mathematics

Article

Mathematical Modeling and Analysis of Credit Scoring Using the LIME Explainer: A Comprehensive Approach

Abdussalam Aljadani [1], Bshair Alharthi [2], Mohammed A. Farsi [3], Hossam Magdy Balaha [4,5], Mahmoud Badawy [6,5] and Mostafa A. Elhosseini [3,5,*]

1 Department of Management, College of Business Administration in Yanbu, Taibah University, Al-Madinah Al-Munawarah 41411, Saudi Arabia; ajadani@taibahu.edu.sa
2 Department of Marketing, College of Business, University of Jeddah, Jeddah 22425, Saudi Arabia; bsalharthi@uj.edu.sa
3 College of Computer Science and Engineering, Taibah University, Yanbu 46421, Saudi Arabia; mafarsi@taibahu.edu.sa
4 Bioengineering Department, J.B. Speed School of Engineering, University of Louisville, Louisville, KY 40208, USA; hmbala01@louisville.edu
5 Department of Computers and Control Systems Engineering, Faculty of Engineering, Mansoura University, Mansoura 35516, Egypt; engbadawy@mans.edu.eg
6 Department of Computer Science and Informatics, Applied College, Taibah University, Al-Madinah Al-Munawarah 41461, Saudi Arabia
* Correspondence: melhosseini@mans.edu.eg

Abstract: Credit scoring models serve as pivotal instruments for lenders and financial institutions, facilitating the assessment of creditworthiness. Traditional models, while instrumental, grapple with challenges related to efficiency and subjectivity. The advent of machine learning heralds a transformative era, offering data-driven solutions that transcend these limitations. This research delves into a comprehensive analysis of various machine learning algorithms, emphasizing their mathematical underpinnings and their applicability in credit score classification. A comprehensive evaluation is conducted on a range of algorithms, including logistic regression, decision trees, support vector machines, and neural networks, using publicly available credit datasets. Within the research, a unified mathematical framework is introduced, which encompasses preprocessing techniques and critical algorithms such as Particle Swarm Optimization (PSO), the Light Gradient Boosting Model, and Extreme Gradient Boosting (XGB), among others. The focal point of the investigation is the LIME (Local Interpretable Model-agnostic Explanations) explainer. This study offers a comprehensive mathematical model using the LIME explainer, shedding light on its pivotal role in elucidating the intricacies of complex machine learning models. This study's empirical findings offer compelling evidence of the efficacy of these methodologies in credit scoring, with notable accuracies of 88.84%, 78.30%, and 77.80% for the Australian, German, and South German datasets, respectively. In summation, this research not only amplifies the significance of machine learning in credit scoring but also accentuates the importance of mathematical modeling and the LIME explainer, providing a roadmap for practitioners to navigate the evolving landscape of credit assessment.

Keywords: credit scoring; empirical analysis; feature selection; LIME explainer; machine learning (ML); particle swarm optimization (PSO)

MSC: 68T20

check for updates

Citation: Aljadani, A.; Alharthi, B.; Farsi, M.A.; Balaha, H.M.; Badawy, M.; Elhosseini, M.A. Mathematical Modeling and Analysis of Credit Scoring Using the LIME Explainer: A Comprehensive Approach. *Mathematics* **2023**, *11*, 4055. https://doi.org/10.3390/math11194055

Academic Editor: Valsamis Ntouskos

Received: 28 August 2023
Revised: 14 September 2023
Accepted: 21 September 2023
Published: 25 September 2023

1. Introduction

In today's dynamic and interconnected global economy, credit plays a pivotal role in facilitating economic activities and fostering financial growth. Lenders and financial institutions rely heavily on credit scoring models to assess the creditworthiness of individuals, businesses, and other entities seeking access to financial products and services.

A credit score is a numerical representation of an individual's creditworthiness, which helps lenders gauge the risk associated with extending credit and making lending decisions. As such, credit scoring has become an indispensable tool in the modern financial landscape, shaping access to credit and influencing financial outcomes for millions of borrowers worldwide. The development and refinement of credit scoring models have evolved significantly over the years, driven by advancements in data analytics, statistical modeling techniques, and the availability of vast amounts of financial and nonfinancial data [1,2]. Traditionally, credit scoring was primarily based on a few key factors, such as payment history, outstanding debt, length of credit history, and new credit applications. However, contemporary credit scoring models have incorporated a more diverse set of variables and sophisticated algorithms to enhance predictive accuracy and provide a more comprehensive assessment of credit risk. The importance of credit scoring cannot be overstated, as it not only affects the availability of credit but also influences interest rates, loan terms, and overall financial inclusion. Access to affordable credit is crucial for individuals and businesses to pursue their aspirations, invest in productive ventures, and contribute to economic growth. Moreover, credit scoring also plays a vital role in mitigating risks for lenders, enabling them to make informed decisions, manage their loan portfolios effectively, and maintain the stability of the financial system [3,4].

Traditionally, credit scoring has heavily relied on manual processes, limited variables, and subjective criteria, leading to inefficiencies and potential biases in the evaluation process. However, the advent of machine learning techniques has revolutionized the credit scoring landscape, offering automated and data-driven approaches that can significantly enhance the accuracy and efficiency of credit assessments. Machine learning algorithms have demonstrated remarkable capabilities in handling complex and high-dimensional data, learning patterns and relationships, and making predictions based on historical information. By training on large-scale credit datasets, machine learning models can capture intricate credit patterns that may be overlooked by traditional methods. Furthermore, these algorithms can adapt and evolve as new data become available, ensuring their relevance in dynamic credit markets. The utilization of machine learning in credit scoring holds the promise of providing lenders with more objective, consistent, and reliable credit assessment models [5–7].

The objective of this research is to conduct experiments and analyses using various machine learning classifiers on different credit approval datasets. The research aims to evaluate and compare the performance of these classifiers in terms of multiple evaluation metrics, including accuracy, sensitivity, specificity, precision, F1 score, receiver operating characteristic (ROC), balanced accuracy, and weighted sum metric (WSM) performance. Through these systematic experiments, the research seeks to assess how well these classifiers can predict credit approval outcomes and identify which classifiers perform best under different dataset conditions. Additionally, the research involves analyzing the impact of various data preprocessing techniques, such as feature selection and scaling, on the classifier's performance. The key research questions to be addressed in this study include the following:

- How do mathematical formulations underpin the machine learning algorithms employed in credit score classification, and which algorithms demonstrate superior predictive performance?
- In the context of credit scoring models, how does the mathematical modeling of feature selection, especially using the PSO metaheuristic optimizer, influence the model's accuracy and efficiency?
- In what ways can the mathematical optimization of hyperparameters (e.g., learning rates, regularization strengths) in machine learning models influence their performance in credit score classification, and are there specific optimization algorithms that are more effective for this domain?

- Through the incorporation of the mathematical model of the LIME explainer, what insights can be gleaned concerning the strengths, limitations, and interpretability of various machine learning approaches in the context of credit scoring?
- Based on the empirical findings and mathematical rigor introduced in this study, how can practitioners be better equipped to choose the most suitable machine learning techniques and feature selection methods for credit score classification?

To achieve these research objectives, a comprehensive experimental analysis is conducted using publicly available credit datasets. Various machine learning algorithms, including but not limited to logistic regression, decision trees, random forests, support vector machines, and neural networks, are implemented and evaluated. The performance of these algorithms is measured using standard evaluation metrics such as accuracy, precision, recall, and F1 score. Furthermore, the impact of feature selection methods are investigated to determine the most relevant variables for credit scoring. The significance of this research extends across several dimensions, providing a comprehensive understanding of credit scoring models due to the following:

- This study introduces a robust mathematical framework underpinning machine learning algorithms and preprocessing techniques in the realm of credit scoring. This framework ensures the solidity of credit scoring models, providing a sound basis for analysis and decision making.
- Through the rigorous mathematical modeling of feature selection, particularly harnessing the Particle Swarm Optimization (PSO) metaheuristic optimizer, this research provides valuable insights into the identification of the most relevant variables for credit scoring. This optimization process not only improves the accuracy of credit scoring models but also enhances their computational efficiency, making them more practical for real-world applications.
- This study places a strong emphasis on the mathematical model underlying the Local Interpretable Model-Agnostic Explanations (LIME) explainer. By delving into the intricacies of LIME's mathematical foundations, it highlights the pivotal role of explainability techniques in enhancing the transparency and interpretability of complex machine learning models used in credit scoring. This transparency is crucial for building trust in the decision-making process.
- This research provides compelling empirical evidence by evaluating the effectiveness of various machine learning techniques for credit score classification. This empirical analysis is conducted across multiple datasets, adding depth and credibility to the findings. The benchmarking of different approaches offers practical insights into their performance under diverse conditions.
- By systematically comparing and analyzing different algorithms, feature selection methods, and the LIME explainer, this research offers valuable guidance to practitioners in the field of credit scoring. It assists them in selecting the most suitable techniques and strategies for specific credit scoring tasks, ultimately leading to better decision-making processes within financial institutions and lending companies.
- The findings of this research not only contribute to the current state of credit scoring but also pave the way for further advancements in the field. By identifying areas where mathematical rigor, feature selection, and interpretability can be enhanced, it opens doors to future research, innovation, and continuous improvement in credit scoring models and practices. This advancement is essential in keeping pace with evolving financial landscapes and data-driven technologies.

The rest of this research paper is structured as follows: Section 2 presents a review of the literature concerned with feature selection and machine learning algorithms for credit scoring. Section 3 presents the datasets utilized in this study. Section 4 provides a detailed discussion of the proposed approach. Section 5 describes the experiments conducted and discusses their outcomes. Lastly, Section 6 concludes this paper and outlines future research directions.

2. Feature Selection, Machine Learning, and Credit Scoring: A Review of Literature

Default risk is a primary concern in online lending, prompting the use of credit scoring models to assess borrower creditworthiness. Existing efforts have mainly focused on improving assessment methods, without adequately addressing the quality of credit data, often plagued by noisy, redundant, or irrelevant features that hinder model accuracy. Effective feature selection methods are crucial for enhancing credit evaluation accuracy. Current feature selection methods in online credit scoring suffer from issues like subjectivity, time consumption, and low accuracy, necessitating the introduction of innovative approaches. Zhang et al. [8] proposed a solution called the local binary social spider algorithm (LBSA), which incorporates two local optimization strategies (i.e., opposition-based learning (OBL) and improved local search algorithm (ILSA)) into BinSSA. These strategies address the aforementioned drawbacks. Comparative experiments conducted on three typical online credit datasets (i.e., Paipaidai (PPD), Renrendai (RRD) in China, and Lending Club (LC) in the United States) concluded that LBSA significantly reduces feature subset redundancy, enhances iterative stability, and improves credit scoring model accuracy and effectiveness.

Tripathi et al. [9] directed their efforts toward enhancing credit scoring models employed by financial institutions and credit industries. Their primary objective was to enhance model performance by introducing a hybrid methodology that combines feature selection with a multilayer ensemble classifier framework. This hybrid model was meticulously crafted in three distinct phases: initial preprocessing and classifier ranking, followed by ensemble feature selection, and ultimately the utilization of the selected features within a multilayer ensemble classifier framework. To further optimize ensemble performance, they introduced a classifier placement algorithm based on the Choquet integral value. Then, the researchers conducted experiments using real-world datasets, including Australian (AUS), Japanese (JPD), German-categorical (GCD), and German-numerical (GND). The findings indicated that the features chosen through their proposed approach exhibited enhanced representativeness, leading to improved classification accuracy across various classifiers such as quadratic discriminant analysis (QDA), Naïve Bayes (NB), multilayer feed-forward neural network (MLFN), time-delay neural network (TDNN), distributed time-delay neural network (DTNN), decision tree (DT), and support vector machine (SVM). Additionally, for all the credit scoring datasets considered, the proposed ensemble model consistently outperformed traditional ensemble models in terms of accuracy, sensitivity, and G-measure.

Furthermore, Zhang et al. [10] introduced a novel multistage ensemble model with enhanced outlier adaptation to enhance credit scoring predictions. To mitigate the impact of outliers in noisy credit datasets, an improved local outlier factor algorithm was employed, incorporating a bagging strategy to identify and integrate outliers into the training set, thereby enhancing base classifier adaptability. Additionally, for improved feature interpretability, a novel dimension-reduced feature transformation method was proposed to hierarchically evolve and extract salient features. To further enhance predictive power, a stacking-based ensemble learning approach with self-adaptive parameter optimization was introduced, automatically optimizing base classifier parameters and constructing a multistage ensemble model. The performance of this model was evaluated across ten datasets (e.g., Australian, Japanese, German, Taiwan, and Polish credit datasets) using six evaluation metrics, and the reported experimental results demonstrated the superior performance and effectiveness of the suggested approach.

A sequential ensemble credit scoring model based on XGBoost, a variation of the gradient boosting machine, was proposed by Xia et al. [11]. The proposed XGBoost-based credit scoring model consists of three phases (i.e., data preprocessing, data scaling, and missing value marking). The redundant features are then removed using a model-based feature selection approach, which enhances performance and lowers computing costs. The final model is trained using the acquired configuration after the hyperparameters have been tuned using the Tree-structured Parzen Estimator (TPE) method. The results show that TPE hyperparameter optimization outperforms grid search, random search, and manual search.

The proposed model also provides feature importance scores and decision charts, which enhance the interpretability of the credit scoring model. Moreover, Liu et al. [12] introduced two tree-based augmented GBDTs, AugBoost-RFS and AugBoost-RFU. These methods incorporate a stepwise feature augmentation mechanism to diversify base classifiers within GBDT, and they maintain interpretability through tree-based embedding techniques. Experimental results on four large-scale credit scoring datasets demonstrated that AugBoost-RFS and AugBoost-RFU outperform standard GBDT. Moreover, their supervised tree-based feature augmentation achieved competitive results compared with neural network-based methods, while significantly improving efficiency.

Chen et al. [13] proposed a multilevel Weighted Voting classification algorithm based on the combination of classifier ranking and the Adaboost algorithm. Four feature selection methods were used to select the features; then, seven commonly used heterogeneous classifiers were used to select five classifiers and calculate their ranks, and then AdaBoost was used to boost the performance of the selected base classifiers and calculate the updated F1 and ranks. The effects of ensemble framework Majority Voting (MV), Weighted Voting (WV), Layered Majority Voting (LMV), and Layered Weighted Voting (LWV) were all evaluated from the aspects of accuracy, sensitivity, specificity, and G-measure. The outcome of the experiments showed that the presented method achieved significant results in Australian credit score data and some progress on the German loan approval data. In Gicić et al. [14], stacked unidirectional and bidirectional LSTM networks were applied to solve credit scoring tasks. The proposed model exploited the full potential of the three-layer stacked LSTM and BiLSTM architecture with the treatment and modeling of public datasets. Attributes of each loan instance were transformed into a sequence of the matrix with a fixed sliding window approach with a one-time step. The proposed models outperformed existing and more complex deep learning models and, thus, succeeded in preserving their simplicity.

Kazemi et al. [15] proposed an approach based on a Genetic Algorithm (GA) and neural networks (NNs) to automatically find customized cut-off values. Since credit scoring is a binary classification problem, two popular credit scoring datasets (i.e., the “Australian” and “German” credit datasets) were used to test the proposed approach. The numerical results reveal that the proposed GA-NN model could successfully find customized acceptance thresholds, considering predetermined performance criteria, including Accuracy, Estimated Misclassification Cost (EMC), and AUC for the tested datasets. Furthermore, the best-obtained results and the paired samples t-test results showed that utilizing the customized cut-off points leads to a more accurate classification than the commonly used threshold value of 0.5. Khatir and Bee [16] aimed to pinpoint the most significant predictors of credit default to construct machine learning classifiers capable of efficiently distinguishing defaulters from nondefaulters. They proposed five machine learning classifiers, and each of them was combined with different feature selection techniques and various data-balancing approaches. Given the imbalance in the used dataset (i.e., German Credit Data), three sample-modifying algorithms were used, and their impact on the performance of the classification models was evaluated. The key findings highlighted that the most effective classifier is a random forest combined with random forest recursive feature elimination and random oversampling. Moreover, it underscored the value of data-balancing algorithms, particularly in enhancing sensitivity.

Khan and Ghosh [17] introduced an improved version of the random wheel classifier. Their proposed approach was evaluated using two datasets (i.e., Australian and South German credit approval datasets). The results showed that their approach not only delivers more accurate and precise recommendations but also offers interpretable confidence levels. Additionally, it provided explanations for each credit application recommendation. This inclusion of recommendation confidence and explanations can instill greater trust in machine-provided intelligence, potentially enhancing the efficiency of the credit approval process. Haldankar [18] discussed the use of data mining techniques to identify fraud in various domains, particularly focusing on risk detection. The study proposed a cost-sensitive classifier for detecting risk using the Statlog (German Credit Data) dataset.

The study demonstrated the effectiveness of proper feature selection combined with an ensemble approach and thresholding in reducing the overall cost. The study reported an ACC of 76% and a SPC of 55%.

Wang et al. [19] focused on ensemble classification. They conducted an analysis and comparison of SVM ensembles using four different ensemble constructing techniques. They reported the highest ACC of 85.35% using the Statlog (Australian credit approval) dataset and 76.41% using the Statlog (German Credit Data) dataset. Additionally, Novakovic et al. [20] presented the performance of the C4.5 decision tree algorithm with wrapper-based feature selection. They conducted tests using eighteen datasets to compare the classification ACC results with the C4.5 decision tree algorithm. The authors demonstrated that wrapper-based feature selection, when applied to the C4.5 decision tree classifier, effectively contributed to the detection and elimination of irrelevant, redundant data, and noise in the data. They reported an ACC of 71.72% using the J48 reduced approach on the Statlog (German Credit Data) dataset.

3. Materials

The current study utilized three public datasets, namely (1) “Statlog (Australian Credit Approval)” [21], (2) “Statlog (German Credit Data)” [22] (presented in Table 1), and (3) “South German Credit ” [23] (presented in Table 2). For the “Statlog (Australian Credit Approval)” dataset, all attribute names and values were changed to meaningless symbols by the original authors to protect the data confidentiality. It contains 690 instances and 14 attributes. The “Statlog (German Credit Data)” dataset contains 1000 instances and 20 attributes. The “South German Credit” dataset contains 1000 instances and 21 attributes.

Table 1. The “Statlog (German Credit Data)” dataset columns and their meaning [22].

Attribute	Description
Attribute 1	Status of existing checking account
Attribute 2	Duration in month
Attribute 3	Credit history
Attribute 4	Purpose
Attribute 5	Credit amount
Attribute 6	Savings account/bonds
Attribute 7	Present employment since
Attribute 8	Installment rate in percentage of disposable income
Attribute 9	Personal status and sex
Attribute 10	Other debtors/guarantors
Attribute 11	Present residence since
Attribute 12	Property
Attribute 13	Age in years
Attribute 14	Other installment plans
Attribute 15	Housing
Attribute 16	Number of existing credits at this bank
Attribute 17	Job
Attribute 18	Number of people accountable for providing maintenance
Attribute 19	Telephone
Attribute 20	Foreign worker

Table 2. Meaning of the columns in the "South German Credit" dataset [23].

Attribute	Description
laufkont	Status of the debtor's checking account with the bank
laufzeit	Duration of the credit in months
moral	History of compliance with previous or concurrent credit contracts
verw	Purpose for which the credit is required
hoehe	Amount of credit in DM
sparkont	Debtor's savings
beszeit	Length of time the debtor has been employed with the current employer
rate	Credit installments as a percentage of the debtor's disposable income
famges	Combined information on the debtor's sex and marital status
buerge	Presence of another debtor or a guarantor for the credit
wohnzeit	Length of time (in years) the debtor has lived in the current residence
verm	The debtor's most valuable property
alter	Age in years
weitkred	Installment plans from providers other than the issuing bank
wohn	Type of housing the debtor resides in
bishkred	Number of credits, including the current one, the debtor has (or had) with this bank
beruf	Quality of the debtor's job
pers	Number of individuals financially dependent on the debtor
telef	Presence of a landline telephone registered under the debtor's name
gastarb	Whether the debtor is a foreign worker
kredit	Compliance status of the credit contract (good or bad)

4. Methodology

The current study proposes the framework depicted in Figure 1. The figure comprises three components: (a) The abstract view of the suggested training and optimization framework. It involves loading the dataset, applying a normalization technique, extracting the most promising features, selecting a model, and tuning the selected model. (b) The flow of the feature selection process utilizing the PSO metaheuristic optimizer. It encompasses initializing the PSO hyperparameters and solutions, calculating fitness scores for different solutions, and updating the solutions. (c) A model explanation using the LIME explainer model takes the instance that requires explanation and the tuned model as input, and it generates an explanation for it.

4.1. Features Scaling Techniques

Scalers, also known as data normalization or feature scaling techniques, are preprocessing methods used to transform the values of features in a dataset to a common scale. Scaling is crucial in machine learning tasks, as it helps to ensure that features with different ranges or units contribute equally to the learning process [24]. In this section, a background on several commonly used scalers is provided, including L1, L2, and max scalers. The L1 scaler, also known as the least absolute deviations scaler, normalizes the features in a dataset by dividing each feature by the sum of their absolute values. This scaler ensures that the sum of absolute feature values is equal to 1. It is particularly useful when the presence or absence of features is important, and their magnitudes are not relevant. The L2 scaler, also known as the Euclidean norm scaler, normalizes the features by dividing each feature by the square root of the sum of their squares. This scaler ensures that the sum of squared feature values is equal to 1. It is commonly used when both the presence or absence of features and their magnitudes are relevant [25,26]. Equations (1) and (2) show how to calculate the L1 and L2 scalers, respectively, where X represents the original feature values, X_{scaled} represents the scaled feature values, and $|(X)|$ represents the absolute values of the elements in X.

$$X_{\text{scaled}} = \frac{X}{\sum |(X)|} \tag{1}$$

$$X_{\text{scaled}} = \frac{X}{\sqrt{\sum (X^2)}} \tag{2}$$

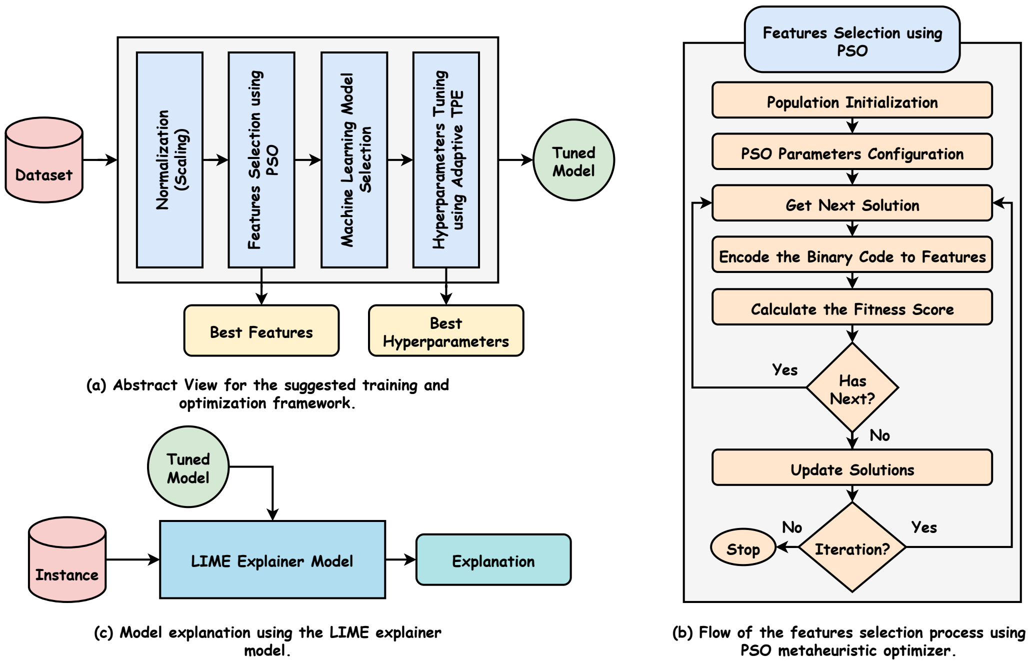
Dataset
Normalization (Scaling)
Features Selection using PSO
Machine Learning Model Selection
Hyperparameters Tuning using Adaptive TPE
Tuned Model
Best Features
Best Hyperparameters
(a) Abstract View for the suggested training and optimization framework.
Tuned Model
Instance
LIME Explainer Model
Explanation
(c) Model explanation using the LIME explainer model.
Features Selection using PSO
Population Initialization
PSO Parameters Configuration
Get Next Solution
Encode the Binary Code to Features
Calculate the Fitness Score
Yes
Has Next?
No
Update Solutions
No
Yes
Stop
Iteration?
(b) Flow of the features selection process using PSO metaheuristic optimizer.

Figure 1. Graphical presentation of the suggested framework in the current study.

The Max scaler, also known as the maximum scaler, scales the features by dividing each feature by the maximum value across the entire feature set. This scaler maps the features into the range $[0,1]$. It is particularly useful when the distribution of the features is highly skewed or contains outliers. Standardization (STD), also known as the Z-score scaler, transforms the features by subtracting the mean of the feature set and dividing by the standard deviation. This scaler ensures that the transformed features have zero mean and unit variance. It is commonly used when the features follow a Gaussian distribution or when algorithms assume standardized input. The MinMax scaler scales the features by subtracting the minimum value and dividing by the difference between the maximum and minimum values. This scaler maps the features into the range $[0,1]$. It preserves the relative relationships and proportions of the feature values and is useful when the distribution of the features is not necessarily Gaussian. The Max-Absolute scaler scales the features by dividing each feature by the maximum absolute value across the entire feature set. This scaler maps the features into the range $[-1,1]$. It is particularly useful when preserving the sign of the data is important, such as in sparse datasets [27,28]. Equations (3)–(6) show how to calculate the Max, STD, MinMax, and Max-Absolute scalers, respectively, where mu represents the mean of the feature set and σ represents the standard deviation of the feature set.

$$X_{\text{scaled}} = \frac{X}{\max (X)} \tag{3}$$

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma} \tag{4}$$

$$X_{\text{scaled}} = \frac{X - \min(X)}{\max(X) - \min(X)} \tag{5}$$

$$X_{\text{scaled}} = \frac{X}{\max|(X)|} \tag{6}$$

4.2. Features Selection Using Particle Swarm Optimization (PSO)

Feature selection plays a crucial role in machine learning and data mining tasks by identifying the most informative and relevant subset of features from a given dataset. It aims to improve model performance, reduce computational complexity, and enhance interpretability by selecting a subset of features that are highly predictive of the target variable. Particle Swarm Optimization (PSO) is a population-based optimization algorithm inspired by the social behavior of bird flocking or fish schooling. It has been widely applied to feature selection due to its ability to efficiently explore high-dimensional search spaces and find near-optimal solutions [29]. The main goal of using PSO for feature selection is to find an optimal subset of features that maximizes the performance of a given machine learning model. The process involves defining a fitness function that quantifies the quality of a feature subset based on its predictive power or some other criterion. The fitness function can be based on classification accuracy, regression error, or any other evaluation metric appropriate for the task at hand. PSO-based feature selection offers several advantages. It can effectively handle high-dimensional feature spaces and explore a large number of possible feature combinations. PSO's ability to balance exploration and exploitation helps in finding near-optimal solutions efficiently. Furthermore, PSO is a versatile technique that can be combined with various machine learning algorithms, making it applicable to different problem domains [30–32].

In PSO-based feature selection, each particle in the swarm represents a potential feature subset. The swarm collectively explores the search space of possible feature combinations by adjusting their positions and velocities. The position of a particle corresponds to a binary string, where each bit represents the presence or absence of a particular feature. The velocity represents the direction and magnitude of change in the binary string. During the optimization process, particles update their velocities and positions based on their own experience (i.e., personal best) and the best solution found by any particle in the swarm (i.e., global best). The personal best represents the best feature subset the particle has encountered so far, while the global best represents the best feature subset found by any particle in the swarm. These best positions guide the movement of particles towards promising regions in the search space. The update equations for PSO-based feature selection involve modifying the velocities and positions of particles based on the current velocities, personal bests, and global best. The specific equations may vary depending on the variant of PSO used and the problem formulation. The iterative optimization process continues until a termination criterion is met, such as reaching a maximum number of iterations or convergence of the particle positions. The resulting global best position represents the selected feature subset that optimizes the performance of the chosen machine learning model [33–35].

Let P be the population of particles in the swarm, where each particle p_i represents a potential feature subset (i.e., ith particle). Each particle p_i has a position vector x_i and a velocity vector v_i, where x_{ij} and v_{ij} represent the jth element of x_i and v_{ij}, respectively. The feature subset is represented as a binary string x_i of length n, where x_{ij} denotes the presence ($x_{ij} = 1$) or absence ($x_{ij} = 0$) of feature j in particle i.

The position of particle p_i is denoted as $x_i = [x_{i1}, x_{i2}, \cdots, x_{in}]$, and the velocity is represented as $v_i = [v_{i1}, v_{i2}, \cdots, v_{in}]$. During the optimization process, particles update their velocities and positions based on their personal best (p_{best_j}) and the global best (p_{best}) solution found by any particle in the swarm.

The velocity update equation for particle p_i is given by Equation (7), where $v_{ij}^{(t+1)}$ is the updated velocity of feature j in particle i at iteration $(t+1)$, w is the inertia weight, c_1 and c_2 are acceleration constants, $r_1^{(t)}$ and $r_2^{(t)}$ are random values at iteration t, $p_{best_{ij}}$ is the personal best value of feature j for particle i, and p_{best_j} is the global best value of feature j among all particles. The position update equation for particle p_i is given by Equation (8)

$$v_{ij}^{(t+1)} = w \times v_{ij}^{(t)} + c_1 \times r_1^{(t)} \times \left(p_{best_{ij}} - x_{ij}^{(t)}\right) + c_2 \times r_2^{(t)} \times \left(p_{best_j} - x_{ij}^{(t)}\right) \tag{7}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)} \tag{8}$$

The personal best value $p_{best_{ij}}$ is updated if the fitness of the current position is better than the previous personal best, as presented in Equation (9). The global best value g_{best_j} is updated by selecting the best position from all particles, as presented in Equation (10), where $i*$ is the index of the particle with the best fitness among all particles.

$$p_{best_{ij}} = \begin{cases} x_{ij}^{(t+1)}, & \text{if fitness} x_i^{(t+1)} > \text{fitness}(p_{best_i}) \\ p_{best_{ij}}, & \text{Otherwise} \end{cases} \tag{9}$$

$$g_{best_j} = x_{i*}(t+1) \tag{10}$$

The process continues iteratively until a termination criterion is met, such as a maximum number of iterations or convergence. The final feature subset is represented by the binary string of the global best position (i.e., g_{best}), which optimizes the performance of the chosen model.

4.3. Machine Learning Classification and Tuning

Machine learning classifiers are algorithms that are designed to learn patterns and make predictions based on labeled training data. They are widely used in various domains, including image recognition, natural language processing, fraud detection, and credit scoring. This study utilized the following machine learning classifiers: LGBM, XGB, KNN, DT, LR, RF, AdaBoost, HGB, and MLP. The LGBM (Light Gradient Boosting Model), as presented in Equation (11), is a gradient boosting framework that uses tree-based learning algorithms. It is known for its high efficiency and scalability, making it suitable for large-scale datasets. LGBM utilizes a gradient-based optimization strategy to construct an ensemble of weak models that sequentially minimize the loss function. It incorporates features such as histogram-based binning and leafwise tree growth to achieve faster training and better accuracy. N is the number of weak models, α_i are the coefficients, and $h_i(x)$ are the weak models. XGB (Extreme Gradient Boosting), as presented in Equation (12), is another popular gradient boosting algorithm that excels in predictive accuracy. It uses a similar approach to LGBM but incorporates additional regularization techniques to prevent overfitting. XGB employs a combination of gradient boosting and decision tree algorithms, optimizing a differentiable loss function through successive iterations. It offers flexibility in terms of customizing the optimization objectives and evaluation metrics [36,37]. $f_i(x)$ are the base models, $T(x;\Theta_t)$ are the decision trees, and γ_t are the step sizes.

$$\text{LGBM:} \quad \text{Ensemble}(X) = \sum_{i=1}^{N} \alpha_i \times h_i(x) \tag{11}$$

$$\text{XGB:} \quad \text{Ensemble}(X) = \sum_{i=1}^{N} f_i(x) + \gamma_t \times T(x;\Theta_t) \tag{12}$$

KNN (K-Nearest Neighbors), as presented in Equation (13), is a nonparametric algorithm that classifies new instances based on their similarity to the labeled training instances.

It operates on the principle that objects with similar attributes tend to belong to the same class. KNN determines the class of an unseen instance by considering the labels of its k-nearest neighbors in the feature space. The choice of k influences the trade-off between model complexity and accuracy. $y_{\text{neighbors}}$ are the class labels of the k-nearest neighbors of x. DT (Decision Trees), as presented in Equation (14), are hierarchical models that recursively partition the feature space based on attribute values. Each internal node represents a decision based on a specific feature, while each leaf node represents a class label or a prediction. Decision trees are interpretable, capable of handling both categorical and numerical features, and they are resistant to outliers. However, they are prone to overfitting, especially when the trees become too complex. N_i are internal nodes, S_i are splits, and θ_i are threshold values.

$$\text{KNN:} \quad \text{Class}(x) = \text{mode}(y_{\text{neighbors}}) \tag{13}$$

$$\text{DT:} \quad T(X) = \{N_i, S_i, \theta_i\}_{i=1}^{M} \tag{14}$$

LR (Logistic Regression), as presented in Equation (15), is a linear classifier that models the relationship between the input features and the probability of belonging to a certain class. It is commonly used for binary classification tasks but can be extended to handle multiclass problems as well. LR applies a Sigmoid function to the linear combination of the input features, mapping the result to a probability between 0 and 1. It learns the optimal weights through maximum likelihood estimation. β_0 to β_n are the factors of the LR equations. RF (Random Forest), as presented in Equation (16), is an ensemble learning method that combines multiple decision trees to make predictions. It constructs each tree by using a random subset of the training data and a random subset of the input features. RF leverages the principle of the "wisdom of crowds" to reduce overfitting and improve generalization performance. It provides feature importance measures and can handle high-dimensional data effectively [38,39]. T is the number of trees, and $T_i(X)$ are individual decision trees.

$$\text{LR:} \quad P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \times x_1 + \cdots + \beta_n \times x_n)}} \tag{15}$$

$$\text{RF:} \quad \text{Ensemble}(X) = \frac{1}{T} \times \sum_{i=1}^{T} T_i(X) \tag{16}$$

AdaBoost (Adaptive Boosting), as presented in Equation (17), is an ensemble learning technique that combines weak classifiers to create a strong classifier. It assigns higher weights to misclassified instances, allowing subsequent classifiers to focus on difficult examples. The final prediction is determined by a weighted vote of all weak classifiers. AdaBoost is particularly effective in handling complex datasets and can achieve high accuracy even with weak base classifiers. α_t are weights and $h_t(x)$ are weak classifiers. HGB (Histogram Gradient Boosting), as presented in Equation (18), is a histogram-based gradient boosting algorithm that combines the advantages of gradient boosting and histogram binning. It discretizes the continuous input features into histograms, allowing for faster training and efficient memory usage. HGB incorporates various optimization techniques, including early stopping and feature subsampling, to enhance performance. $f_i(x)$ are base models, $H(x;\Theta_t)$ are histograms, and γ_t are step sizes.

$$\text{AdaBoost:} \quad F(x) = \sum_{t=1}^{T} \alpha_t \times h_t(x) \tag{17}$$

$$\text{HGB:} \quad \text{Ensemble}(X) = \sum_{i=1}^{N} f_i(x) + \gamma_t \times H(x;\Theta_t) \tag{18}$$

MLP (Multilayer Perceptron), as presented in Equation (19), is a type of artificial neural network that consists of multiple layers of interconnected nodes (neurons). It learns by adjusting the weights and biases associated with each connection, enabling it to approximate complex nonlinear functions. MLP is a versatile classifier capable of handling a wide range of problem domains. It requires careful architecture design and appropriate activation functions to achieve good performance [37,40].

$$\text{MLP:} \quad y = f_{out}(W_{out} \cdot f_{hidden}(W_{hidden} \cdot x + b_{hidden}) + b_{out}) \tag{19}$$

Hyperparameter tuning is a critical aspect of machine learning model development, as it involves selecting the optimal configuration of hyperparameters that govern the behavior of the model. Hyperparameters significantly impact the model's performance, and finding the best combination can be a challenging and time-consuming process. One popular technique for hyperparameter optimization is the Tree-structured Parzen Estimator (TPE). The TPE algorithm is a sequential model-based optimization approach that uses Bayesian optimization to efficiently search the hyperparameter space. It models the relationship between hyperparameters and the performance metric of interest, typically using a Gaussian Process. TPE divides the search space into two parts: the exploration space, where hyperparameters are randomly sampled, and the exploitation space, where the most promising hyperparameters are selected based on their expected improvement. Adaptive TPE takes the TPE algorithm further by incorporating adaptive mechanisms to dynamically adjust the search process based on the observed performance. It continuously learns from the optimization process and adapts the exploration–exploitation trade-off accordingly. This adaptivity allows Adaptive TPE to focus the search on promising regions of the hyperparameter space and efficiently explore different configurations [41–43].

The Adaptive TPE algorithm follows these key steps: **Initialization**: The search process begins with an initial set of hyperparameter configurations randomly sampled from the search space. **Evaluation**: Each configuration is evaluated using cross-validation or another appropriate evaluation method to obtain the performance metric. **Modeling**: A probabilistic model, such as a Gaussian Process, is constructed to capture the relationship between hyperparameters and the performance metric. **Selection**: Based on the probabilistic model, the next set of hyperparameter configurations is selected using the expected improvement or another acquisition function. This balances the exploration of unexplored regions and the exploitation of promising configurations. **Update**: The selected configurations are evaluated, and the performance results are used to update the probabilistic model. **Iteration**: Steps 4 and 5 are repeated iteratively until a stopping criterion is met, such as a maximum number of iterations or convergence of the performance metric [41,44].

The advantages of Adaptive TPE for hyperparameter tuning include its ability to efficiently explore the search space, adapt to the observed performance, and converge to promising configurations. It balances exploration and exploitation to find the best hyperparameters within a reasonable computational budget. Adaptive TPE is applicable to various machine learning algorithms and can significantly improve model performance compared with using default or suboptimal hyperparameter settings [41,42,44].

As mentioned, the TPE aims to maximize the conditional probability of hyperparameters given the performance metric, y as presented in Equation (20), where x represents hyperparameter configurations, y , $p(x|y)$ represents the conditional probability of hyperparameters given the performance metric, $p(y|x)$ represents the conditional probability of the performance metric given hyperparameters, $p(x)$ represents the prior probability of hyperparameters, and $p(y)$ represents the marginal probability of the performance metric.

$$\text{TPE:} \quad \arg\max_x (p(x|y)) = \arg\max_x \left(\frac{p(y|x) \times p(x)}{p(y)} \right) \tag{20}$$

The Adaptive TPE algorithm follows the key steps presented in Equation (21). In it, the Gaussian Process (GP), as presented in Equation (22), models the underlying function $f(x)$ mapping hyperparameters x to the performance metric y, considering Gaussian noise ε. The expected improvement (EI) acquisition function, as presented in Equation (23), measures the potential improvement over the current best performance metric $f(x_{min})$ for a given hyperparameter configuration x.

$$\text{Adaptive TPE:} \quad \begin{aligned}&\text{Initialization} \rightarrow \text{Evaluation} \rightarrow \text{Modeling} \\ &\rightarrow \text{Selection} \rightarrow \text{Update} \rightarrow \text{Iteration}\end{aligned} \tag{21}$$

$$\text{Gaussian Process:} \quad y = f(x) + \varepsilon \tag{22}$$

$$\text{Expected Improvement:} \quad \mathrm{EI}(x) = \mathbb{E}[\max(f(x) - f(x_{\min}), 0)] \tag{23}$$

4.4. Cross-Validation and Evaluation Metrics

k-Fold Cross-Validation is a widely used technique in machine learning and model evaluation. It provides a robust and unbiased estimate of a model's performance by partitioning the available data into k subsets or folds. The process involves training and testing the model k times, each time using a different fold as the validation set and the remaining folds as the training set. The major benefits of it are as follows: (1) k-Fold Cross-Validation provides a more reliable estimate of a model's performance compared with a single train–test split. By using multiple validation sets and averaging the results, it reduces the potential bias and variability that can arise from a particular data split. (2) k-Fold Cross-Validation makes efficient use of available data by utilizing all instances in both the training and validation phases. This maximizes the amount of information used for model training and evaluation, resulting in more robust and accurate performance estimates. (3) k-Fold Cross-Validation is commonly used in model selection and hyperparameter tuning. It allows for comparing different models or different hyperparameter settings by evaluating their performance across multiple iterations and providing a fair comparison. (4) k-Fold Cross-Validation is beneficial when dealing with imbalanced datasets, where the distribution of classes is uneven. It ensures that each fold contains a representative distribution of instances, reducing the potential for biased evaluation due to class imbalance. (5) k-Fold Cross-Validation helps in identifying overfitting, which occurs when a model performs well on the training set but fails to generalize to new, unseen data. By evaluating the model's performance on multiple validation sets, it provides insights into the model's generalization ability and potential overfitting issues [45,46]. k-Fold Cross-Validation can be expressed as presented in Equation (24), where k is the number of folds, M_i is the model trained on the ith fold, and Performance (M_i) is the performance of the model on the validation set.

$$k\text{-Fold Cross-Validation:} \quad \text{Performance Estimate} = \frac{1}{k} \times \sum_{i=1}^{k} \text{Performance}(M_i) \tag{24}$$

Performance metrics are essential tools used to evaluate the effectiveness and quality of machine learning models. They provide quantitative measures to assess how well a model performs on classification or prediction tasks. In this section, a background on several commonly used performance metrics is provided, including accuracy, recall, precision, F1 score, specificity, balanced accuracy, and receiver operating characteristic (ROC) curve. Accuracy, as presented in Equation (25), is a widely used performance metric that measures the proportion of correctly predicted instances out of the total number of instances. It provides a general overview of how well a model performs across all classes. However, accuracy may not be suitable for imbalanced datasets, where the majority class dominates the performance evaluation. Recall (Sensitivity or True Positive Rate), as presented in Equation (26), measures the proportion of correctly predicted positive instances (true positives) out of all actual positive instances. It focuses on identifying as many positive instances as possible and is particularly useful when the goal is to minimize false negatives.

In medical diagnostics or fraud detection, recall is crucial to ensure the identification of all relevant cases, even at the cost of higher false positives [47,48].

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Instances}} = \frac{\text{TP + TN}}{\text{TP + TN + FP + FN}} \tag{25}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} = \frac{\text{TP}}{\text{TP + FN}} \tag{26}$$

Precision, as presented in Equation (27), measures the proportion of correctly predicted positive instances (true positives) out of all predicted positive instances (true positives and false positives). It focuses on the accuracy of positive predictions and is particularly useful when minimizing false positives is crucial. Precision is important in scenarios where false positives have significant consequences, such as in spam email filtering or legal systems. F1 Score, as presented in Equation (28), combines precision and recall into a single metric, providing a balanced measure of a model's performance. It is the harmonic mean of precision and recall, offering a single value that represents both metrics. The F1 score is suitable when there is an imbalance between precision and recall, and a balance between the two is desired. Specificity (True Negative Rate), as presented in Equation (29), measures the proportion of correctly predicted negative instances (true negatives) out of all actual negative instances. It is the complement of the false positive rate (FPR) and provides a measure of how well a model identifies negative instances. Specificity is particularly relevant when minimizing false positives is critical, such as in medical testing or manufacturing quality control [46,48].

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} = \frac{\text{TP}}{\text{TP + FP}} \tag{27}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{28}$$

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}} = \frac{\text{TN}}{\text{TN + FP}} \tag{29}$$

Balanced Accuracy, as presented in Equation (30), takes into account the proportion of correctly predicted instances for each class, providing an overall measure of model performance that accounts for class imbalance. It calculates the average of sensitivity (recall) across all classes. Balanced accuracy is valuable when there are significant differences in the number of instances among different classes. ROC Curve (receiver operating characteristic) is a graphical representation of the trade-off between the true positive rate (i.e., sensitivity) and the false positive rate (i.e., 1—specificity) for different classification thresholds. It helps evaluate the performance of a model across various thresholds and provides a visual tool to compare different models [46,47].

$$\text{Balanced Accuracy} = \frac{\text{Specificity} \times \text{Recall}}{2} \tag{30}$$

4.5. Model Explainability Using LIME

Model explainability is a crucial aspect of machine learning, particularly in domains where decisions have significant implications, such as healthcare, finance, and legal systems. While complex machine learning models, such as deep neural networks, often deliver high predictive accuracy, they lack interpretability, making it challenging to understand how they arrive at their predictions. Local Interpretable Model-agnostic Explanations (LIME) is a popular technique that addresses this issue by providing post hoc explanations for black-box models. LIME (Local Interpretable Model-Agnostic Explanations) aims to explain the predictions of any machine learning model by approximating its behavior locally. The key idea behind LIME is to create interpretable models, such as linear models or decision trees, that are locally faithful to the predictions of the black-box model. By explaining the model's predictions in a human-understandable manner, LIME helps users comprehend and trust the decisions made by the machine learning model. The advantages of LIME include its model-agnostic nature, as it can be applied to any black-box model without requiring knowledge of its internal workings. LIME also provides interpretable explanations at the local level, which can enhance trust and understanding of the model's decisions. Additionally, LIME can handle various types of data, including text, images, and structured data [49–51].

The LIME process involves the following steps: **Selection of Instances**: Initially, a set of instances or data points for which explanations are required is selected. These instances represent the inputs for which the model's predictions need to be explained. **Perturbation**: For each selected instance, LIME generates perturbed versions by randomly sampling data points near the original instance while preserving the important features. The perturbed instances are created to assess the model's behavior in the local neighborhood of the original instance. **Model Prediction**: The black-box model's predictions are obtained for the perturbed instances, capturing the output of the model within the local neighborhood of the original instance. **Feature Selection**: LIME identifies the important features for the selected instance by employing a technique such as sparse linear regression or decision tree induction. These features play a significant role in the model's predictions within the local context. **Weights and Explanations**: LIME assigns weights to the perturbed instances based on their proximity to the original instance and uses these weights to learn an interpretable model. This interpretable model approximates the behavior of the black-box model in the local neighborhood, providing explanations for the original instance's prediction. **Explanation Generation**: Finally, LIME generates explanations by highlighting the important features and their contributions to the prediction. This can be visualized as feature importance scores or rules that indicate the influence of each feature on the model's output [50,52]. The LIME process can be summarized mathematically as follows:

- **Step 1: Model Explainability:** Understanding decisions made by complex machine learning models, e.g., $f_{ML}(x)$, is critical in domains like healthcare and finance.
- **Step 2: LIME Approximation:** $f_{\text{LIME}}(x') \approx f_{ML}(x)$, where f_{LIME} is an interpretable model and x' is a local perturbation. The LIME objective is to find an approximate of the black-box model $f_{ML}(x)$ with an interpretable model $f_{\text{LIME}}(x')$ by minimizing the loss between them (Equation (31)).

$$\min_{f_{\text{LIME}}} \sum_{i=1}^{n} \mathcal{L}(f_{\text{LIME}}(x'), f_{ML}(x)) \tag{31}$$

- **Step 3: LIME Process:**
  - **Selection of Instances**: $\mathcal{D} = x_1, x_2, \ldots, x_n$, where x_i are instances to be explained.
  - **Perturbation**: $x'_i = x_i + \epsilon_i$, where ϵ_i is a small perturbation.
  - **Model Prediction**: $f_{ML}(x'_i)$, predicting using the black-box model.
  - **Feature Selection**: Identify significant features such as $x'_i = [x_{i1}, x_{i2}, \ldots, x_{im}]$ from the perturbed instances using a regression model, such as Lasso regression (Equation (32)).

$$\min_{w_j} \left(\sum_{i=1}^{n} (f_{ML}(x_i) - f_{\text{LIME}}(x'_i))^2 + \lambda \times \sum_{j=1}^{m} |w_j| \right) \tag{32}$$

  - **Weights and Explanations**: Learn weights w_j for features and create the interpretable model (Equation (33)).

$$f_{\text{LIME}}(x') = \sum_{j=1}^{m} w_j x'_{ij} \tag{33}$$

- **Explanation Generation**: Generate explanations such as feature importance scores or rules.
- **Feature Importance Scores (Explanation Generation)**: Calculate feature importance scores based on the absolute values of feature weights (Equation (34)).

$$\text{Feature Importance} = |w_j| \tag{34}$$

- **Rule Extraction (Explanation Generation)**: Extract rules from the interpretable model. For example, "If x_1 is significantly different from x_1', it strongly influences the prediction".

As noted, LIME often uses a kernel function to assign weights to perturbed samples based on their proximity to the original instance. The choice of kernel function (e.g., Gaussian kernel) and the bandwidth parameter can affect the weighting mathematically. The kernel weighting formula typically looks like Equation (35), where the weight w_i assigned to a perturbed sample x_i' based on its distance to the original instance x and the bandwidth parameter σ in the context of kernel weighting in LIME. $d(x, x_i')$ represents the distance between the original instance x and the perturbed sample x_i' [53].

$$w_i = \frac{e^{-\frac{d(x,x_i')^2}{\sigma^2}}}{\sum_{j=1}^{n} e^{-\frac{d(x,x_j')^2}{\sigma^2}}} \tag{35}$$

The computational complexity of LIME can be analyzed by considering the number of operations required for its various steps. Firstly, in the step of generating perturbed samples, LIME creates these samples for each of the n instances. For each instance, small random values are added to the features, resulting in k perturbed samples per instance. This operation's complexity is $O(n \times m \times k)$, where n is the number of instances, m is the dimensionality of the feature space, and k is the number of perturbed samples per instance.

Next, when evaluating the black-box model for each perturbed sample, the computational cost depends on the complexity of the black-box model itself. If the evaluation of the black-box model has a complexity of $O(f)$, then the total complexity for this step is $O(n \times m \times k \times f)$.

Moving on to feature selection and the creation of an interpretable model, techniques like Lasso regression are employed, involving optimization problems. The complexity of solving these optimization problems depends on factors like the number of iterations (t) and the number of features selected (s). Training the interpretable model, typically a linear regression model, has a complexity of $O(n \times s \times m)$, where n is the number of instances.

Finally, generating explanations, such as feature importance scores, is typically a linear time operation with respect to the number of features. The complexity for generating explanations is $O(s \times m)$. The overall computational complexity of LIME can be approximated as formulated in Equation (36).

$$O(n \times m \times k \times f + t \times s \times m + n \times s \times m + s \times m) \tag{36}$$

The most significant factors affecting complexity include the number of instances (n), the dimensionality of the feature space (m), the number of perturbed samples per instance (k), the complexity of the black-box model evaluation (f), the number of iterations in feature selection (t), and the number of selected features (s) [54].

5. Experiments and Discussions

The present study utilized the Python 3.11.4 programming language to conduct experiments, employing various packages, including scikit-learn 0.24.2 and LIME 0.2.0.1. The experiments were executed on a device equipped with 128 GB of RAM, a 4 GB NVIDIA graphics card, and the Windows 11 operating system.

5.1. Performance Report Using "Statlog (Australian Credit Approval)" Dataset

Table 3 provides a performance report for various classifiers using the "Statlog (Australian Credit Approval)" dataset. The table includes several evaluation metrics, such as accuracy (ACC), sensitivity (SNS), specificity (SPC), precision (PRC), F1 score (F1), receiver operating characteristic (ROC), balanced accuracy (BAC), and weighted sum metric (WSM) performance.

Among the classifiers, AdaBoost achieved an ACC of 87.54%. It demonstrated good SNS (88.27%) and SPC (86.95%), indicating its ability to correctly identify both positive and negative instances. However, its PRC (84.42%) was slightly lower compared with other classifiers. The F1 score (86.31%) and ROC (87.61%) were reasonably high, reflecting a balanced performance. The BAC was 87.61%, and the WSM performance was 86.96%. AdaBoost utilized L1 regularization, and the hyperparameters included a logistic regression value of approximately 0.87672 and an estimate of 26. The selected features were X1, X3, X5, X8, X9, X11, and X12.

The Decision Tree (DT) classifier achieved an ACC of 87.10%. It exhibited a slightly lower SNS (84.04%) and higher SPC (89.56%). The PRC (86.58%) and F1 score (85.29%) were relatively good, while the ROC (86.84%) and BAC (86.80%) were also satisfactory. The maximum scaling method was applied, and the DT classifier used entropy as the splitting criterion, a maximum depth of 5, and the "best" splitter. The selected features were X5, X6, X8, X9, and X12. The Histogram-Based Gradient Boosting (HGB) classifier achieved the highest ACC of 88.12%. It showed good SNS (88.93%) and SPC (87.47%). The PRC (85.05%) was slightly lower, but the F1 score (86.94%) and ROC (88.20%) were relatively high. The BAC was 88.20%, and the WSM performance was 87.56%. HGB used standard scaling, a logistic regression value of approximately 0.06022, and a maximum depth of 7. The selected features were X1, X2, X3, X5, X8, X9, X12, and X13.

The K-Nearest Neighbors (KNN) classifier achieved an ACC of 88.12%. It demonstrated good SNS (85.99%) and high SPC (89.82%). The PRC (87.13%) and F1 score (86.56%) were relatively good, while the ROC (87.93%) and BAC (87.91%) were also satisfactory. KNN used standard scaling, the KDTree algorithm, the Manhattan metric, 11 neighbors, and uniform weights. The selected features were X5, X8, X9, X10, X12, and X13. The LightGBM (LGBM) classifier achieved an ACC of 87.54%. It showed good SNS (88.93%) and SPC (86.42%). The PRC (84.00%) was slightly lower compared with other classifiers. The F1 score (86.39%) and ROC (87.68%) were relatively high, indicating a balanced performance. The BAC was 87.67%, and the WSM performance was 86.95%. LGBM utilized MinMax scaling, a logistic regression value of approximately 0.52510, a maximum depth of 1, and an estimate of 61. The selected features were X1, X3, X4, X5, X8, X9, X12, and X13.

The Logistic Regression (LR) classifier achieved an ACC of 87.54%. It exhibited good SNS (90.23%) and reasonable SPC (85.38%). The PRC (83.18%) was the lowest among the classifiers, but the F1 score (86.56%) and ROC (87.84%) were relatively high. The BAC was 87.80%, and the WSM performance was 86.93%. LR used standard scaling, a logistic regression value of approximately 0.05388, and the LBFGS solver. The selected features were X1, X4, X5, X6, X8, X10, X12, and X13. The MultiLayer Perceptron (MLP) classifier achieved an ACC of 87.97%. It demonstrated good SNS (87.62%) and SPC (88.25%). The PRC (85.67%) and F1 score (86.63%) were relatively good, while the ROC (87.94%) and BAC (87.94%) were also satisfactory. The WSM performance was 87.43%. MLP used standard scaling, the ReLU activation function, 64 hidden layers, an adaptive learning rate, and the Adam solver. The selected features were X3, X4, X5, X7, X8, X10, X13, and X14.

Table 3. Performance report using the "Statlog (Australian Credit Approval)" dataset.

Classifier	ACC	SNS	SPC	PRC	F1	ROC	BAC	Mean	Scaler	Hyperparameters	Selected Features
AdaBoost	87.54%	88.27%	86.95%	84.42%	86.31%	87.61%	87.61%	86.96%	L1	LR: ≈0.87672, Est. #: 26	[X1, X3, X5, X8, X9, X11, X12]
DT	87.10%	84.04%	89.56%	86.58%	85.29%	86.84%	86.80%	86.60%	Max	Criterion: Entropy, Max Depth: 5, Splitter: Best	[X5, X6, X8, X9, X12]
HGB	88.12%	88.93%	87.47%	85.05%	86.94%	88.20%	88.20%	87.56%	STD	LR: ≈0.06022, Max Depth: 7	[X1, X2, X3, X5, X8, X9, X12, X13]
KNN	88.12%	85.99%	89.82%	87.13%	86.56%	87.93%	87.91%	87.63%	STD	Algo.: KDTree, Metric: Manhattan, Neighbors #: 11, Weights: Uniform	[X5, X8, X9, X10, X12, X13]
LGBM	87.54%	88.93%	86.42%	84.00%	86.39%	87.68%	87.67%	86.95%	MinMax	LR: ≈0.52510, Max Depth: 1, Est. #: 61	[X1, X3, X4, X5, X8, X9, X12, X13]
LR	87.54%	90.23%	85.38%	83.18%	86.56%	87.84%	87.80%	86.93%	STD	C: ≈0.05388, Solver: LBFGS	[X1, X4, X5, X6, X8, X10, X12, X13]
MLP	87.97%	87.62%	88.25%	85.67%	86.63%	87.94%	87.94%	87.43%	STD	Activation: ReLU, Hidden Layers #: 64, LR: Adaptive, Solver: Adam	[X3, X4, X5, X7, X8, X10, X13, X14]
RF	88.84%	84.69%	92.17%	89.66%	87.10%	88.51%	88.43%	88.48%	L2	Criterion: Gini, Max Depth: 2, Est. #: 90	[X4, X5, X8, X9, X10, X11, X13]
XGB	87.83%	85.67%	89.56%	86.80%	86.23%	87.63%	87.61%	87.33%	L2	LR: ≈0.21922, Max Depth: 23, Est. #: 31, Subsample: ≈0.81806	[X1, X4, X7, X8, X9, X10, X11, X13, X14]

The Random Forest (RF) classifier achieved the highest ACC of 88.84%. It showed relatively low SNS (84.69%) but the highest SPC (92.17%). The PRC (89.66%) was the highest among the classifiers, and the F1 score (87.10%) and ROC (88.51%) were also high. The BAC was 88.43%, and the WSM performance was 88.48%. RF utilized L2 regularization, the Gini criterion, a maximum depth of 2, and an estimate of 90. The selected features were X4, X5, X8, X9, X10, X11, and X13. The XGBoost (XGB) classifier achieved an ACC of 87.83%. It exhibited good SNS (85.67%) and SPC (89.56%). The PRC (86.80%) and F1 score (86.23%) were relatively good, while the ROC (87.63%) and BAC (87.61%) were also satisfactory. XGB used L2 regularization, a logistic regression value of approximately 0.21922, a maximum depth of 23, an estimate of 31, and a subsample of approximately 0.81806. The selected features were X1, X4, X7, X8, X9, X10, X11, X13, and X14.

Based on the evaluation metrics, the best classifier varies depending on the metric considered. For overall ACC, the RF classifier achieved the highest ACC of 88.84%. For SNS, the LR classifier performed the best with 90.23%. For SPC, the RF classifier achieved the highest value of 92.17%. The RF classifier also obtained the highest PRC (89.66%). The F1 score, which considers both PRC and recall, was the highest for the RF classifier at 87.10%. The ROC was the highest for the HGB classifier with 88.20%. Finally, the BAC was also the highest for the RF classifier at 88.43%.

5.2. Performance Report Using "South German Credit" Dataset

Table 4 provides a performance report for various classifiers using the "South German Credit" dataset. The table includes several evaluation metrics, such as ACC, SNS, SPC, PRC, F1, receiver operating characteristic (ROC), BAC, and WSM performance.

Among the classifiers, AdaBoost achieved an ACC of 75.60%, SNS of 84.86%, SPC of 54.00%, PRC of 81.15%, F1 score of 82.96%, ROC of 71.12%, BAC of 69.43%, and WSM performance of 74.16%. It utilized the MinMax scaler, and the selected features were laufkont, laufzeit, moral, verw, hoehe, sparkont, rate, famges, buerge, wohnzeit, wohn, and gastarb. The DT classifier achieved an ACC of 70.70%, SNS of 73.86%, SPC of 63.33%, PRC of 82.46%, F1 score of 77.92%, ROC of 68.80%, BAC of 68.60%, and WSM performance of 72.24%. It utilized the L2 scaler, and the selected features were laufkont, laufzeit, moral, sparkont, verm, wohn, and gastarb.

The HGB classifier achieved an ACC of 77.60%, SNS of 87.86%, SPC of 53.67%, PRC of 81.56%, F1 score of 84.59%, ROC of 72.80%, BAC of 70.76%, and WSM performance of 75.55%. It utilized the STD scaler, and the selected features were laufkont, laufzeit, moral, verw, hoehe, sparkont, beszeit, buerge, verm, weitkred, pers, telef. The KNN classifier achieved an ACC of 73.00%, SNS of 77.86%, SPC of 61.67%, PRC of 82.58%, F1 score of 80.15%, ROC of 70.23%, BAC of 69.76%, and WSM performance of 73.61%. It utilized the STD scaler, and the selected features were laufkont, laufzeit, moral, hoehe, buerge, pers, telef, and gastarb. The LGBM classifier achieved an ACC of 76.60%, SNS of 87.29%, SPC of 51.67%, PRC of 80.82%, F1 score of 83.93%, ROC of 71.72%, BAC of 69.48%, and WSM performance of 74.50%. It utilized the MinMax scaler, and the selected features were laufkont, laufzeit, moral, hoehe, sparkont, beszeit, buerge, verm, alter, wohn, pers, and telef.

The LR classifier achieved an ACC of 75.70%, SNS of 88.29%, SPC of 46.33%, PRC of 79.33%, F1 score of 83.57%, ROC of 70.50%, BAC of 67.31%, and WSM performance of 73.00%. It utilized the MinMax scaler, and the selected features were laufkont, laufzeit, moral, hoehe, sparkont, beszeit, verm, alter, and gastarb. The MLP classifier achieved an ACC of 77.80%, SNS of 88.00%, SPC of 54.00%, PRC of 81.70%, F1 score of 84.73%, ROC of 73.01%, BAC of 71.00%, and WSM performance of 75.75%. It utilized the STD scaler, and the selected features were laufkont, laufzeit, moral, hoehe, beszeit, buerge, verm, pers, and telef.

Table 4. Performance report using the "South German Credit" dataset.

Classifier	ACC	SNS	SPC	PRC	F1	ROC	BAC	Mean	Scaler	Hyperparameters	Selected Features
AdaBoost	75.60%	84.86%	54.00%	81.15%	82.96%	71.12%	69.43%	74.16%	MinMax	LR: ≈1.07005, Est. #: 44	laufkont, laufzeit, moral, verw, hoehe, sparkont, rate, famges, buerge, wohnzeit, wohn, and gastarb
DT	70.70%	73.86%	63.33%	82.46%	77.92%	68.80%	68.60%	72.24%	L2	Criterion: Gini, Max Depth: 1, Splitter: Best	laufkont, laufzeit, moral, sparkont, verm, wohn, and gastarb
HGB	77.60%	87.86%	53.67%	81.56%	84.59%	72.80%	70.76%	75.55%	STD	LR: ≈0.07715, Max Depth: 10	laufkont, laufzeit, moral, verw, hoehe, sparkont, beszeit, buerge, verm, weitkred, pers, and telef
KNN	73.00%	77.86%	61.67%	82.58%	80.15%	70.23%	69.76%	73.61%	STD	Algo.: KDTree, Metric: Manhattan, Neighbors #: 4, Weights: Uniform	laufkont, laufzeit, moral, hoehe, buerge, pers, telef, and gastarb
LGBM	76.60%	87.29%	51.67%	80.82%	83.93%	71.72%	69.48%	74.50%	MinMax	LR: ≈0.25021, Max Depth: 4, Est. #: 69	laufkont, laufzeit, moral, hoehe, sparkont, beszeit, buerge, verm, alter, wohn, pers, and telef
LR	75.70%	88.29%	46.33%	79.33%	83.57%	70.50%	67.31%	73.00%	MinMax	C: ≈1.16623, Solver: LibLinear	laufkont, laufzeit, moral, hoehe, sparkont, beszeit, verm, alter, and gastarb
MLP	77.80%	88.00%	54.00%	81.70%	84.73%	73.01%	71.00%	75.75%	STD	Activation: ReLU, Hidden Layers #: 272, LR: Const., Solver: Adam	laufkont, laufzeit, moral, hoehe, beszeit, buerge, verm, pers, and telef
RF	76.80%	89.71%	46.67%	79.70%	84.41%	71.51%	68.19%	73.85%	MaxAbs	Criterion: Gini, Max Depth: 11, Est. #: 80	laufkont, laufzeit, moral, hoehe, sparkont, rate, buerge, verm, alter, bishkred, and telef
XGB	75.90%	87.71%	48.33%	79.84%	83.59%	70.82%	68.02%	73.46%	STD	LR: ≈0.04300, Max Depth: 10, Est. #: 55, Subsample: ≈0.82929	laufkont, laufzeit, moral, hoehe, sparkont, beszeit, buerge, verm, alter, beruf, pers, telef, and gastarb

The RF classifier achieved an ACC of 76.80%, SNS of 89.71%, SPC of 46.67%, PRC of 79.70%, F1 score of 84.41%, ROC of 71.51%, BAC of 68.19%, and WSM performance of 73.85%. It utilized the MaxAbs scaler, and the selected features were laufkont, laufzeit, moral, hoehe, sparkont, rate, buerge, verm, alter, bishkred, and telef. The XGB classifier achieved an ACC of 75.90%, SNS of 87.71%, SPC of 48.33%, PRC of 79.84%, F1 score of 83.59%, ROC of 70.82%, BAC of 68.02%, and WSM performance of 73.46%. It utilized the STD scaler, and the selected features were laufkont, laufzeit, moral, hoehe, sparkont, beszeit, buerge, verm, alter, beruf, pers, telef, and gastarb.

Based on the evaluation metrics, the best classifier varies depending on the metric considered. For ACC, the best classifier is AdaBoost with 75.60%. For SNS, the best classifier is RF with 89.71%. For SPC, the best classifier is DT with 63.33%. For PRC, the best classifier is KNN with 82.58%. For the F1 score, the best classifier is MLP with 84.73%. For the ROC, the best classifier is MLP with 73.01%. For balanced ACC, the best classifier is MLP with 71.00%. Lastly, for WSM performance, the best classifier is MLP with 75.75%.

5.3. Performance Report Using "Statlog (German Credit Data)" Dataset

Table 5 provides a performance report for various classifiers using the "Statlog (German Credit Data)" dataset. The table includes several evaluation metrics, such as ACC, SNS, SPC, PRC, F1, receiver operating characteristic (ROC), BAC, and WSM performance.

Among the classifiers, AdaBoost achieved an ACC of 77.60%, SNS of 51.67%, SPC of 88.71%, PRC of 66.24%, F1 score of 58.05%, ROC of 72.59%, BAC of 70.19%, and a WSM performance of 69.29%. It used the maximum scaling method, and the hyperparameters LR: $\approx$0.85523 and Est. #: 80. The selected features for AdaBoost were C1, C2, C3, C4, C5, C7, C8, C12, C16, C17, C18, C19, C20. The DT classifier achieved an ACC of 75.70%, SNS of 52.33%, SPC of 85.71%, PRC of 61.09%, F1 score of 56.37%, ROC of 71.01%, BAC of 69.02%, and a WSM performance of 67.32%. It used the maximum scaling method, and the hyperparameters Criterion: Gini, Max Depth: 6, and Splitter: Best. The selected features for DT were C1, C2, C3, C4, C6, C7, C11, C14, C15, C17, and C19.

The HGB classifier achieved an ACC of 78.30%, SNS of 53.33%, SPC of 89.00%, PRC of 67.51%, F1 score of 59.59%, ROC of 73.37%, BAC of 71.17%, and a WSM performance of 70.32%. It used the standard scaling method, and the hyperparameters LR: $\approx$0.08319 and Max Depth: 7. The selected features for HGB were C1, C2, C3, C5, C6, C10, C11, C13, C14, C15, C17, and C19. The KNN classifier achieved an ACC of 77.60%, SNS of 47.67%, SPC of 90.43%, PRC of 68.10%, F1 score of 56.08%, ROC of 72.28%, BAC of 69.05%, and a WSM performance of 68.74%. It used the maximum scaling method, and the hyperparameters Algo.: KDTree, Metric: Euclidean, Neighbors #: 7, and Weights: Distance. The selected features for KNN were C1, C2, C3, C6, C7, C8, C10, C11, C14, C15, C16, C18, and C20.

The LGBM classifier achieved an ACC of 78.30%, SNS of 55.00%, SPC of 88.29%, PRC of 66.80%, F1 score of 60.33%, ROC of 73.55%, BAC of 71.64%, and a WSM performance of 70.56%. It used the standard scaling method, and the hyperparameters LR: $\approx$0.15232, Max Depth: 9, and Est. #: 81. The selected features for LGBM were C1, C2, C3, C5, C6, C7, C9, C10, C11, C12, C13, C17, and C20. The LR classifier achieved an ACC of 77.10%, SNS of 49.33%, SPC of 89.00%, PRC of 65.78%, F1 score of 56.38%, ROC of 71.95%, BAC of 69.17%, and a WSM performance of 68.39%. It used the standard scaling method, and the hyperparameters C: $\approx$1.55321 and Solver: LibLinear. The selected features for LR were C1, C2, C3, C5, C6, C7, C8, C9, C10, C12, C13, C14, C15, C17, C19, and C20.

Table 5. Performance report using the "Statlog (German Credit Data)" dataset.

Classifier	ACC	SNS	SPC	PRC	F1	ROC	BAC	Mean	Scaler	Hyperparameters	Selected Features
AdaBoost	77.60%	51.67%	88.71%	66.24%	58.05%	72.59%	70.19%	69.29%	Max	LR: ≈0.85523, Est. #: 80	C1, C2, C3, C4, C5, C7, C8, C12, C16, C17, C18, C19, and C20
DT	75.70%	52.33%	85.71%	61.09%	56.37%	71.01%	69.02%	67.32%	Max	Criterion: Gini, Max Depth: 6, Splitter: Best	C1, C2, C3, C4, C6, C7, C11, C14, C15, C17, and C19
HGB	78.30%	53.33%	89.00%	67.51%	59.59%	73.37%	71.17%	70.32%	STD	LR: ≈0.08319, Max Depth: 7	C1, C2, C3, C5, C6, C10, C11, C13, C14, C15, C17, and C19
KNN	77.60%	47.67%	90.43%	68.10%	56.08%	72.28%	69.05%	68.74%	Max	Algo.: KDTree, Metric: Euclidean, Neighbors #: 7, Weights: Distance	C1, C2, C3, C6, C7, C8, C10, C11, C14, C15, C16, C18, and C20
LGBM	78.30%	55.00%	88.29%	66.80%	60.33%	73.55%	71.64%	70.56%	STD	LR: ≈0.15232, Max Depth: 9, Est. #: 81	C1, C2, C3, C5, C6, C7, C9, C10, C11, C12, C13, C17, and C20
LR	77.10%	49.33%	89.00%	65.78%	56.38%	71.95%	69.17%	68.39%	STD	C: ≈1.55321, Solver: LibLinear	C1, C2, C3, C5, C6, C7, C8, C9, C10, C12, C13, C14, C15, C17, C19, and C20
MLP	77.60%	54.00%	87.71%	65.32%	59.12%	72.83%	70.86%	69.64%	MinMax	Activation: ReLU, Hidden Layers #: 176, LR: Inv. Scaling, Solver: Adam	C1, C2, C3, C4, C6, C7, C8, C9, C10, C13, C15, C16, C17, C18, C19, and C20
RF	78.70%	48.67%	91.57%	71.22%	57.82%	73.33%	70.12%	70.20%	STD	Criterion: Entropy, Max Depth: 11, Est. #: 94	C1, C2, C3, C5, C6, C7, C8, C10, C12, C13, C15, and C16
XGB	78.50%	51.33%	90.14%	69.06%	58.89%	73.35%	70.74%	70.29%	STD	LR: ≈0.10056, Max Depth: 13, Est. #: 67, Subsample: ≈0.34291	C1, C2, C3, C5, C6, C9, C10, C11, C12, C13, C14, C15, C17, C19, and C20

The MLP classifier achieved an ACC of 77.60%, SNS of 54.00%, SPC of 87.71%, PRC of 65.32%, F1 score of 59.12%, ROC of 72.83%, BAC of 70.86%, and a WSM performance of 69.64%. It used the MinMax scaling method, and the hyperparameters Activation: ReLU, Hidden Layers #: 176, LR: Inv. Scaling, and Solver: Adam. The selected features for MLP were C1, C2, C3, C4, C6, C7, C8, C9, C10, C13, C15, C16, C17, C18, C19, and C20. The RF classifier achieved an ACC of 78.70%, SNS of 48.67%, SPC of 91.57%, PRC of 71.22%, F1 score of 57.82%, ROC of 73.33%, BAC of 70.12%, and a WSM performance of 70.20%. It used the standard scaling method, and the hyperparameters Criterion: Entropy, Max Depth: 11, and Est. #: 94. The selected features for RF were C1, C2, C3, C5, C6, C7, C8, C10, C12, C13, C15, and C16.

The XGB classifier achieved an ACC of 78.50%, SNS of 51.33%, SPC of 90.14%, PRC of 69.06%, F1 score of 58.89%, ROC of 73.35%, BAC of 70.74%, and a WSM performance of 70.29%. It used the standard scaling method, and the hyperparameters LR: ≈0.10056, Max Depth: 13, Est. #: 67, and Subsample: ≈0.34291. The selected features for XGB were C1, C2, C3, C5, C6, C9, C10, C11, C12, C13, C14, C15, C17, C19, and C20. Based on the evaluation metrics, the best classifier varies depending on the metric considered. For ACC, the best classifier is RF with 78.70%. For SNS, the best classifier is LGBM with 55.00%. For SPC, the best classifier is RF with 91.57%. For PRC, the best classifier is RF with 71.22%. For F1, the best classifier is LGBM with 60.33%. For ROC, the best classifier is LGBM with 73.55%. For BAC, the best classifier is LGBM with 71.64%. Finally, for WSM performance, the best classifier is LGBM with 70.56%.

5.4. Overall Discussion and Explainability

Table 6 presents the performance of the best classifiers on three different datasets after conducting 100 trials. The results are reported in terms of mean values and their corresponding standard deviations. The first metric, accuracy, displays the mean accuracy values followed by their standard deviations for the datasets. For example, the mean accuracy value for the Australian Credit dataset is 87.57, with a standard deviation of 0.45. Similarly, the subsequent metrics include the other metrics, where each metric provides the mean values and standard deviations for the corresponding datasets.

Table 6. Performance of the best classifiers on the different datasets after running 100 trials. The results are reported as mean (standard deviation).

Metric	Australian Credit	South German Credit	Statlog (German Credit Data)
ACC	87.57 (0.45)	77.35 (0.4)	78.3 (0.0)
SNS	83.28 (1.1)	87.92 (0.39)	55.0 (0.0)
SPC	91.0 (0.79)	52.68 (0.94)	88.29 (0.0)
PRC	88.14 (0.85)	81.26 (0.31)	66.8 (0.0)
F1	85.63 (0.56)	84.46 (0.28)	60.33 (0.0)
ROC	87.23 (0.47)	72.48 (0.43)	73.55 (0.0)
BAC	87.14 (0.48)	70.3 (0.52)	71.64 (0.0)
WSM	87.14 (0.47)	75.2 (0.42)	70.56 (0.0)

For the "Statlog (Australian Credit Approval)" dataset, the features were X4, X5, X8, X9, X10, X11, X13, utilizing the L2 scaler and RF classifier (with 90 estimators, max depth of 2, and Gini criterion). For the "South German Credit" dataset, the features were laufkont, laufzeit, moral, hoehe, beszeit, buerge, verm, pers, telef, utilizing the STD scaler and MLP classifier (with ReLU activation, Adam optimizer, 272 hidden layers, and constant learning rate). For the "Statlog (German Credit Data)" dataset, the features were C1, C2, C3, C5, C6, C7, C9, C10, C11, C12, C13, C17, and C20, utilizing the STD scaler and LGBM classifier (with 81 estimators, max depth of 9, and learning rate of ≈0.15232).

In the "Statlog (Australian Credit Approval)" dataset, Figure 2 provides the LIME explanation for the model's positive decision (i.e., Yes) with a 97% confidence regarding a testing instance. The figure demonstrates that this decision was primarily influenced by the high confidence value of X8, which was 1 (within the range of −1 to 1).

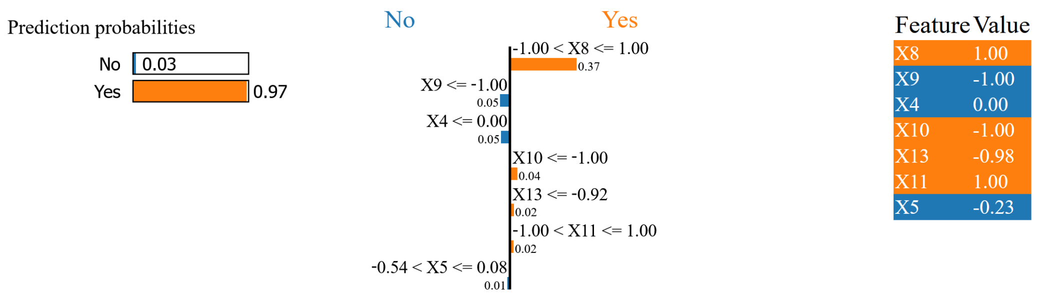

Figure 2. LIME explanation of a decision taken for a testing instance from the "Statlog (Australian Credit Approval)" dataset.

In the "South German Credit" dataset, Figure 3 provides the LIME explanation for the model's positive decision (i.e., Yes) with a 75% confidence regarding a testing instance. The figure demonstrates that this decision was primarily influenced by the high confidence value of laufkont, which was 4 (within the range of 2 to 4), and buerge, which was 1 (less than or equal to 1).

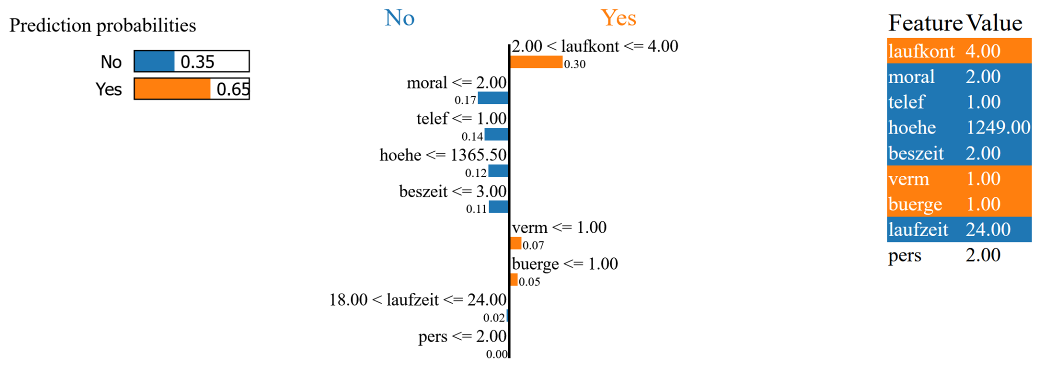

Figure 3. LIME explanation of a decision taken for a testing instance from the "South German Credit" dataset.

In the "Statlog (German Credit Data)" dataset, Figure 4 provides the LIME explanation for the model's negative decision (i.e., No) with a 96% confidence regarding a testing instance. The figure demonstrates that this decision was primarily influenced by the high confidence value of C1, which was 3 (within the range of 1 to 3).

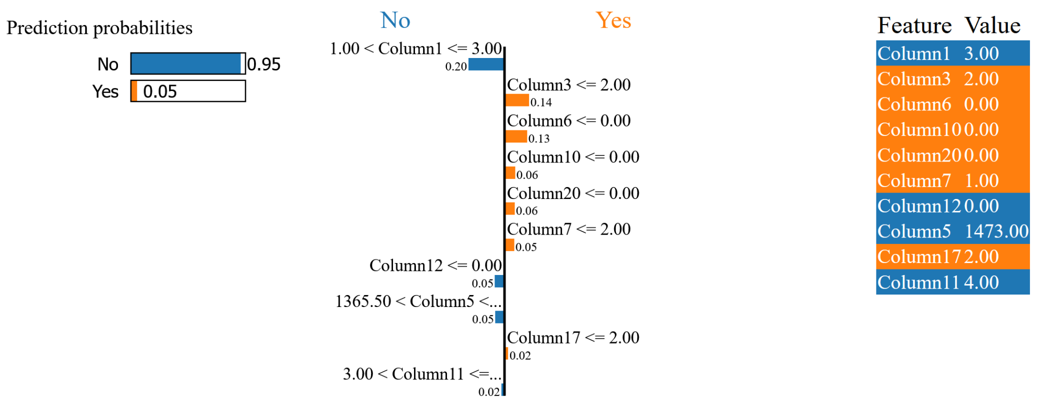

Figure 4. LIME explanation of a decision taken for a testing instance from the "Statlog (German Credit Data)" dataset.

5.5. Related Studies Comparison

Table 7 shows a comparison between the suggested approach and related studies concerning the same used datasets. It can be observed from the literature that there are different research works for credit scoring. For “Statlog (Australian Credit Approval)” [21], the current study achieved an accuracy of 88.84%. This result is competitive with the highest accuracy reported for this dataset, which was 91.91% by Kazemi et al. [15]. In the case of the “Statlog (German Credit Data)” [22], the current study achieved an accuracy of 78.30%. While this accuracy is an improvement over some earlier studies, it falls short of the highest accuracy reported for this dataset, which was 88.89% by Gicić et al. [14]. For “South German Credit ” [23], this study achieved an accuracy of 77.80%. In comparison, Khan and Ghosh [17] reported a slightly higher accuracy of 80.50%. In summary, the current work demonstrates a consistent and competitive performance across all three datasets. The credit scoring model developed in this study appears to be effective in predicting credit risk for various datasets, with accuracy values ranging from 77.80% to 88.84%.

While some related works may surpass the performance of our current study on individual datasets, our approach demonstrates superior performance when applied to specific datasets. For instance, despite Kazemi et al. [15] achieving an impressive 91.91% accuracy in their study on the “Statlog (Australian Credit Approval)” [21] dataset, our model excels with an accuracy of 78.30% when evaluated on the “Statlog German Credit Data" dataset, surpassing Kazemi et al.’s results (accuracy: 67.49%). This underscores the adaptability and competitiveness of our credit scoring model, showcasing its ability to outperform others in various scenarios across diverse datasets. It is worth noting that the relative performance depends on the specific dataset and the benchmarks set by each work.

Table 7. Comparison between the current study and the related studies.

Dataset	Study	Year	Best Performance
Statlog Australian Dataset	Wang et al. [19]	2009	Accuracy: 85.35%
	Haldankar [18]	2016	Accuracy: 76%. Specificity: 55%
	Chen et al. [13]	2022	Accuracy: 88.94%
	Kazemi et al. [15]	2023	Accuracy: 91.91%
	Khan and Ghosh [17]	2023	Accuracy: 86.81%
	This Study	2023	Accuracy: 88.84% (Table 3)
Statlog German Dataset	Wang et al. [19]	2009	Accuracy: 76.41%
	Novakovic et al. [20]	2017	Accuracy: 71.72%
	Chen et al. [13]	2022	Accuracy: 74.33%
	Khatir and Bee [16]	2022	Accuracy: 84.3%
	Gicić et al. [14]	2023	Accuracy: 88.89%
	Kazemi et al. [15]	2023	Accuracy: 67.49%
	This Study	2023	Accuracy: 78.30% (Table 5)
South German Credit	Khan and Ghosh [17]	2023	Accuracy: 80.50%
	This Study	2023	Accuracy: 77.80% (Table 4)

6. Conclusions and Future Work

Credit scoring models have evolved significantly, transitioning from traditional manual processes to sophisticated machine learning techniques. This transformation has been catalyzed by the availability of vast datasets and advancements in data analytics. Contemporary models now leverage intricate algorithms and diverse variables, ensuring more accurate credit risk assessments. These advancements not only shape access to credit but also bolster the stability of the financial system. This research has delved deep into the mathematical underpinnings of machine learning techniques in credit scoring. Central to the exploration is the LIME explainer, providing a granular understanding of machine learning intricacies, thereby enhancing transparency and interpretability. The methodology, grounded in mathematical rigor, encompasses preprocessing techniques and algorithms such as Particle Swarm Optimization (PSO) and Extreme Gradient Boosting (XGB), as well as feature selection methods. Utilizing datasets such as "Statlog (Australian Credit Approval)", "Statlog (German Credit Data)", and "South German Credit", a comprehensive framework is proposed that integrates training, optimization, feature selection via the PSO optimizer, and model explanation using the LIME explainer. This approach ensures a holistic and rigorous credit score classification. Looking forward, the field presents numerous avenues for exploration: (1) Delving into advanced machine learning techniques like deep learning. (2) Incorporating alternative data sources, such as social media metrics. (3) Addressing data imbalances and biases to ensure equitable credit evaluations. (4) Enhancing model interpretability and real-time credit scoring capabilities. (5) Exploring the transferability of models across financial contexts. (6) Prioritizing ethical considerations, ensuring fairness and responsible AI application in credit scoring. In essence, this research serves as a beacon for practitioners, illuminating the path towards more accurate, transparent, and ethical credit scoring methodologies.

Author Contributions: Conceptualization, M.A.E.; Methodology, H.M.B.; Software, H.M.B.; Validation, A.A., B.A., M.A.F. and M.A.E.; Formal analysis, M.B. and M.A.E.; Writing—original draft, H.M.B., M.B. and M.A.E.; Writing—review & editing, A.A., B.A., M.A.F., M.B. and M.A.E.; Visualization, H.M.B. and M.B.; Supervision, M.B. and M.A.E.; Project administration, M.A.E.; Funding acquisition, A.A., B.A. and M.A.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia, project number 445-9-367.

Data Availability Statement: Data are available upon request.

Acknowledgments: The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through project number 445-9-367.

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

References

1. Mays, E. *Handbook of Credit Scoring*; Global Professional Publishig: Chicago, IL, USA; London, UK, 1995.
2. Jensen, H.L. Using neural networks for credit scoring. *Manag. Financ.* **1992**, *18*, 15–26. [CrossRef]
3. Levine, R. Foreign banks, financial development, and economic. In *International Financial Markets: Harmonization versus Competition*; AEI Press: Washington, DC, USA, 1996; pp. 224–255.
4. Torvekar, N.; Game, P.S. Predictive analysis of credit score for credit card defaulters. *Int. J. Recent Technol. Eng.* **2019**, *7*, 4.
5. Thomas, L.; Crook, J.; Edelman, D. *Credit Scoring and Its Applications*; SIAM: Philadelphia, PA, USA, 2017.
6. West, D. Neural network credit scoring models. *Comput. Oper. Res.* **2000**, *27*, 1131–1152. [CrossRef]
7. Abdou, H.A.; Pointon, J. Credit scoring, statistical techniques and evaluation criteria: A review of the literature. *Intell. Syst. Account. Financ. Manag.* **2011**, *18*, 59–88. [CrossRef]
8. Zhang, Z.; Li, Y.; Liu, Y.; Liu, S. A local binary social spider algorithm for feature selection in credit scoring model. *Appl. Soft Comput.* **2023**, *144*, 110549. [CrossRef]
9. Tripathi, D.; Edla, D.R.; Cheruku, R.; Kuppili, V. A novel hybrid credit scoring model based on ensemble feature selection and multilayer ensemble classification. *Comput. Intell.* **2019**, *35*, 371–394. [CrossRef]
10. Zhang, W.; Yang, D.; Zhang, S.; Ablanedo-Rosas, J.H.; Wu, X.; Lou, Y. A novel multi-stage ensemble model with enhanced outlier adaptation for credit scoring. *Expert Syst. Appl.* **2021**, *165*, 113872. [CrossRef]
11. Xia, Y.; Liu, C.; Li, Y.; Liu, N. A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. *Expert Syst. Appl.* **2017**, *78*, 225–241. [CrossRef]
12. Liu, W.; Fan, H.; Xia, M. Credit scoring based on tree-enhanced gradient boosting decision trees. *Expert Syst. Appl.* **2022**, *189*, 116034. [CrossRef]

13. Chen, R.; Ju, C.; Tu, F.S. A Credit Scoring Ensemble Framework using Adaboost and Multi-layer Ensemble Classification. In Proceedings of the 2022 International Conference on Pattern Recognition and Intelligent Systems, Wuhan, China, 29–31 July 2022; pp. 72–79.
14. Gicić, A.; Đonko, D.; Subasi, A. Intelligent credit scoring using deep learning methods. *Concurr. Comput. Pract. Exp.* **2023**, *35*, e7637. [CrossRef]
15. Kazemi, H.R.; Khalili-Damghani, K.; Sadi-Nezhad, S. Estimation of optimum thresholds for binary classification using genetic algorithm: An application to solve a credit scoring problem. *Expert Syst.* **2023**, *40*, e13203. [CrossRef]
16. Hussin Adam Khatir, A.A.; Bee, M. Machine learning models and data-balancing techniques for credit scoring: What is the best combination? *Risks* **2022**, *10*, 169. [CrossRef]
17. Khan, A.; Ghosh, S.K. Machine assistance for credit approval? Random wheel can recommend and explain. *Expert Syst. Appl.* **2023**, *215*, 119231. [CrossRef]
18. Haldankar, A.N.; Bhowmick, K. A cost sensitive classifier for Big Data. In Proceedings of the 2016 IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT), Pune, India, 2–3 December 2016; pp. 122–127.
19. Wang, S.J.; Mathew, A.; Chen, Y.; Xi, L.F.; Ma, L.; Lee, J. Empirical analysis of support vector machine ensemble classifiers. *Expert Syst. Appl.* **2009**, *36*, 6466–6476. [CrossRef]
20. Novakovic, J.Đ.; Veljovic, A.; Ilic, S.S.; Veljovic, V. Application wrapper-based feature selection on C4. 5 decision tree classifier. In Proceedings of the International Scientific Conference, UNITECH 2017, Gabrovo, Bulgaria, 17–18 November 2017.
21. Quinlan, R. Statlog (Australian Credit Approval). UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu/dataset/143/statlog+australian+credit+approval (accessed on 24 May 2023). [CrossRef]
22. Hofmann, H. Statlog (German Credit Data). UCI Machine Learning Repository. 1994. Available online: https://archive.ics.uci.edu/dataset/144/statlog+german+credit+data (accessed on 24 May 2023). [CrossRef]
23. South German Credit. UCI Machine Learning Repository. 2019. Available online: https://archive.ics.uci.edu/dataset/522/south+german+credit (accessed on 24 May 2023). [CrossRef]
24. Balaha, H.M.; Hassan, A.E.S.; El-Gendy, E.M.; ZainEldin, H.; Saafan, M.M. An aseptic approach towards skin lesion localization and grading using deep learning and harris hawks optimization. In *Multimedia Tools and Applications*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 1–29.
25. Shinohara, R.T.; Sweeney, E.M.; Goldsmith, J.; Shiee, N.; Mateen, F.J.; Calabresi, P.A.; Jarso, S.; Pham, D.L.; Reich, D.S.; Crainiceanu, C.M.; et al. Statistical normalization techniques for magnetic resonance imaging. *NeuroImage Clin.* **2014**, *6*, 9–19. [CrossRef]
26. Huang, L.; Qin, J.; Zhou, Y.; Zhu, F.; Liu, L.; Shao, L. Normalization techniques in training dnns: Methodology, analysis and application. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 10173–10196. [CrossRef]
27. Patro, S.; Sahu, K.K. Normalization: A preprocessing stage. *arXiv* **2015**, arXiv:1503.06462.
28. Balaha, H.M.; Ali, H.A.; Youssef, E.K.; Elsayed, A.E.; Samak, R.A.; Abdelhaleem, M.S.; Tolba, M.M.; Shehata, M.R.; Mahmoud, M.R.; Abdelhameed, M.M.; et al. Recognizing arabic handwritten characters using deep learning and genetic algorithms. *Multimed. Tools Appl.* **2021**, *80*, 32473–32509. [CrossRef]
29. Garcia-Gonzalo, E.; Fernandez-Martinez, J.L. A brief historical review of particle swarm optimization (PSO). *J. Bioinform. Intell. Control* **2012**, *1*, 3–16. [CrossRef]
30. Marini, F.; Walczak, B. Particle swarm optimization (PSO). A tutorial. *Chemom. Intell. Lab. Syst.* **2015**, *149*, 153–165. [CrossRef]
31. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. [CrossRef]
32. Eberhart, R.; Kennedy, J. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
33. Tu, C.J.; Chuang, L.Y.; Chang, J.Y.; Yang, C.H. Feature Selection using PSO-SVM. *IAENG Int. J. Comput. Sci.* **2007**, *33*.
34. Chuang, L.Y.; Chang, H.W.; Tu, C.J.; Yang, C.H. Improved binary PSO for feature selection using gene expression data. *Comput. Biol. Chem.* **2008**, *32*, 29–38. [CrossRef]
35. Amoozegar, M.; Minaei-Bidgoli, B. Optimizing multi-objective PSO based feature selection method using a feature elitism mechanism. *Expert Syst. Appl.* **2018**, *113*, 499–514. [CrossRef]
36. Kotsiantis, S.B.; Zaharakis, I.; Pintelas, P. Supervised machine learning: A review of classification techniques. *Emerg. Artif. Intell. Appl. Comput. Eng.* **2007**, *160*, 3–24.
37. Kotsiantis, S.B.; Zaharakis, I.D.; Pintelas, P.E. Machine learning: A review of classification and combining techniques. *Artif. Intell. Rev.* **2006**, *26*, 159–190. [CrossRef]
38. Soofi, A.A.; Awan, A. Classification techniques in machine learning: Applications and issues. *J. Basic Appl. Sci.* **2017**, *13*, 459–465. [CrossRef]
39. Michie, D.; Spiegelhalter, D.J.; Taylor, C.C. *Machine Learning, Neural and Statistical Classification*; Ellis Horwood: London, UK, 1994.
40. Maxwell, A.E.; Warner, T.A.; Fang, F. Implementation of machine-learning classification in remote sensing: An applied review. *Int. J. Remote Sens.* **2018**, *39*, 2784–2817. [CrossRef]
41. Zhao, M.; Li, J. Tuning the hyper-parameters of CMA-ES with tree-structured Parzen estimators. In Proceedings of the 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI), Xiamen, China, 29–31 March 2018; pp. 613–618.
42. Ozaki, Y.; Tanigaki, Y.; Watanabe, S.; Nomura, M.; Onishi, M. Multiobjective tree-structured Parzen estimator. *J. Artif. Intell. Res.* **2022**, *73*, 1209–1250. [CrossRef]

43. Rong, G.; Li, K.; Su, Y.; Tong, Z.; Liu, X.; Zhang, J.; Zhang, Y.; Li, T. Comparison of tree-structured parzen estimator optimization in three typical neural network models for landslide susceptibility assessment. *Remote Sens.* **2021**, *13*, 4694. [CrossRef]
44. Watanabe, S. Tree-structured Parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv* **2023**, arXiv:2304.11127.
45. Ménard, R.; Deshaies-Jacques, M. Evaluation of analysis by cross-validation. Part I: Using verification metrics. *Atmosphere* **2018**, *9*, 86. [CrossRef]
46. Wardhani, N.W.S.; Rochayani, M.Y.; Iriany, A.; Sulistyono, A.D.; Lestantyo, P. Cross-validation metrics for evaluating classification performance on imbalanced data. In Proceedings of the 2019 International Conference on Computer, Control, Informatics and Its Applications (IC3INA), Tangerang, Indonesia, 23–24 October 2019; pp. 14–18.
47. Wienold, J.; Iwata, T.; Sarey Khanie, M.; Erell, E.; Kaftan, E.; Rodriguez, R.G.; Yamin Garretón, J.A.; Tzempelikos, T.; Konstantzos, I.; Christoffersen, J.; et al. Cross-validation and robustness of daylight glare metrics. *Light. Res. Technol.* **2019**, *51*, 983–1013. [CrossRef]
48. Dalianis, H.; Dalianis, H. Evaluation metrics and evaluation. In *Clinical Text Mining: Secondary Use of Electronic Patient Records*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 45–53.
49. Magesh, P.R.; Myloth, R.D.; Tom, R.J. An explainable machine learning model for early detection of Parkinson's disease using LIME on DaTSCAN imagery. *Comput. Biol. Med.* **2020**, *126*, 104041. [CrossRef]
50. Bhattacharya, A. *Applied Machine Learning Explainability Techniques: Make ML Models Explainable and Trustworthy for Practical Applications Using LIME, SHAP, and More*; Packt Publishing Ltd.: Birmingham, UK, 2022.
51. Zhang, Y.; Song, K.; Sun, Y.; Tan, S.; Udell, M. "Why Should You Trust My Explanation?" Understanding Uncertainty in LIME Explanations. *arXiv* **2019**, arXiv:1904.12991.
52. Garreau, D.; Luxburg, U. Explaining the explainer: A first theoretical analysis of LIME. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Virtually, 26–28 August 2020; pp. 1287–1296.
53. Zhao, X.; Huang, W.; Huang, X.; Robu, V.; Flynn, D. Baylime: Bayesian local interpretable model-agnostic explanations. In Proceedings of the Uncertainty in Artificial Intelligence, Online, 27–29 July 2021; pp. 887–896.
54. Pedersen, T.L.; Benesty, M. *Lime: Local Interpretable Model-Agnostic Explanations*; R Package Version 0.4; GitHub: San Francisco, CA, USA, 2018.