electronics

MDPI

Article

Enhanced Credit Card Fraud Detection Model Using Machine Learning

Noor Saleh Alfaiz * and Suliman Mohamed Fati

College of Computer and Information Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia; smfati@yahoo.com or sgaber@psu.edu.sa

* Correspondence: mrnoor220@gmail.com

Abstract: The COVID-19 pandemic has limited people's mobility to a certain extent, making it difficult to purchase goods and services offline, which has led the creation of a culture of increased dependence on online services. One of the crucial issues with using credit cards is fraud, which is a serious challenge in the realm of online transactions. Consequently, there is a huge need to develop the best approach possible to using machine learning in order to prevent almost all fraudulent credit card transactions. This paper studies a total of 66 machine learning models based on two stages of evaluation. A real-world credit card fraud detection dataset of European cardholders is used in each model along with stratified K-fold cross-validation. In the first stage, nine machine learning algorithms are tested to detect fraudulent transactions. The best three algorithms are nominated to be used again in the second stage, with 19 resampling techniques used with each one of the best three algorithms. Out of 330 evaluation metric values that took nearly one month to obtain, the All K-Nearest Neighbors (AllKNN) undersampling technique along with CatBoost (AllKNN-CatBoost) is considered to be the best proposed model. Accordingly, the AllKNN-CatBoost model is compared with related works. The results indicate that the proposed model outperforms previous models with an AUC value of 97.94%, a Recall value of 95.91%, and an *F*1-Score value of 87.40%.

Keywords: credit card fraud; fraud detection; machine learning; CatBoost; XGBoost; random forest; class imbalance

check for updates

Citation: Alfaiz, N.S.; Fati, S.M. Enhanced Credit Card Fraud Detection Model Using Machine Learning. *Electronics* **2022**, *11*, 662. https://doi.org/10.3390/electronics11040662

Academic Editors: Amir H. Gandomi, Fang Chen and Laith Abualigah

Received: 19 December 2021
Accepted: 14 February 2022
Published: 21 February 2022

1. Introduction

As the world is heading to a cashless society, there will be more and more dependency on making online transactions. Modern fraud does not require fraudsters to be physically in the crime locations. They can perform their diabolical activities at the comfort of their homes with many ways of hiding their identities. Such identity hiding techniques include using a VPN, routing the victim's traffic through the Tor network, etc., and it is not easy to trace them back.

The impact of online financial losses cannot be underestimated. Once fraudsters steal card details, they can use the cards themselves or sell the card details to other people, as is the case in India, where the card details of around 70 million people are being sold on the dark web [1]. One of the most serious credit card fraud incidents in recent memory that took place in the UK resulted in GBP 17 million total in financial losses. The incident occurred after a group of international fraudsters conspired to steal the detail information of more than 32,000 credit cards in the mid-2000s [2]. This incident is considered to be the biggest card fraud in history. Thus, the lack of effective security systems results in billion-dollar losses due to credit card fraud [3]. Both cardholders, while using their cards, and card issuers, while processing the transactions, are being reassured that all transactions are benign. Conversely to this belief, fraudsters intend to deceive financial institutions and cardholders into believing that the fraudulent transactions are legitimate.

In addition, there are some fraudulent transactions that happen continuously to obtain financial gain without the knowledge of both card issuers and cardholders. Both authorized institutions and cardholders sometimes do not know that they have fraudulent transactions, and this is the darkest side of credit card transactions. With that in mind, it is a very challenging process to detect fraudulent activity amongst thousands of genuine transactions, especially when fraudulent transactions are significantly less than the genuine ones [4].

There are many fraud detection techniques that help to prevent fraud in the financial industry, such as predictive analytics and data mining, especially modeling algorithms that incorporate clustering techniques and anomaly detection [5]. However, all these techniques cannot be performed without machine learning algorithms, whether they are supervised or unsupervised, which can be effective in credit card fraud classification [6]. However, those machine learning algorithms encounter countless numbers of challenges when trying to detect all fraudulent activity [7].

In the ideal machine learning model, the commonly used evaluation metrics must be at the highest values. In the hopes of moving closer to this ideal model, there are many improvements needed in this arena. The challenges facing credit card fraud detection depend on many factors, such as machine learning algorithms, cross-validation techniques, and resampling techniques. Considering these factors can enhance the performance of the model that can be validated by the evaluation metrics.

In a real-world problem, it is extremely rare to have a balanced dataset to work with, which means that the classification algorithm undermines the importance of the minority class in the dataset in most cases. As a matter of fact, the minority class is the most significant aspect of the classification process, especially in credit card fraud detection. Due to the unbalanced distribution of the classes in the dataset, the proposed approach highlights the imbalance class issue using various resampling techniques after choosing the best machine learning algorithms. Not only are the resampling techniques considered in this paper, but so too are the improved cross-validation (CV) techniques as well.

This paper proposes an advanced approach in terms of choosing the best machine learning algorithm in combination with the best resampling technique. This approach is based on an analysis at two stages using performance evaluation metrics. The first stage aims to analyze nine machine learning algorithms with their default parameters. The nine algorithms are Logistic Regression (LR) [8], K-Nearest Neighbors (KNN) [9], Decision Tree (DT) [10], Naïve Bayes (NB) [11], Random Forest (RF) [9], Gradient Boosting Machines (GBM) [12], Light Gradient Boosting Machine (LightGBM) [13], Extreme Gradient Boosting (XGBoost) [14], and Category Boosting (CatBoost) [15]. Out of these nine algorithms, only the best three algorithms are nominated to be use in the second stage. The second stage aims to analyze 19 resampling techniques with each one of the selected three algorithms from the first stage. These 19 resampling techniques are categorized as follows: 11 undersampling, 6 oversampling, and 2 with combinations of both undersampling and oversampling techniques at once. Furthermore, this stage aims to select the best combination of both algorithm and resampling technique to obtain the best proposed model based on the best overall performance.

This innovative approach stands out by exploring different ways to address the class imbalance issue in the dataset. This is shown in terms of comparing the best machine learning algorithms and using stratified K-fold CV and resampling techniques. Using this number of various algorithms and techniques gives a promising result, especially given that it took nearly one month just to obtain all the evaluation metric values.

The rest of the paper is structured as follows. In the second section, related work is reviewed. The third section describes the proposed approach. In the fourth section, the experimental results are discussed. Finally, the conclusion and future work are summarized in the fifth section.

2. Related Work

Given the importance of credit card fraud, there are renowned techniques to impede this diabolical activity. Not only do financial institutions and banks provide the convenience of financial services, but they also do not hesitate to be the front line protectors of cardholders. In addition, they invest and develop various techniques, including the state-of-the-art machine learning techniques that many systems heavily depend on.

One of these techniques used is DT. It is easy to implement, but it needs to check each transaction one by one [16]. Khatri et al. [17] analyzed various models with an imbalanced European credit card fraud detection (ECCFD) dataset. They did not consider using any resampling techniques. The results indicated that DT was generally the best, with good Recall (79.21%), Precision (85.11%), and time (5 s), while KNN was better in regard to Recall (81.19%), Precision (91.11%), but not with time (463 s).

Another technique involves using LightGBM. Taha and Malebary [18] conducted their experiment on two datasets using LightGBM. The first one is the ECCFD dataset, and the second one is the UCSD-FICO Data Mining Contest 2009 dataset. Using a 5-fold version of the K-Fold CV, they calculated the average values. They compared it to their Optimized Light Gradient Boosting Machine (OLightGBM), which included hyper-parameter tuning with other state-of-the-art algorithms. They found that OLightGBM achieved the highest scores in both datasets. In the first dataset, OLightGBM achieved 90.94% in Area under the Receiver Operating Characteristic Curve (AUC) measures, 98.40% in Accuracy, 40.59% in Recall, 97.34% in Precision, and 56.95% in $F1$-Score. Similarly, in the second dataset, OLightGBM achieved 92.88% in AUC, 98.35% in Accuracy, 28.33% in Recall, 91.72% in Precision, and 43.27% in $F1$-Score.

Another research direction focused on LR and KNN, whereby Vengatesan et al. [19] examined the performance of LR and KNN on the imbalanced ECCFD dataset. The findings were that KNN achieved the best Precision of 95%, Recall of 72%, and $F1$-Score of 82%. In addition, Puh and Brkić [20] studied the performance of different algorithms, namely, RF, the Support Vector Machine (SVM), and LR, on the dataset of European cardholders. They tackled the imbalance class issue in the dataset using the Synthetic Minority Oversampling Technique (SMOTE). SMOTE and LR were used to create their models with some changes in the parameters of the algorithm. The LR parameter C was set to 100 and L2-Regulation was used. They created two models using LR in terms of the learning process. The first one involves static learning and the other one involves incremental learning. The results showed that the AUC score was 91.14% with static learning and the AUC score was 91.07% with incremental learning. The Average Precision score was 73.37% with static learning, and the Average Precision score was 84.13% with incremental learning.

Other researchers focused on RF. Hema [21] evaluated the ECCFD dataset without addressing the imbalanced class issue using three algorithms, which were RF, LR, and Category Boosting (CatBoost). Hema found that RF gave better overall results in terms of Accuracy (99.95%), Precision (91.95%), Recall (79.2%), $F1$-Score (85.1%), MCC (85.31%), and AUC (89%). Kumar et al. [22] conducted a basic study using RF on the ECCFD dataset. They found that the accuracy of RF was 90%.

Some other researchers considered using an Artificial Neural Network (ANN), which simulates how a human brain works [23]. Asha and KR [24] used SVM, KNN, and ANN models on the ECCFD dataset. The results showed that the ANN was the best among the other models, with an Accuracy of 99.92%, a Precision of 81.15%, and a Recall of 76.19%. Dubey et al. [1] conducted an experiment on the Credit Card Customer dataset with the use of ANN. That processed the data in the first layer, which was the input layer, and then the hidden layer, which had 15 neurons and the use of the RELU activation function, and then the output layer, which used the Sigmoid activation function. Eventually, their approach resulted in 99.92% Accuracy, 99.96% Recall, 99.96% Precision, and 99.96% $F1$-Score values. Varmedja et al. [25] conducted their research by splitting the ECCFD in a 80:20 ratio. The algorithms they used were LR, RF, NB, Multilayer Perceptron, and ANN. The imbalanced issue of the dataset was addressed using SMOTE. Each model was updated through multiple epochs depending on the tolerance for the optimization. Their findings indicated that RF achieved the best results in terms of Accuracy (99.96%), Recall (81.63%), and Precision (96.38%).

Local Outlier Factor (LOF) and Isolation Forest (iForest) have been used in some studies. LOF detects outliers based on the local density [26], while iForest also detects outliers based on a tree structure [27]. One of these studies was performed by John and Naaz [28], as they used both LOF and iForest on the ECCFD dataset. They did not address the imbalanced class issue in the dataset, and the results indicated that LOF showed the highest accuracy rate of 97%.

Looking at related work, there are some aspects need to be considered to detect fraudulent activity in credit card transactions. Each approach has its own methodology to enhance the overall performance of their proposed models. However, a machine learning algorithm can have a certain result in one approach, and different results in the other ones. To obtain a better idea on which algorithm performs the best, increasing the number of algorithms used in the experiment should be considered. The imbalance class issue is very common in datasets. Thus, not addressing this issue can lead to poor performance. This issue can be tackled by using stratified CV and resampling techniques, with a significant number of resampling techniques that can be used in experiments. Moreover, the number of evaluation metrics is important for evaluating the model's performance from different angles. Some previous works lack one or more of these aspects. Consequently, an unprecedented approach is proposed.

3. Proposed Approach

The approach is divided into two stages, where the results of the first stage are used as inputs for the second one. The two stages are illustrated in Figure 1.

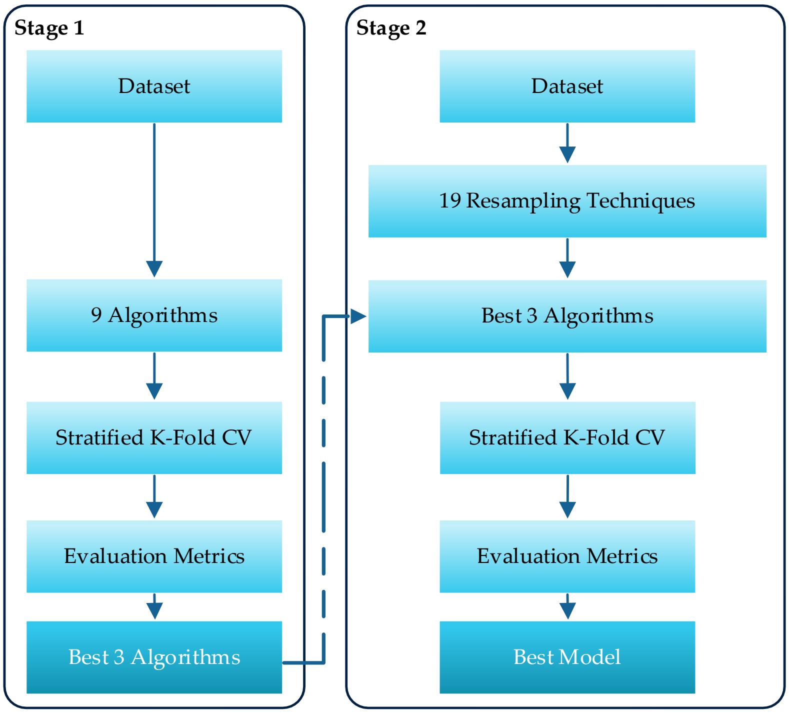

Figure 1. Overall proposed approach.

3.1. Simulation Environment

The simulation environment can be categorized as follows:

3.1.1. Software

The experiment is performed using a 64-bit Windows 10 virtual machine on a server that is equipped with Anaconda Navigator 1.10.0, Jupyter Notebook 6.1.4, and Python 3.8. The libraries used in the Anaconda Navigator environment are Scikit-Learn, Pandas, Numpy, Seaborn, Matplotlib, and Imbalanced-Learn, along with machine learning classifiers.

3.1.2. Hardware

The following points reflect the hardware environment:

- Processor: Intel(R) Xeon(R) CPU D-1527 @ 2.20 GHz 2.19 GHz.
- RAM: 7.00 GB.

The server itself has four CPU cores, and the virtual machine only uses three CPU cores with six threats.

3.2. Dataset

The dataset that is used with this proposed approach is a real-world dataset obtained from Kaggle [29]. It contains transactions made by credit cards in September 2013 by European cardholders that occurred over two days published by Universite Libre de Bruxelles. There are 492 instances of fraud out of 284,807 transactions with 31 features, namely ‘Time’, ‘V1′ to ‘V28′, ‘Amount’, and ‘Class’. This dataset is widely used by many researchers and practitioners found in the related work section; hence, this dataset is chosen to compare the evaluation metric values of our proposed model with theirs.

Preprocessing is not always needed when the dataset meets certain criteria. One of these criteria is to check if there are any missing values that may affect the prediction. When exploring the data, we see that each feature in the dataset has 284,807 values, which means that none of these features is missing any values. Thus, preprocessing is not needed. Figure 2 shows the correlation matrix along with the heat map. The correlation matrix is an advantageous technique that helps us to decide if there is any need to remove a certain feature. Since the correlation matrix shows that all features are related to the ‘Class’ feature regardless of whether the correlation is strong or weak, this leads us to the conclusion that there is no need to pull out any feature; hence, there is no need to preprocess the dataset. Another reason is that the features from ‘V1′ to ‘V28′ are the result of a PCA dimensionality reduction transformation. This was performed because the original data of these features contain sensitive information. Due to the fact that the dataset is already processed, the preprocessing stage is deliberately avoided to obtain a more realistic approach.

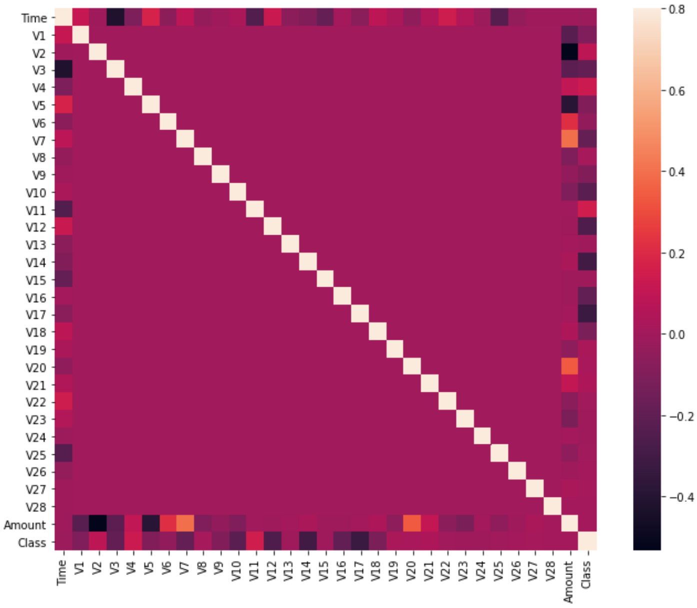

Figure 2. Correlation matrix.

3.3. Algorithms

There are nine machine learning algorithms that are being used in the first stage of the proposed approach. They are LR, KNN, DT, NB, RF, GBM, LightGBM, XGBoost, and finally CatBoost. Each one of these machine learning algorithms’ parameters is set to default, except KNN, where the value of “n_neighbors” is set to 3. Only the best 3 algorithms that achieve the highest performance from the first stage are used again in the second stage, as is shown in Figure 1. Each algorithm is explained as follows.

3.3.1. Logistic Regression

This is one of the traditional machine learning algorithms that is still used today due to its quick analysis and simple method of processing the features of a class. The ability of LR to associate various factors, especially with strong ones, and its ability to adjust to different factors depend on predictor variables and the outcome. LR uses values that are greater than 1 and less than 0 to treat the anomalies in the dataset, and it is not limited to classifying and predicting binominal outcomes, but also multinomial outcomes as well, and it uses the sigmoid function to estimate the values of parameters' coefficients [7]. When LR examines the values of the attributes during an ongoing transaction, it tells us whether the transaction should continue or not, as it is used for clustering [30].

3.3.2. K-Nearest Neighbors

This is a classifier that is used for classification and regression problems. One of its advantages is that it increases the credit card detection rate and decreases false alarm rates. It works based on similarity measures. Thus, it stores all instances that are accessible and arranges new ones [9]. It uses statistical learning techniques, and it works well in supervised machine learning systems, as there is a learning stage that is used to obtain the necessary knowledge, which enables it to classify major differences. However, in unsupervised learning techniques, the training stage is not required. There are three factors in KNN: the distance metrics, the distance rule, and the K value. Distance metrics help to locate the nearest neighbor of any incoming data point, the distance rule classifies the new data point into a class when it is processing its features in relation to the data points in its neighborhood, and the K value is the number of neighbors with which to compare it [7].

3.3.3. Decision Tree

This is a computational method that aims to classify and predict. It has a tree-like process, including a root node, leaf node, and branch. Each internal node indicates a test based on its attributes, the outcome of the test indicates each branch, and the class label holds each leaf node [10]. The process happens recursively when a similarity tree is defined, where the nodes are labelled with attribute names and edges. Those labelled values are attributes that satisfy some conditions and leaves that contain an intensity factor, which is defined as the ratio of the number of transactions that satisfy these conditions over the total number of legitimate transaction in the behavior [31].

3.3.4. CatBoost

There are some classifiers that belong to the DT family, such as CatBoost, which is a cutting-edge, open-source form of gradient boosting for the DT library developed by Yandex researchers and engineers [15]. CatBoost is highly versatile and can be used with a wide range of applications and problems.

3.3.5. XGBoost

Another classifier is XGBoost, which belongs to the DT family, and it is a decision tree-based ensemble machine learning classifier. XGBoost uses a gradient boosting framework that consists of a set of classification and regression trees (CART) [14].

3.3.6. GBM

GBM is another classifier that belongs to the DT family that uses an ensemble technique. It aims to improve Accuracy using an ensemble of trees instead of a single tree, and this algorithm is used for regression and classification, as outlined by Friedman [12,32].

3.3.7. LightGBM

Another algorithm is LightGBM, which is commonly used as a fast, distributed, high-performance, and open-source gradient boosting framework. LightGBM is developed by Microsoft and is based on DT algorithms [13].

3.3.8. Naïve Bayes

This is a supervised machine learning classifier that can be trained to predict future instances of the target class. NB is known to be a powerful probabilistic method that takes advantage of feature and class information from the dataset, which enables it to predict instances in the future [33,34]. "Naïve" is a description of how this technique works because it treats each attribute independently based on the class variable, whether it is present or absent, and "Bayes" is a description of how it calculates the probability of the class correctness [11]. Although NB has a simple mechanism, this algorithm produces good results in many complicated real-world problems.

3.3.9. Random Forest

When encountering a classification or regression problem, the ensemble method, also known as the RF method, can deal with both by growing many decision trees in a way that each tree acts as a weak learner, and those trees are added together, and they become a robust learner [9]. One of the advantages of an RF is that it is effective and fast when handling imbalanced datasets, even when the imbalanced datasets have thousands of features [35]. The way that an RF works is that each tree provides a classification vote for a class. The new object is created, and it is given maximum votes into the class.

3.4. Resampling Techniques

Resampling techniques are commonly used to tackle the imbalance class issue in a dataset [36]. When looking at the dataset used here, the total number of valid cases is 284,315 and the total number of fraud cases is 492. That means that the valid cases are 99.827% of the total number of cases; meanwhile, the fraud cases are only 0.173% of the total number of cases. Certainly, the dataset is highly unbalanced. Thus, resampling techniques come in handy, as the imbalance class issue is related to the performance of the algorithms [37]. There are 3 main categories of resampling techniques: undersampling, oversampling, and the combination of both undersampling and oversampling.

3.4.1. Undersampling

Undersampling techniques are known to provide a compact balanced training set, and one of the advantages of this kind of technique is that it reduces the cost of the learning phase [38]. One of the issues of undersampling techniques is the removal of a large chunk of the training set, especially when the majority class instances are tremendously huge, which leads to the loss of significant cases that would, in turn, lead to difficulties in classification and prediction.

3.4.2. Oversampling

Unlike undersampling, the development of oversampling methods aims to preserve the majority class instances and replicate the minority class instances in order to tackle the issue of imbalanced training set. The issue with this kind of technique is that it may lead to poor performance of the model in some cases because it may be hard to generate the minority data in the training set [39,40].

3.4.3. Combination of Both Undersampling and Oversampling

This combination aims to use both undersampling and oversampling techniques at the same time. By merging these techniques, the imbalance class issue is addressed differently.

3.4.4. The 19 Resampling Techniques

Table 1 states the 19 resampling techniques that are used in the proposed approach, and they are as follows:

- Eleven undersampling techniques.
- Six oversampling techniques.
- Two combinations of both undersampling and oversampling techniques at once.

Table 1. Nineteen Resampling Techniques.

Type	Technique	Abbreviation
Undersampling	Random Undersampling	RUS
	Condensed Nearest Neighbor	CNN
	Tomek	Tomek
	One-Sided Selection	OSS
	Edited Nearest Neighbors	ENN
	Repeated Edited Nearest Neighbors	RENN
	All K-Nearest Neighbors	AllKNN
	Neighborhood Cleaning Rule	NCR
	Near Miss 1	NM1
	Near Miss 2	NM2
	Instance Hardness Threshold	IHT
Oversampling	Random Oversampling	ROS
	Synthetic Minority Oversampling Technique	SMOTE
	Adaptive Synthetic	ADASYN
	Borderline SMOTE 1	Border1
	Borderline SMOTE 2	Border2
	SVM SMOTE	SVMSM
Both Undersampling and Oversampling	SMOTEENN	SMENN
	SMOTETomek	SMTomek

There are two versions of Near Miss, and that is the reason why both NM1 and NM2 are used. Similarly, there are two versions of Borderline SMOTE; the first one is Borderline1, and the second one is Borderline2.

3.5. Stratified K-Fold Cross-Validation

Due to the high sensitivity of the credit card fraud issue, evaluating machine learning models must involve rigorous procedures. A single iteration of evaluating a model may not provide an accurate validation of the model. Thus, the multiple iteration method is the right method to ensure that the model goes through a rigid examination to obtain robust performance.

Provided that multiple iterations are necessary to examine the proposed models, one cannot simply use K-fold cross-validation (CV) when there is an improved version of it. The improved version is stratified K-fold CV. This technique comes in handy, especially when considering that a highly unbalanced dataset may cause a major issue for machine learning algorithms. Despite the fact that K-fold CV is widely used, stratified K-fold CV is used to validate the models of the proposed approach, as it is recommended by Scikit-Learn Developers [41]. Stratified K-fold CV aims to consider the proportion of the minority class in each fold; thus, the minority class is similarly distributed.

Figure 3 shows how stratified K-fold CV is used throughout the proposed approach in both stages. The data are randomly divided in each iteration, with 80% forming part of the training set and 20% forming part of the test set in a very organized manner, which does not lead to data leakage when training the model. The average value of the total of the 5 iterations is considered to be the final value when evaluating the model using evaluation metrics.

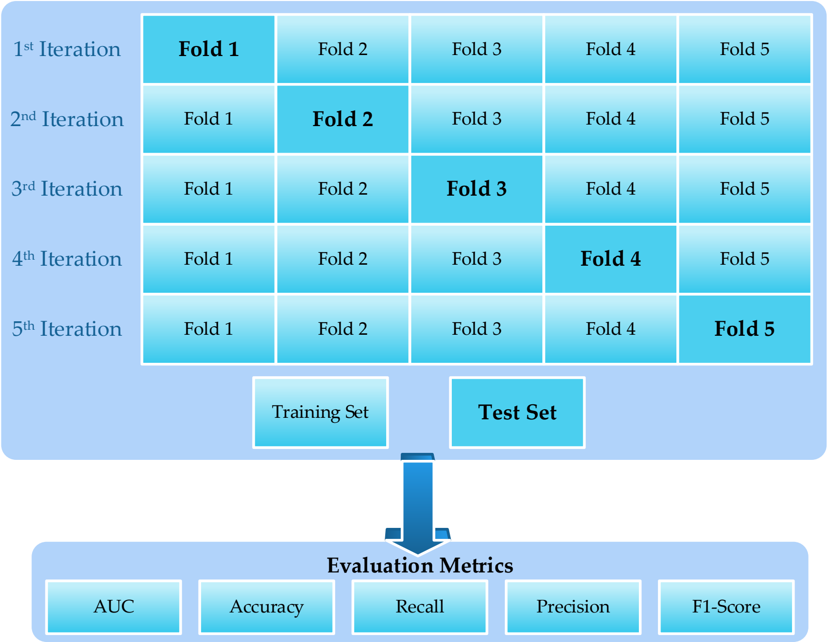

Figure 3. Stratified K-fold cross-validation with evaluation metrics.

3.6. Evaluation Metrics

Each one of the 66 models in both stages is evaluated based on the Area under the Receiver Operating Characteristic Curve (AUC), *Accuracy*, *Recall*, *Precision*, and *F*1-Score. All evaluation metrics used in the proposed approach depend on a confusion matrix in one way or another [42]. A confusion matrix, sometimes referred to as an error matrix [43], is one of the traditional ways to evaluate machine learning model performance that provides four outcomes. They are True Positive (*TP*), False Negative (*FN*), True Negative (*TN*), and False Positive (*FP*) [44–46].

The Receiver Operating Characteristics (ROC) curve provides a graphical representation of the classification model performance. The False Positive Rate (FPR) is plotted on the x-axis and the True Positive Rate (TPR) is on the y-axis in the ROC space. The AUC calculates the whole two-dimensional area underneath the whole ROC curve. This can be achieved by calculating the collective performance of all potential classification thresholds [47].

Accuracy, which is also known as the error rate, is one method to find out how often the classifier classifies a data point properly [48]. Equation (1) [49] shows the Accuracy calculated as the ratio of the correct classified cases (fraud (*TP*) and non-fraud (*TN*)) to the whole pool of cases.

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{1}$$

Precision and *Recall* are evaluation metrics that work in different ways to achieve different results. There is often a trade-off between *Precision* and *Recall*. If *Precision* is higher, *Recall* becomes lower, and if *Precision* is lower, *Recall* becomes higher [50]. *Precision*, also called the Positive Predictive Value, evaluates the correct prediction of the positive cases out of the total positive cases, as is shown in Equation (2):

$$Precison = Positive\ Predicted\ Value = \frac{TP}{TP + FP} \tag{2}$$

Recall, also known as True Positive Rate (*TPR*) and *Sensitivity* [51], is one of the most important evaluation metrics used in detecting fraudulent credit card transactions. Its importance relies on its ability to catch positive cases. The higher the *Recall* value is, the more the detection of fraudulent activity is increased. As a result, it is important to obtain the higher *Recall* value as much as possible to avoid missing any cases of fraud, even though the proposed model may obtain a reasonable amount of *FP*, but not at the expense of *FN* as far as possible. *Recall* is expressed in Equation (3):

$$Recall = Sensitivity = TPR = \frac{True\ Positive}{Total\ Positives} = \frac{TP}{TP + FN} \tag{3}$$

Precision is important, but it is not as important as *Recall*. *Precision* is concerned about the actual *TP* cases of all predicted positive cases. Nonetheless, having a reasonable amount of *FP* cases made by the proposed model is acceptable as long as the proposed model does not compromise on *FN* cases as much as possible. For this reason, *Recall* is more important than *Precision* in credit card fraud detection. On the other hand, *F*1-Score evaluates the performance of the model by combining the *Precision* and *Recall* values. When comparing between two or more models, the *F*1-Score is considered among other evaluation metrics [52]; hence, the classifier with the highest *F*1-Score must be chosen, as is stated in [6]. Equation (4) is the expression of *F*1-Score:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

The AUC, *Accuracy*, *Recall*, *Precision*, and *F*1-Score evaluation metrics give 330 values for each one of the 66 models that are used in the proposed approach across both stages.

4. Results

4.1. The First Stage: Algorithm Comparison

Referring to Figure 1, the first stage provides an initial assessment of how well each machine learning algorithm performs. These algorithms are LR, KNN, DT, NB, RF, GBM, LightGBM, XGBoost, and CatBoost.

Table 2 shows that CatBoost, XGBoost, and RF achieve the highest AUC values. With regard to Accuracy, all algorithms achieve similar values above 99%. However, this evaluation metric does not reflect the overall performance of these nine algorithms. Recall, which is a significant evaluation metric due to its intolerance to any fraudulent activity, gives similar results to the AUC. Regarding Precision, CatBoost, XGBoost, RF, and DT achieve the highest values. With regard to *F*1-Score, CatBoost, XGBoost, and RF achieve the highest values. NB achieves the lowest values in terms of AUC and Recall, whereas KNN achieves the lowest values in terms of Precision and *F*1-Score.

Table 2. Algorithms without Resampling Techniques.

Classifier	**AUC**	**Accuracy**	**Recall**	**Precision**	***F*1-Score**
LR	0.8483	0.9989	0.6971	0.6462	0.6688
KNN	0.9180	0.9984	0.8375	0.0956	0.1711
DT	0.8722	0.9991	0.7449	0.7724	0.7581
NB	0.5736	0.9929	0.1478	0.6485	0.2405
RF	0.9742	0.9996	0.9487	0.7846	0.8588
GBM	0.8841	0.9990	0.7688	0.6034	0.6615
LightGBM	0.6246	0.9959	0.2499	0.5612	0.3410
XGBoost	0.9760	0.9996	0.9523	**0.8008**	0.8698
Catboost	**0.9804**	**0.9996**	**0.9612**	0.7967	**0.8711**

The takeaway from the first stage of assessment is that CatBoost achieved high scores virtually on each evaluation metric. Nonetheless, XGBoost is competing aggressively against CatBoost, and achieves a slightly higher Precision metric than CatBoost. RF is closely behind CatBoost and XGBoost, as it achieves an equal Accuracy metric value to CatBoost and XGBoost. Therefore, the chosen three machine learning algorithms from the first stage are CatBoost, XGBoost, and RF.

4.2. The Second Stage: Resampling Techniques

Tables 3–5 show the results of the second stage when integrating the chosen three algorithms from the first stage with the three resampling technique types. These types are undersampling, oversampling, and the combination of both undersampling and oversampling techniques at the same time. The first row of Tables 3–5 shows the evaluation metric results of the algorithm without any resampling techniques. The next 11 rows show the resulting values of the undersampling techniques, which are RUS, CNN, Tomek, OSS, ENN, RENN, AllKNN, NCR, NM1, NM2, and IHT. The next six rows show the resulting values of the oversampling techniques including ROS, SMOTE, ADASYN, Border1, Border2, and SVMSM. The last two rows of Tables 3–5 show the resulting values of the combined undersampling and oversampling techniques. The two techniques are SMENN and SMTomek.

Table 3. CatBoost with Resampling Techniques.

Technique	AUC	Accuracy	Recall	Precision	*F*1-Score
Without	0.9804	**0.9996**	0.9612	0.7967	0.8711
RUS	0.5331	0.9776	0.0663	0.9065	0.1234
CNN	0.7496	0.9982	0.4994	0.8597	0.6306
Tomek	0.9768	**0.9996**	0.9539	0.7947	0.8670
OSS	**0.9806**	**0.9996**	**0.9616**	0.7948	0.8699
ENN	0.9791	**0.9996**	0.9586	0.7947	0.8689
RENN	0.9782	**0.9996**	0.9568	0.8029	0.8730
AllKNN	0.9794	**0.9996**	0.9591	0.8028	**0.8740**
NCR	0.9792	**0.9996**	0.9587	0.7967	0.8702
NM1	0.5037	0.7698	0.0075	0.9350	0.0149
NM2	0.5005	0.0160	0.0017	**0.9939**	0.0035
IHT	0.5007	0.0300	0.0018	**0.9939**	0.0035
ROS	0.9045	0.9994	0.8093	0.8435	0.8256
SMOTE	0.8627	0.9992	0.7256	0.8476	0.7811
ADASYN	0.8327	0.9990	0.6657	0.8435	0.7431
Border1	0.9139	0.9994	0.8281	0.8231	0.8253
Border2	0.8869	0.9993	0.7741	0.8313	0.8014
SVMSM	0.9291	0.9995	0.8585	0.8231	0.8403
SMENN	0.8486	0.9991	0.6975	0.8414	0.7617
SMTomek	0.8295	0.9990	0.6593	0.8435	0.7400

To find out the best model, going through each one of these evaluation metrics can help us obtain a better idea.

4.2.1. AUC

When looking at Tables 3–5 in regard to AUC, OSS improves all three algorithms, especially CatBoost and XGBoost. Unlike RENN, which improves RF, it does not improve both CatBoost and XGBoost the same way. Therefore, the best model for AUC is CatBoost with OSS.

4.2.2. Accuracy

Tables 3–5 show that the maximum Accuracy value is 99.96%, which is nearly 100%. There are many models that achieve an extremely high score for Accuracy. As such, the best model cannot be determined based on Accuracy. In spite of the inconclusive high scores achieved, IHT and NM2 are extremely poor in this arena. Therefore, this is an alarming indication that these two resampling techniques should not be considered.

4.2.3. Recall

It is shown in Tables 3–5 that CatBoost achieves the highest value, while XGBoost and RF are closely behind CatBoost, in regard to Recall. The highest Recall score is 96.16%, which is achieved by CatBoost with OSS. Significantly, OSS always seems to improve all three chosen algorithms, which is a good indication of this undersampling technique.

Table 4. XGBoost with Resampling Techniques.

Technique	AUC	Accuracy	Recall	Precision	*F*1-Score
Without	0.9760	**0.9996**	0.9523	0.8008	0.8698
RUS	0.5210	0.9638	0.0422	0.9146	0.0806
CNN	0.7058	0.9976	0.4119	0.8638	0.5558
Tomek	0.9748	**0.9996**	0.9499	0.8008	0.8689
OSS	**0.9769**	**0.9996**	**0.9541**	0.7947	0.8669
ENN	0.9727	**0.9996**	0.9457	0.8049	0.8695
RENN	0.9737	**0.9996**	0.9477	0.8029	0.8692
AllKNN	0.9725	**0.9996**	0.9453	0.8008	0.8670
NCR	0.9714	**0.9996**	0.9431	0.8009	0.8659
NM1	0.5026	0.6754	0.0054	0.9472	0.0107
NM2	0.5004	0.0105	0.0017	**0.9959**	0.0035
IHT	0.5008	0.0413	0.0018	0.9939	0.0036
ROS	0.9598	**0.9996**	0.9200	0.8252	0.8697
SMOTE	0.9306	0.9995	0.8614	0.8394	0.8501
ADASYN	0.9203	0.9995	0.8408	0.8435	0.8420
Border1	0.9647	**0.9996**	0.9298	0.8231	**0.8730**
Border2	0.9458	0.9995	0.8920	0.8231	0.8561
SVMSM	0.9650	**0.9996**	0.9304	0.8089	0.8652
SMENN	0.9319	0.9995	0.8640	0.8333	0.8483
SMTomek	0.9227	0.9995	0.8457	0.8415	0.8434

Table 5. RF with Resampling Techniques.

Technique	AUC	Accuracy	Recall	Precision	*F*1-Score
Without	0.9742	**0.9996**	0.9487	0.7846	0.8588
RUS	0.5303	0.9753	0.0608	0.9065	0.1140
CNN	0.7911	0.9987	0.5824	0.8577	0.6924
Tomek	0.9776	**0.9996**	0.9556	0.7825	0.8604
OSS	0.9756	**0.9996**	0.9516	0.7927	0.8648
ENN	0.9740	0.9995	0.9484	0.7765	0.8537
RENN	**0.9789**	**0.9996**	**0.9582**	0.7886	0.8651
AllKNN	0.9759	**0.9996**	0.9521	0.7907	0.8636
NCR	0.9733	**0.9996**	0.9470	0.7927	0.8629
NM1	0.5040	0.7802	0.0081	0.9350	0.0160
NM2	0.5011	0.2504	0.0023	0.9918	0.0046
IHT	0.5008	0.0332	0.0018	**0.9959**	0.0035
ROS	0.9766	**0.9996**	0.9536	0.7886	0.8632
SMOTE	0.9495	0.9995	0.8994	0.8313	0.8639
ADASYN	0.9453	0.9995	0.8908	0.8272	0.8578
Border1	0.9712	**0.9996**	0.9427	0.7988	0.8647
Border2	0.9640	**0.9996**	0.9282	0.8110	0.8656
SVMSM	0.9767	**0.9996**	0.9537	0.7927	**0.8657**
SMENN	0.9453	0.9995	0.8909	0.8273	0.8578
SMTomek	0.9466	0.9995	0.8935	0.8313	0.8612

4.2.4. Precision

Precision is also important in order to obtain a high *F*1-Score, as is the case with Recall. When a model gives an extremely low score in Recall and it gives an extremely high score in Precision, this indicates that this model is not able to classify the real positive cases out of all classified positive cases accurately because the model has a high assumption of classified positive cases. That is, there are significantly more stipulated positive cases than the real number of positive cases. This is the reason why the model gives a low Recall value. However, high values demonstrate the improvement in the classification of the real positive cases as positives. That is the reason why the model gives a high value in Precision. After all, the models' results in both Recall and Precision achieved by IHT and NM2 followed by NM1 must not be considered to be reflective of good performance. That is because they achieve extremely low Recall values, even though they achieve extremely high Precision values, as is shown in Tables 3–5.

4.2.5. *F*1-Score

This evaluation metric is tremendously valuable. It can give a significant indication of the best model so far. CatBoost still achieves the highest results virtually on all the evaluation metrics, including the *F*1-Score. The highest *F*1-Score value is 87.40% and is achieved by the AllKNN undersampling technique along with CatBoost, among other *F*1-Score values in Tables 3–5.

It is noticeable that some resampling techniques perform well, such as RENN and AllKNN, while other ones perform badly, such as RUS, NM1, NM2, and IHT. This is related to how each technique resamples the dataset. This study not only shows the best combination of a machine learning algorithm and resampling technique, but it also shows the ones that should not be considered in future.

4.3. Comparison with Previous Works

After the careful analysis of 66 models and 330 evaluation metric values that took nearly a month to be obtained, the best proposed model is AllKNN along with CatBoost (AllKNN-CatBoost). To put this into perspective, the proposed model is compared with previous works with the same dataset and similar approaches in Table 6.

Table 6. RF with Resampling Techniques.

Paper	Yr.	Model	AUC	Acc.	Recall	Prc.	*F*1
[21]	2021	RF	0.8900	0.9995	0.7920	0.9195	0.8510
[18]	2020	LGBM+Hyper-Parameter	0.9094	0.9840	0.4059	**0.9734**	0.5695
[19]	2020	KNN			0.7200	0.9500	0.8200
[17]	2020	KNN			0.8119	0.9111	
[24]	2021	ANN		0.9992	0.7619	0.8115	
[25]	2019	RF+SMOTE		**0.9996**	0.8163	0.9638	
[20]	2019	LR+SMOTE+ StaticLearning	0.9114				
[28]	2019	LOF		0.9700			
[22]	2019	RF		0.9000			
Ours	2021	AllKNN-CatBoost	**0.9794**	**0.9996**	**0.9591**	0.8028	**0.8740**

As one of main objectives of this paper is to address the imbalance class issue, and this can be looked at in terms of CV and resampling techniques. None of the previous works mentioned in Table 6 used stratified CV. The proposed model uses stratified CV, which can help to address the imbalance class issue, whereas [18] considered using K-fold CV, which overlooks the proportion of minority class in each fold. The other aspect of addressing the imbalance class issue is the use of resampling techniques. When using resampling techniques such as SMOTE in [20,25], the overall performance is very reasonable. However, when the resampling techniques are not used, the overall performance may be adversely impacted, as is the case in [21,28].

When looking at Table 6, the proposed model indeed outperforms other models in terms of AUC, Recall, and *F*1-Score. In regard to Accuracy, AllKNN-CatBoost still achieves a very good value, even though it is tied with the model found in [25]. However, AllKNN-CatBoost does not outperform the other models in terms of Precision despite the outstanding performance. However, it is reasonably acceptable to generate some FP cases, but not at the expense of FN cases as far as possible. In other words, it is bearable to be extra cautious and classify a reasonable number of valid cases as fraudulent, but not to classify fraudulent cases as valid. Thus, obtaining a very high Recall value shows that AllKNN-CatBoost does improve credit card fraud detection without compromising on the detection of fraudulent cases as much as possible. Unlike some previous works that were concerned with one or more evaluation metrics, the proposed model is evaluated based on several evaluation metrics.

The importance of this work lies in comparing different algorithms while addressing the imbalance class issue using stratified CV and resampling techniques. In addition, it considers using multiple evaluation metrics to obtain a better idea on how well the proposed model performs. Considering these aspects enhances the detection of fraudulent activity in credit card transactions. However, this work is limited in considering only one dataset, unlike some other works that have considered more than one dataset, as is the case in [18].

5. Conclusions and Future Work

With increased dependency on online transactions and credit cards, fraudsters and criminals are developing their means to seize other people's money. Nevertheless, a proactive approach must be considered by harnessing artificial intelligence and machine learning tools to aggressively tackle this issue, regardless of how sophisticated the countermeasures are.

The proposed approach is constructed based on two stages. The first stage aims to nominate the best three machine learning algorithms out of nine algorithms. The second stage aims to integrate the best three algorithms with nineteen resampling techniques. Each model in both stages is evaluated based on the Area under the Receiver Operating Characteristic Curve (AUC), Accuracy, Recall, Precision, and $F1$-Score. In the first stage, the nine algorithms are Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT), Naïve Bayes (NB), Random Forest (RF), Gradient Boosting Machines (GBM), Light Gradient Boosting Machine (LightGBM), Extreme Gradient Boosting (XGBoost), and Category Boosting (CatBoost). In the second stage, the 19 resampling techniques are divided as follows: 11 undersampling, 6 oversampling, and 2 combinations of both undersampling and oversampling techniques. The total number of models in both stages are 66, with their 330 evaluation metric values that took nearly one month to obtain. The best model out of all these is AllKNN along with CatBoost (AllKNN-CatBoost). Finally, AllKNN-CatBoost is compared with previous works with the same dataset and similar approaches. Indeed, AllKNN-CatBoost outperforms previous models in terms of AUC (97.94%), Recall (95.91%), and $F1$-Score (87.40%).

Future work may include using another dataset and other optimization algorithms. Some of these algorithms are Monarch Butterfly Optimization (MBO) [53], Earthworm Optimization Algorithm (EWA) [54], Elephant Herding Optimization (EHO) [55], Moth Search (MS) algorithm [56], Slime Mold Algorithm (SMA), and Harris Hawks Optimization (HHO) [57].

Author Contributions: Conceptualization, methodology, validation, formal analysis, investigation, and visualization, N.S.A. and S.M.F.; software and writing—original draft preparation, N.S.A.; writing—review and editing, S.M.F. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges (APC) for this publication.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dubey, S.C.; Mundhe, K.S.; Kadam, A.A. Credit Card Fraud Detection using Artificial Neural Network and BackPropagation. In Proceedings of the 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Rasayani, India, 13–15 May 2020; pp. 268–273. [CrossRef]
2. Martin, T. Credit Card Fraud: The Biggest Card Frauds in History. Available online: https://www.uswitch.com/credit-cards/guides/credit-card-fraud-the-biggest-card-frauds-in-history/ (accessed on 22 January 2022).
3. Zhang, X.; Han, Y.; Xu, W.; Wang, Q. HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture. *Inf. Sci.* **2019**, *557*, 302–316. [CrossRef]
4. Makki, S.; Assaghir, Z.; Taher, Y.; Haque, R.; Hacid, M.-S.; Zeineddine, H. An experimental study with imbalanced classification approaches for credit card fraud detection. *IEEE Access* **2019**, *7*, 93010–93022. [CrossRef]

5. McCue, C. *Advanced Topics. Data Mining and Predictive Analysis*; Butterworth-Heinemann: Oxford, UK, 2015; pp. 349–365.
6. Berad, P.; Parihar, S.; Lakhani, Z.; Kshirsagar, A.; Chaudhari, A. A Comparative Study: Credit Card Fraud Detection Using Machine Learning. *J. Crit. Rev.* **2020**, *7*, 1005.
7. Jain, Y.; Namrata, T.; Shripriya, D.; Jain, S. A comparative analysis of various credit card fraud detection techniques. *Int. J. Recent Technol. Eng.* **2019**, *7*, 402–403.
8. Tolles, J.; Meurer, W.J. Logistic regression: Relating patient characteristics to outcomes. *JAMA* **2016**, *316*, 533–534. [CrossRef] [PubMed]
9. Shirodkar, N.; Mandrekar, P.; Mandrekar, R.S.; Sakhalkar, R.; Kumar, K.C.; Aswale, S. Credit card fraud detection techniques–A survey. In Proceedings of the 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Shiroda, India, 13–15 May 2020; pp. 1–7. [CrossRef]
10. Gaikwad, J.R.; Deshmane, A.B.; Somavanshi, H.V.; Patil, S.V.; Badgujar, R.A. Credit Card Fraud Detection using Decision Tree Induction Algorithm. *Int. J. Innov. Technol. Explor. Eng. IJITEE* **2014**, *4*, 66–67.
11. Zareapoor, M.; Seeja, K.; Alam, M.A. Analysis on credit card fraud detection techniques: Based on certain design criteria. *Int. J. Comput. Appl.* **2012**, *52*, 35–42. [CrossRef]
12. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
13. Microsoft. LightGBM. Available online: https://github.com/microsoft/LightGBM (accessed on 22 January 2021).
14. XGBoost Developers. Introduction to Boosted Trees. Available online: https://xgboost.readthedocs.io/en/latest/tutorials/model.html (accessed on 22 January 2022).
15. Yandex Technologies. CatBoost. Available online: https://yandex.com/dev/catboost/ (accessed on 22 January 2022).
16. Delamaire, L.; Abdou, H.; Pointon, J. Credit card fraud and detection techniques: A review. *Banks Bank Syst.* **2009**, *4*, 61.
17. Khatri, S.; Arora, A.; Agrawal, A.P. Supervised machine learning algorithms for credit card fraud detection: A comparison. In Proceedings of the 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 29–31 January 2020; pp. 680–683. [CrossRef]
18. Taha, A.A.; Malebary, S.J. An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine. *IEEE Access* **2020**, *8*, 25579–25587. [CrossRef]
19. Vengatesan, K.; Kumar, A.; Yuvraj, S.; Kumar, V.; Sabnis, S. Credit card fraud detection using data analytic techniques. *Adv. Math. Sci. J.* **2020**, *9*, 1185–1196. [CrossRef]
20. Puh, M.; Brkić, L. Detecting credit card fraud using selected machine learning algorithms. In Proceedings of the 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Zagreb, Croatia, 20–24 May 2019; pp. 1250–1255. [CrossRef]
21. Hema, A. Machine Learning methods for Discovering Credit Card Fraud. *Int. Res. J. Comput. Sci.* **2020**, *8*, 1–6.
22. Kumar, M.S.; Soundarya, V.; Kavitha, S.; Keerthika, E.; Aswini, E. Credit card fraud detection using random forest algorithm. In Proceedings of the 2019 3rd International Conference on Computing and Communications Technologies (ICCCT), Chennai, India, 21–22 February 2019; pp. 149–153. [CrossRef]
23. Patidar, R.; Sharma, L. Credit card fraud detection using neural network. *Int. J. Soft Comput. Eng. IJSCE* **2011**, *1*, 32–38.
24. Asha, R.; KR, S.K. Credit card fraud detection using artificial neural network. *Glob. Trans. Proc.* **2021**, *2*, 35–41. [CrossRef]
25. Varmedja, D.; Karanovic, M.; Sladojevic, S.; Arsenovic, M.; Anderla, A. Credit card fraud detection-machine learning methods. In Proceedings of the 2019 18th International Symposium Infoteh-Jahorina (Infoteh), Novi Sad, Serbia, 20–22 March 2019; pp. 1–5. [CrossRef]
26. Breunig, M.M.; Kriegel, H.P.; Ng, R.T.; Sander, J. LOF: Identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX, USA, 15–18 May 2000; pp. 93–104. [CrossRef]
27. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Ballarat, VIC, Australia, 15–19 December 2008; pp. 413–422. [CrossRef]
28. John, H.; Naaz, S. Credit card fraud detection using local outlier factor and isolation forest. *Int. J. Comput. Sci. Eng* **2019**, *7*, 1060–1064. [CrossRef]
29. Dal Pozzolo, A.; Caelen, O.; Johnson, R.A.; Bontempi, G. Calibrating probability with undersampling for unbalanced classification. In Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence, Cape Town, South Africa, 7–10 December 2015; pp. 159–166. [CrossRef]
30. Sahin, Y.; Duman, E. Detecting credit card fraud by ANN and logistic regression. In Proceedings of the 2011 International Symposium on Innovations in Intelligent Systems and Applications, Istanbul, Turkey, 15–18 June 2011; pp. 315–319.
31. Kokkinaki, A.I. On atypical database transactions: Identification of probable frauds using machine learning for user profiling. In Proceedings of the 1997 IEEE Knowledge and Data Engineering Exchange Workshop, Nicosia, Cyprus, 4 November 1997; p. 109.
32. Piryonesi, S.M.; El-Diraby, T.E. Data analytics in asset management: Cost-effective prediction of the pavement condition index. *J. Infrastruct. Syst.* **2020**, *26*, 04019036. [CrossRef]
33. Maes, S.; Tuyls, K.; Vanschoenwinkel, B.; Manderick, B. Credit card fraud detection using Bayesian and neural networks. In Proceedings of the 1st International Naiso Congress on Neuro Fuzzy Technologies, Brussel, Belgium, 16–19 January 2002; pp. 261–270.

34. Syeda, M.; Zhang, Y.Q.; Pan, Y. Parallel granular neural networks for fast credit card fraud detection. In Proceedings of the 2002 IEEE World Congress on Computational Intelligence. 2002 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE'02. Proceedings (Cat. No. 02CH37291), Atlanta, GA, USA, 12–17 May 2002; pp. 572–577. [CrossRef]
35. Seeja, K.; Zareapoor, M. Fraudminer: A novel credit card fraud detection model based on frequent itemset mining. *Sci. World J.* **2014**, *2014*, 1–10. [CrossRef]
36. Scikit-Learn-Contrib. Imbalanced-Learn. Available online: https://github.com/scikit-learn-contrib/imbalanced-learn (accessed on 22 January 2022).
37. He, H.; Garcia, E.A. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1263–1284. [CrossRef]
38. Dal Pozzolo, A.; Caelen, O.; Bontempi, G. When is undersampling effective in unbalanced classification tasks? In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Porto, Portugal, 7–11 September 2015; pp. 200–215.
39. García, V.; Mollineda, R.A.; Sánchez, J.S. On the k-NN performance in a challenging scenario of imbalance and overlapping. *Pattern Anal. Appl.* **2008**, *11*, 269–280. [CrossRef]
40. Cieslak, D.A.; Chawla, N.V. Start globally, optimize locally, predict globally: Improving performance on imbalanced data. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Notre Dame, IN, USA, 15–19 December 2008; pp. 143–152.
41. Scikit-Learn Developers. 3.1. Cross-validation: Evaluating Estimator Performance. Available online: https://scikit-learn.org/stable/modules/cross_validation.html (accessed on 22 January 2022).
42. Hanley, J.A.; McNeil, B.J. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* **1982**, *143*, 29. [CrossRef]
43. Stehman, S.V. Selecting and interpreting measures of thematic classification accuracy. *Remote Sens. Environ.* **1997**, *62*, 77. [CrossRef]
44. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 862. [CrossRef]
45. Powers, D.M. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv* **2020**, arXiv:2010.16061.
46. Chicco, D.; Tötsch, N.; Jurman, G. The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. *BioData Min.* **2021**, *14*, 1–22. [CrossRef] [PubMed]
47. Google Developers. Classification: ROC Curve and AUC. Available online: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc (accessed on 22 January 2022).
48. DeepAI. Accuracy (Error Rate). Available online: https://deepai.org/machine-learning-glossary-and-terms/accuracy-error-rate (accessed on 22 January 2022).
49. Guido, S.; Müller, A.C. *Introduction to Machine Learning with Python A Guide for Data Scientists*; O'Reilly: Sebastopol, CA, USA, 2021; p. 282.
50. C3, AI. Precision. Available online: https://c3.ai/glossary/machine-learning/precision/ (accessed on 22 January 2022).
51. Masís, S. *Interpretable Machine Learning with Python: Learn to Build Interpretable High-Performance Models with Hands-On Real-World Examples*; Packt Publishing Ltd.: Birmingham, UK, 2021; p. 81.
52. Prusti, D.; Rath, S.K. Fraudulent transaction detection in credit card by applying ensemble machine learning techniques. In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Rourkela, India, 6–8 July 2019; pp. 1–6. [CrossRef]
53. Wang, G.-G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2019**, *31*, 1995–2014. [CrossRef]
54. Ghosh, I.; Roy, P.K. Application of earthworm optimization algorithm for solution of optimal power flow. In Proceedings of the 2019 International Conference on Opto-Electronics and Applied Optics (Optronix), Kolkata, India, 18–20 March 2019; pp. 1–6. [CrossRef]
55. Wang, G.-G.; Deb, S.; Coelho, L.d.S. Elephant herding optimization. In Proceedings of the 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI), Xuzhou, China, 7–9 December 2015; pp. 1–5. [CrossRef]
56. Wang, G.-G. Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Comput.* **2018**, *10*, 151–164. [CrossRef]
57. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]