Scientific African 8 (2020) e00464

Contents lists available at ScienceDirect

Scientific African

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

A framework for detecting credit card fraud with cost-sensitive meta-learning ensemble approach

Toluwase Ayobami Olowookere[a,b,*], Olumide Sunday Adewale[c]

[a] Department of Computer Science, Ekiti State University, Ado-Ekiti, Nigeria
[b] Department of Computer Science, Redeemer's University, Ede, Nigeria
[c] School of Computing, Federal University of Technology, Akure, Nigeria

ARTICLE INFO

Article history:
Received 29 February 2020
Revised 29 May 2020
Accepted 23 June 2020

Keywords:
Ensemble
Cost-sensitive
Credit card
Fraud detection
Meta-learning
Receiver operating characteristic

ABSTRACT

Electronic payment systems continue to seamlessly aid business transactions across the world, and credit cards have emerged as a means of making payments in E-payment systems. Fraud due to credit card usage has, however, remained a major global threat to financial institutions with several reports and statistics laying bare the extent of this challenge. Several machine learning techniques and approaches have been established to mitigate this prevailing menace in payment systems, effective amongst which are ensemble methods and cost-sensitive learning techniques. This paper proposes a framework that combines the potentials of meta-learning ensemble techniques and cost-sensitive learning paradigm for fraud detection. The approach of the proposed framework is to allow base-classifiers to fit traditionally while the cost-sensitive learning is incorporated in the ensemble learning process to fit the cost-sensitive meta-classifier without having to enforce cost-sensitive learning on each of the base-classifiers. The predictive accuracy of the trained cost-sensitive meta-classifier and base classifiers were evaluated using Area Under the Receiver Operating Characteristic curve (AUC). Results obtained from classifying unseen data show that the cost-sensitive ensemble classifier maintains an excellent AUC value indicating consistent performance across different fraud rates in the dataset. These results indicate that the cost-sensitive ensemble framework is efficient in producing cost-sensitive ensemble classifiers that are capable of effectively detecting fraudulent transactions in different databases of payment systems irrespective of the proportion of fraud cases as compared to the performances of ordinary ensemble classifiers.

Introduction

The growing need for effective, efficient, and reliable methods for mitigating the rising occurrence of fraud in electronic payment systems particularly via the instrument of credit cards, cannot be over-emphasized. This position is bolstered by the rate of our reliance on electronic payment means in transacting businesses across the world. Quite prominent among

* Corresponding author at: Department of Computer Science, Redeemer's University, Ede, Nigeria.
E-mail addresses: toluwase.olowookere@eksu.edu.ng, olowookereta@run.edu.ng (T.A. Olowookere).

https://doi.org/10.1016/j.sciaf.2020.e00464

these methods of electronic payment are credit cards, whose transactions keep growing in size on day to day basis in commerce and this has attracted fraudsters to developed various mechanisms of committing fraud via this channel. The prevalence in credit card usage in electronic commerce comes along with some significant security and economic challenges, as fraudulent usage of credit cards has become an attractive source of ill-gotten revenue for criminally-minded persons. Also, the apparent exposure of the security loopholes of traditional credit card processing systems has dramatically increased the occurrence of credit card fraud, and this has resulted in a yearly loss of billions of dollars globally. Most fraudsters have devised sophisticated methods to commit credit card fraud [1]. Frauds due to credit cards are deemed to come in various ways. At the inception of credit card usage, the most prevalent and commonly known type of fraud was counterfeit card fraud in which cards are being cloned. Due to prevention measures implemented today, the counterfeit fraud has faded away and is overthrown by a modern method known as Cardholder-Not-Present (CNP) fraud. This fraud type is due to remote payments and accounts for a large proportion of frauds committed in electronic payment in the world today [2].

These worldwide fraudulent activities present a common global challenge to banks and other financial institutions involved in the issuance of credit cards. A study conducted in 1996 by the American Bankers Association reveals that the estimated gross fraud loss in 1995 was a whopping $790 million [3]. Due to the fact that about 71% of all credit cards are issued in the USA, the majority of losses that emanate from the usage of credit card fraud are suffered by the USA alone. In 2005, the total fraud loss in the USA stood at $2.7 billion, and it went up to $3.2 billion in 2007. By the end of the year 2018, credit card loss due to remote purchases (card-not-present fraud) has risen to 76% [4]. A survey carried out on over 160 companies revealed that online fraud (remote purchases) was twelve times higher than offline fraud (committed by using a stolen physical card) [5]. Many papers have reported massive amounts of losses in different countries [6]. Whenever there is an occurrence of credit card fraud, the loss involved is often enormous and does affect the concerned parties. This loss is usually grave because once a fraudster has obtained the needed information on a credit card, he moves swiftly to start making quite large purchases. The fraudster usually does this in order to exhaust the credit granted to the cardholder to his own advantage before his fraudulent activity gets noticed.

To curtail the problems as mentioned earlier, several prevention measures including chip and PIN verification, card activation, card verification code/value, and address verification services, were devised to curb fraud. Because smart fraudsters can successfully bypass prevention measures, detection mechanisms have, over the years, been developed to tackle fraud in credit card transactions. More of the so far developed fraud detection schemes have however, continued to rely on advanced machine learning techniques in mitigating credit card fraud.

Research motivation

With the increasing reliance on electronic payment systems, particularly through the globally rising usage of credit cards by individuals and business owners in recent times, all major stakeholders such as electronic card manufacturers, payment operators, and cardholders have continued to seek for effective measures to tackle the threats posed by credit card frauds. Hence there appears to be a regular need to continue to improve on efficiency and effectiveness of previous fraud detection schemes with the aim of achieving a more reliable and secure payment system. Though several related works have been presented in the literature to effectively handle this menace, with a vast majority based on advanced machine learning techniques, more needs to be done to curb credit card fraud.

Pun [7] proposed the use of a meta-classifier to act as a filter for the transaction data of credit cards. The meta-classifier used the predictions obtained from three base classifiers to arrive at the final prediction of any transaction. The dataset used was derived from a Canadian bank's eleven months' credit card transactions. The approach of the model includes three base classifiers that were constructed via the naïve Bayesian, decision tree, and k-nearest neighbor algorithms. The naïve Bayesian algorithm was also used to combine the base classifier predictions at the meta-level, to construct the final classifier. However, in pursuing to minimize the misclassification of transactions, the Bayesian algorithm used to build the meta-classifier assumes that all misclassification errors carry the same cost. In this assumption, therefore, the technique treats each misclassification errors equally. In reality, this assumption is not the case for the real-world credit card fraud problem in which the costs involved in every transaction vary.

Sahin et al. [8] proposed a cost-sensitive decision tree approach for detecting credit card fraud. The cost-sensitive tree approach was developed to minimize the total misclassification cost during the process of selecting the splitting attribute that must be used at each non-terminal node of a tree. In the approach, varying misclassification costs were thus employed. The result of using the strategy on credit card data shows that the approach outperforms the existing traditional methods with respect to accuracy and true positive rate.

Bahnsen et al. [9] in work "Fraud Detection by stacking Cost-Sensitive Decision Trees" proposed a meta-learning approach to an existing cost-sensitive decision trees algorithm, by training an ensemble of such cost-sensitive decision trees, and then combining them through stacking ensemble approach. The work further asserted, and correctly so that problem of detecting fraud due to the credit card is by nature an example dependent and cost-sensitive classification problem, in which misclassifying a fraudulent transaction does have some financial impacts of varying degrees in amount involved. Therefore, costs due to misclassification actually vary between transaction instances and not only within classes (be it fraud/non-fraud) result from deploying the method on real-world credit card dataset provided by a European card processing firm show that the method achieves savings of about 73.3% (above 2% points) more than that which can be obtained by a single cost-sensitive decision tree.

Bahnsen et al. [10] in putting forward an Example-Dependent Cost-Sensitive Logistic Regression, proposed an instance-based Example-Dependent cost matrix for credit scoring. They did introduce these example-dependent costs into a logistic regression algorithm and used the proposed approach on both Kaggle Credit Dataset and PAKDD Credit Dataset. The results obtained revealed the importance of using real example-dependent financial costs due to every financial credit transaction. The evaluation of the approach further confirmed that the inclusion of the costs of every individual transaction instance, and employing example-dependent cost-sensitive techniques such as Bayes minimum risk classifier, can lead to better results in terms of higher savings regardless of algorithms used in the probabilities estimations.

An observation according to Bahnsen et al. [11] reveals that as at the year 2013, the state-of-the-art techniques for credit card fraud detection fail to capture the real costs involved in credit card fraud as a measure to evaluate the techniques. Bahnsen et al. [11] therefore proposed a measure that realistically represents the monetary gains and losses due to fraud by defining the cost of false negative in a cost matrix to be the amount involved in the transaction of interest. The paper did not only use the proposed cost matrix in evaluating algorithms but goes further to use it to develop a cost-sensitive classification algorithm, which is based on Bayes minimum risk. To determine how various algorithms perform on different class distributions, five databases of credit card transactions were created, with each one having a different proportion of frauds in this order 1%, 5%, 10%, 20%, and 50%. Results obtained by the proposed algorithm and state of the art algorithms were compared, and the proposed method shows improvements up to 23% measured by cost.

In 2015, Bahnsen et al. [12] in the research "Ensemble of Example-Dependent Cost-Sensitive Decision Trees" proposed a framework of ensembles of example-dependent cost-sensitive decision trees. The framework enlists the creation of different example-dependent cost-sensitive decision trees to form randomly selected subsamples of the training samples. These decision trees were then combined using ensemble approaches. The framework was evaluated on a credit card fraud detection database and some other real-world applications. They reported that the proposed method performed much better in the sense of higher savings compared to Bayes minimum risk and cost-sensitive decision trees.

Improvement of the detection accuracy of detection algorithms is a sought after because, any slight miss of a fraudulent transaction can be a catastrophic loss. From the above review, it is observed that within the domain of credit card fraud detection, apart from a few identified in this paper, the vast modern existing techniques and approaches have either focused on creating ensembles of classifiers and failed to credibly take into consideration, the varying misclassification costs in their learning procedure or pursuing to build cost-sensitive classifiers while ignoring to harness the powers of ensemble meta-learning paradigm. This paper, therefore, identifies that the two approaches can result in a robust classifier and proposes a framework based on a merger of the two apparently compatible methods and harnessing their potentials in fraud detection. It is quite important to note the fact that the cost-sensitive learning approach and meta-learning ensemble method are two machine learning concepts with high potentials in classification problems.

The rest of this paper is structured as follows: Section 3 presents the architecture of the proposed framework, while Section 4 describes the proposed workflow depicting the proposed framework. Section 5 gives the mathematical model formulation for the proposed framework, while Section 6 describes the dataset used, and Section 7 shows the experimental setup. Test results and performance evaluation are presented in Section 8, while Section 9 discusses the presented results in detail. Section 10 draws a conclusion on the work.

The architecture of the proposed framework

The proposed framework for Credit Card Fraud Detection is a three-phase architecture. The architecture consists of the Data Preparation phase, Heterogeneous Base-Classifiers Learning phase, and Cost-sensitive Meta-Classifier Learning and a Transaction detection phase. Fig. 1 illustrates the architecture of the proposed framework.

The data preparation phase

This phase details the preprocessing of the dataset obtained from payment systems and serves as the preprocessing phase of the proposed framework. Data preprocessing remains a critical step in any machine learning and data mining task. Here, the credit card transaction dataset is split into two disjoint datasets; the training and the test sets. Another essential step in this phase is to check for missing values in the dataset and handle them appropriately if there are missing values found. Also, feature importance is assessed via Pearson correlation with a view of determining a need for possible dimensionality reduction in the dataset.

The heterogeneous base-classifiers learning phase (Level-0 training)

Based on the concept of meta-learning ensemble methodology, two training phases were designed for inducing classifier; the Base-Level and the Meta-Level training phases. The heterogeneous base-classifiers learning phase of the proposed architecture constitutes the base-level training phase of the model building, also referred to as level-0 learning. Here, different first level classifiers are induced from transaction data (the training set) that have already been prepared to be fed into machine learning algorithms at the data preparation phase. In the classification task, a learned model is referred to as Classifier. The use of classification learning algorithms accomplishes the process of inducing classifiers from training data. At this

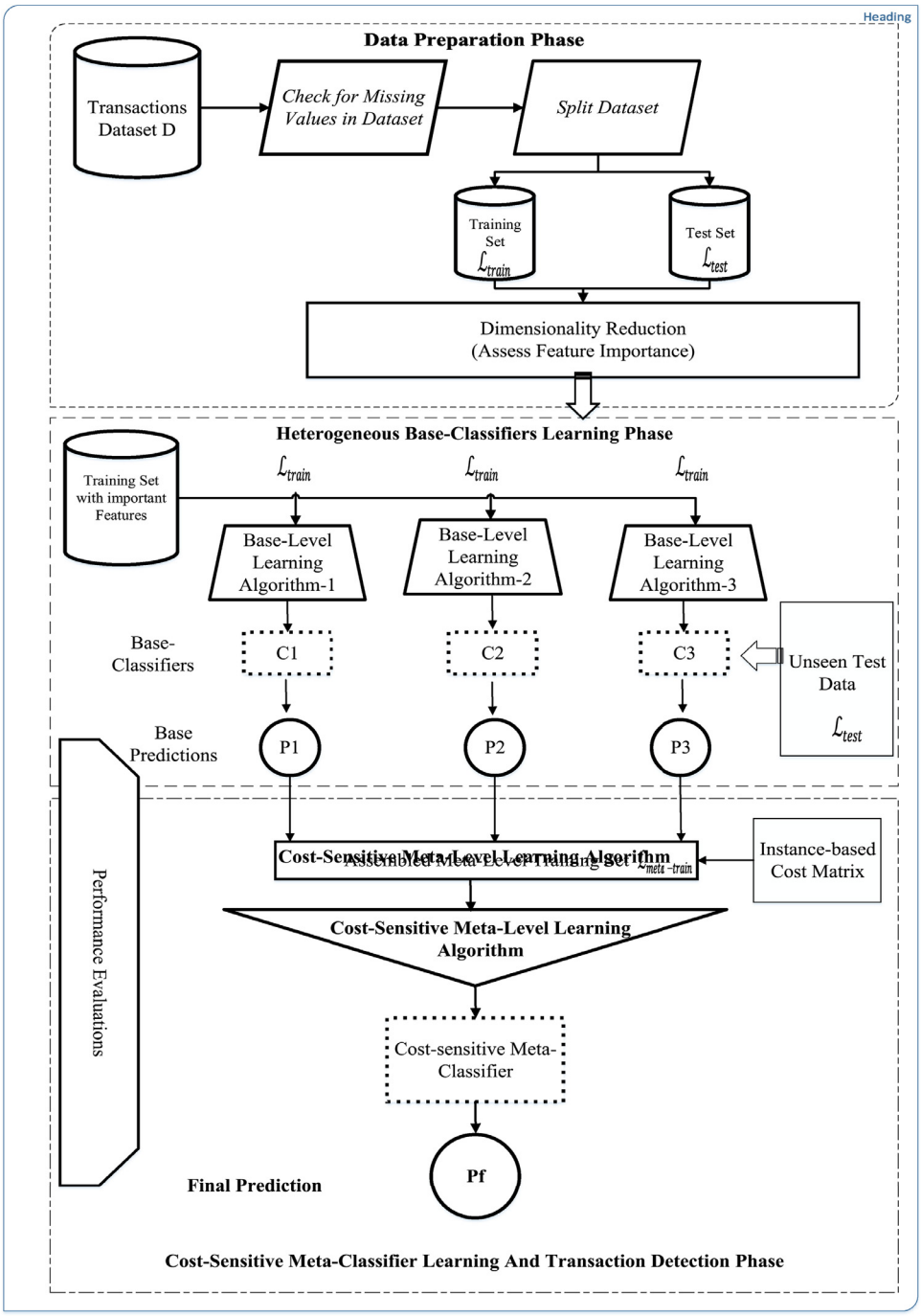

Fig. 1. Architecture of the proposed framework.

base-classifiers learning phase, three (3) different classification learning algorithms (also referred to as base-level generalizers) are used to induce diverse base-classifiers from the training set. This is to ensure that the induced base-classifiers are heterogeneous in nature. The three classification learning algorithms that are used to build heterogeneous base-classifiers from the training set in this phase are K-Nearest Neighbour, Decision Tree, and Multilayer Perceptron algorithms. However, a 10-fold cross-validation process is used at this base-level training phase, in which the training set is divided into ten folds, and during the ten iterations, 9-folds are used for training, and 1-fold used as the validation set for the model evaluation within the base-level training phase.

The three base-classifiers are used to classify unseen transaction instances in the validation set (as derived from the training set during the cross-validation process). The predictions (classifications) made on the unseen transaction instances by these base classifiers are then combined along with their respective expected target class values. Also aggregated with these predictions is the cost matrix specifying the realistic misclassification cost penalty attached to each transaction instance, which primarily includes the amount involved in the individual transactions. These are assembled together as a new meta-level training set and passed on as input to the meta-level (level-1) learning phase of the architecture in order to induce the desired meta-classifier from them.

Cost-Sensitive meta-classifier learning and transaction detection phase

In this phase of the proposed architecture, principle of the meta-learning ensemble is used to induce a meta-classifier by combining the classification outputs (predictions) of the three base-classifiers. The ensemble methodology adopted in this phase is a Stacked generalization (commonly referred to as Stacking), which is a meta-combining method used to combine the predictions from multiple lower-level classifiers and to train a meta-classifier from them by applying a meta-level algorithm on them.

At the meta-level training phase, using the Stacking construct, the classification outputs (predictions) of the base-classifiers, the expected target class values for each classified transaction instance, and the introduced instance-based cost matrix are together taken as the training set from which a cost-sensitive meta-classifier would be induced. These are now considered as the new features for each transaction instances involved in the training set while retaining the original target class as the target attribute. A meta-level learning algorithm trains meta-classifier from this meta-level training set. The meta-level learning algorithm used in this phase is the Cost-Sensitive Logistic Regression algorithm [10]. This algorithm incorporates the principle of cost-sensitive learning in the training phase of the meta-classifier by taking into account the misclassification cost of each transaction instance, as defined in an instance-based cost matrix. This gives rise to the cost-sensitive meta-classifier referred to as the Cost-Sensitive Ensemble classifier.

Finally, for the classification (detection of the class) of the new unseen transactions from the test set, the base classifiers are first used to classify the new transaction and these classifications are passed to the cost-sensitive ensemble classifier which makes the final classification of that transaction as either fraud or non-fraud.

The workflow of the proposed framework

The proposed Cost-Sensitive Meta-learning Ensemble framework for detecting credit card fraud is represented as a workflow using a Unified Modelling Language (UML) diagram. The UML diagram used to depict the idea of the Cost-Sensitive Meta-learning Ensemble approach in a workflow is the UML Statechart diagram.

Fig. 2 shows the workflow of the proposed framework in a UML Statechart diagram. This is closely related to the architecture shown in Fig. 1. The designed statechart diagram (sometimes referred to as state machine diagram) is typically a behavioral diagram that shows the transitions between the various objects in the proposed framework. Each state relates to the different combinations of information that an object can hold, not how the object behaves. The transition arrows show how each state (represented by a rounded-edge rectangle) changes in the proposed model.

Mathematical model formulation for the proposed framework

Machine learning algorithms are used in formulating the cost-sensitive ensemble detection model since the pattern explaining the relationship between known transactions input variables and the respective occurrence or not of fraud (the target variable) is required to classify an unseen transaction instance.

In machine learning, this objective is formulated as the supervised learning task of inducing from a given dataset, a predictive model that predicts the value of a target variable based on the observed values of some input variables. As such, the task of deriving an appropriate model is based on the assumption that the target variable does not take its value at random and that a relationship exists between the input variables and the target variable.

For any supervised learning technique proposed for the formulation of a predictive model, a mapping function can be used to express the general expression for the formulation of the predictive model. In credit card fraud detection problem, the goal is to derive a classification pattern (that is, a model, referred to as classifier in this context) from a set of historical credit card transaction cases (that is, dataset) for determining the legitimacy (that is, non-fraud or fraud) of an incoming transaction (that is, the target value) given a set of feature values.

To give a precise formulation, let X in (1) be defined as an input space that represents the set of transaction instances carried out using credit cards in a payment system and Y in (2) be defined as the output space that denotes the set of corresponding class labels (targets) of these transaction instances.

Mathematically defined thus:

$$X = \{x_1,\ x_2,\ \ldots,\ x_N\} \tag{1}$$

$$Y = \{y_1,\ y_2,\ \ldots,\ y_N\} \tag{2}$$

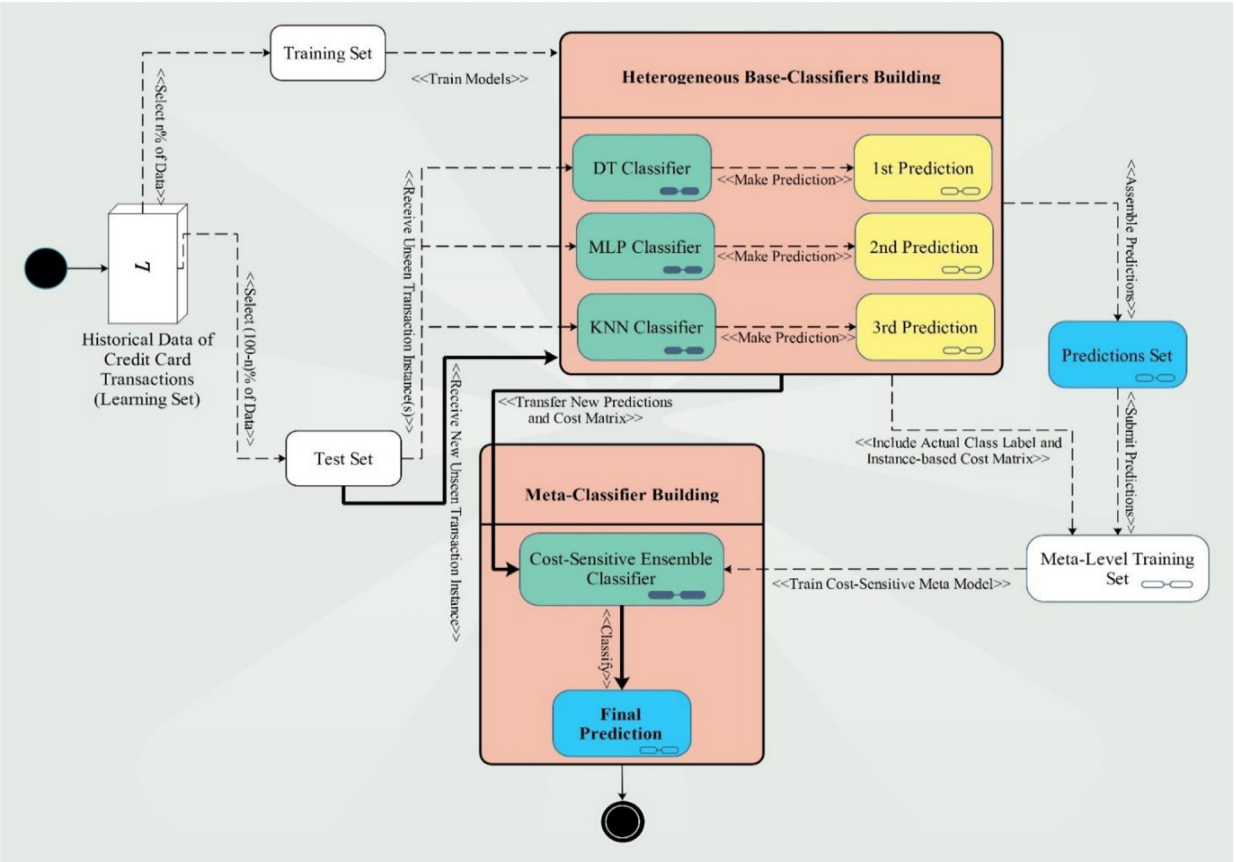

Fig. 2. Workflow of the proposed framework using UML Statechart Diagram {with three (3) specified base classifiers}.

Where, $x_n = (x_{n1}, x_{n2}, \ldots, x_{nm}) \; \forall \, n(1, 2, \ldots, N)$ is the input vector representing the m-dimensional feature values of the nth transaction instance in the input set, and;

$y_n \in \{0, 1\} \; \forall \, n(1, 2, \ldots, N)$ is the target or class label (classification value) for the nth transaction instance, which can take any of the two possible values 0 (implying non-fraud transaction) or 1 (implying fraudulent transaction).

The dataset $\mathcal{L}$ (that is, the historical credit card transaction data) is shown in (3) as the set of N pairs of the transaction instances (input vectors) and class label;

$$\mathcal{L} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \tag{3}$$

where $x_n \in X$ and $y_n \in Y$

In this framework then, the supervised learning task can be stated as inducing from the dataset $\mathcal{L}$, a mapping function h, that describes the relationship between the input vectors and the target values as depicted in (4).

$$h : X \rightarrow Y \tag{4}$$

The objective is to learn the predictive classifier (or model) h, such that given a new unseen transaction, its corresponding target value (that is, its class label) can be predicted or determined as thus:

$$h(x) = \hat{y} = \begin{cases} 1 \;\; or \;\; fraud \\ 0 \;\; or \;\; non\_fraud \end{cases} \tag{5}$$

In order to learn the desired classifier in this study, the dataset $\mathcal{L}$ is randomly split into two parts; the training set, denoted $\mathcal{L}_{train}$ and the test set, denoted $\mathcal{L}_{test}$ such that;

$$\mathcal{L}_{train} = p\% \; of \; \mathcal{L} \tag{6}$$

$$\mathcal{L}_{test} = (100 - p)\% \; of \mathcal{L} \tag{7}$$

(In the experiment carried out in this study, p in Eqs. (6) and (7) is set to 67, that is, the training set is allotted 67% of the transaction instances in the dataset).

Learning the base classifiers (Level-0 learning)

The process of learning classifiers from the training set $\mathcal{L}_{train}$ in the proposed *CostEns* model involves two-level learning. Here, the principle of ensemble learning is employed whereby a high-level model (level-1 classifier) is trained from the prediction outputs of some lower-level models (level-0 classifiers) to achieve greater predictive accuracy than can be produced by the individual models separately. Stacked generalization (Stacking) is the ensemble method to be employed in combining the level-0 models.

Given a predetermined set of learning algorithms, which in this context is referred to as the level-0 generalizer, invoke the ath learning algorithm in that set on the training dataset $\mathcal{L}_{train}$ to induce a classifier: $h_{a_{\mathcal{L}_{train}}}$ for $a = \{DT, MLP, KNN\}$; that is, each $h_{a_{\mathcal{L}_{train}}}$ classifier is built from the $\mathcal{L}_{train}$ training set.

These classifiers are referred to as the base-classifiers (the level-0 models). In this study, the three learning algorithms used as the level-0 generalizers are Decision Tree (DT) based on Classification and Regression Tree CART algorithm, Multilayer Perceptron (MLP) algorithm, and k-Nearest Neighbors (KNN) algorithm; which are invoked on the training set $\mathcal{L}_{train}$ to learn three heterogeneous base classifiers.

Base-classifier 1

The Decision Tree (DT) classifier is built using the Classification and Regression Tree (CART) algorithm on the training set $\mathcal{L}_{train}$. The decision tree classifier for the training set as derived using the Classification and Regression Tree (CART) algorithm uses a Gini index as the cost function to evaluate splits in the dataset. It is computed by subtracting the sum of the squared probabilities of each of the binary classes (0 and 1) from one. The mathematical expression for the Gini index cost function is as thus:

$$h_{DT_{\mathcal{L}_{train}}}(x) = Gini = \sum_{i=0}^{i=1} (P_i)^2 \tag{8}$$

The Gini index used as metric/cost function in the Classification and Regression algorithm evaluates split in feature selection in for classification tree. The derived tree is used to classify a new observation into the appropriate class.

Base-classifier 2

The second base-classifier is obtained by invoking the Multilayer Perceptron algorithm on the training set $\mathcal{L}_{train}$. The Multilayer Perceptron (MLP) algorithm is an Artificial Neural Network that has a topology which is characterized by grouping neurons of the same category in substructures of input, hidden, and output layers [15]. MLP is a universal approximation of functions with logistic function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{9}$$

The MLP learns if it can establish that the error found in its output is minimal. Therefore, it minimizes Error function E [15]. The function E is usually the mean squared error between the recent output and the desired output as thus:

$$E = \frac{1}{2} \sum_{i=1}^{k} (S_i - Y_i)^2 \tag{10}$$

where, s_i are the test values, the y_i are the neuron outputs of the output layer k.

The backpropagation technique is typically used to compute the descending gradient. In this technique, the sigmoid function given in (9) is adopted as an activation function such that:

$$h_{MLP_{\mathcal{L}_{train}}}(x) = y_k = \sigma\left(\sum_{i=1}^{L} b_i^L w_{ik}^L\right) \tag{11}$$

where, L is the total number of hidden layers of the MLP, b_i^L denotes the output of the ith neuron of the L-th hidden layer, and w_{ik} represents the weight assigned to the neuron i of layer k [15].

Base-classifier 3

The third base-classifier is obtained by invoking the k-Nearest Neighbor (KNN) algorithm on the training set $\mathcal{L}_{train}$. The principle of the k-nearest neighbor learning approach is in finding k transaction instances from the training set $\mathcal{L}_{train}$ that are closest to the new transaction instance to be classified. KNN uses a similarity measure *dist* (most commonly, the Euclidean distance function) to determine the distance between two transaction instances; say a new transaction instance x in the test set $\mathcal{L}_{test}$ and each of the transaction instance x_n in the training set $\mathcal{L}_{train}$. This Euclidean distance measure [16] is as represented in the following formula in (12):

$$dist(x, x_n) \equiv \sqrt{\sum_{r=1}^{m} (x_r - x_{nr})^2} \tag{12}$$

Table 1
Classification cost matrix (C_n) (Source: Bahnsen et al., 2015).

	Actual Positives $y_n = 1$	Actual Negatives $y_n = 0$
Classified Positives $\hat{y}_n = 1$	$C_{TP_n} = C_a = 100$	$C_{FP_n} = C_a = 100$
Classified Negatives $\hat{y}_n = 0$	$C_{FN_n} = Amount_n$	$C_{TN_n} = 0$

where x_r denotes the value of the r-th feature of the new transaction x, and x_{nr} denotes the r-th feature of a transaction x_n in the training set.

Then the Euclidean distances between the new transaction instance considered for classification, and all the transaction instances in the training set are gathered. These gathered distances are sorted from the smallest to the largest distance, and k transaction instances that are nearest to the new transaction instance are selected.

Let the k selected nearest transactions to the new transaction instance x be denoted by the set $\{(x_s, y_s),\ s = 1, \ldots, k\}$, then the classification (target class value) for the new transaction instance (x) is computed as the majority target class value among the k-nearest transactions as depicted in (13) thus:

$$h_{KNN_{\mathcal{L}_{train}}}(x) = \text{argmax} \sum_{s=1}^{k} y_s \tag{13}$$

The classifications (predictions) made on the unseen new transaction instances by all theses base-classifiers at the end of the level-0 learning, along with the actual (expected) target classes of these classified transaction instances are assembled into a new of training set, $\mathcal{L}_{meta-train}$; from which a meta-classifier would be learned by invoking a meta-level learning algorithm on it.

In the context of cost-sensitive learning, misclassification costs vary across transaction instances; a framework referred to as instance-based cost-sensitive classification. To aid cost-sensitivity in the learning process of the meta-level learning algorithm used as the level-1 generalizer, and so learn an example dependent cost-sensitive classifier, a cost measure defined in Bahnsen et al. [12] was adopted. This measure takes into account the different actual costs $C_n = [C_{TP_n},\ C_{FP_n}, C_{FN_n}, C_{TN_n}]$ of each example x_n. In order to take these varying costs that each transaction instance carries into account, a cost matrix per transaction instance is considered as given in the Table 1.

The cost matrix table defines the cost of True Negatives as zero (in which an actual negative transaction instance is correctly classified as negative). The cost of False Positive (in which an actual negative transaction instance is incorrectly classified as positive) is defined as the administrative cost C_a, related to analyzing the transaction instance and contacting the cardholder. The same cost is assigned to the True Positive (in which an actual positive transaction instance is correctly classified as positive), because in most cases, the cardholder may be contacted. In the case of False Negative (in which fraud is missed or undetected; that is, positive (fraud) transaction instance is incorrectly classified as negative (non-fraud)), the cost is defined as the amount, $Amount_n$, that would constitute loss in that particular transaction.

Having determined the cost matrix per transaction instance C_n, the new augmented training set, $\mathcal{L}_{meta-train}$, obtained by assembling the classifications ($h_{DT_{\mathcal{L}_{train}}}(x),\ h_{MLP_{\mathcal{L}_{train}}}(x),\ \ and\ \ h_{KNN_{\mathcal{L}_{train}}}(x)$) made on the unseen new transaction instances by each of the base-classifiers, along with the actual (expected) target classes of these classified transaction instances and the cost matrix per transaction instance C_n is $(h_{a_{\mathcal{L}_{train}}}, C_N, y_N)$ and is thus given as:

$$\mathcal{L}_{meta-train} = \left\{ \begin{matrix} \left(h_{DT_{\mathcal{L}_{train}}}(x_1),\ h_{MLP_{\mathcal{L}_{train}}}(x_1),\ h_{KNN_{\mathcal{L}_{train}}}(x_1),\ C_1, y_1\right), \ldots, \\ \left(h_{DT_{\mathcal{L}_{train}}}(x_N),\ h_{MLP_{\mathcal{L}_{train}}}(x_N),\ h_{KNN_{\mathcal{L}_{train}}}(x_N), C_N, y_N\right) \end{matrix} \right\} \tag{14}$$

Learning the cost-sensitive meta-classifier (Level-1 learning)

A cost-sensitive meta-classifier is built from the augmented training set $\mathcal{L}_{meta-train}$ by invoking a cost-sensitive Logistic Regression algorithm on the training set. The essence is to incorporate cost-sensitive learning into the ensemble learning to fit the cost-sensitive meta-classifier while allowing the individual base-classifiers to fit traditionally without having to introduce cost-sensitive learning to the base-classifiers in the first place. The logistic regression algorithm is in its standard form does not take into consideration the varying misclassification costs (that is, it is cost-insensitive), as will be shown. However, to achieve cost-sensitivity in its learning procedure, misclassification costs are directly introduced into the process of learning the weights of the logistic cost function.

Logistic regression is a classification model that performs well on linearly separable classes (a linear model for binary classification) [13]. It can be used to express a linear relationship between feature values of the training set $\mathcal{L}_{meta-train}$ and

a log-odd as a logit function:

$$logit\,(P(y_n = 1|x_n)) = w_1x_{n1} + w_2x_{n2} + w_mx_{nm} = \sum_{n=1}^{N} w_mx_{nm} = w^Tx_n \tag{15}$$

where $P(y_n = 1|x_n)$ is the conditional probability that a transaction instance x_n belongs to target class 1 (that is, fraud) given its features.

In this specific context of binary classification that denotes fraud detection, the problem of interest is in predicting the probability estimation that a particular instance of transaction belongs to the target class 1 (that is, fraud) given its features and parametrized by the weights w. This is typically represented as the inverse form of the logit function called the logistic sigmoid function [13]:

$$H_{CE_{\mathcal{L}_{meta-train}}} = \phi(z) = \frac{1}{1+e^{-z}} \tag{16}$$

where z is the net input, that is, the linear combination of the weights and transaction instance features which is represented as:

$$z = w^Tx_n = w_1x_{n1} + w_2x_{n2} + \ldots + w_mx_{nm} \tag{17}$$

According to Bahnsen et al. [13], the problem then becomes that of learning the right weight w parameters that minimizes a given cost function. The logistic regression cost function $J(w)$ is usually taken as the negative logarithm of likelihood such that:

$$J(w) = \frac{1}{N}\sum_{n=1}^{N} J_n(w) \tag{18}$$

where, $J_n(w) = -y_n \log(\phi(z)) - (1-y_n)\log(1-\phi(z))$

Based on the work of Bahnsen et al. [13], this cost function $J(w)$ actually assigns the same weight to different misclassification errors, the false positive and the false negative, which in fact is not the case in the real world credit card fraud detection problem. In particular,

$$J_n(w) \approx \begin{cases} 0\ if\ y_n \approx \phi(z) \\ \infty\ if\ y_n \approx (1-\phi(z)) \end{cases} \tag{19}$$

which, according to [12], in the context of cost-sensitive learning, means that: $C_{TP_n} = C_{TN_n} \approx 0$ and $C_{FP_n} = C_{FN_n} \approx \infty$.

In order to infuse the varying costs stated in the Table 1 into the process of learning the weights of the logistic cost function, the expected costs of classifying each transaction instance into a target class are first analyzed thus [13]:

$$J_n^c(w) = \begin{cases} C_{TP_n}\ if\ y_n = 1\ and\ \phi(z)\ \approx 1 \\ C_{TN_n}\ if\ y_n = 0\ and\ \phi(z)\ \approx 0 \\ C_{FP_n}\ if\ y_n = 0\ and\ \phi(z)\ \approx 1 \\ C_{FN_n}\ if\ y_n = 1\ and\ \phi(z)\ \approx 0 \end{cases} \tag{20}$$

These different costs are then being merged into a cost function which is dependent on the new costs [13]:

$$J^c(w) = \frac{1}{N}\sum_{n=1}^{N} (y_n(\phi(z)C_{TP_n} + (1-\phi(z))C_{FN_n}) + (1-y_n)(\phi(z)C_{FP_n} + (1-\phi(z))C_{TN_n})) \tag{21}$$

The cost function $J^c(w)$ in (21) is applied to adjust the weights w in (17) in a backward propagation mechanism using gradient descent as given thus [13]:

$$w = w - \phi(z)\frac{\partial J^c(w)}{\partial w} \tag{22}$$

Having adjusted the weights w using Eq. (22), the prediction probability of a new transaction instance obtained by using the sigmoid function in Eq. (16) can then be simply converted into a binary outcome through a quantizer thus:

$$H_{CE_{\mathcal{L}_{meta-train}}}(x) = \begin{cases} 1\ if\ \phi(z) \geq 0.5 \\ 0\ if\ \phi(z)\ otherwise \end{cases} \tag{23}$$

Dataset description

The dataset used for experiments in this research was retrieved from the Kaggle website [13]. Kaggle is a leading platform and online community for Data Mining and Machine Learning competitions. The dataset was reported to have been collected and analyzed during a research collaboration of Worldline and the Machine Learning Group of Université Libre de Bruxelles on big data mining and fraud detection [13]. The dataset contains transactions that were made using credit cards in September 2013 by some European credit cardholders [14]. The transactions were said to have occurred in two days,

Table 2
Composition of the Six Dataset Fragments.

	Dataset DF_10	Dataset DF_5	Dataset DF_2	Dataset DF_1	Dataset DF_0.75	Dataset DF_0.37
Fraud Transactions	492	492	492	492	492	492
Legitimate Transactions	4428	9348	24,108	48,708	64,374	130,889
Total Transactions	4920	9840	24,600	49,200	64,866	131,381
Fraud Rate	10%	5%	2%	1%	0.75%	0.37%

containing 492 fraudulent transactions out of 284,807 total transactions. The dataset is a highly unbalanced one, with the positive class (that is, frauds represented by integer "1") accounting for minority 0.172% of the entire transactions.

The dataset, as obtained from the Kaggle database, contains only numerical input variables, which are the result of a Principal Components Analysis (PCA) transformation of the original data. Due to confidentiality issues and the sensitive nature of the data, the original features and more background information about the data were not provided. The only two features that were not transformed with Principal Component Analysis are 'Time' and 'Amount'. The feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' refers to the transaction Amount, and this feature can be used for example-dependent cost-sensitive learning. The feature 'Class' is the target attribute, and it takes the value "1" in case of a fraudulent transaction and the value "0" otherwise. The features V1, V2, ... V28 are the principal components obtained by using Principal Components Analysis. There are thirty-one (31) features in the dataset. The dataset, however, does not contain any missing values whatsoever.

Experimental setup

Using the rich packages of the python programming language for handling and manipulating data, the dataset was pre-processed and fed into machine learning algorithms for predictive learning. However, the original dataset was partitioned into six (6) dataset fragments having different proportions of fraud cases in them. The motivation behind creating these six (6) datasets is to observe and compare the performance of the framework's classifiers on different proportions of fraud. Each fragmented dataset was split into training and test sets in six separate experiments. At each experiment, base classifiers were trained from the training sets, and predictions evaluations were made on unseen data using 10-fold cross-validation. Afterward, the predictions of the learned base classifiers are combined into an ensemble using a stacked generalization strategy, and a cost-sensitive learning algorithm was used to learn for training at this stage. The predictive performance of the resulting cost-sensitive classifier from the ensemble was evaluated on the test set. This operation was performed on each of the six dataset fragments, and results of performance evaluations were obtained.

The dataset was partitioned into six different fragments sampled at different rates to obtain different training data with different reasonable proportions of fraudulent transactions to non-fraudulent cases, though still maintaining the class imbalance. The transaction instances were sampled such that every fraudulent transaction instance is retained in each partition without replication of the non-fraudulent transaction instances in each partition to obtain the dataset fragments with varying proportions of fraud cases. This resulted in six different dataset fragments having 10%, 5%, 2%, 1%, 0.75% and 0.37% proportions of fraudulent transactions. These are labeled DF_10, DF_5, DF_2, DF_1, DF_0.75 and DF_0.37 respectively. The dataset was fragmented into these different fraud rates so as to examine the performance trend of the machine learning algorithms on the six dataset fragments containing different fraud rates. Table 2 shows the compositions of the dataset fragments.

Test results and performance evaluation

This section presents results from the experiments carried out by employing the proposed framework, showing the performances obtained by applying the cost-sensitive meta-learning framework's ensemble-classifier and comparing these performances with those of the performances of the base classifiers (Decision Tree, Multi-Layer Perceptron and K-Nearest Neighbors) as trained on the training dataset fragments carrying varying degrees of occurrences of fraud. Consequently, for each technique, results are obtained for the six data fragments carrying different proportions of fraud.

In the experiment, 33 percent each of the dataset fragments were set aside as test data and used to check if the proposed framework's meta-classifier generalizes well on unseen data. Since the test set is not to be used for model training or selection, its original essence is to aid the report of an unbiased estimate of the generalization performance of a classifier. The Receiver Operating Characteristic Curve standard performance evaluation metric was considered for rating the performance of the proposed framework's cost-sensitive ensemble-classifier and the individual base classifiers on unseen test data. The area under the ROC curve (AUC) performance measure is often taken as a better measure of overall performance in this domain than other traditional metrics. The AUC measures the area under the Receiver Operating Characteristics (ROC) curve, and it is independent of any specific classification cutoff values.

The base classifiers and the proposed framework's Cost-Sensitive Ensemble classifier were trained on the six dataset fragments, and the ROC evaluations obtained. The ROC curves obtained from these classifiers trained on the DF_5, DF_2, DF_1, DF_0.75, and DF_0.37 dataset fragments are as shown in Figs. 3–8 respectively.

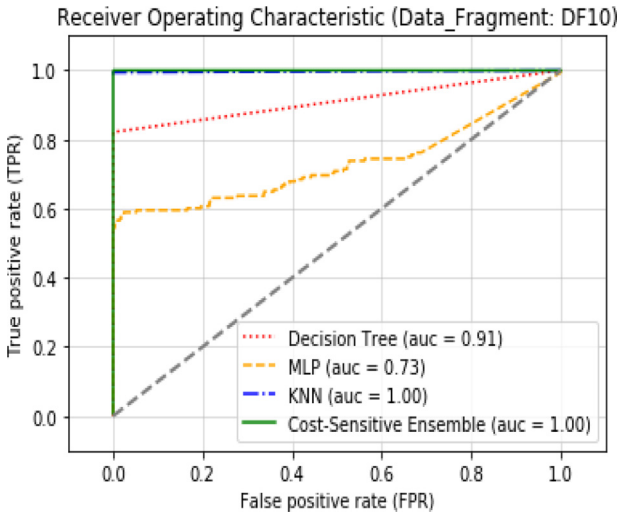

Fig. 3. Receiver Operating Characteristic Curve (using classifiers trained on the DF10 Dataset Fragment).

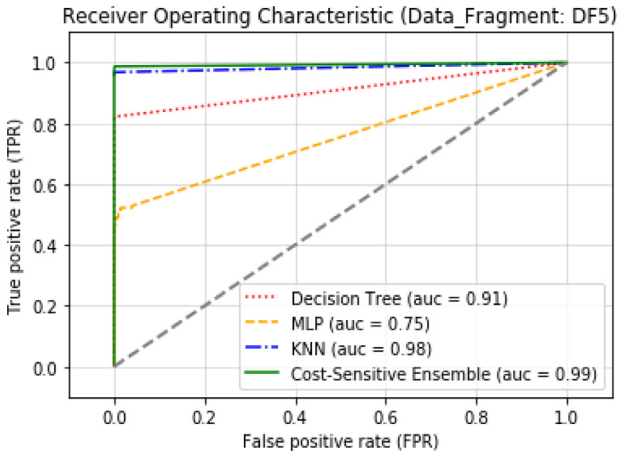

Fig. 4. Receiver Operating Characteristic Curve (using classifiers trained on the DF5 Dataset Fragment).

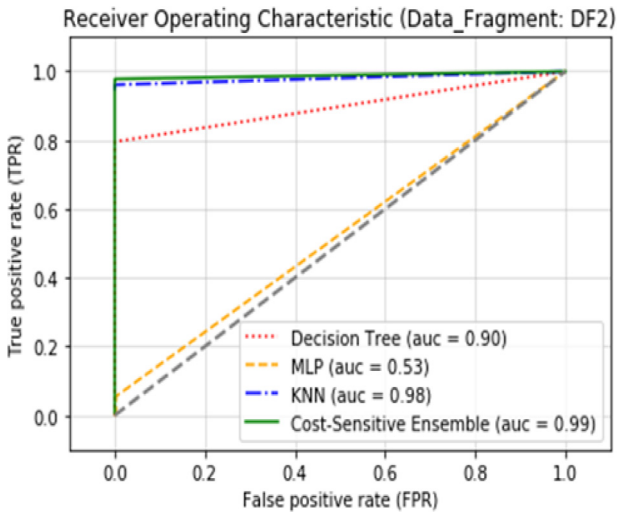

Fig. 5. Receiver Operating Characteristic Curve (using classifiers trained on the DF2 Dataset Fragment).

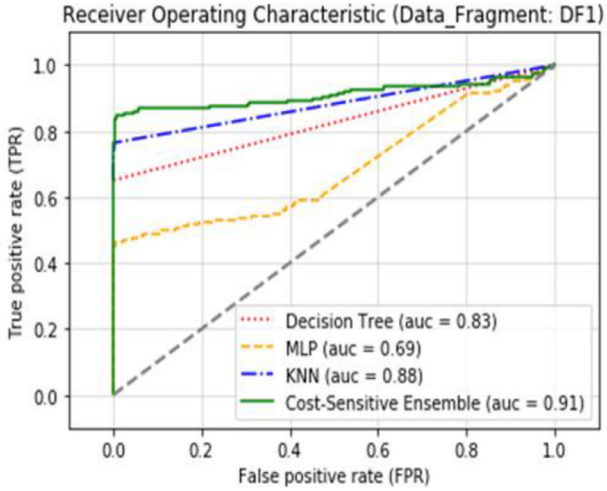

Fig. 6. Receiver Operating Characteristic Curve (using classifiers trained on the D1 Dataset Fragment).

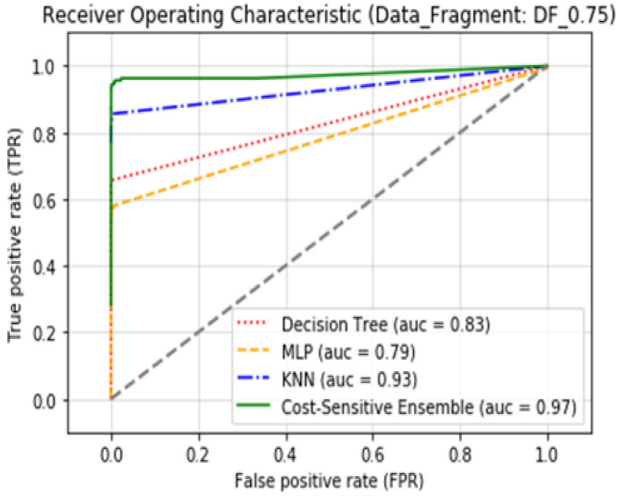

Fig. 7. Receiver Operating Characteristic Curve (using classifiers trained on the DF_0.75 Dataset Fragment).

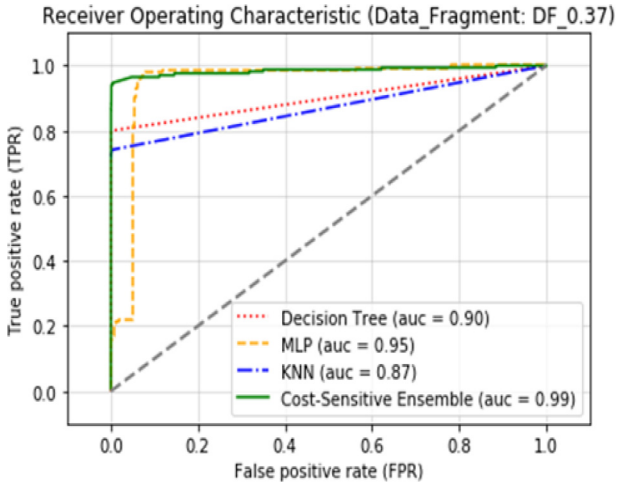

Fig. 8. Receiver Operating Characteristic Curve (using classifiers trained on the DF_0.37 Dataset Fragment).

Results discussion

The classification performances (based on ROC measures) of the Cost-Sensitive Ensemble, Decision Tree (DT), Multi-Layer Perceptron (MLP) and K-Nearest Neighbors (KNN) classifiers on the 33% test data as graphically shown in Figs. 3–8 highlight the ROC curve comparison among the classifiers across the different dataset fragments. The ROC curve plots the percentage of true positive (TP) against the false positive (FP) rate. The closer the curves are to the upper left-hand corner, the higher the model accuracy.

For each classifier trained on each of the six dataset fragments carrying different proportions of fraud, the Area Under ROC curve (AUC) values obtained upon classifying the 33% test data are shown. The green line depicts the ROC curve for the Cost-sensitive ensemble classifier; the broken blue lines show the ROC curve for the K-nearest neighbor classifier, the yellow dash lines show the ROC curve for multilayer perceptron classifier while the red dotted lines represent the ROC curve for decision tree classifier. Also, the diagonal dash lines represent random guess in the ROC curve performance rating (having an AUC value of 0.5) below which the performance of a classifier is said to be poor.

The ROC curve in Fig. 3 presents the area under the ROC curve (AUC) performance of each of the classifiers that were trained on the dataset fragment consisting of 10% fraud proportion. As seen in the ROC curve, the AUC performance of the Cost-Sensitive Ensemble and the KNN classifiers stands at the peak of 1.00, showing excellent classification accuracies. Trailing these classifiers are the DT classifier with a good AUC of 0.91, while the MLP classifier shows a lower AUC of 0.73.

The ROC curve in Fig. 4 shows the area under the ROC curve (AUC) performance of each of the classifiers that were trained on the dataset fragment that consists of a 5% fraud proportion. From the curve, the obtained AUC value of the Cost-Sensitive Ensemble classifier is 0.99, performing much better than base classifiers. However, the KNN classifier (with 0.98) compares competitively with the Cost-Sensitive Ensemble classifier, while the DT classifier with AUC of 0.91 and the MLP classifier with AUC of 0.73 perform lower than the duo of KNN and Cost-Sensitive Ensemble classifiers.

Fig. 5 shows the area under the ROC curve (AUC) performance of each of the classifiers that were trained on the dataset fragment that consists of a 2% proportion of fraudulent transaction records. It can be observed from the curve that both the Cost-Sensitive Ensemble classifier and KNN classifier maintain AUC value of 0.99 and 0.98, respectively, as obtained by the classifiers trained on the 5% fraud rate dataset fragment. In this regard, however, the DT classifier and MLP classifier suffer AUC values decline to 0.90 and 0.53, respectively.

The ROC curve in Fig. 6 shows the area under the ROC curve (AUC) performance of each of the classifiers that were trained on the dataset fragment consisting of 1% fraud proportion. From the ROC curve, the AUC performance of the Cost-Sensitive Ensemble classifier is seen to have declined to an AUC value of 0.91, though this still depicts a significantly good performance. The KNN classifiers show AUC value of 0.88 while the consistently low-performing classifiers (DT and MLP) among the four classifiers have AUC values of 0.83 and 0.69 respectively.

From Fig. 7 which shows the area under ROC curve (AUC) performance of each of the classifiers that were trained on the dataset fragment that consists of 0.75% proportion of fraudulent transaction records, it can be observed the Cost-Sensitive Ensemble classifier has a sharp rise in AUC value (0.97) as compared to its previous performance on the 1% fraud rate dataset fragment. Its competing counterpart, KNN, is also seen to rise to AUC value of 0.93 compared to its performance on the 1% fraud rate dataset fragment too. In the same manner, there is an increase in the AUC value of the low-performing MLP classifier with an AUC value of 0.79, while the DT classifier maintains an AUC value of 0.83, as seen in the previous dataset fragment.

Finally, from Fig. 8, showing the area under ROC curve (AUC) performance of each of the classifiers that were trained on the dataset fragment with the lowest fraud rate (that is, 0.75% proportion of fraudulent transaction records), it can be observed the Cost-Sensitive Ensemble classifier maintains high AUC value of 0.99 while the MLP classifier with an AUC value of 0.95 outperforms the KNN classifier which has AUC value of 0.87 and DT classifier with an AUC value of 0.90 is also seen to exceed the KNN classifier too.

Of all the classifiers considered in the framework, the Cost-Sensitive Ensemble classifiers trained from the different training data fragments carrying varying fraud rates generally show the best performance on the ROC curves obtained by classifying unseen test data using these classifiers. This indicates that the Cost-Sensitive Ensemble classifier is able to more accurately detect fraud cases irrespective of the proportion of available fraud rate in the dataset. It is closely followed by the KNN classifier and then the DT classifier, while MLP classifier performs least in most cases. However, for all the classifiers trained from the training dataset segment with the lowest fraud rate (that is, DF_0.37 with 0.37% fraud), the performance of the MLP classifier surpasses that of the KNN classifier. The Cost-Sensitive Ensemble classifier shows to be slightly but significantly higher than the KNN classifiers. It can be asserted from Figs. 3–8 that the cost-sensitive ensemble classifier produced in this framework performs better under high class-imbalance in all cases.

Existing Ensemble method by Sood (2019) [17], applied on the same credit card dataset shows that logistic regression had the highest AUC area (with 98.48 AUC value), hence proving to be better than the ensemble model (with 93.04 AUC value), as shown in Fig. 9. Also, an ensemble approach comparing post-tuned Randon Forest and XGBoost [18] shows that their AUC values stood at 0.83 and 0.84, respectively, making the XGBoost ensemble a better classifier in that comparison. An ensemble of decision trees in the random forest used on the same dataset in [19] shows an AUC value of 0.97 for the best performing model. However, as shown in the above ROC curves, the AUC values of Cost-Sensitive Ensemble model proposed in this paper outperforms both the best performing model and the pure ensemble models in [17, 18] and [19] specifically on

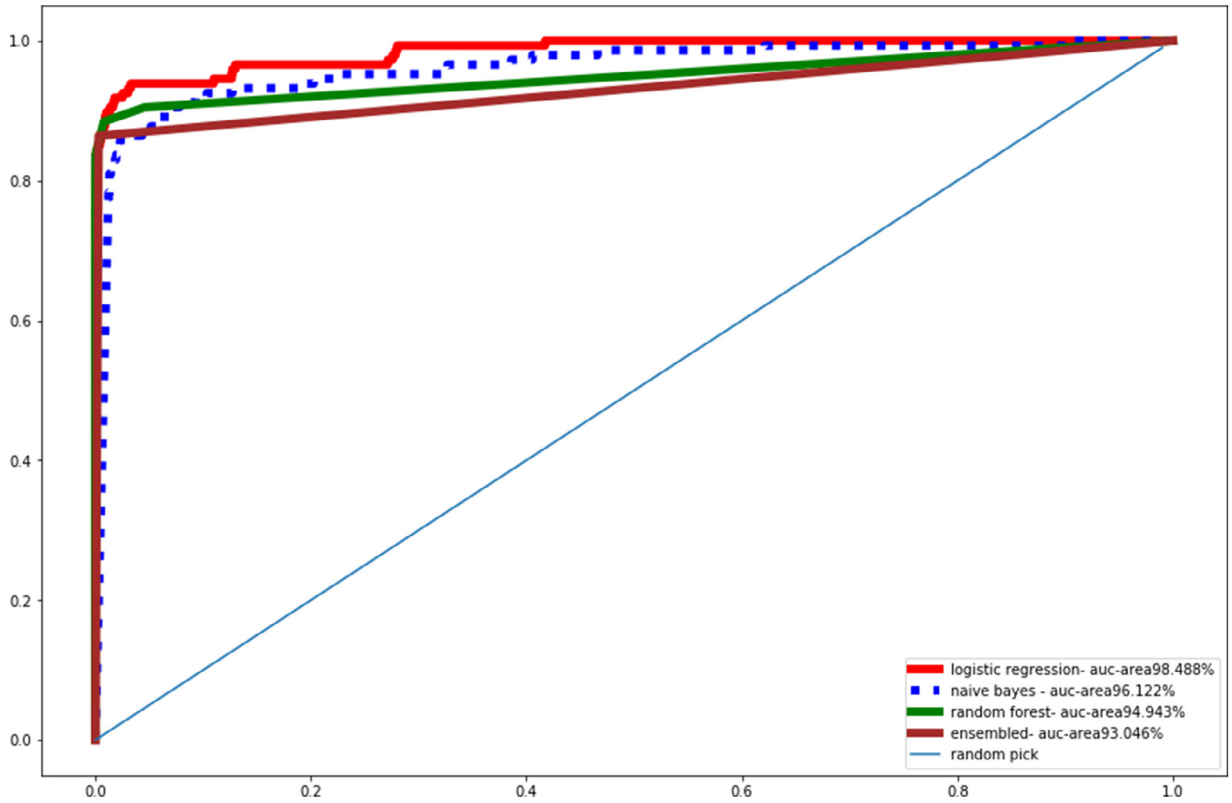

Fig. 9. Receiver Operating Characteristic Curve of a Related Approach on the dataset (Source: Sood, 2019).

low fraud rate dataset fragment. The performance of the cost-sensitive ensemble model remains steadily higher in majority of the dataset fragments showing that it performs well in different proportions of class imbalance ranging from 0.91 to 1.00 across the different fraud proportions in the dataset.

Conclusion

In this paper, a framework for detecting credit card fraud by using a cost-sensitive meta-learning ensemble approach is proposed and a resultant cost-sensitive ensemble classifier built from an ensemble of base-classifiers by using three learning algorithms (k-nearest neighbors, decision tree and multilayer perceptron) as base-level generalizers presented. In the framework, the base-classifiers were combined into a meta-classifier with a cost-sensitive logistic regression algorithm as the meta-level generalizer, thereby incorporating cost-sensitive learning only in the meta-level training phase. The performance of the base-classifiers and subsequent cost-sensitive ensemble classifiers trained on different dataset fragments based on the proposed framework was evaluated using ROC curves. Also, the results of some pure ensemble methods applied on the same dataset were compared with the results of the proposed cost-sensitive ensemble method.

The contributions of the paper include the presentation of a framework that consists of both ensemble learning paradigm and cost-sensitive learning paradigm to improve the detection of fraud in varying imbalanced credit card dataset fragments. The approach of the proposed framework is to allow base-classifiers to fit traditionally while the cost-sensitive learning is incorporated in the ensemble learning process to fit the cost-sensitive meta-classifier without having to enforce cost-sensitive learning on each of the base-classifiers. This approach differs from those already presented and used in literature. It also underscores the strength of cost-sensitive classifiers in handling highly negatively skewed data.

As it was observed from the resulting ROC curves upon predicting unseen data in the test set, the cost-sensitive ensemble classifier maintains an excellent AUC value showing consistent performance across different fraud rates in datasets. This indicates that the cost-sensitive ensemble framework is efficient in producing meta-classifiers that are capable of effectively detecting fraudulent transactions in different databases of payment systems irrespective of the proportion of available historical fraud cases data to train on. It can be concluded from the results obtained that the cost-sensitive ensemble classifier produced in this framework performs better under high class-imbalance in all cases compared with the pure ensemble methods presented in [17, 18], and [19].

Declaration of Competing Interest

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

References

[1] S. Panigrahi, A. Kundu, S. Sural, A.K. Majumdar, Credit card fraud detection: a fusion approach using Dempster–Shafer theory and Bayesian learning, Inf. Fusion 10 (2009) 354–363.
[2] J. Dara and L. Gundemoni, "Credit card security and e-payment: enquiry into credit card fraud in e-payment," 2006.
[3] W. Roberds, The impact of fraud on new methods of retail payment, Fed. Reserve Bank Atlanta Econ. Rev. (1998).
[4] "Statistics for general and online card fraud," 20 June 2007. [Online]. Available: http://epaynews.com/statistics/fraud.html.
[5] , Online Fraud is 12 Times Higher Than Offline Fraud, sellitontheweb, 2016 [Online]. Available: http://sellitontheweb.com/ezine/news0434.shtml [Accessed 29 January.
[6] R.J. Bolton, D.J. Hand, Statistical fraud detection: a review, Stat. Sci. 28 (3) (2002) 235–255.
[7] J.K. Pun, "Improving Credit Card Fraud Detection using a Meta-Learning Strategy," 2011.
[8] Y. Sahin, S. Bulkan, E. Duman, A cost-sensitive decision tree approach for fraud detection, Expert Syst. Appl. 40 (2013) 5916–5923.
[9] A.C. Bahnsen, S. Villegas, D. Aouaday, B. Ottersten, Fraud detection by stacking cost-sensitive decision trees, Data Sci. Cyber-Secur. (2017) 1–15.
[10] A.C. Bahnsen, D. Aouada, B. Ottersten, Example-dependent cost-sensitive logistic regression for credit scoring, 13th International Conference on Machine Learning and Applications, 2014.
[11] A.C. Bahnsen, A. Stojanovic, D. Aouada, B. Ottersten, Cost sensitive credit card fraud detection using bayes minimum risk, 12th International Conference on Machine Learning and Applications, 2013.
[12] A.C. Bahnsen, D. Aouada and B. Ottersten, "arXiv: Cornell University," 18 May 2015. [Online]. Available: https://arxiv.org/abs/1505.04637. [Accessed 29 January 2018].
[13] "Credit Card Fraud," 2016. [Online]. Available: http://www.kaggle.com/mlg-ulb/creditcardfraud. [Accessed 15 December 2016].
[14] A.D. Pozzolo, G. Boracchi, O. Caelen, C. Alippi, G. Bontempi, Credit card fraud detection and concept-drift adaptation with delayed supervised information, 2015 IEEE International Joint Conference on Neural Networks (IJCNN), 2015.
[15] D. López, N. Vera, L. Pedraza, Analysis of multilayer neural network modeling and long short-term memory, Int. J. Math. Comput. Sci. 10 (12) (2016) 697–702.
[16] N. Dutta, U. Subramaniam, S. Padmanaban, Mathematical models of classification algorithm of machine learning, International Meeting on Advanced Technologies in Energy and Electrical Engineering, 2019.
[17] A. Sood, "Credit Card Ensemble Detectors," 2019. [Online]. Available: https://www.kaggle.com/manisood001/credit-card-ensemble-detectors. [Accessed 20 April 2020].
[18] "Using Ensemble Methods and SMOTE Samping," 2017. [Online]. Available: https://www.kaggle.com/ganakar/using-ensemble-methods-and-smote-sampling. [Accessed 20 April 2020].
[19] K.L. Kurien, A. Chikkamannur, Detection and prevention of credit card fraud transactions, Int. J. Eng. Sci. Res. 8 (3) (2019) 1–11.