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.01	0.7302 ± 0.0733	0.7242 ± 0.1391	0.5703 ± 0.1056	0.4802 ± 0.1100	0.7315 ± 0.0904	0.8145 ± 0.0777
	'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.1	0.7900 ± 0.0686	0.8155 ± 0.1400	0.6569 ± 0.1116	0.5593 ± 0.1168	0.7808 ± 0.0800	**0.8602 ± 0.0813**
MLP	'hidden_layer_sizes': (100,), 'activation': 'relu', 'solver': 'adam'	0.7902 ± 0.0621	0.8316 ± 0.1184	0.6620 ± 0.0959	0.5587 ± 0.1078	0.7770 ± 0.0739	0.8559 ± 0.0748
	'hidden_layer_sizes': (100,), 'activation': 'relu', 'solver': 'sgd'	0.7664 ± 0.0686	0.7695 ± 0.1402	0.6195 ± 0.0987	0.5321 ± 0.1109	0.7663 ± 0.0907	0.8465 ± 0.0658
	'hidden_layer_sizes': (100,), 'activation': 'tanh', 'solver': 'adam'	0.7850 ± 0.0673	0.8175 ± 0.1328	0.6527 ± 0.1032	0.5523 ± 0.1077	0.7742 ± 0.0796	0.8535 ± 0.0719
	'hidden_layer_sizes': (100,), 'activation': 'tanh', 'solver': 'sgd'	0.7409 ± 0.0823	0.7121 ± 0.1388	0.5783 ± 0.1074	0.5003 ± 0.1196	0.7513 ± 0.1003	0.8149 ± 0.0818
	'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'solver': 'adam'	**0.7975 ± 0.0682**	0.8195 ± 0.1303	**0.6672 ± 0.1058**	**0.5738 ± 0.1238**	**0.7895 ± 0.0810**	0.8603 ± 0.0722
	'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'solver': 'sgd'	0.7764 ± 0.0665	0.7906 ± 0.1403	0.6362 ± 0.0979	0.5426 ± 0.1064	0.7719 ± 0.0813	0.8464 ± 0.0764
	'hidden_layer_sizes': (50, 50), 'activation': 'tanh', 'solver': 'adam'	0.7916 ± 0.0712	**0.8400 ± 0.1456**	0.6661 ± 0.1102	0.5636 ± 0.1187	0.7760 ± 0.0890	**0.8717 ± 0.0753**
	'hidden_layer_sizes': (50, 50), 'activation': 'tanh', 'solver': 'sgd'	0.7573 ± 0.0641	0.7856 ± 0.1192	0.6164 ± 0.0796	0.5163 ± 0.0877	0.7481 ± 0.0871	0.8406 ± 0.0754
PyTorch NN	'lr': 0.01, 'max_epochs': 10	0.5293 ± 0.2126	0.4971 ± 0.3912	0.2756 ± 0.1832	0.2542 ± 0.2199	0.5419 ± 0.3967	0.5278 ± 0.1672
	'lr': 0.01, 'max_epochs': 20	0.5120 ± 0.2123	0.5364 ± 0.3969	0.2825 ± 0.1728	0.2587 ± 0.2255	0.5065 ± 0.3986	0.5622 ± 0.1325
	'lr': 0.01, 'max_epochs': 50	0.5098 ± 0.2047	0.6657 ± 0.3365	0.3798 ± 0.1249	0.3456 ± 0.1931	0.4550 ± 0.3705	0.6567 ± 0.1287
	'lr': 0.1, 'max_epochs': 10	0.5686 ± 0.1802	0.6875 ± 0.2741	0.4352 ± 0.1093	0.3700 ± 0.1435	0.5306 ± 0.3069	0.7304 ± 0.1025
	'lr': 0.1, 'max_epochs': 20	0.6536 ± 0.1249	**0.7858 ± 0.1717**	0.5342 ± 0.0982	0.4252 ± 0.1121	0.6104 ± 0.1890	0.7925 ± 0.0873
	'lr': 0.1, 'max_epochs': 50	**0.7268 ± 0.0742**	0.7101 ± 0.1585	**0.5607 ± 0.1065**	**0.4754 ± 0.1054**	**0.7322 ± 0.0957**	**0.8074 ± 0.0853**

Table 6
This table presents a comparative analysis of ML models for predicting credit outcomes from the German credit dataset. The evaluation encompasses metrics such as Accuracy, Recall, F1-Score, Precision, Specificity, and AUC, offering a detailed perspective on each model efficacy. Bootstrap methods were used for estimating the ranges.

Model	Hyperparameters	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
LogisticRegression	**'C': 0.1, 'solver': 'lbfgs'**	**0.7258 ± 0.0357**	0.7305 ± 0.0576	**0.6154 ± 0.0426**	**0.5338 ± 0.0472**	**0.7238 ± 0.0480**	0.7955 ± 0.0304
	'C': 0.1, 'solver': 'liblinear'	0.7221 ± 0.0357	**0.7438 ± 0.0628**	0.6161 ± 0.0459	0.5273 ± 0.0443	0.7127 ± 0.0411	0.7945 ± 0.0382
	'C': 1, 'solver': 'lbfgs'	0.7217 ± 0.0346	0.7324 ± 0.0560	0.6124 ± 0.0421	0.5282 ± 0.0466	0.7171 ± 0.0463	0.7934 ± 0.0299
	'C': 1, 'solver': 'liblinear'	0.7229 ± 0.0307	0.7299 ± 0.0530	0.6124 ± 0.0382	0.5290 ± 0.0401	0.7199 ± 0.0392	0.7932 ± 0.0324
	'C': 10, 'solver': 'lbfgs'	0.7231 ± 0.0320	0.7365 ± 0.0619	0.6145 ± 0.0403	0.5291 ± 0.0399	0.7176 ± 0.0414	0.7970 ± 0.0346
	'C': 10, 'solver': 'liblinear'	0.7241 ± 0.0325	0.7349 ± 0.0653	0.6147 ± 0.0447	0.5304 ± 0.0453	0.7195 ± 0.0411	**0.7973 ± 0.0342**
	'C': 100, 'solver': 'lbfgs'	0.7167 ± 0.0326	0.7327 ± 0.0554	0.6080 ± 0.0412	0.5210 ± 0.0419	0.7098 ± 0.0393	0.7887 ± 0.0325
	'C': 100, 'solver': 'liblinear'	0.7182 ± 0.0310	0.7256 ± 0.0575	0.6069 ± 0.0398	0.5232 ± 0.0411	0.7152 ± 0.0385	0.7921 ± 0.0340
SVM	'C': 1, 'kernel': 'linear', 'gamma': 'auto'	0.7174 ± 0.0357	0.7458 ± 0.0620	0.6128 ± 0.0450	0.5221 ± 0.0467	0.7052 ± 0.0461	0.7903 ± 0.0354
	'C': 1, 'kernel': 'linear', 'gamma': 'scale'	0.7216 ± 0.0350	0.7427 ± 0.0664	0.6152 ± 0.0466	0.5271 ± 0.0473	0.7125 ± 0.0444	0.7939 ± 0.0345
	'C': 1, 'kernel': 'rbf', 'gamma': 'auto'	0.7735 ± 0.0314	0.7944 ± 0.0591	0.6777 ± 0.0427	0.5930 ± 0.0467	0.7645 ± 0.0408	0.8607 ± 0.0325
	'C': 1, 'kernel': 'rbf', 'gamma': 'scale'	0.7798 ± 0.0335	0.8081 ± 0.0536	0.6881 ± 0.0401	0.6012 ± 0.0465	0.7676 ± 0.0437	0.8676 ± 0.0288
	'C': 10, 'kernel': 'linear', 'gamma': 'auto'	0.7244 ± 0.0319	0.7445 ± 0.0584	0.6183 ± 0.0390	0.5308 ± 0.0428	0.7156 ± 0.0451	0.7952 ± 0.0343
	'C': 10, 'kernel': 'linear', 'gamma': 'scale'	0.7183 ± 0.0401	0.7376 ± 0.0589	0.6116 ± 0.0459	0.5245 ± 0.0514	0.7098 ± 0.0527	0.7865 ± 0.0382
	'C': 10, 'kernel': 'rbf', 'gamma': 'auto'	**0.8027 ± 0.0302**	**0.8260 ± 0.0545**	**0.7152 ± 0.0420**	**0.6325 ± 0.0481**	**0.7928 ± 0.0372**	**0.8807 ± 0.0304**
	'C': 10, 'kernel': 'rbf', 'gamma': 'scale'	0.7932 ± 0.0277	0.8191 ± 0.0502	0.7037 ± 0.0375	0.6189 ± 0.0452	0.7821 ± 0.0383	0.8743 ± 0.0276
RandomForest	'n_estimators': 100, 'max_depth': 10, 'min_samples_split': 10	0.8090 ± 0.0300	0.8441 ± 0.0586	0.7260 ± 0.0414	0.6396 ± 0.0504	0.7940 ± 0.0415	0.8966 ± 0.0253
	'n_estimators': 100, 'max_depth': 10, 'min_samples_split': 2	0.8233 ± 0.0290	**0.8573 ± 0.0458**	0.7446 ± 0.0383	0.6597 ± 0.0467	0.8086 ± 0.0371	0.9172 ± 0.0210
	'n_estimators': 100, 'max_depth': None, 'min_samples_split': 10	0.8124 ± 0.0335	0.8315 ± 0.0524	0.7271 ± 0.0442	0.6484 ± 0.0548	0.8043 ± 0.0422	0.8965 ± 0.0281
	'n_estimators': 100, 'max_depth': None, 'min_samples_split': 2	**0.8301 ± 0.0282**	0.8516 ± 0.0490	**0.7505 ± 0.0387**	**0.6737 ± 0.0529**	0.8211 ± 0.0400	**0.9248 ± 0.0213**
	'n_estimators': 200, 'max_depth': 10, 'min_samples_split': 10	0.8063 ± 0.0310	0.8344 ± 0.0538	0.7212 ± 0.0409	0.6374 ± 0.0498	0.7945 ± 0.0403	0.8923 ± 0.0276
	'n_estimators': 200, 'max_depth': 10, 'min_samples_split': 2	0.8255 ± 0.0296	0.8556 ± 0.0592	0.7462 ± 0.0412	0.6639 ± 0.0469	0.8127 ± 0.0361	0.9185 ± 0.0248
	'n_estimators': 200, 'max_depth': None, 'min_samples_split': 10	0.8097 ± 0.0351	0.8288 ± 0.0595	0.7236 ± 0.0459	0.6454 ± 0.0589	0.8014 ± 0.0484	0.8958 ± 0.0273
	'n_estimators': 200, 'max_depth': None, 'min_samples_split': 2	0.8291 ± 0.0298	0.8468 ± 0.0554	0.7480 ± 0.0441	0.6721 ± 0.0524	**0.8216 ± 0.0362**	0.9236 ± 0.0220
GradientBoosting	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.01	0.7088 ± 0.0353	0.7972 ± 0.0477	0.6221 ± 0.0362	0.5117 ± 0.0418	0.6708 ± 0.0515	0.8093 ± 0.0326
	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1	0.8018 ± 0.0303	0.8342 ± 0.0514	0.7166 ± 0.0391	0.6302 ± 0.0479	0.7880 ± 0.0404	0.8831 ± 0.0264
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.7569 ± 0.0347	0.8300 ± 0.0536	0.6725 ± 0.0387	0.5672 ± 0.0448	0.7257 ± 0.0488	0.8504 ± 0.0316
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	**0.8218 ± 0.0295**	**0.8468 ± 0.0535**	**0.7403 ± 0.0409**	**0.6606 ± 0.0541**	**0.8112 ± 0.0415**	0.9060 ± 0.0240
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.01	0.7319 ± 0.0379	0.8059 ± 0.0566	0.6438 ± 0.0417	0.5379 ± 0.0467	0.6997 ± 0.0534	0.8322 ± 0.0292
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1	0.8145 ± 0.0296	0.8436 ± 0.0549	0.7316 ± 0.0435	0.6472 ± 0.0464	0.8018 ± 0.0327	0.8883 ± 0.0283
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.7793 ± 0.0356	0.8376 ± 0.0578	0.6951 ± 0.0450	0.5961 ± 0.0504	0.7542 ± 0.0459	0.8720 ± 0.0327
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	0.8185 ± 0.0261	0.8415 ± 0.0546	0.7354 ± 0.0366	0.6552 ± 0.0433	0.8085 ± 0.0355	**0.9090 ± 0.0225**
XGBoost	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.01	0.7000 ± 0.0363	0.7889 ± 0.0641	0.6121 ± 0.0403	0.5024 ± 0.0439	0.6617 ± 0.0550	0.7958 ± 0.0343
	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1	0.7881 ± 0.0288	0.8142 ± 0.0540	0.6974 ± 0.0386	0.6121 ± 0.0458	0.7769 ± 0.0393	0.8695 ± 0.0259
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.7405 ± 0.0381	0.8153 ± 0.0625	0.6537 ± 0.0430	0.5484 ± 0.0500	0.7086 ± 0.0552	0.8315 ± 0.0338
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	**0.8160 ± 0.0280**	**0.8500 ± 0.0469**	**0.7350 ± 0.0366**	**0.6498 ± 0.0494**	0.8014 ± 0.0398	0.8918 ± 0.0252
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.01	0.7190 ± 0.0346	0.7874 ± 0.0565	0.6272 ± 0.0376	0.5233 ± 0.0426	0.6896 ± 0.0506	0.8156 ± 0.0316
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1	0.8014 ± 0.0317	0.8211 ± 0.0556	0.7129 ± 0.0417	0.6322 ± 0.0495	0.7930 ± 0.0412	0.8753 ± 0.0287
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.7626 ± 0.0339	0.8229 ± 0.0550	0.6756 ± 0.0393	0.5754 ± 0.0473	0.7366 ± 0.0494	0.8538 ± 0.0338
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	0.8123 ± 0.0298	0.8350 ± 0.0517	0.7274 ± 0.0411	0.6466 ± 0.0507	**0.8025 ± 0.0387**	**0.8921 ± 0.0260**

(*continued on next page*)

Table 6 (*continued*).

Model	Hyperparameters	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
LightGBM	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.7295 ± 0.0353	0.7735 ± 0.0583	0.6320 ± 0.0394	0.5367 ± 0.0461	0.7107 ± 0.0510	0.8134 ± 0.0351
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	0.8106 ± 0.0330	0.8394 ± 0.0547	0.7270 ± 0.0431	0.6434 ± 0.0519	0.7985 ± 0.0407	0.8840 ± 0.0318
	'n_estimators': 100, 'max_depth': 7, 'learning_rate': 0.01	0.7414 ± 0.0331	0.7740 ± 0.0641	0.6421 ± 0.0430	0.5510 ± 0.0470	0.7275 ± 0.0455	0.8260 ± 0.0284
	'n_estimators': 100, 'max_depth': 7, 'learning_rate': 0.1	**0.8172 ± 0.0280**	0.8396 ± 0.0479	**0.7338 ± 0.0386**	**0.6540 ± 0.0507**	**0.8075 ± 0.0384**	0.8882 ± 0.0250
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.7514 ± 0.0330	0.7901 ± 0.0557	0.6561 ± 0.0406	0.5626 ± 0.0428	0.7346 ± 0.0429	0.8377 ± 0.0332
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	0.8156 ± 0.0280	**0.8423 ± 0.0499**	0.7328 ± 0.0377	0.6505 ± 0.0463	0.8041 ± 0.0377	0.8907 ± 0.0292
	'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.01	0.7678 ± 0.0337	0.7847 ± 0.0549	0.6699 ± 0.0436	0.5865 ± 0.0502	0.7607 ± 0.0429	0.8457 ± 0.0317
	'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.1	0.8154 ± 0.0319	0.8418 ± 0.0535	0.7324 ± 0.0434	0.6502 ± 0.0515	0.8040 ± 0.0398	**0.8943 ± 0.0297**
MLP	'hidden_layer_sizes': (100,), 'activation': 'relu', 'solver': 'adam'	0.8053 ± 0.0295	0.8306 ± 0.0576	0.7189 ± 0.0417	0.6358 ± 0.0474	0.7945 ± 0.0373	0.8657 ± 0.0308
	'hidden_layer_sizes': (100,), 'activation': 'relu', 'solver': 'sgd'	0.7659 ± 0.0316	0.7917 ± 0.0545	0.6699 ± 0.0408	0.5823 ± 0.0446	0.7549 ± 0.0390	0.8438 ± 0.0308
	'hidden_layer_sizes': (100,), 'activation': 'tanh', 'solver': 'adam'	**0.8142 ± 0.0319**	0.8335 ± 0.0502	**0.7294 ± 0.0429**	**0.6504 ± 0.0524**	**0.8060 ± 0.0397**	**0.8893 ± 0.0279**
	'hidden_layer_sizes': (100,), 'activation': 'tanh', 'solver': 'sgd'	0.7175 ± 0.0349	0.7392 ± 0.0654	0.6106 ± 0.0455	0.5220 ± 0.0455	0.7083 ± 0.0419	0.7970 ± 0.0397
	'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'solver': 'adam'	0.8063 ± 0.0335	**0.8418 ± 0.0545**	0.7229 ± 0.0451	0.6357 ± 0.0535	0.7911 ± 0.0419	0.8691 ± 0.0329
	'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'solver': 'sgd'	0.7735 ± 0.0318	0.7906 ± 0.0598	0.6767 ± 0.0430	0.5938 ± 0.0485	0.7661 ± 0.0415	0.8488 ± 0.0288
	'hidden_layer_sizes': (50, 50), 'activation': 'tanh', 'solver': 'adam'	0.8089 ± 0.0292	0.8239 ± 0.0538	0.7210 ± 0.0424	0.6430 ± 0.0500	0.8024 ± 0.0365	0.8727 ± 0.0303
	'hidden_layer_sizes': (50, 50), 'activation': 'tanh', 'solver': 'sgd'	0.7537 ± 0.0331	0.7691 ± 0.0572	0.6521 ± 0.0408	0.5681 ± 0.0457	0.7471 ± 0.0436	0.8299 ± 0.0302
PyTorch NN	'lr': 0.01, 'max_epochs': 10	0.4993 ± 0.1371	0.5493 ± 0.3475	0.3392 ± 0.1608	0.3085 ± 0.1468	0.4787 ± 0.3372	0.5427 ± 0.0777
	'lr': 0.01, 'max_epochs': 20	0.5116 ± 0.1453	0.5792 ± 0.3482	0.3659 ± 0.1528	0.3274 ± 0.1264	0.4816 ± 0.3479	0.5772 ± 0.0708
	'lr': 0.01, 'max_epochs': 50	0.5681 ± 0.1171	0.6697 ± 0.2441	0.4672 ± 0.0990	0.3973 ± 0.0897	0.5245 ± 0.2509	0.6603 ± 0.0659
	'lr': 0.1, 'max_epochs': 10	0.6410 ± 0.0815	0.7045 ± 0.1456	0.5388 ± 0.0702	0.4555 ± 0.0766	0.6140 ± 0.1581	0.7297 ± 0.0482
	'lr': 0.1, 'max_epochs': 20	0.6900 ± 0.0489	**0.7517 ± 0.0799**	0.5930 ± 0.0489	0.4943 ± 0.0545	0.6639 ± 0.0764	0.7719 ± 0.0429
	'lr': 0.1, 'max_epochs': 50	**0.7265 ± 0.0314**	0.7477 ± 0.0663	**0.6209 ± 0.0392**	**0.5333 ± 0.0399**	**0.7175 ± 0.0449**	**0.8051 ± 0.0348**

Table 7
This table presents a comparative analysis of ML models for predicting breast cancer outcomes. The evaluation encompasses metrics such as Accuracy, Recall, F1-Score, Precision, Specificity, and AUC, offering a detailed perspective on each model efficacy.

Model	Hyperparameters	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
LogisticRegression	'C': 0.1, 'solver': 'lbfgs'	0.9783 ± 0.0132	0.9858 ± 0.0156	0.9827 ± 0.0105	0.9800 ± 0.0139	0.9660 ± 0.0245	**0.9961 ± 0.0041**
	'C': 0.1, 'solver': 'liblinear'	**0.9795 ± 0.0138**	**0.9871 ± 0.0133**	**0.9836 ± 0.0110**	0.9804 ± 0.0164	0.9667 ± 0.0284	0.9956 ± 0.0052
	'C': 1, 'solver': 'lbfgs'	0.9754 ± 0.0168	0.9769 ± 0.0223	0.9802 ± 0.0137	0.9838 ± 0.0133	0.9727 ± 0.0225	0.9959 ± 0.0050
	'C': 1, 'solver': 'liblinear'	0.9787 ± 0.0148	0.9831 ± 0.0157	0.9830 ± 0.0118	0.9832 ± 0.0179	0.9715 ± 0.0316	0.9953 ± 0.0075
	'C': 10, 'solver': 'lbfgs'	0.9743 ± 0.0144	0.9768 ± 0.0199	0.9793 ± 0.0118	0.9822 ± 0.0144	0.9699 ± 0.0248	0.9948 ± 0.0059
	'C': 10, 'solver': 'liblinear'	0.9761 ± 0.0168	0.9773 ± 0.0210	0.9807 ± 0.0135	0.9844 ± 0.0132	0.9739 ± 0.0224	0.9935 ± 0.0077
	'C': 100, 'solver': 'lbfgs'	0.9736 ± 0.0168	0.9724 ± 0.0242	0.9786 ± 0.0137	**0.9854 ± 0.0154**	**0.9757 ± 0.0259**	0.9924 ± 0.0089
	'C': 100, 'solver': 'liblinear'	0.9728 ± 0.0145	0.9731 ± 0.0176	0.9781 ± 0.0118	0.9834 ± 0.0162	0.9721 ± 0.0285	0.9922 ± 0.0081
SVM	'C': 1, 'kernel': 'linear', 'gamma': 'auto'	0.9739 ± 0.0167	0.9755 ± 0.0241	0.9790 ± 0.0138	0.9829 ± 0.0146	0.9715 ± 0.0245	0.9947 ± 0.0066
	'C': 1, 'kernel': 'linear', 'gamma': 'scale'	0.9755 ± 0.0148	0.9782 ± 0.0199	0.9803 ± 0.0120	0.9828 ± 0.0153	0.9709 ± 0.0265	0.9955 ± 0.0058
	'C': 1, 'kernel': 'rbf', 'gamma': 'auto'	**0.9778 ± 0.0154**	**0.9844 ± 0.0167**	**0.9823 ± 0.0122**	0.9805 ± 0.0162	0.9669 ± 0.0281	0.9960 ± 0.0049
	'C': 1, 'kernel': 'rbf', 'gamma': 'scale'	0.9761 ± 0.0150	0.9800 ± 0.0209	0.9809 ± 0.0119	0.9823 ± 0.0154	0.9700 ± 0.0263	0.9964 ± 0.0054
	'C': 10, 'kernel': 'linear', 'gamma': 'auto'	0.9714 ± 0.0162	0.9720 ± 0.0230	0.9769 ± 0.0132	0.9823 ± 0.0152	0.9706 ± 0.0260	0.9927 ± 0.0080
	'C': 10, 'kernel': 'linear', 'gamma': 'scale'	0.9716 ± 0.0172	0.9712 ± 0.0220	0.9771 ± 0.0141	0.9833 ± 0.0148	0.9721 ± 0.0249	0.9919 ± 0.0081
	'C': 10, 'kernel': 'rbf', 'gamma': 'auto'	0.9754 ± 0.0139	0.9751 ± 0.0219	0.9802 ± 0.0113	**0.9857 ± 0.0140**	**0.9757 ± 0.0247**	0.9966 ± 0.0048
	'C': 10, 'kernel': 'rbf', 'gamma': 'scale'	0.9767 ± 0.0139	0.9776 ± 0.0184	0.9812 ± 0.0112	0.9852 ± 0.0147	0.9751 ± 0.0252	**0.9969 ± 0.0041**
RandomForest	'n_estimators': 100, 'max_depth': 10, 'min_samples_split': 10	0.9725 ± 0.0168	0.9715 ± 0.0233	0.9778 ± 0.0137	0.9847 ± 0.0150	0.9744 ± 0.0259	0.9956 ± 0.0052
	'n_estimators': 100, 'max_depth': 10, 'min_samples_split': 2	**0.9754 ± 0.0172**	**0.9751 ± 0.0234**	**0.9802 ± 0.0141**	0.9857 ± 0.0151	0.9757 ± 0.0267	**0.9962 ± 0.0059**
	'n_estimators': 100, 'max_depth': None, 'min_samples_split': 10	0.9660 ± 0.0169	0.9661 ± 0.0225	0.9726 ± 0.0135	0.9798 ± 0.0172	0.9658 ± 0.0298	0.9939 ± 0.0076
	'n_estimators': 100, 'max_depth': None, 'min_samples_split': 2	0.9733 ± 0.0181	0.9715 ± 0.0226	0.9785 ± 0.0145	**0.9859 ± 0.0169**	**0.9763 ± 0.0291**	0.9960 ± 0.0057
	'n_estimators': 200, 'max_depth': 10, 'min_samples_split': 10	0.9711 ± 0.0182	0.9734 ± 0.0214	0.9767 ± 0.0148	0.9804 ± 0.0168	0.9673 ± 0.0279	0.9953 ± 0.0056
	'n_estimators': 200, 'max_depth': 10, 'min_samples_split': 2	0.9739 ± 0.0180	0.9738 ± 0.0235	0.9791 ± 0.0143	0.9849 ± 0.0159	0.9741 ± 0.0286	0.9958 ± 0.0054
	'n_estimators': 200, 'max_depth': None, 'min_samples_split': 10	0.9697 ± 0.0179	0.9712 ± 0.0244	0.9756 ± 0.0146	0.9805 ± 0.0157	0.9676 ± 0.0264	0.9948 ± 0.0065
	'n_estimators': 200, 'max_depth': None, 'min_samples_split': 2	0.9722 ± 0.0176	0.9710 ± 0.0238	0.9776 ± 0.0143	0.9846 ± 0.0157	0.9742 ± 0.0268	0.9955 ± 0.0064
GradientBoosting	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.01	0.9568 ± 0.0212	0.9558 ± 0.0278	0.9651 ± 0.0169	0.9752 ± 0.0203	0.9587 ± 0.0353	0.9851 ± 0.0143
	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1	0.9695 ± 0.0183	0.9697 ± 0.0235	0.9754 ± 0.0148	0.9817 ± 0.0159	0.9692 ± 0.0275	0.9946 ± 0.0074
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.9602 ± 0.0222	0.9581 ± 0.0315	0.9676 ± 0.0185	0.9780 ± 0.0191	0.9634 ± 0.0325	0.9755 ± 0.0208
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	0.9582 ± 0.0208	0.9562 ± 0.0286	0.9661 ± 0.0170	0.9769 ± 0.0209	0.9617 ± 0.0349	0.9765 ± 0.0201
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.01	0.9611 ± 0.0196	0.9586 ± 0.0273	0.9685 ± 0.0158	0.9793 ± 0.0179	0.9654 ± 0.0309	0.9894 ± 0.0111
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1	**0.9713 ± 0.0185**	**0.9699 ± 0.0243**	**0.9768 ± 0.0150**	**0.9843 ± 0.0171**	**0.9738 ± 0.0295**	**0.9954 ± 0.0063**
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.9576 ± 0.0228	0.9552 ± 0.0325	0.9656 ± 0.0188	0.9771 ± 0.0204	0.9624 ± 0.0336	0.9714 ± 0.0216
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	0.9612 ± 0.0239	0.9625 ± 0.0276	0.9688 ± 0.0190	0.9758 ± 0.0203	0.9598 ± 0.0348	0.9910 ± 0.0094
XGBoost	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.01	0.9526 ± 0.0223	0.9501 ± 0.0308	0.9615 ± 0.0186	0.9740 ± 0.0236	0.9574 ± 0.0386	0.9882 ± 0.0113
	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1	0.9727 ± 0.0173	0.9700 ± 0.0254	0.9779 ± 0.0142	0.9864 ± 0.0148	0.9774 ± 0.0248	0.9957 ± 0.0055
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.9592 ± 0.0220	0.9582 ± 0.0274	0.9670 ± 0.0181	0.9764 ± 0.0203	0.9609 ± 0.0341	0.9892 ± 0.0111
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	0.9736 ± 0.0162	0.9710 ± 0.0228	0.9787 ± 0.0131	0.9870 ± 0.0147	**0.9782 ± 0.0252**	0.9956 ± 0.0059
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.01	0.9618 ± 0.0205	0.9576 ± 0.0310	0.9688 ± 0.0175	0.9809 ± 0.0157	0.9688 ± 0.0260	0.9923 ± 0.0073
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1	0.9741 ± 0.0163	0.9722 ± 0.0213	0.9791 ± 0.0133	0.9864 ± 0.0137	0.9772 ± 0.0228	0.9959 ± 0.0059
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.9655 ± 0.0197	0.9618 ± 0.0283	0.9720 ± 0.0163	0.9831 ± 0.0163	0.9719 ± 0.0278	0.9932 ± 0.0071
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	**0.9746 ± 0.0165**	**0.9730 ± 0.0218**	**0.9795 ± 0.0135**	**0.9865 ± 0.0133**	0.9775 ± 0.0221	**0.9962 ± 0.0055**
LightGBM	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.9469 ± 0.0262	0.9415 ± 0.0368	0.9567 ± 0.0216	0.9734 ± 0.0210	0.9559 ± 0.0368	0.9892 ± 0.0096
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	**0.9778 ± 0.0161**	**0.9810 ± 0.0189**	**0.9822 ± 0.0129**	0.9837 ± 0.0161	0.9727 ± 0.0272	0.9962 ± 0.0052
	'n_estimators': 100, 'max_depth': 7, 'learning_rate': 0.01	0.9491 ± 0.0215	0.9486 ± 0.0320	0.9586 ± 0.0181	0.9697 ± 0.0217	0.9491 ± 0.0383	0.9897 ± 0.0111
	'n_estimators': 100, 'max_depth': 7, 'learning_rate': 0.1	0.9762 ± 0.0169	0.9781 ± 0.0189	0.9809 ± 0.0135	0.9841 ± 0.0177	0.9734 ± 0.0302	0.9961 ± 0.0049
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.9616 ± 0.0175	0.9623 ± 0.0243	0.9689 ± 0.0143	0.9762 ± 0.0184	0.9606 ± 0.0308	0.9935 ± 0.0054
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	0.9770 ± 0.0152	0.9797 ± 0.0182	0.9816 ± 0.0121	0.9838 ± 0.0161	0.9730 ± 0.0274	0.9960 ± 0.0056
	'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.01	0.9652 ± 0.0185	0.9669 ± 0.0243	0.9718 ± 0.0153	0.9773 ± 0.0179	0.9621 ± 0.0306	0.9937 ± 0.0068
	'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.1	0.9774 ± 0.0154	0.9776 ± 0.0228	0.9817 ± 0.0127	**0.9864 ± 0.0140**	**0.9770 ± 0.0238**	**0.9967 ± 0.0047**

(*continued on next page*)

Table 7 (*continued*).

Model	Hyperparameters	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
MLP	'hidden_layer_sizes': (100,), 'activation': 'relu', 'solver': 'adam'	0.9814 ± 0.0139	0.9806 ± 0.0186	0.9850 ± 0.0113	**0.9896 ± 0.0119**	**0.9827 ± 0.0206**	0.9974 ± 0.0041
	'hidden_layer_sizes': (100,), 'activation': 'relu', 'solver': 'sgd'	0.9735 ± 0.0144	0.9740 ± 0.0207	0.9787 ± 0.0116	0.9837 ± 0.0131	0.9728 ± 0.0227	0.9963 ± 0.0041
	'hidden_layer_sizes': (100,), 'activation': 'tanh', 'solver': 'adam'	0.9814 ± 0.0135	**0.9860 ± 0.0147**	0.9851 ± 0.0108	0.9845 ± 0.0154	0.9739 ± 0.0261	0.9966 ± 0.0050
	'hidden_layer_sizes': (100,), 'activation': 'tanh', 'solver': 'sgd'	0.9725 ± 0.0174	0.9736 ± 0.0202	0.9779 ± 0.0142	0.9825 ± 0.0167	0.9706 ± 0.0281	0.9957 ± 0.0053
	'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'solver': 'adam'	0.9813 ± 0.0133	0.9822 ± 0.0184	0.9850 ± 0.0107	0.9881 ± 0.0137	0.9797 ± 0.0238	0.9969 ± 0.0042
	'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'solver': 'sgd'	0.9732 ± 0.0177	0.9747 ± 0.0225	0.9784 ± 0.0145	0.9825 ± 0.0173	0.9710 ± 0.0280	0.9951 ± 0.0057
	'hidden_layer_sizes': (50, 50), 'activation': 'tanh', 'solver': 'adam'	**0.9825 ± 0.0121**	0.9837 ± 0.0174	**0.9860 ± 0.0098**	0.9886 ± 0.0134	0.9804 ± 0.0241	**0.9975 ± 0.0036**
	'hidden_layer_sizes': (50, 50), 'activation': 'tanh', 'solver': 'sgd'	0.9749 ± 0.0163	0.9794 ± 0.0212	0.9798 ± 0.0132	0.9806 ± 0.0146	0.9675 ± 0.0248	0.9956 ± 0.0044
PyTorch NN	'lr': 0.01, 'max_epochs': 10	0.6646 ± 0.1896	0.5987 ± 0.3807	0.6054 ± 0.3341	0.7440 ± 0.3110	0.7751 ± 0.2764	0.8633 ± 0.1240
	'lr': 0.01, 'max_epochs': 20	0.7654 ± 0.1707	0.7066 ± 0.3151	0.7385 ± 0.2732	0.8955 ± 0.1748	0.8631 ± 0.1848	0.9499 ± 0.0362
	'lr': 0.01, 'max_epochs': 50	0.9054 ± 0.0601	0.8926 ± 0.0994	0.9191 ± 0.0620	0.9572 ± 0.0404	0.9273 ± 0.0847	0.9797 ± 0.0164
	'lr': 0.1, 'max_epochs': 10	0.9371 ± 0.0249	0.9396 ± 0.0373	0.9489 ± 0.0209	0.9598 ± 0.0239	0.9328 ± 0.0423	0.9852 ± 0.0092
	'lr': 0.1, 'max_epochs': 20	0.9621 ± 0.0213	0.9707 ± 0.0259	0.9698 ± 0.0168	0.9696 ± 0.0212	0.9478 ± 0.0382	0.9923 ± 0.0082
	'lr': 0.1, 'max_epochs': 50	**0.9768 ± 0.0154**	**0.9772 ± 0.0197**	**0.9813 ± 0.0125**	**0.9857 ± 0.0137**	**0.9761 ± 0.0232**	**0.9964 ± 0.0046**

Table 8
This table presents a comparative analysis of ML models for predicting car evaluation outcomes. The evaluation includes metrics such as Accuracy, Recall, F1-Score, Precision, Specificity, and AUC, providing a detailed perspective on each model efficacy.

Model	Hyperparameters	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
LogisticRegression	**'C': 0.1, 'solver': 'lbfgs'**	0.6747 ± 0.0297	0.6811 ± 0.0483	**0.5015 ± 0.0342**	**0.3977 ± 0.0318**	0.6727 ± 0.0359	**0.7656 ± 0.0277**
	'C': 0.1, 'solver': 'liblinear'	0.6726 ± 0.0277	**0.6850 ± 0.0630**	0.5007 ± 0.0406	0.3954 ± 0.0340	0.6687 ± 0.0306	0.7635 ± 0.0268
	'C': 1, 'solver': 'lbfgs'	0.6711 ± 0.0265	0.6778 ± 0.0560	0.4971 ± 0.0373	0.3934 ± 0.0331	0.6691 ± 0.0309	0.7612 ± 0.0260
	'C': 1, 'solver': 'liblinear'	**0.6760 ± 0.0262**	0.6672 ± 0.0563	0.4969 ± 0.0354	0.3968 ± 0.0313	**0.6788 ± 0.0326**	0.7629 ± 0.0255
	'C': 10, 'solver': 'lbfgs'	0.6729 ± 0.0279	0.6718 ± 0.0613	0.4962 ± 0.0398	0.3943 ± 0.0345	0.6733 ± 0.0323	0.7620 ± 0.0270
	'C': 10, 'solver': 'liblinear'	0.6731 ± 0.0209	0.6805 ± 0.0554	0.4993 ± 0.0334	0.3951 ± 0.0282	0.6706 ± 0.0270	0.7622 ± 0.0249
	'C': 100, 'solver': 'lbfgs'	0.6694 ± 0.0290	0.6735 ± 0.0657	0.4940 ± 0.0439	0.3909 ± 0.0369	0.6680 ± 0.0313	0.7598 ± 0.0264
	'C': 100, 'solver': 'liblinear'	0.6718 ± 0.0277	0.6653 ± 0.0550	0.4931 ± 0.0384	0.3924 ± 0.0342	0.6737 ± 0.0318	0.7616 ± 0.0280
SVM	'C': 1, 'kernel': 'linear', 'gamma': 'auto'	0.6607 ± 0.0310	0.7356 ± 0.0570	0.5101 ± 0.0389	0.3913 ± 0.0350	0.6371 ± 0.0359	0.7599 ± 0.0304
	'C': 1, 'kernel': 'linear', 'gamma': 'scale'	0.6629 ± 0.0269	0.7473 ± 0.0592	0.5153 ± 0.0353	0.3940 ± 0.0305	0.6360 ± 0.0349	0.7636 ± 0.0289
	'C': 1, 'kernel': 'rbf', 'gamma': 'auto'	0.8951 ± 0.0195	0.9923 ± 0.0124	0.8202 ± 0.0287	0.7002 ± 0.0428	0.8644 ± 0.0264	0.9907 ± 0.0044
	'C': 1, 'kernel': 'rbf', 'gamma': 'scale'	0.8929 ± 0.0207	**0.9953 ± 0.0113**	0.8176 ± 0.0306	0.6949 ± 0.0441	0.8605 ± 0.0272	0.9906 ± 0.0046
	'C': 10, 'kernel': 'linear', 'gamma': 'auto'	0.6580 ± 0.0294	0.7410 ± 0.0570	0.5098 ± 0.0352	0.3894 ± 0.0308	0.6317 ± 0.0378	0.7565 ± 0.0270
	'C': 10, 'kernel': 'linear', 'gamma': 'scale'	0.6614 ± 0.0302	0.7427 ± 0.0582	0.5129 ± 0.0399	0.3924 ± 0.0349	0.6356 ± 0.0356	0.7596 ± 0.0282
	'C': 10, 'kernel': 'rbf', 'gamma': 'auto'	**0.9725 ± 0.0141**	0.9949 ± 0.0095	**0.9460 ± 0.0265**	**0.9028 ± 0.0461**	**0.9655 ± 0.0181**	**0.9969 ± 0.0033**
	'C': 10, 'kernel': 'rbf', 'gamma': 'scale'	0.9708 ± 0.0122	0.9941 ± 0.0105	0.9425 ± 0.0236	0.8970 ± 0.0406	0.9635 ± 0.0155	0.9965 ± 0.0034
RandomForest	'n_estimators': 100, 'max_depth': 10, 'min_samples_split': 10	0.9415 ± 0.0177	0.9941 ± 0.0131	0.8915 ± 0.0301	0.8096 ± 0.0500	0.9249 ± 0.0235	0.9953 ± 0.0033
	'n_estimators': 100, 'max_depth': 10, 'min_samples_split': 2	0.9565 ± 0.0147	0.9944 ± 0.0106	0.9168 ± 0.0271	0.8516 ± 0.0457	0.9446 ± 0.0191	0.9975 ± 0.0019
	'n_estimators': 100, 'max_depth': None, 'min_samples_split': 10	0.9412 ± 0.0177	**0.9965 ± 0.0072**	0.8910 ± 0.0309	0.8071 ± 0.0499	0.9238 ± 0.0229	0.9955 ± 0.0028
	'n_estimators': 100, 'max_depth': None, 'min_samples_split': 2	**0.9605 ± 0.0163**	0.9954 ± 0.0111	**0.9241 ± 0.0299**	**0.8638 ± 0.0505**	**0.9495 ± 0.0210**	**0.9979 ± 0.0019**
	'n_estimators': 200, 'max_depth': 10, 'min_samples_split': 10	0.9423 ± 0.0177	0.9951 ± 0.0101	0.8927 ± 0.0308	0.8108 ± 0.0492	0.9257 ± 0.0227	0.9957 ± 0.0034
	'n_estimators': 200, 'max_depth': 10, 'min_samples_split': 2	0.9568 ± 0.0134	0.9952 ± 0.0104	0.9173 ± 0.0249	0.8515 ± 0.0411	0.9446 ± 0.0171	0.9978 ± 0.0020
	'n_estimators': 200, 'max_depth': None, 'min_samples_split': 10	0.9403 ± 0.0187	0.9965 ± 0.0081	0.8897 ± 0.0318	0.8053 ± 0.0522	0.9226 ± 0.0247	0.9959 ± 0.0029
	'n_estimators': 200, 'max_depth': None, 'min_samples_split': 2	0.9556 ± 0.0141	0.9962 ± 0.0085	0.9154 ± 0.0252	0.8479 ± 0.0429	0.9427 ± 0.0187	0.9978 ± 0.0017
GradientBoosting	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.01	0.8422 ± 0.0169	1.0000 ± 0.0000	0.7529 ± 0.0234	0.6042 ± 0.0302	0.7924 ± 0.0219	0.9509 ± 0.0138
	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1	0.9668 ± 0.0105	0.9982 ± 0.0066	0.9353 ± 0.0202	0.8806 ± 0.0353	0.9569 ± 0.0137	0.9969 ± 0.0031
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.9248 ± 0.0175	0.9958 ± 0.0180	0.8645 ± 0.0294	0.7651 ± 0.0453	0.9024 ± 0.0229	0.9874 ± 0.0061
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	0.9819 ± 0.0105	0.9966 ± 0.0095	0.9636 ± 0.0208	0.9335 ± 0.0368	0.9773 ± 0.0131	**0.9985 ± 0.0018**
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.01	0.8613 ± 0.0257	**0.9991 ± 0.0094**	0.7768 ± 0.0335	0.6366 ± 0.0446	0.8177 ± 0.0339	0.9765 ± 0.0109
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1	0.9814 ± 0.0093	0.9984 ± 0.0050	0.9629 ± 0.0185	0.9303 ± 0.0335	0.9761 ± 0.0119	0.9981 ± 0.0030
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.9406 ± 0.0174	0.9982 ± 0.0090	0.8904 ± 0.0294	0.8050 ± 0.0478	0.9225 ± 0.0228	0.9940 ± 0.0035
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	**0.9826 ± 0.0102**	0.9950 ± 0.0091	**0.9651 ± 0.0200**	**0.9374 ± 0.0340**	**0.9786 ± 0.0123**	0.9984 ± 0.0027
XGBoost	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.01	0.8436 ± 0.0215	**1.0000 ± 0.0000**	0.7549 ± 0.0280	0.6071 ± 0.0361	0.7942 ± 0.0281	0.9471 ± 0.0168
	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1	0.9571 ± 0.0141	0.9991 ± 0.0043	0.9185 ± 0.0252	0.8509 ± 0.0431	0.9439 ± 0.0186	0.9958 ± 0.0038
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.9087 ± 0.0227	0.9957 ± 0.0142	0.8406 ± 0.0344	0.7288 ± 0.0503	0.8812 ± 0.0295	0.9795 ± 0.0093
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	0.9723 ± 0.0112	0.9989 ± 0.0041	0.9455 ± 0.0213	0.8984 ± 0.0384	0.9638 ± 0.0148	0.9981 ± 0.0022
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.01	0.8528 ± 0.0232	**1.0000 ± 0.0000**	0.7661 ± 0.0308	0.6218 ± 0.0402	0.8063 ± 0.0302	0.9720 ± 0.0118
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1	0.9707 ± 0.0124	0.9993 ± 0.0045	0.9428 ± 0.0231	0.8932 ± 0.0410	0.9617 ± 0.0163	0.9978 ± 0.0032
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.9139 ± 0.0226	0.9991 ± 0.0043	0.8488 ± 0.0356	0.7396 ± 0.0545	0.8871 ± 0.0298	0.9847 ± 0.0084
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	**0.9790 ± 0.0110**	0.9955 ± 0.0103	**0.9581 ± 0.0214**	**0.9243 ± 0.0394**	**0.9737 ± 0.0145**	**0.9983 ± 0.0017**
LightGBM	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.8664 ± 0.0197	0.9859 ± 0.0254	0.7803 ± 0.0286	0.6467 ± 0.0382	0.8286 ± 0.0270	0.9685 ± 0.0094
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	0.9786 ± 0.0103	0.9967 ± 0.0107	0.9574 ± 0.0202	0.9218 ± 0.0353	0.9729 ± 0.0131	0.9984 ± 0.0023
	'n_estimators': 100, 'max_depth': 7, 'learning_rate': 0.01	0.8750 ± 0.0219	0.9862 ± 0.0240	0.7918 ± 0.0312	0.6628 ± 0.0434	0.8398 ± 0.0297	0.9728 ± 0.0092
	'n_estimators': 100, 'max_depth': 7, 'learning_rate': 0.1	0.9790 ± 0.0111	0.9974 ± 0.0065	0.9582 ± 0.0211	0.9229 ± 0.0387	0.9732 ± 0.0146	0.9980 ± 0.0021
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.9074 ± 0.0263	0.9971 ± 0.0094	0.8394 ± 0.0396	0.7268 ± 0.0589	0.8790 ± 0.0346	0.9847 ± 0.0078
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	0.9811 ± 0.0103	0.9965 ± 0.0107	0.9621 ± 0.0206	0.9306 ± 0.0358	0.9763 ± 0.0127	0.9982 ± 0.0025
	'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.01	0.9170 ± 0.0260	0.9967 ± 0.0079	0.8536 ± 0.0401	0.7486 ± 0.0611	0.8919 ± 0.0339	0.9880 ± 0.0063
	'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.1	**0.9842 ± 0.0098**	**0.9977 ± 0.0059**	**0.9683 ± 0.0189**	**0.9412 ± 0.0353**	**0.9799 ± 0.0129**	**0.9989 ± 0.0017**

(*continued on next page*)

Table 8 (*continued*).

Model	Hyperparameters	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
MLP	'hidden_layer_sizes': (100,), 'activation': 'relu', 'solver': 'adam'	0.9652 ± 0.0163	**0.9920 ± 0.0126**	0.9322 ± 0.0298	0.8806 ± 0.0490	0.9568 ± 0.0207	**0.9957 ± 0.0041**
	'hidden_layer_sizes': (100,), 'activation': 'relu', 'solver': 'sgd'	0.8723 ± 0.0252	0.9798 ± 0.0197	0.7870 ± 0.0387	0.6593 ± 0.0519	0.8384 ± 0.0316	0.9613 ± 0.0129
	'hidden_layer_sizes': (100,), 'activation': 'tanh', 'solver': 'adam'	0.9097 ± 0.1034	0.9380 ± 0.1275	0.8454 ± 0.1581	0.7742 ± 0.1728	0.9007 ± 0.0975	0.9570 ± 0.0826
	'hidden_layer_sizes': (100,), 'activation': 'tanh', 'solver': 'sgd'	0.6710 ± 0.0243	0.6945 ± 0.0598	0.5028 ± 0.0392	0.3948 ± 0.0330	0.6633 ± 0.0275	0.7618 ± 0.0279
	'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'solver': 'adam'	**0.9659 ± 0.0180**	0.9867 ± 0.0138	**0.9334 ± 0.0322**	**0.8875 ± 0.0553**	**0.9593 ± 0.0236**	0.9950 ± 0.0051
	'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'solver': 'sgd'	0.9066 ± 0.0197	0.9838 ± 0.0182	0.8354 ± 0.0324	0.7273 ± 0.0477	0.8823 ± 0.0258	0.9788 ± 0.0085
	'hidden_layer_sizes': (50, 50), 'activation': 'tanh', 'solver': 'adam'	0.9501 ± 0.0487	0.9853 ± 0.0388	0.9086 ± 0.0692	0.8462 ± 0.0894	0.9390 ± 0.0547	0.9869 ± 0.0339
	'hidden_layer_sizes': (50, 50), 'activation': 'tanh', 'solver': 'sgd'	0.7389 ± 0.0958	0.7842 ± 0.1426	0.5958 ± 0.1396	0.4829 ± 0.1315	0.7245 ± 0.0848	0.8269 ± 0.0914
PyTorch NN	'lr': 0.01, 'max_epochs': 10	0.5077 ± 0.1932	0.5929 ± 0.3667	0.3220 ± 0.1334	0.2823 ± 0.1129	0.4804 ± 0.3616	0.5908 ± 0.0858
	'lr': 0.01, 'max_epochs': 20	0.5652 ± 0.1758	0.5501 ± 0.3192	0.3455 ± 0.1140	0.3252 ± 0.1127	0.5722 ± 0.3228	0.6399 ± 0.0672
	'lr': 0.01, 'max_epochs': 50	0.6161 ± 0.1109	0.7145 ± 0.1812	0.4714 ± 0.0589	0.3721 ± 0.0665	0.5847 ± 0.1907	0.7255 ± 0.0505
	'lr': 0.1, 'max_epochs': 10	0.6622 ± 0.0407	0.7596 ± 0.0852	0.5190 ± 0.0418	0.3962 ± 0.0364	0.6309 ± 0.0640	0.7639 ± 0.0332
	'lr': 0.1, 'max_epochs': 20	0.6852 ± 0.0299	0.7418 ± 0.0646	0.5306 ± 0.0363	0.4144 ± 0.0332	0.6673 ± 0.0409	0.7848 ± 0.0273
	'lr': 0.1, 'max_epochs': 50	**0.7485 ± 0.0397**	**0.8756 ± 0.0587**	**0.6267 ± 0.0485**	**0.4903 ± 0.0535**	**0.7084 ± 0.0513**	**0.8669 ± 0.0346**

Table 9
This table presents a comparative analysis of ML models for predicting Santander customer satisfaction outcomes. The evaluation includes metrics such as Accuracy, Recall, F1-Score, Precision, Specificity, and AUC, providing a detailed perspective on each model efficacy.

Model	Hyperparameters	Accuracy	Recall	F1-Score	Precision	Specificity	AUC
LogisticRegression	'C': 0.1, 'solver': 'lbfgs'	0.6856 ± 0.0101	0.7630 ± 0.0179	0.1605 ± 0.0052	0.0897 ± 0.0032	0.6824 ± 0.0107	**0.8003 ± 0.0077**
	'C': 0.1, 'solver': 'liblinear'	0.6868 ± 0.0110	0.7603 ± 0.0161	0.1606 ± 0.0062	0.0898 ± 0.0038	0.6838 ± 0.0116	0.7997 ± 0.0082
	'C': 1, 'solver': 'lbfgs'	**0.6888 ± 0.0102**	0.7615 ± 0.0195	0.1616 ± 0.0051	**0.0904 ± 0.0031**	**0.6858 ± 0.0108**	0.7975 ± 0.0087
	'C': 1, 'solver': 'liblinear'	0.6845 ± 0.0109	0.7523 ± 0.0201	0.1582 ± 0.0070	0.0884 ± 0.0041	0.6817 ± 0.0111	0.7924 ± 0.0093
	'C': 10, 'solver': 'lbfgs'	0.6868 ± 0.0081	0.7671 ± 0.0205	**0.1617 ± 0.0060**	0.0904 ± 0.0035	0.6836 ± 0.0083	0.7966 ± 0.0086
	'C': 10, 'solver': 'liblinear'	0.6823 ± 0.0100	**0.7692 ± 0.0184**	0.1602 ± 0.0052	0.0894 ± 0.0032	0.6788 ± 0.0107	0.7964 ± 0.0092
	'C': 100, 'solver': 'lbfgs'	0.6879 ± 0.0104	0.7604 ± 0.0196	0.1610 ± 0.0045	0.0900 ± 0.0028	0.6850 ± 0.0111	0.7943 ± 0.0105
	'C': 100, 'solver': 'liblinear'	0.6886 ± 0.0132	0.7608 ± 0.0181	0.1615 ± 0.0083	0.0904 ± 0.0050	0.6857 ± 0.0136	0.7953 ± 0.0103
SVM	'C': 1, 'kernel': 'linear', 'gamma': 'auto'	0.6744 ± 0.0079	0.7576 ± 0.0184	0.1548 ± 0.0041	0.0862 ± 0.0025	0.6710 ± 0.0086	0.7946 ± 0.0070
	'C': 1, 'kernel': 'linear', 'gamma': 'scale'	0.6791 ± 0.0105	0.7528 ± 0.0260	0.1559 ± 0.0051	0.0870 ± 0.0030	0.6760 ± 0.0115	0.7956 ± 0.0099
	'C': 1, 'kernel': 'rbf', 'gamma': 'auto'	**0.6986 ± 0.0113**	0.7271 ± 0.0233	0.1597 ± 0.0053	0.0897 ± 0.0033	**0.6974 ± 0.0124**	0.7986 ± 0.0089
	'C': 1, 'kernel': 'rbf', 'gamma': 'scale'	0.6941 ± 0.0080	0.7406 ± 0.0256	0.1601 ± 0.0065	0.0898 ± 0.0038	0.6922 ± 0.0085	0.8014 ± 0.0111
	'C': 10, 'kernel': 'linear', 'gamma': 'auto'	0.6778 ± 0.0079	0.7411 ± 0.0269	0.1533 ± 0.0051	0.0855 ± 0.0029	0.6752 ± 0.0088	0.7876 ± 0.0100
	'C': 10, 'kernel': 'linear', 'gamma': 'scale'	0.6769 ± 0.0072	0.7460 ± 0.0217	0.1538 ± 0.0039	0.0857 ± 0.0023	0.6741 ± 0.0080	0.7902 ± 0.0067
	'C': 10, 'kernel': 'rbf', 'gamma': 'auto'	0.6815 ± 0.0076	0.7763 ± 0.0215	0.1610 ± 0.0060	0.0898 ± 0.0035	0.6776 ± 0.0078	0.8115 ± 0.0120
	'C': 10, 'kernel': 'rbf', 'gamma': 'scale'	0.6877 ± 0.0106	**0.7856 ± 0.0187**	**0.1654 ± 0.0058**	**0.0924 ± 0.0036**	0.6837 ± 0.0113	**0.8177 ± 0.0091**
RandomForest	'n_estimators': 100, 'max_depth': 10, 'min_samples_split': 10	0.7528 ± 0.0214	0.8010 ± 0.0323	**0.2038 ± 0.0098**	**0.1169 ± 0.0068**	0.7508 ± 0.0233	0.8320 ± 0.0080
	'n_estimators': 100, 'max_depth': 10, 'min_samples_split': 2	**0.7555 ± 0.0069**	0.7826 ± 0.0179	0.2013 ± 0.0057	0.1155 ± 0.0037	**0.7544 ± 0.0073**	0.8370 ± 0.0086
	'n_estimators': 100, 'max_depth': None, 'min_samples_split': 10	0.7467 ± 0.0256	0.8101 ± 0.0266	0.2021 ± 0.0127	0.1156 ± 0.0085	0.7441 ± 0.0274	**0.8378 ± 0.0103**
	'n_estimators': 100, 'max_depth': None, 'min_samples_split': 2	0.7531 ± 0.0060	0.7758 ± 0.0168	0.1983 ± 0.0056	0.1137 ± 0.0036	0.7521 ± 0.0065	0.8323 ± 0.0060
	'n_estimators': 200, 'max_depth': 10, 'min_samples_split': 10	0.7460 ± 0.0205	0.7967 ± 0.0294	0.1986 ± 0.0090	0.1135 ± 0.0061	0.7439 ± 0.0223	0.8210 ± 0.0109
	'n_estimators': 200, 'max_depth': 10, 'min_samples_split': 2	0.7477 ± 0.0081	0.7944 ± 0.0167	0.1987 ± 0.0069	0.1136 ± 0.0045	0.7458 ± 0.0087	0.8283 ± 0.0063
	'n_estimators': 200, 'max_depth': None, 'min_samples_split': 10	0.7435 ± 0.0202	**0.8123 ± 0.0250**	0.2001 ± 0.0090	0.1142 ± 0.0062	0.7407 ± 0.0219	0.8282 ± 0.0054
	'n_estimators': 200, 'max_depth': None, 'min_samples_split': 2	0.7466 ± 0.0098	0.7858 ± 0.0207	0.1964 ± 0.0073	0.1122 ± 0.0046	0.7450 ± 0.0104	0.8279 ± 0.0106
GradientBoosting	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.01	0.7500 ± 0.0211	0.7677 ± 0.0296	0.1953 ± 0.0112	0.1120 ± 0.0074	0.7493 ± 0.0227	0.8295 ± 0.0091
	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1	0.7663 ± 0.0089	0.7979 ± 0.0176	0.2120 ± 0.0069	0.1222 ± 0.0046	0.7650 ± 0.0097	0.8545 ± 0.0078
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.7710 ± 0.0111	0.7698 ± 0.0216	0.2094 ± 0.0081	0.1212 ± 0.0054	0.7710 ± 0.0119	0.8446 ± 0.0074
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	0.7756 ± 0.0085	0.8196 ± 0.0213	0.2235 ± 0.0070	0.1294 ± 0.0046	0.7738 ± 0.0093	0.8685 ± 0.0067
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.01	0.7518 ± 0.0150	0.7731 ± 0.0267	0.1972 ± 0.0072	0.1130 ± 0.0048	0.7509 ± 0.0165	0.8370 ± 0.0071
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1	0.7676 ± 0.0094	0.8029 ± 0.0183	0.2139 ± 0.0059	0.1234 ± 0.0040	0.7661 ± 0.0102	0.8563 ± 0.0071
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.7695 ± 0.0081	0.7901 ± 0.0196	0.2126 ± 0.0075	0.1229 ± 0.0048	0.7687 ± 0.0085	0.8512 ± 0.0071
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	**0.7791 ± 0.0059**	**0.8409 ± 0.0150**	**0.2307 ± 0.0069**	**0.1337 ± 0.0045**	**0.7766 ± 0.0063**	**0.8773 ± 0.0073**
XGBoost	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.01	0.7389 ± 0.0143	0.7685 ± 0.0209	0.1884 ± 0.0077	0.1074 ± 0.0051	0.7376 ± 0.0154	0.8267 ± 0.0082
	'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1	0.7611 ± 0.0068	0.7878 ± 0.0183	0.2062 ± 0.0056	0.1186 ± 0.0036	0.7600 ± 0.0074	0.8491 ± 0.0081
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.7700 ± 0.0156	0.7705 ± 0.0171	0.2092 ± 0.0106	0.1211 ± 0.0072	0.7700 ± 0.0166	0.8427 ± 0.0067
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	0.7734 ± 0.0055	0.7983 ± 0.0189	0.2171 ± 0.0047	0.1257 ± 0.0031	0.7724 ± 0.0059	0.8618 ± 0.0073
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.01	0.7477 ± 0.0161	0.7750 ± 0.0268	0.1951 ± 0.0088	0.1117 ± 0.0059	0.7466 ± 0.0175	0.8360 ± 0.0078
	'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1	0.7672 ± 0.0110	0.8012 ± 0.0209	0.2134 ± 0.0074	0.1231 ± 0.0050	0.7658 ± 0.0121	0.8573 ± 0.0068
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.7689 ± 0.0088	0.7803 ± 0.0189	0.2101 ± 0.0053	0.1214 ± 0.0036	0.7685 ± 0.0097	0.8495 ± 0.0051
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	**0.7774 ± 0.0070**	**0.8242 ± 0.0222**	**0.2257 ± 0.0056**	**0.1308 ± 0.0037**	**0.7755 ± 0.0079**	**0.8704 ± 0.0059**
LightGBM	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.01	0.7716 ± 0.0116	0.7621 ± 0.0243	0.2082 ± 0.0085	0.1206 ± 0.0055	0.7720 ± 0.0125	0.8413 ± 0.0099
	'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1	0.7720 ± 0.0108	0.8075 ± 0.0168	0.2182 ± 0.0077	0.1262 ± 0.0052	0.7705 ± 0.0117	0.8622 ± 0.0068
	'n_estimators': 100, 'max_depth': 7, 'learning_rate': 0.01	0.7708 ± 0.0093	0.7759 ± 0.0148	0.2106 ± 0.0072	0.1219 ± 0.0048	0.7706 ± 0.0099	0.8453 ± 0.0075
	'n_estimators': 100, 'max_depth': 7, 'learning_rate': 0.1	0.7762 ± 0.0070	0.8229 ± 0.0132	0.2246 ± 0.0081	0.1301 ± 0.0053	0.7743 ± 0.0071	0.8706 ± 0.0070
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01	0.7652 ± 0.0147	0.7862 ± 0.0174	0.2090 ± 0.0110	0.1206 ± 0.0073	0.7643 ± 0.0155	0.8497 ± 0.0084
	'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1	0.7789 ± 0.0091	0.8245 ± 0.0197	0.2271 ± 0.0087	0.1317 ± 0.0058	0.7770 ± 0.0097	0.8712 ± 0.0078
	'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.01	0.7671 ± 0.0087	0.7924 ± 0.0190	0.2114 ± 0.0063	0.1220 ± 0.0042	0.7661 ± 0.0095	0.8537 ± 0.0054
	'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.1	**0.7806 ± 0.0054**	**0.8424 ± 0.0193**	**0.2321 ± 0.0067**	**0.1346 ± 0.0043**	**0.7781 ± 0.0058**	**0.8788 ± 0.0094**
MLP	'hidden_layer_sizes': (100,), 'activation': 'relu', 'solver': 'adam'	0.7353 ± 0.0115	0.7672 ± 0.0240	0.1859 ± 0.0069	0.1058 ± 0.0044	0.7340 ± 0.0122	0.8388 ± 0.0075
	'hidden_layer_sizes': (100,), 'activation': 'relu', 'solver': 'sgd'	0.7499 ± 0.0193	0.7758 ± 0.0180	0.1969 ± 0.0109	0.1129 ± 0.0074	0.7489 ± 0.0205	0.8467 ± 0.0056
	'hidden_layer_sizes': (100,), 'activation': 'tanh', 'solver': 'adam'	0.7987 ± 0.0064	0.8304 ± 0.0120	0.2453 ± 0.0077	0.1439 ± 0.0051	0.7974 ± 0.0065	0.8843 ± 0.0053
	'hidden_layer_sizes': (100,), 'activation': 'tanh', 'solver': 'sgd'	0.8082 ± 0.0084	0.8509 ± 0.0170	0.2590 ± 0.0098	0.1528 ± 0.0066	0.8064 ± 0.0088	0.9016 ± 0.0076
	'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'solver': 'adam'	0.7430 ± 0.0086	0.7646 ± 0.0120	0.1899 ± 0.0060	0.1084 ± 0.0040	0.7421 ± 0.0093	0.8399 ± 0.0071
	'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'solver': 'sgd'	0.7421 ± 0.0159	0.7658 ± 0.0145	0.1899 ± 0.0105	0.1085 ± 0.0069	0.7411 ± 0.0167	0.8431 ± 0.0071
	'hidden_layer_sizes': (50, 50), 'activation': 'tanh', 'solver': 'adam'	0.7983 ± 0.0063	0.8280 ± 0.0166	0.2443 ± 0.0074	0.1433 ± 0.0050	0.7971 ± 0.0068	0.8837 ± 0.0067
	'hidden_layer_sizes': (50, 50), 'activation': 'tanh', 'solver': 'sgd'	**0.8085 ± 0.0068**	**0.8568 ± 0.0172**	**0.2606 ± 0.0094**	**0.1537 ± 0.0064**	**0.8065 ± 0.0071**	**0.9034 ± 0.0066**
PyTorch NN	'max_epochs': 100, 'module__activation': ELU(alpha=1.0)	0.7122 ± 0.0629	0.8147 ± 0.0543	0.1824 ± 0.0120	0.1027 ± 0.0077	0.7080 ± 0.0674	0.8286 ± 0.0130
	'max_epochs': 100, 'module__activation': LeakyReLU(negative_slope=0.01)	0.7725 ± 0.0111	0.7663 ± 0.0235	0.2097 ± 0.0055	0.1215 ± 0.0033	0.7727 ± 0.0121	0.8212 ± 0.0116
	'max_epochs': 100, 'module__activation': ReLU()	**0.7770 ± 0.0120**	**0.8114 ± 0.0235**	**0.2228 ± 0.0054**	**0.1292 ± 0.0032**	**0.7756 ± 0.0130**	**0.8388 ± 0.0114**
	'max_epochs': 50, 'module__activation': ELU(alpha=1.0)	0.7670 ± 0.0398	0.7496 ± 0.0386	0.2022 ± 0.0165	0.1169 ± 0.0109	0.7677 ± 0.0426	0.8289 ± 0.0111
	'max_epochs': 50, 'module__activation': LeakyReLU(negative_slope=0.01)	0.7668 ± 0.0401	0.7846 ± 0.0472	0.2095 ± 0.0157	0.1209 ± 0.0105	0.7660 ± 0.0432	0.8314 ± 0.0131
	'max_epochs': 50, 'module__activation': ReLU()	0.7462 ± 0.0443	0.8097 ± 0.0439	0.2009 ± 0.0170	0.1147 ± 0.0112	0.7436 ± 0.0474	0.8351 ± 0.0189

References

Adadi, A., Berrada, M., 2018. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). IEEE Access 6, 52138–52160. http://dx.doi.org/10.1109/ACCESS.2018.2870052, URL: https://ieeexplore.ieee.org/document/8466590/.

Adams, D., 1995. The Hitch Hiker's Guide to the Galaxy Omnibus. Random House.

Adebayo, J., Gilmer, J., Goodfellow, I., Kim, B., 2018. Local explanation methods for deep neural networks lack sensitivity to parameter values. http://dx.doi.org/10.48550/arXiv.1810.03307, arXiv preprint arXiv:1810.03307.

Agarwal, C., Dong, B., Schonfeld, D., Hoogs, A., 2018. An explainable adversarial robustness metric for deep learning neural networks. http://dx.doi.org/10.48550/arXiv.1806.01477, arXiv preprint arXiv:1806.01477.

Alam, T.M., Shaukat, K., Hameed, I.A., Luo, S., Sarwar, M.U., Shabbir, S., Li, J., Khushi, M., 2020. An investigation of credit card default prediction in the imbalanced datasets. IEEE Access 8, 201173–201198. http://dx.doi.org/10.1109/ACCESS.2020.3033784.

Alexandrov, V.N., Martel, C.G., Straßburg, J., 2011. Monte Carlo scalable algorithms for computational finance. Procedia Comput. Sci. 4, 1708–1715. http://dx.doi.org/10.1016/j.procs.2011.04.185, Publisher: Elsevier.

Altmann, A., Toloşi, L., Sander, O., Lengauer, T., 2010. Permutation importance: a corrected feature importance measure. Bioinformatics 26 (10), 1340–1347. http://dx.doi.org/10.1093/bioinformatics/btq134, Publisher: Oxford University Press.

Alvarez-Melis, D., Jaakkola, T.S., 2018. On the robustness of interpretability methods. http://dx.doi.org/10.48550/arXiv.1806.08049, arXiv preprint arXiv:1806.08049.

Alvarez Melis, D., Jaakkola, T., 2018. Towards robust interpretability with self-explaining neural networks. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), In: Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/3e9f0fc9b2f89e043bc6233994dfcf76-Paper.pdf.

Amiri, S.S., Weber, R.O., Goel, P., Brooks, O., Gandley, A., Kitchell, B., Zehm, A., 2020. Data representing ground-truth explanations to evaluate xai methods. http://dx.doi.org/10.48550/arXiv.2011.09892, arXiv preprint arXiv:2011.09892.

Amparore, E.G., Perotti, A., Bajardi, P., 2021. To trust or not to trust an explanation: using LEAF to evaluate local linear XAI methods. CoRR abs/2106.00461, arXiv:2106.00461, URL: https://arxiv.org/abs/2106.00461.

Ancona, M., Oztireli, C., Gross, M., 2019. Explaining deep neural networks with a polynomial time algorithm for Shapley value approximation. In: Chaudhuri, K., Salakhutdinov, R. (Eds.), Proceedings of the 36th International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 97, PMLR, pp. 272–281, URL: https://proceedings.mlr.press/v97/ancona19a.html.

Apley, D.W., Zhu, J., 2020. Visualizing the effects of predictor variables in black box supervised learning models. J. R. Stat. Soc. Ser. B Stat. Methodol. 82 (4), 1059–1086. http://dx.doi.org/10.1111/rssb.12377.

Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Inf. Fusion 58, 82–115. http://dx.doi.org/10.1016/j.inffus.2019.12.012, Publisher: Elsevier.

Azar, A.T., El-Metwally, S.M., 2013. Decision tree classifiers for automated medical diagnosis. Neural Comput. Appl. 23, 2387–2403. http://dx.doi.org/10.1007/s00521-012-1196-7.

Azodi, C.B., Tang, J., Shiu, S.H., 2020. Opening the black box: interpretable machine learning for geneticists. TIG 36 (6), 442–455. http://dx.doi.org/10.1016/j.tig.2020.03.005.

Barbado, A., Corcho, Ó., 2022. Interpretable machine learning models for predicting and explaining vehicle fuel consumption anomalies. Eng. Appl. Artif. Intell. 115, 105222. http://dx.doi.org/10.1016/j.engappai.2022.105222, URL: https://www.sciencedirect.com/science/article/pii/S0952197622003049.

Belaid, M.K., Hüllermeier, E., Rabus, M., Krestel, R., 2022. Do we need another explainable AI method? Toward unifying post-hoc XAI evaluation methods into an interactive and multi-dimensional benchmark. http://dx.doi.org/10.48550/arXiv.2207.14160, arXiv:2207.14160 [cs] URL: http://arxiv.org/abs/2207.14160.

Bishop, C.M., Nasrabadi, N.M., 2006. Pattern Recognition and Machine Learning, vol. 4, Springer.

Bohanec, M., 1997. Car evaluation. http://dx.doi.org/10.24432/C5JP48, UCI Machine Learning Repository.

Bommer, P.L., Kretschmer, M., Hedström, A., Bareeva, D., Höhne, M.M.C., 2024. Finding the right XAI method—a guide for the evaluation and ranking of explainable AI methods in climate science. Artif. Intell. Earth Syst. 3 (3), e230074. http://dx.doi.org/10.1175/AIES-D-23-0074.1.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24, 123–140. http://dx.doi.org/10.1007/BF00058655.

Breiman, L., 2001. Random forests. Mach. Learn. 45, 5–32. http://dx.doi.org/10.1023/A:1010933404324.

Van den Broeck, G., Lykov, A., Schleich, M., Suciu, D., 2022. On the tractability of SHAP explanations. J. Artificial Intelligence Res. 74, 851–886. http://dx.doi.org/10.1613/jair.1.13283.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., et al., 2013. API design for machine learning software: experiences from the scikit-learn project. http://dx.doi.org/10.48550/arXiv.1309.0238, arXiv preprint arXiv:1309.0238.

Burkart, N., Huber, M.F., 2021. A survey on the explainability of supervised machine learning. J. Artificial Intelligence Res. 70, 245–317. http://dx.doi.org/10.1613/jair.1.12228.

Carvalho, D.V., Pereira, E.M., Cardoso, J.S., 2019. Machine learning interpretability: A survey on methods and metrics. Electronics 8 (8), 832. http://dx.doi.org/10.3390/electronics8080832.

Chakraborty, S., Tomsett, R., Raghavendra, R., Harborne, D., Alzantot, M., Cerutti, F., Srivastava, M., Preece, A., Julier, S., Rao, R.M., et al., 2017. Interpretability of deep learning models: A survey of results. In: 2017 IEEE Smartworld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (Smartworld/SCALCOM/UIC/ATC/CBDcom/IOP/SCI). IEEE, pp. 1–6. http://dx.doi.org/10.1109/UIC-ATC.2017.8397411.

Chalé, M., Bastian, N.D., Weir, J., 2020. Algorithm selection framework for cyber attack detection. In: Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning. WiseML '20, Association for Computing Machinery, New York, NY, USA, pp. 37–42. http://dx.doi.org/10.1145/3395352.3402623.

Chatterjee, S., Hadi, A.S., 1986. Influential observations, high leverage points, and outliers in linear regression. Statist. Sci. 379–393, URL: https://www.jstor.org/stable/2245477.

Chelouah, R., Siarry, P., 2022. Optimization and Machine Learning: Optimization for Machine Learning and Machine Learning for Optimization. John Wiley & Sons.

Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining. pp. 785–794. http://dx.doi.org/10.1145/2939672.2939785.

Chen, J., Song, L., Wainwright, M.J., Jordan, M.I., 2018. L-shapley and c-shapley: Efficient model interpretation for structured data. http://dx.doi.org/10.48550/arXiv.1808.02610, arXiv preprint arXiv:1808.02610.

Confalonieri, R., Coba, L., Wagner, B., Besold, T.R., 2021. A historical perspective of explainable Artificial Intelligence. WIREs Data Min. Knowl. Discov. 11 (1), e1391, URL: https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1391, 10.1002/widm.1391.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Mach. Learn. 20, 273–297. http://dx.doi.org/10.1007/BF00994018.

Covert, I., Lee, S.I., 2021. Improving KernelSHAP: Practical Shapley value estimation using linear regression. In: Banerjee, A., Fukumizu, K. (Eds.), Proceedings of the 24th International Conference on Artificial Intelligence and Statistics. In: Proceedings of Machine Learning Research, vol. 130, PMLR, pp. 3457–3465, URL: https://proceedings.mlr.press/v130/covert21a.html.

Covert, I., Lundberg, S.M., Lee, S.I., 2020. Understanding global feature contributions with additive importance measures. Adv. Neural Inf. Process. Syst. 33, 17212–17223, URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/c7bf0b7c1a86d5eb3be2c722cf2cf746-Paper.pdf.

Covert, I., Lundberg, S., Lee, S.I., 2021. Explaining by removing: A unified framework for model explanation. J. Mach. Learn. Res. 22 (209), 1–90, URL: http://jmlr.org/papers/v22/20-1316.html.

Doran, D., Schulz, S., Besold, T.R., 2017. What does explainable AI really mean? A new conceptualization of perspectives. http://dx.doi.org/10.48550/arXiv.1710.00794, arXiv preprint arXiv:1710.00794.

Doshi-Velez, F., Kim, B., 2017. Towards a rigorous science of interpretable machine learning. http://dx.doi.org/10.48550/arXiv.1702.08608, arXiv preprint arXiv:1702.08608.

Došilović, F.K., Brčić, M., Hlupić, N., 2018. Explainable artificial intelligence: A survey. In: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics. MIPRO, IEEE, pp. 0210–0215. http://dx.doi.org/10.23919/MIPRO.2018.8400040.

Du, M., Liu, N., Hu, X., 2019. Techniques for interpretable machine learning. Commun. ACM 63 (1), 68–77. http://dx.doi.org/10.1145/3359786.

Ebers, M., 2020. Regulating explainable AI in the European Union. An overview of the current legal framework (s). In: Colonna, L., Greenstein, S. (Eds.), An Overview of the Current Legal Framework (s)(August 9, 2021). Nordic Yearbook of Law and Informatics, http://dx.doi.org/10.2139/ssrn.3901732.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D., 2018. Robust physical-world attacks on deep learning visual classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1625–1634, URL: https://openaccess.thecvf.com/content_cvpr_2018/papers/Eykholt_Robust_Physical-World_Attacks_CVPR_2018_paper.pdf.

Fawcett, T., 2006. An introduction to ROC analysis. Pattern Recognit. Lett. 27 (8), 861–874. http://dx.doi.org/10.1016/j.patrec.2005.10.010.

Friedman, J.H., 1991. Multivariate adaptive regression splines. Ann. Statist. 19 (1), 1–67. http://dx.doi.org/10.1214/aos/1176347963.

Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Ann. Stat. 1189–1232, URL: https://www.jstor.org/stable/2699986.

Frye, C., de Mijolla, D., Begley, T., Cowton, L., Stanley, M., Feige, I., 2020. Shapley explainability on the data manifold. http://dx.doi.org/10.48550/arXiv.2006.01272, arXiv preprint arXiv:2006.01272.

Gaddam, D.K.R., Ansari, M.D., Vuppala, S., Gunjan, V.K., Sati, M.M., 2022. A performance comparison of optimization algorithms on a generated dataset. In: Proceedings of the 2nd International Conference on Data Science, Machine Learning and Applications. ICDSMLA 2020, Springer, pp. 1407–1415. http://dx.doi.org/10.1007/978-981-16-3690-5_135.

Gómez-Talal, I., Bote-Curiel, L., Rojo-Álvarez, J.L., 2024. Understanding the disparities in mathematics performance: An interpretability-based examination. Eng. Appl. Artif. Intell. 133, 108109. http://dx.doi.org/10.1016/j.engappai.2024.108109, URL: https://www.sciencedirect.com/science/article/pii/S0952197624002677.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D., 2018. A survey of methods for explaining black box models. ACM Comput. Surv. 51 (5), 1–42. http://dx.doi.org/10.1145/3236009.

Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A., 2008. Feature Extraction: Foundations and Applications, vol. 207, Springer, http://dx.doi.org/10.1007/978-3-540-35488-8_10.

Hanawa, K., Yokoi, S., Hara, S., Inui, K., 2020. Evaluation of similarity-based explanations. http://dx.doi.org/10.48550/arXiv.2006.04528, arXiv preprint arXiv:2006.04528.

Harris, C.R., Millman, K.J., Van Der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., et al., 2020. Array programming with NumPy. Nature 585 (7825), 357–362. http://dx.doi.org/10.1038/s41586-020-2649-2.

Hoffman, R.R., Mueller, S.T., Klein, G., Litman, J., 2018. Metrics for explainable AI: Challenges and prospects. http://dx.doi.org/10.48550/arXiv.1812.04608, arXiv preprint arXiv:1812.04608.

Hofmann, H., 1994. Statlog (german credit data). http://dx.doi.org/10.24432/C5NC77, UCI Machine Learning Repository.

Hosmer, Jr., D.W., Lemeshow, S., Sturdivant, R.X., 2013. Applied Logistic Regression, vol. 398, John Wiley & Sons.

Hu, B., Vasu, B., Hoogs, A., 2022. X-MIR: Explainable medical image retrieval. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. WACV, pp. 440–450, URL: https://openaccess.thecvf.com/content/WACV2022/papers/Hu_X-MIR_EXplainable_Medical_Image_Retrieval_WACV_2022_paper.pdf.

Jean-Quartier, C., Bein, K., Hejny, L., Hofer, E., Holzinger, A., Jeanquartier, F., 2023. The cost of understanding—XAI algorithms towards sustainable ML in the view of computational cost. Computation 11 (5), 92. http://dx.doi.org/10.3390/computation11050092.

Jethani, N., Sudarshan, M., Covert, I.C., Lee, S.I., Ranganath, R., 2022. FastSHAP: Real-time Shapley value estimation. In: International Conference on Learning Representations. URL: https://openreview.net/forum?id=Zq2G_VTV53T.

Jimenez, S., Cukierski, W., 2016. Santander customer satisfaction. URL: https://kaggle.com/competitions/santander-customer-satisfaction.

Joblib Development Team, 2024. Joblib: Efficiently run python functions as pipeline jobs. URL: https://joblib.readthedocs.io/en/stable/ (Accessed 25 June 2024).

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y., 2017. Lightgbm: A highly efficient gradient boosting decision tree. Adv. Neural Inf. Process. Syst. 30, URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Koh, P.W., Liang, P., 2017. Understanding black-box predictions via influence functions. In: International Conference on Machine Learning. PMLR, pp. 1885–1894, URL: https://proceedings.mlr.press/v70/koh17a.html.

Krishnan, R., Sivakumar, G., Bhattacharya, P., 1999. Extracting decision trees from trained neural networks. Pattern Recognit. 32 (12), 1999–2009. http://dx.doi.org/10.1016/S0031-3203(98)00181-2, URL: https://www.sciencedirect.com/science/article/pii/S0031320398001812.

Kumar, I.E., Venkatasubramanian, S., Scheidegger, C., Friedler, S., 2020. Problems with Shapley-value-based explanations as feature importance measures. In: International Conference on Machine Learning. PMLR, pp. 5491–5500, URL: https://proceedings.mlr.press/v119/kumar20e.html.

Lakkaraju, H., Kamar, E., Caruana, R., Leskovec, J., 2019. Faithful and customizable explanations of black box models. In: Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society. pp. 131–138. http://dx.doi.org/10.1145/3306618.3314229.

Leavitt, M.L., Morcos, A., 2020. Towards falsifiable interpretability research. http://dx.doi.org/10.48550/arXiv.2010.12016, arXiv preprint arXiv:2010.12016.

Lin, Y.S., Lee, W.C., Celik, Z.B., 2021a. What do you see? Evaluation of explainable artificial intelligence (XAI) interpretability through neural backdoors. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 1027–1035. http://dx.doi.org/10.1145/3447548.3467213.

Lin, Y.S., Liu, Z.Y., Chen, Y.A., Wang, Y.S., Chang, Y.L., Hsu, W.H., 2021b. Xcos: An explainable cosine metric for face verification task. ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) 17 (3s), 1–16. http://dx.doi.org/10.1145/3469288.

Lipton, Z.C., 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue 16 (3), 31–57.

Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.I., 2019. Explainable AI for trees: From local explanations to global understanding. http://dx.doi.org/10.48550/arXiv.1905.04610, arXiv preprint arXiv:1905.04610.

Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.I., 2020. From local explanations to global understanding with explainable AI for trees. Nat. Mach. Intell. 2 (1), 56–67. http://dx.doi.org/10.1038/s42256-019-0138-9.

Lundberg, S.M., Lee, S.I., 2017. A unified approach to interpreting model predictions. Adv. Neural Inf. Process. Syst. 30, URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

Lyuu, Y.D., 2002. Financial Engineering and Computation: Principles, Mathematics, Algorithms. Cambridge University Press.

Merrick, L., Taly, A., 2020. The explanation game: Explaining machine learning models using shapley values. In: Machine Learning and Knowledge Extraction: 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020, Dublin, Ireland, August 25–28, 2020, Proceedings 4. Springer, pp. 17–38. http://dx.doi.org/10.1007/978-3-030-57321-8_2.

Messalas, A., Kanellopoulos, Y., Makris, C., 2019. Model-agnostic interpretability with Shapley values. In: 2019 10th International Conference on Information, Intelligence, Systems and Applications. IISA, IEEE, PATRAS, Greece, pp. 1–7. http://dx.doi.org/10.1109/IISA.2019.8900669, URL: https://ieeexplore.ieee.org/document/8900669/.

Minh, D., Wang, H.X., Li, Y.F., Nguyen, T.N., 2022. Explainable artificial intelligence: a comprehensive review. Artif. Intell. Rev. 55 (5), 3503–3568. http://dx.doi.org/10.1007/s10462-021-10088-y, URL: https://link.springer.com/10.1007/s10462-021-10088-y.

Molnar, C., Casalicchio, G., Bischl, B., 2020a. Interpretable machine learning–a brief history, state-of-the-art and challenges. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, pp. 417–431. http://dx.doi.org/10.1007/978-3-030-65965-3_28.

Molnar, C., Casalicchio, G., Bischl, B., 2020b. Quantifying model complexity via functional decomposition for better post-hoc interpretability. In: Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I. Springer, pp. 193–204.

Mothilal, R.K., Sharma, A., Tan, C., 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. pp. 607–617. http://dx.doi.org/10.1145/3351095.3372850.

Ng, C.H., Abuwala, H.S., Lim, C.H., 2022. Towards more stable lime for explainable ai. In: 2022 International Symposium on Intelligent Signal Processing and Communication Systems. ISPACS, IEEE, pp. 1–4. http://dx.doi.org/10.1109/ISPACS57703.2022.10082810.

Nguyen, H.T.T., Cao, H.Q., Nguyen, K.V.T., Pham, N.D.K., 2021. Evaluation of explainable artificial intelligence: Shap, lime, and cam. In: Proceedings of the FPT AI Conference. pp. 1–6.

Pasquale, F., 2015. The Black Box Society. Harvard University Press, URL: http://www.jstor.org/stable/j.ctt13x0hch.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A., 2017. Automatic differentiation in PyTorch. In: NIPS-W.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, P., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. PyTorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., dÁlché-Buc, F., Fox, E., Garnett, R. (Eds.), In: Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., pp. 8024–8035, URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Pawlicki, M., 2023. Towards quality measures for xAI algorithms: Explanation stability. In: 2023 IEEE 10th International Conference on Data Science and Advanced Analytics. DSAA, IEEE, pp. 1–10. http://dx.doi.org/10.1109/DSAA60987.2023.10302535.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Plumb, G., Molitor, D., Talwalkar, A.S., 2018. Model agnostic supervised local explanations. Adv. Neural Inf. Process. Syst. 31, URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/b495ce63ede0f4efc9eec62cb947c162-Paper.pdf.

Powers, D.M., 2020. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. http://dx.doi.org/10.48550/arXiv.2010.16061, arXiv preprint arXiv:2010.16061.

Prusty, S., Patnaik, S., Dash, S.K., 2022. SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer. Front. Nanotechnol. 4, 972421. http://dx.doi.org/10.3389/fnano.2022.972421.

Purushotham, S., Tripathy, B., 2011. Evaluation of classifier models using stratified tenfold cross validation techniques. In: International Conference on Computing and Communication Systems. Springer, pp. 680–690. http://dx.doi.org/10.1007/978-3-642-29216-3_74.

Ribeiro, M.T., Singh, S., Guestrin, C., 2016a. Model-agnostic interpretability of machine learning. http://dx.doi.org/10.48550/arXiv.1606.05386, arXiv preprint arXiv:1606.05386.

Ribeiro, M.T., Singh, S., Guestrin, C., 2016b. "Why should I trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, San Francisco California USA, pp. 1135–1144. http://dx.doi.org/10.1145/2939672.2939778.

Ribeiro, M.T., Singh, S., Guestrin, C., 2018. Anchors: High-precision model-agnostic explanations. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, (1), http://dx.doi.org/10.1609/aaai.v32i1.11491.

Rojat, T., Puget, R., Filliat, D., Del Ser, J., Gelin, R., Díaz-Rodríguez, N., 2021. Explainable artificial intelligence (xai) on timeseries data: A survey. http://dx.doi.org/10.48550/arXiv.2104.00950, arXiv preprint arXiv:2104.00950.

Roth, A.E., 1988. Introduction to the Shapley value. In: The Shapley Value: Essays in Honor of Lloyd S. Shapley. Cambridge University Press, pp. 1–28.

Rudin, C., 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat. Mach. Intell. 1 (5), 206–215. http://dx.doi.org/10.1038/s42256-019-0048-x.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323 (6088), 533–536. http://dx.doi.org/10.1038/323533a0.

Ryman-Tubb, N.F., Krause, P., Garn, W., 2018. How Artificial Intelligence and machine learning research impacts payment card fraud detection: A survey and industry benchmark. Eng. Appl. Artif. Intell. 76, 130–157. http://dx.doi.org/10.1016/j.engappai.2018.07.008, URL: https://www.sciencedirect.com/science/article/pii/S0952197618301520.

Schwab, P., Karlen, W., 2019. Cxplain: Causal explanations for model interpretation under uncertainty. Adv. Neural Inf. Process. Syst. 32, URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/3ab6be46e1d6b21d59a3c3a0b9d0f6ef-Paper.pdf.

Setiono, R., Liu, H., 1995. Understanding neural networks via rule extraction. In: IJCAI, vol. 1, Citeseer, pp. 480–485, URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=68aaa9b0ec2d08822972a29926af648786049e71.

Shapley, L., 1997. 7. a value for n-person games. Contributions to the theory of games ii (1953) 307-317. In: Classics in Game Theory. Princeton University Press, Princeton, pp. 69–79. http://dx.doi.org/10.1515/9781400829156-012.

Shrikumar, A., Greenside, P., Shcherbina, A., Kundaje, A., 2016. Not just a black box: Learning important features through propagating activation differences. CoRR abs/1605.01713, arXiv:1605.01713, URL: http://arxiv.org/abs/1605.01713.

Slack, D., Hilgard, A., Singh, S., Lakkaraju, H., 2021. Reliable post hoc explanations: Modeling uncertainty in explainability. Adv. Neural Inf. Process. Syst. 34, 9391–9404, URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/4e246a381baf2ce038b3b0f82c7d6fb4-Paper.pdf.

Thomas, T., Mehta, P., Kalinin, K., 2018. Skorch: A scikit-learn compatible neural network library that wraps Pytorch. https://github.com/skorch-dev/skorch.

Van Der Walt, S., Colbert, S.C., Varoquaux, G., 2011. The NumPy array: a structure for efficient numerical computation. Comput. Sci. Eng. 13 (2), 22–30. http://dx.doi.org/10.1109/MCSE.2011.37.

Velmurugan, M., Ouyang, C., Moreira, C., Sindhgatta, R., 2021. Developing a fidelity evaluation approach for interpretable machine learning. http://dx.doi.org/10.48550/arXiv.2106.08492, arXiv preprint arXiv:2106.08492.

Vieira, C.P., Digiampietri, L.A., 2022. Machine learning post-hoc interpretability: a systematic mapping study. In: Proceedings of the XVIII Brazilian Symposium on Information Systems. pp. 1–8. http://dx.doi.org/10.1145/3535511.3535512.

Wachter, S., Mittelstadt, B., Russell, C., 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. Harv. JL Tech. 31, 841, Publisher: HeinOnline.

Wang, R., Wang, X., Inouye, D.I., 2021. Shapley explanation networks. CoRR abs/2104.02297 arXiv:2104.02297, URL: https://arxiv.org/abs/2104.02297.

Wojke, N., Bewley, A., 2018. Deep cosine metric learning for person re-identification. In: 2018 IEEE Winter Conference on Applications of Computer Vision. WACV, IEEE, pp. 748–756. http://dx.doi.org/10.1109/WACV.2018.00087.

Wolberg, W.H., Mangasarian, O.L., Street, N., Street, W.H., 1992. Breast cancer wisconsin (original) data set. URL: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original). (Accessed 25 June 2024).

Wolberg, W.H., Mangasarian, O.L., Street, N., Street, W., 1995. Breast cancer wisconsin (diagnostic). http://dx.doi.org/10.24432/C5DW2B, UCI Machine Learning Repository.

Wong, T.T., Yang, N.Y., 2017. Dependency analysis of accuracy estimates in k-fold cross validation. IEEE Trans. Knowl. Data Eng. 29 (11), 2417–2427. http://dx.doi.org/10.1109/TKDE.2017.2740926.

Xing, E.P., Ho, Q., Xie, P., Wei, D., 2016. Strategies and principles of distributed machine learning on big data. Engineering 2 (2), 179–195. http://dx.doi.org/10.1016/J.ENG.2016.02.008, Publisher: Elsevier.

Zeng, X., Martinez, T.R., 2000. Distribution-balanced stratified cross-validation for accuracy estimation. J. Exp. Theor. Artif. Intell. 12 (1), 1–12. http://dx.doi.org/10.1080/095281300146272.

Zhou, L., Pan, S., Wang, J., Vasilakos, A.V., 2017. Machine learning on big data: Opportunities and challenges. Neurocomputing 237, 350–361. http://dx.doi.org/10.1016/j.neucom.2017.01.026, Publisher: Elsevier.