International Journal of Smart Business and Technology
Vol.9, No.2 (2021), pp.33-48
http://dx.doi.org/10.21742/ijsbt.2021.9.2.03

Ensemble Techniques for Credit Card Fraud Detection

Satya Dileep Penmetsa[1] and Sabah Mohammed[2*]

[1,2]Computer Science Department, Lakehead University, Canada*
[1]penmetsas@lakeheadu.ca, [2]sabah.mohammed@lakeheadu.ca*

Abstract

Credit card fraud is a problem that has grown by great danger and has a huge impact on the financial sector. The challenges of credit card fraud are the availability of public data, high imbalance in data, and volatility of the fraud nature. Over the years ensemble learning has gained more importance and proved to give better performance. Here we try to do a comparative study of various ensemble approaches using various learning algorithms on the credit card fraud data and to understand multiple models based on various evaluation and performance metrics using the SMOTE balancing technique.

Keywords: *Credit card fraud, Machine learning, Ensemble learning, Oversampling, SMOTE*

1. Introduction

Over the years, due to the rise of e-commerce, the use of debit cards for purchases has increased drastically. The present unprecedented just added more and has increases the use of credit cards manifold. Credit card payments have become one of the popular methods of purchase and have revolutionized the way of payments. Financial institutions issue these credit cards for the customers to use, the customers can use these cards for purchases and doing credit transactions by using the details imprinted upon the cards.

When the information of your credit card is used to make purchases without your knowledge that implies there is a fraudulent transaction which can adversely cost you money and can also affect your credit score. Some individuals try to exploit this information making identity thefts and cause of huge losses to credit card owners by doing fraudulent transactions and this problem is something to be addressed with the increased e-commerce and the ease of making payments and transactions, it is now important than ever before for establish proper

fraud detection and fraud prevention techniques. While fraud prevention works by setting some thresholds and security methods to prevent fraud, fraud detection is totally on understanding the patterns of fraud transactions by Machine Learning and Deep Learning techniques to check for the possibility of a fraudulent transaction. As the methods and strategies used by the fraudsters change constantly there is a high and constant need for ML techniques in the sector. And over the years these machine learning techniques have been widely used in fraud detection and achieved favorable performances.

Article history:
Received (April 14, 2021), Review Result (May 16, 2021), Accepted (August 30, 2021)

Print ISSN: 2288-8969, eISSN: 2207-516X IJSBT

2. Essential concepts

2.1. Simple ensemble techniques

(1) Max Voting.

A technique used for classification problems. In these multiple models are used to make predictions for each data point. These predictions are then voted by each model, the majority vote of the models is considered the final prediction.

(2) Averaging.

Similar to the above method multiple predictions are made based on a single data point. The average of all the predictions from the models is considered the final prediction.

(3) Weighted Averaging

This acts as an extension to the above method but differs in assigning weights or importance to models for making the predictions and then getting the final average.

2.2. Advanced ensemble techniques

(1) Stacking. In this technique, a combination of multiple machine learning models takes place on the same dataset. Stacking uses heterogeneous weak learners for predictions.

The architecture usually involves two models:

(a) Base-Model

(b) Meta-Model

A pool of base models is trained using the training data and the predictions are taken. The outputs of these models are used to train a different meta-model to give the final prediction. K-fold cross-validation technique may be used to train the base model.

(2) Bagging. Also called Bootstrapping is a sampling technique where we create bags of subsets from a dataset. A base learner is assigned to each of these subsets. These models run independently and parallel to each other. The final predictions are then made by combining the predictions.

(3) Boosting. Boosting refers to a family of algorithms that converts weak learners to strong learners. Boosting is a method for improving the model predictions of any given learning algorithm. It trains weak learners sequentially each time correcting its predecessor.

3. Related research studies

Credit Card Fraud detection has proved to be a challenging problem because of mainly two reasons which it poses - both the profiles of fraudulent and normal behaviors change and data sets used are highly skewed, the data is so much imbalanced that the majority class is almost 99% or more which makes the problem more critical where the traditional machine learning methods become less significant to use. The performance of fraud detection is affected by the variables used and the technique used to detect fraud.

Bagga et al. [1] in their research compares the performance of several traditional ensemble machine learning methods such as logistic regression, K-nearest neighbors, random forest, naive Bayes, multilayer perceptron, Adaboost, quadrant discriminative analysis, pipelining, and ensemble learning on the credit card fraud data. They successfully investigated and proposed their findings on the performance of Logistic Regression, Naïve Bayes, K nearest

neighbors, Multilayer Perceptron, Ada Boost, Quadrant Discriminant Analysis, Random Forests, Pipelining and Ensemble Learning in determining fraudulent credit card transactions. They used different classifier models which were trained on the real-life dataset and their performances were evaluated based on various parameters and metrics. Since the datasets were highly imbalanced, they used the ADASYN method to make the dataset balanced. Finally, they evaluated the performance of the classifiers using precision, accuracy, recall, F1 score, Matthews correlation coefficient, and Balanced Classification Rate.

Sahony et al. [2] in their research presented an ensemble machine learning approach as a possible solution to Credit Card Fraud Detection. They observed that Random Forest was more accurate in detecting normal instances, and Neural Network was for detecting fraud instances and finally presented an ensemble method - based on a combination of random forest and neural network - which was able to predict with high accuracy and confidence. Zareapoor et al. [3] in their research examined three state-of-the-art ensemble techniques for detecting the frauds in credit cards transactions using a bagging classifier based on the decision tree algorithm. They found that Bagging ensembles perform the best as compared to the traditional machine learning algorithms in detecting frauds. They used two metrics for the evaluation of the ensemble methods which were Balanced Classification Rate (BCR) and Matthew's Correlation Coefficient (MCC) [3][19].

Barahim et al. [4] in their study examined the three widely used machine learning techniques using Weka: Decision Tree, SVM, and Naïve Bayes each technique was applied individually in the dataset and was enhanced with boosting ensemble technique. The credit card fraud detection model was constructed with Boosting and Decision Tree resulted with the highest accuracy of 98.37% and F-Measure of 94.49% while Decision Tree only resulted with the accuracy of 98.27%and F-Measure of 93.98%. The results can show the impact of applying Boosting ensemble technique and how it enhances the performance of the machine learning techniques [4].

Randhawa [9] in their study on credit card fraud detection using machine learning algorithms presented several standard models which include NB, SVM, and DL. They used a publicly available credit card data set has been used for evaluation using individual (standard) models and hybrid models using AdaBoost and majority voting combination methods. The MCC metric was adopted as a performance measure, as it takes into account the true and false positive and negative predicted outcomes. The best MCC score is 0.823, achieved using majority voting. A perfect MCC score of 1 was achieved using AdaBoost and majority voting methods. To further evaluate the hybrid models, noise from 10% to 30% has been added into the data samples. The majority voting method yielded the best MCC score of 0.942 for 30% noise added to the data set. This shows that the majority voting method offers robust performance in the presence of noise. The use of ensemble techniques is very significant in the prediction of faulty credit card transactions from normal credit card transactions.

4. Application design and component diagram

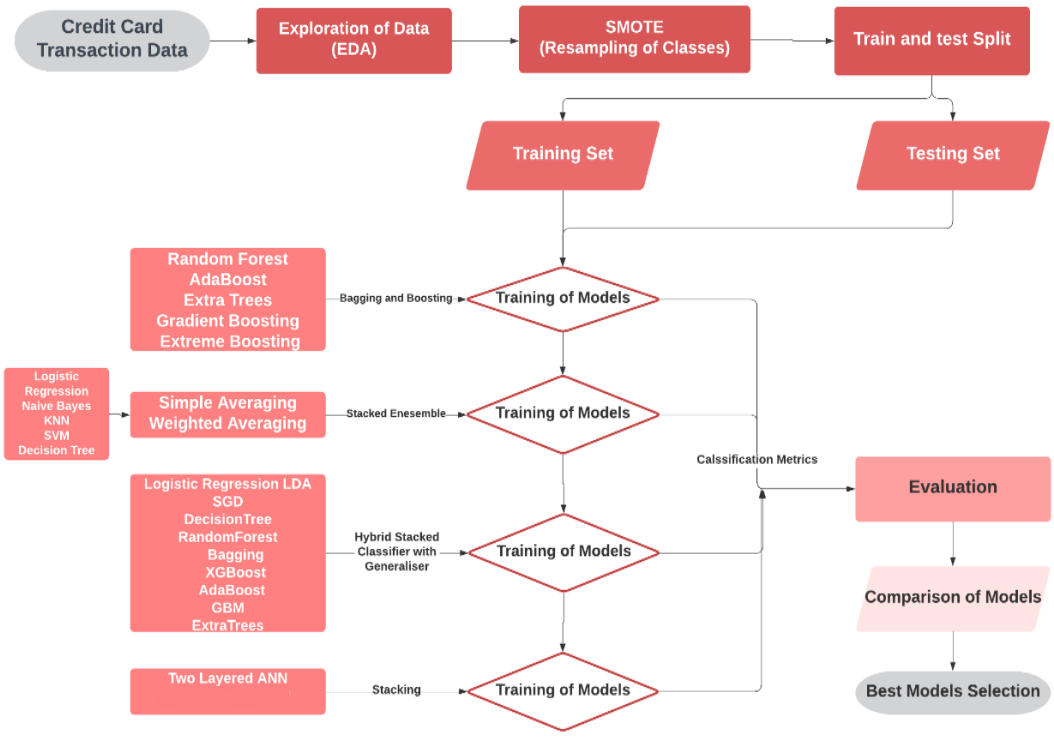

Figure 1. Application design

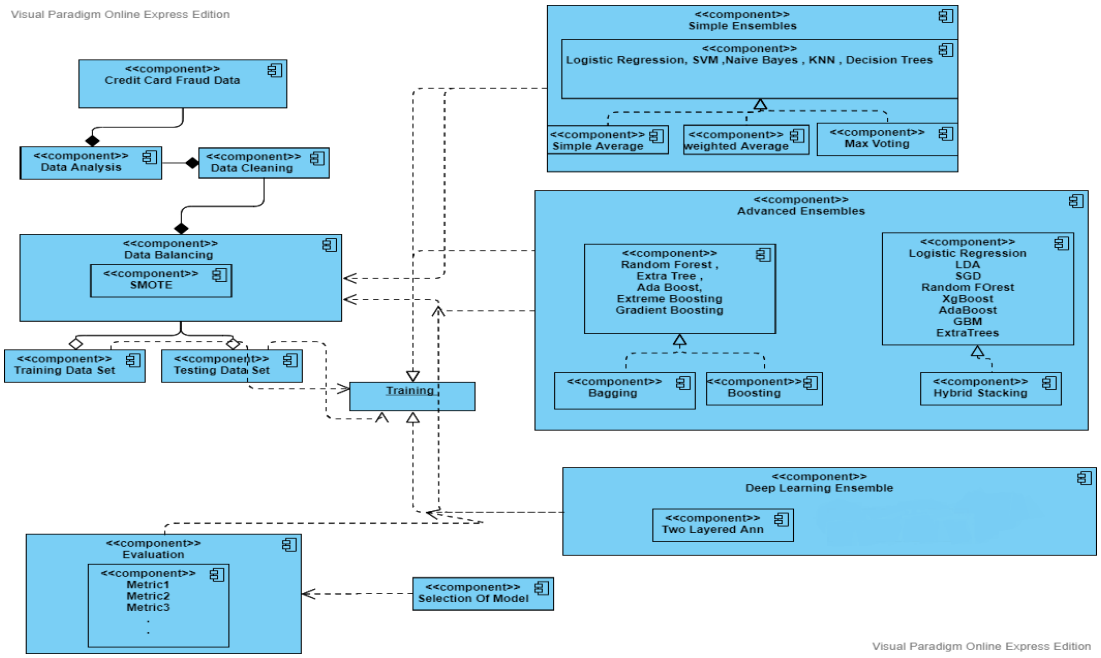

Figure 2. Component diagram

5. Algorithms

Simple ensembles:

Algorithm 1

```
procedure SimpleEnmsebles
    Algorithms = [LogisticRegression, DecisionTreeClassification, ...]
    for Each Algorithm do
        Train Data
        Test Data
        Get Predictions
procedure SimpleEnsembles With Simple Averaging
    for Each Prediction do
        Mean(AllPredictions)
        Get Mean Result
procedure SimpleEnsembles With Weighted Averaging
    weights=[]
    Predictions=[]
    for Each Prediction do
        Mean(Prediction)*Weight
        Get Mean Result
```

[1]

Advance ensembles:

Algorithm 2 AdvancedEnsemble:Bagging

```
procedure BootStrapSampling(OriginalDataset,M)
    m=Number Of SubSets SubDataSet=[]
    Step1 :
    for in range of 1 to M do
        CreateRandomData From Orginal Datset
    Step2 :
procedure Bagging(N,M,BaseAlgo,TrainData,Target,TestData)
    prediction=matrix(RowLength=len(target),ColumnLength=N)
    for in range of 1 to N do
        SubDataSet=BootStrapSample(TrainData,M)
        Predictions[,i]=BaseAlgorithm.fit(OriginalDatset, target).predict(TestData)
        FinalPredictions = voting(predictions)
```

[2]

Algorithm 3 AdvancedEnsemble:Stacking

```
procedure Stacking
    BaseAlgorithms = [LogisticRegression, DecisionTreeClassification, ...]
    StackTrainData=matrix(RowLength=len(target),ColumnLength=len(BaseAlgorithms
    StackTestData=matrix(RowLength=len(target),ColumnLength=len(BaseAlgorithms)
    for i,BaseAlgorithm in BaseAlgoritms do
        StackingTrainDataset[,i]=BaseAlgorithm.fit(train,target).predict(train)
        StackingTestDataset[,i]=BaseAlgorithm.predict(test)          FinalPredic-
tions=CombinerAlgorithm.fit(StackTrainData, target).predict(StackTestData)
```

[3]

6. The dataset

The datasets contain transactions made by credit cards in September 2013 by European cardholders. There are 492 frauds out of 284,807 transactions which make the data highly unbalanced, where we have the positive class for the transactions which are fraudulent account for 0.172% of all the total transactions in the dataset. Because of confidentiality issues, we are provided with the numerical input variables after the PCA transformation on the dataset. We are also given the outcome variable which is a binary variable. The other two are 28 features which are the PCA transforms (V1, V2, V28). The feature 'Amount' is the transaction Amount, another feature is given as 'Time'. The feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise. There were no missing values in the dataset.

7. Ensemble techniques

We used the different ensemble techniques using Machine Learning and Deep Learning models, which are discussed below.

7.1. Simple ensemble techniques

Voting Based Classifiers includes the three techniques of voting. There were five base classifiers in all three techniques (Logistic Regression, Naïve Bayes, Support Vector Machine (SVM), K-Nearest Neighbour (KNN), and Decision Tree).

a.) Averaging
b.) Hard Voting
c.) Soft Voting

7.2. Advanced ensemble techniques

a.) Bagging (Random Forest, Extra Trees Classifier)
b.) Boosting (AdaBoost, Gradient Boosting Machine, XGBOOST)
c.) Stacking (Logistic Regression, Naïve Bayes, Support Vector Machine (SVM), K-Nearest Neighbour (KNN) and Decision Tree)

7.3. Deep learning-based ensembles

a.) ANN (An ensemble of two Model Architectures for Artificial Neural Networks)

8. Evaluation metrics

We used the following evaluation metrics for evaluating the ensemble models [Table 1].
- Accuracy Score
- Confusion Matrix
- Precision
- Recall
- F1-score
- AUC score
- Matthew Correlation Coefficient Score
- Cohen Kappa Score

All the different models were trained and evaluated using a train-test split of 70%-30% and the different evaluation metrics were used to evaluate the different ensemble models. [Figure 1] shows the flow diagram of the machine learning pipeline which we used to conduct this study.

Table 1. Typres of evaluation metrics

Evaluation Metrics	Description
Accuracy score	Percentage of Correctly Classified observations.
Confusion matrix	Representation of correctly and incorrectly classified instances in test set for the classes in a matrix. The confusion matrix is one of the most important representations in classification which describes all performance and can be considered as the base of all evaluation metrics.
Precision	The sum of True Positives divided by the combined number of True Positives and False Positives equals precision. In other words, it's the total number of positive class values predicted divided by the total number of positive predictions. It's also known as the Positive Predictive Value (PPV). Precision can be thought of as a metric for how accurate a classifier is. A high number of False Positives indicates poor precision.
Sensitivity/Recall	Sensitivity is known as the True Positive Rate, also known as Recall which is the proportion of actual positive cases which are predicted as positive by our model. Recall, in simple words, is the fraction of examples classified as positive, among the total number of positive examples or the number of true positives divided by the number of true positives plus false negatives. Sensitivity = True Positive / (True Positive + False negative)
Specificity	Specificity is also known as the True Negative Rate which is the proportion of actual negative cases which are predicted as negative by our model. Specificity = True Negative / (True Negative + False Positive)
Area Under Curve (AUC) Score	At different threshold conditions, the AUC score is an output measurement for classification problems. The degree or metric of separability is represented by the AUC. It indicates how well the model can differentiate between classes. The higher the AUC, the more accurate the model.
F1 Score	The F-score is a way of combining the precision and recall of the model, and it is defined as the harmonic mean of the model's precision and recall.
Matthews correlation coefficient	In machine learning, the Matthews Correlation Coefficient is used to determine the consistency of binary (two-class) classifications. It accounts for true and false positives and negatives. The MCC is a correlation coefficient that returns a value between -1 and +1 for observed and expected binary classifications. A coefficient of +1 indicates a perfect prediction, a coefficient of 0 indicates little more than a random prediction, and a coefficient of -1 indicates disagreement between prediction and observation. It is highly used for the unbalanced dataset.
Cohen Kappa	Cohen's kappa coefficient is a statistic that measures inter-rater agreement for qualitative (categorical) items. Cohen's kappa statistic is a very good measure that can handle very well both multi-class and imbalanced class problems. Cohen's kappa is always less than or equal to 1. Values of 0 or less, indicate that the classifier is useless. There is no standardized way to interpret its values.

[Sourced From Google]

9. Implementation

9.1. Data analytics

Initially, we loaded the data into the panda's data frame and performed basic exploration of the data, we also checked the missing values. It was found that there were no missing values in the dataset. We conducted the Exploratory Data Visualization (EDA) for all the variables present in the dataset. The purpose of EDA was to check the distribution of all the variables.

We used the univariate as well as the bivariate analysis to perform the EDA. We used the line plots for the independent variables and the count plot or frequency plot for the outcome variable. For bivariate analysis, we consider the different independent variables together with the outcome variable, in the scatter plots and box plots we looked into the relationship of the different independent variables with the outcome variable. [Figure 3] shows the distribution of classes in the dataset.

Figure 3. Distribution of classes in the dataset

It can be observed that only 492 (or 0.172%) of the transaction are fraudulent. That means the data is highly unbalanced for the target variable Class. [Figure 4] shows the time density plot of the normal and fraudulent transactions.

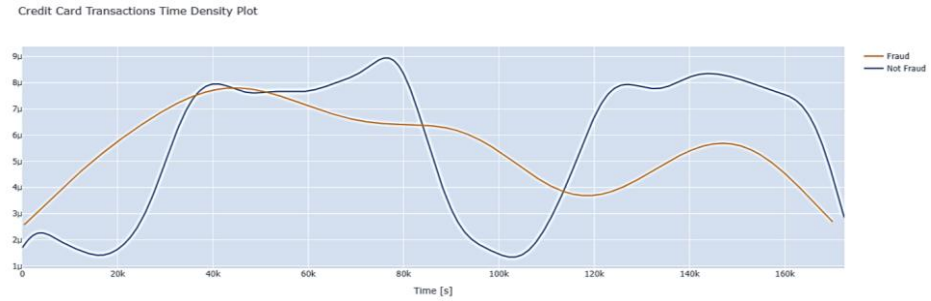

Figure 4. Time density plot of the normal and fraudulent transactions

It can be observed that fraudulent transactions have a distribution more even than valid transactions - are equally distributed in time, including the low real transaction times, during the night in the European time zone. [Figure 5] shows the number of fraudulent transactions over time.

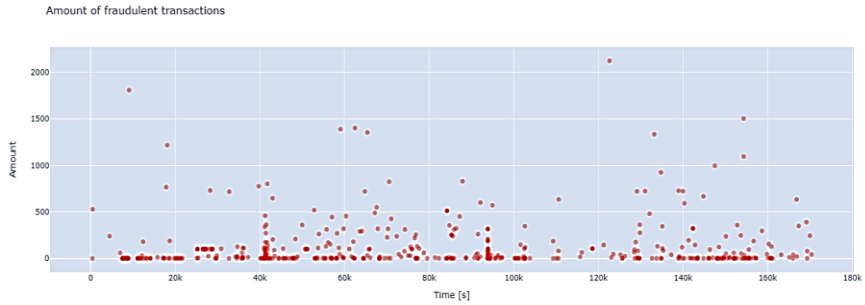

Figure 5. Number of fraudulent transactions over the time

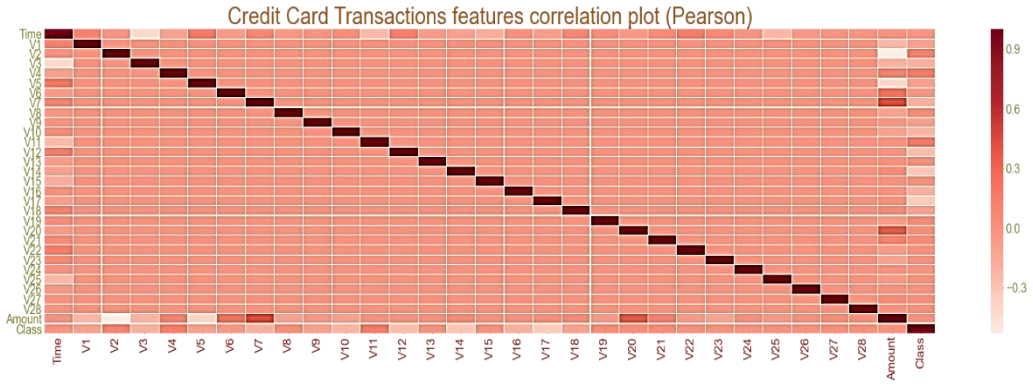

Figure 6. Feature correlation plot

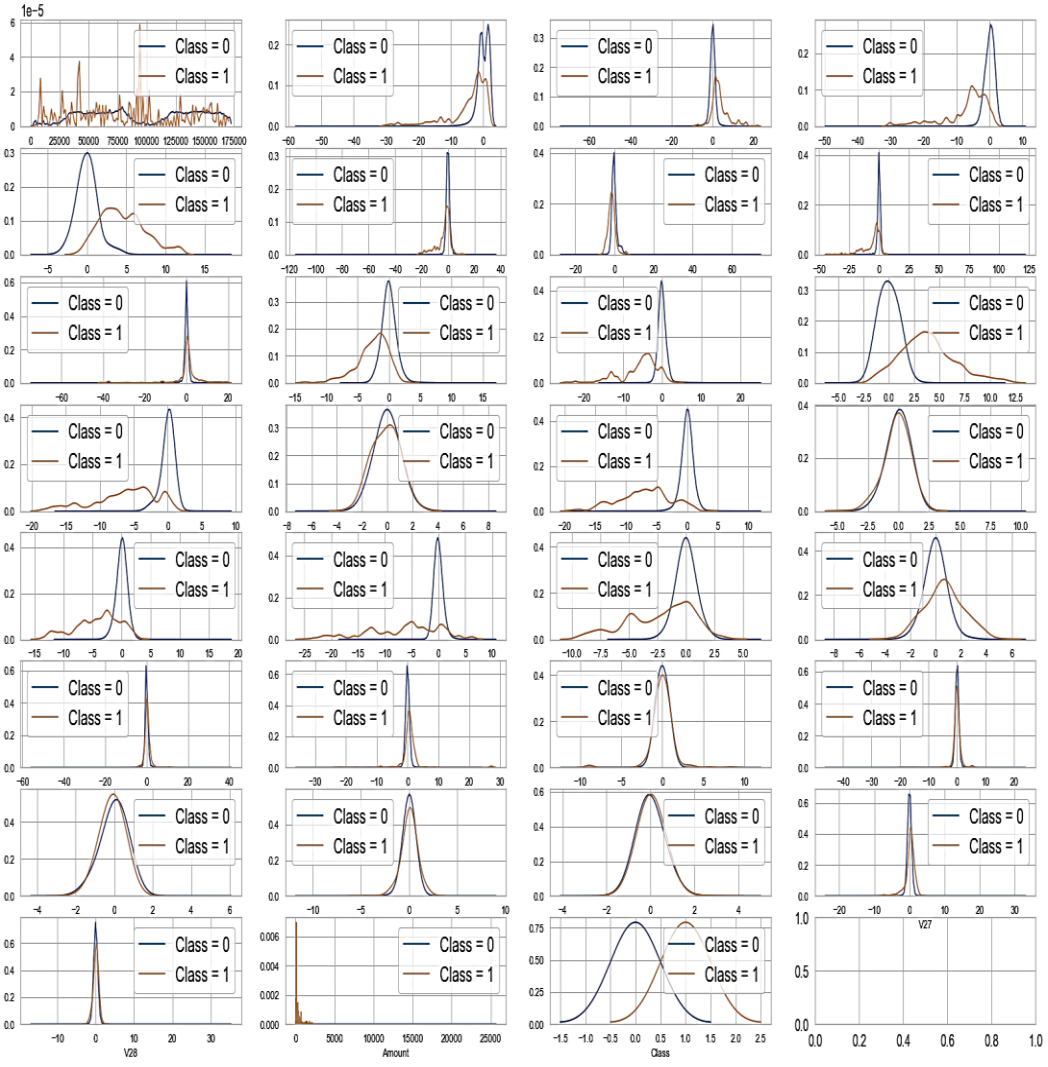

Figure 7. Density plot of each variable

Feature correlation is another important analysis to check for the correlation among the variables, which we also used to check for the correlation among the variables, with which we can reduce the dimension of the features. We used the line plots and the density plot after the correlation analysis. There was not any high correlation among the feature variables. It can be seen from [Figure 6] that there is no notable correlation in features V1-V28 as well as with the target variable.

[Figure 7] shows that there is strong selectivity in terms of distribution for some of the features for the two values of Class: For Class values 0 and 1, V4, V11 has distinctly differentiated distributions, V12, V14, V18 is partly separated, V1, V2, V3, V10 has a distinct profile, and V25, V26, V28 has identical profiles for the two values of Class. With just a few exceptions (Time and Amount), the distribution of the features for valid transactions (Class = 0) is generally centered on 0, with a long queue at one of the extremes. Around the same time, the distribution of illegitimate transactions (values of Class = 1) is distorted (asymmetric).

9.2. Balancing using SMOTE

The next step was to balance the data before performing machine learning modeling. [Figure 8] and [Figure 9] show the distribution of classes in the dataset before applying Over-Sampling and after applying Over-Sampling for balancing the classes. It can be seen that the dataset was highly imbalanced, so the next step was to balance the data. We used SMOTE Over Sampling Method (Up Sampling) to sample the dataset. Over Sampling Method or Up Sampling deals with generating synthetic data to the Minority Class, to make the number of observations equal with both the classes.

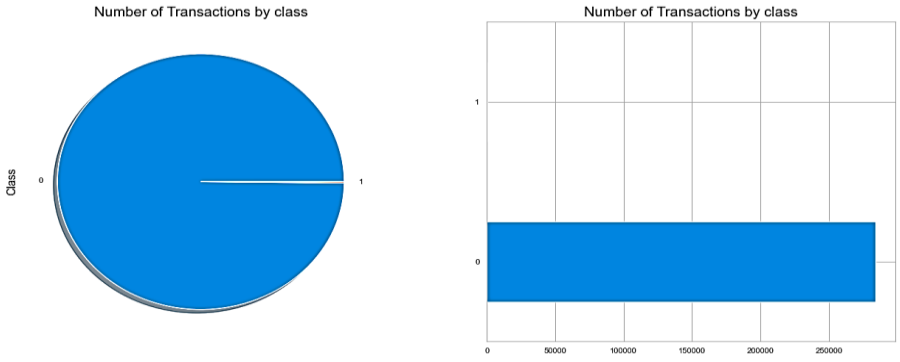

Figure 8. Before applying oversampling for balancing the classes

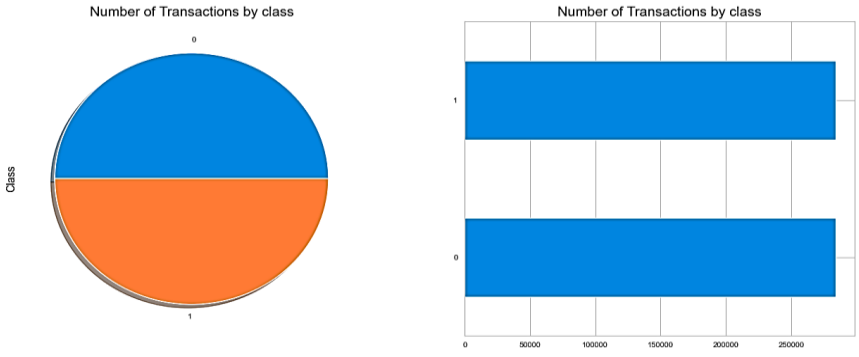

Figure 9. After applying oversampling for balancing the classes

The Synthetic Minority Oversampling Technique, or SMOTE for short, is the most commonly used method for synthesizing new instances or oversampling. Nitesh Chawla described this technique in their paper titled "SMOTE: Synthetic Minority Over-sampling Technique". SMOTE functions by choosing samples in the feature space that are close together, drawing a line in the feature space between the examples, and drawing a new sample at a point along the line. To be more specific, a random example from the minority class is selected first. Then, for that example, k of the closest neighbors is identified (typically k=5). A randomly selected neighbor is chosen, and a synthetic example is generated at a randomly chosen point in feature space between the two examples. The oversampling method doubled the number of observations for the majority class and we had 568,630 observations. For making a robust model with a considerable number of observations, we used the over-sampling method.

9.3. Models and results

The following are the models which we used in this study [Table 2].

Table 2. Ensemble techniques models

Ensemble Techniques	Description	Model Type and Name	API Used
Simple Ensemble Techniques	Voting Based Classifiers includes the three techniques of Voting. There will be 5 base classifiers in all three techniques (Logistic Regression, Naïve Bayes, SVM, KNN, and Decision Tree).	Max Voting Averaging Weighted Average	SciKit Learn
Advanced Ensemble Techniques	These comprise Stacking, Bagging, and Boosting.	Stacking (Logistic Regression, Naïve Bayes, SVM, KNN and Decision Tree) Bagging (Random Forest, Extra Trees Classifier) Boosting (Adaboost, Gradient Boosting Machine, XGboost)	SciKit Learn XGBOOST
Deep Learning-Based Ensembles	This comprises Deep Learning-based models which are stacked with another based-on voting criteria.	ANN (An ensemble of two Model Architectures for Artificial Neural Networks)	Keras Tensor flow

The final selection of the model was based on the multiple evaluations and overall performance check across the various evaluation metrics, to get the best ensemble model for predicting Credit Card fraud with robustness. The ensembles were compared with a traditional model to examine how well they performed. [Figure 10] shows the comparison of Bagging and Boosting-based models using 5-fold Cross-Validation. It can be seen that XGBOST, Extra Trees, Random Forest, and AdaBoost performed well in comparison to the Gradient Boosting Machine Classifier.

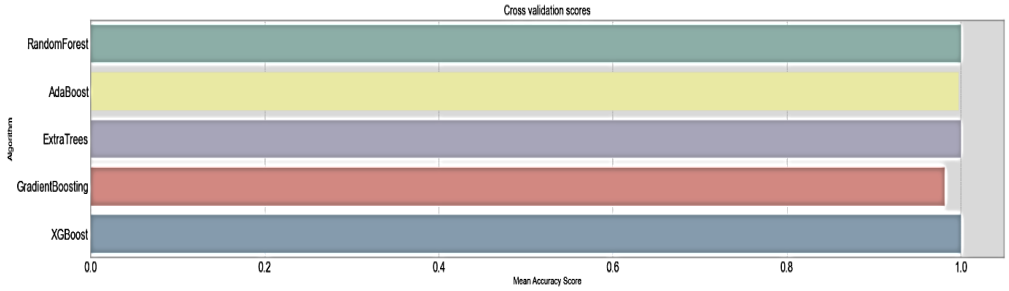

Figure 10. Comparison of ensemble models - Cross-validation

We also tested the Bagging and Boosted models using the train-test split method for which the evaluation results are shown in [Figure 11]. It can be seen that in the train-test split method Xgboost and GBM are performing the best whereas the Random Forest model did not give good results.

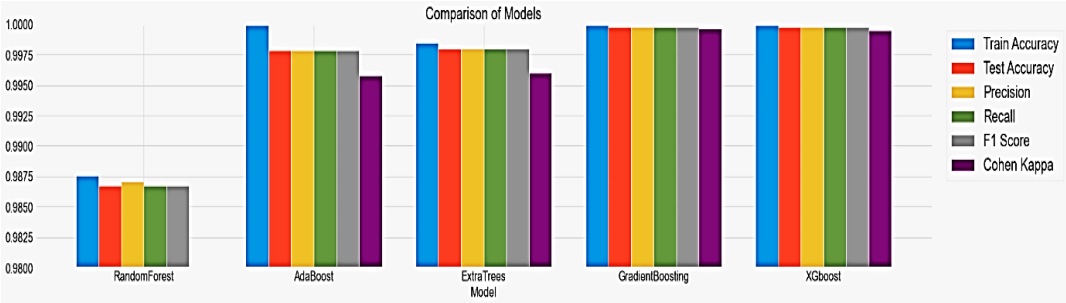

Figure 11. Comparison of ensemble models - train test split

We also used Averaging and Voting-based ensembles as shown in [Figure 12].

1. Simple Averaging - Predict the class with the largest sum of votes from models
2. Soft Voting - Predict the class with the largest summed probability from models.
3. Hard Voting - Predict the class with the largest sum of votes from models.

The simple Averaging based model did not perform well as compare to the weighted averaging-based models. Soft Voting-based models outperformed the hard Voting-based ensemble models across the various evaluation metrics except for the Cohen Kappa score.

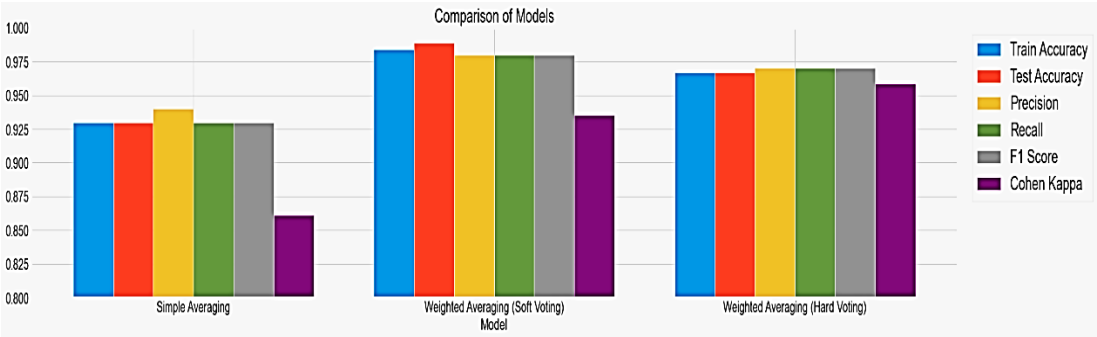

Figure 12. The ensemble on traditional models – voting ensemble

Finally, we compared all the models with the Stacking-based ensemble using multiple evaluation metrics for which the results are shown in [Figure 13] and [Table 1].

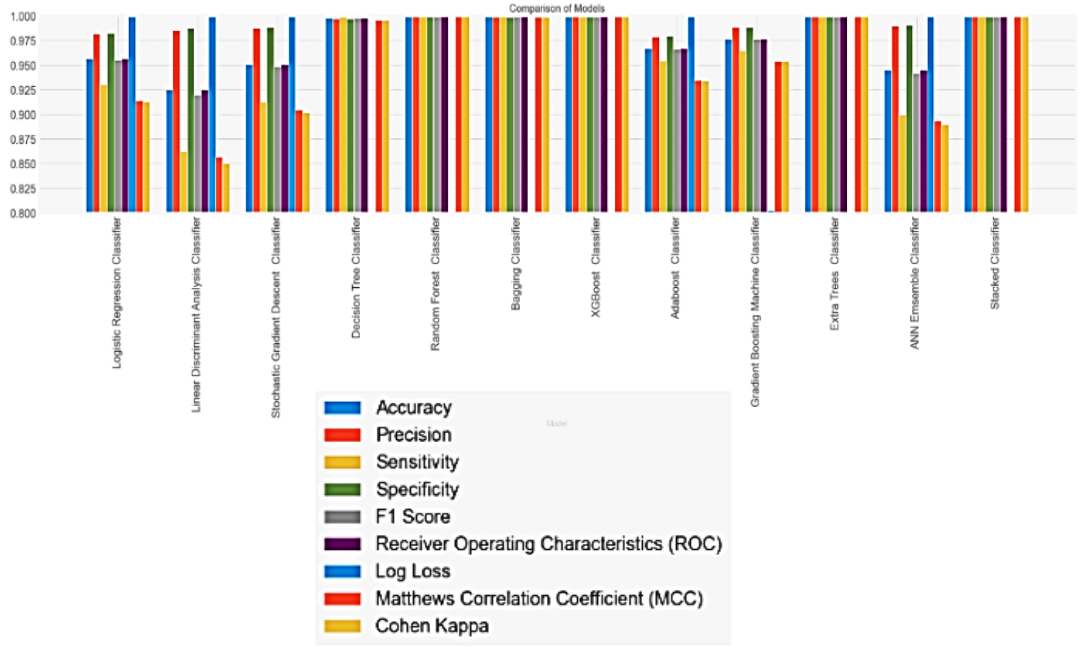

Figure 12. Comparison of all models with stacking-based ensembles

Table 3. Comparison of models

Model	Accuracy	Precision	Sensitivity	Specificity	F1 Score	ROC	Log Loss	MCC	Cohen Kappa
Logistic Regression Classifier	0.95653	0.98186	0.93029	0.98279	0.95538	0.95654	1.50151	0.91432	0.91306
Linear Discriminant Analysis Classifier	0.92497	0.98576	0.86247	0.98753	0.92001	0.925	2.59159	0.85668	0.84994
Stochastic Gradient Descent Classifier	0.95088	0.98794	0.91295	0.98884	0.94897	0.9509	1.69668	0.90437	0.90176
Decision Tree Classifier	0.99804	0.99722	0.99888	0.99721	0.99804	0.99804	0.06763	0.99609	0.99608
Random Forest Classifier	0.99992	0.99984	1	0.99984	0.99992	0.99992	0.00344	0.99984	0.99984
Bagging Classifier	0.99937	0.99916	0.99958	0.99916	0.99937	0.99937	0.02187	0.99873	0.99873
XGBOOST Classifier	0.99983	0.99966	1	0.99966	0.99983	0.99983	0.00587	0.99966	0.99966

AdaBoost Classifier	0.96711	0.97926	0.95447	0.97976	0.96671	0.96712	1.13605	0.93452	0.93422
Gradient Boosting Machine Classifier	0.97678	0.98856	0.96475	0.98882	0.97651	0.97679	0.80198	0.95384	0.95356
Extra Trees Classifier	0.99992	0.99984	1	0.99984	0.99992	0.99992	0.00283	0.99984	0.99984
ANN Ensemble Classifier	0.94507	0.98983	0.89945	0.99074	0.94248	0.9451	1.89713	0.89389	0.89015
Stacked Classifier	0.99992	0.99985	1	0.99985	0.99992	0.99992	0.00263	0.99985	0.99985

Matthew's correlation coefficient produces a more informative and truthful measure in evaluating binary classifications than accuracy and F1 score. We were more interested in the Matthews correlation coefficient (MCC), as it is a more accurate evaluation metric for problems such as Fraud Detection which only yields a high score if the model performed well in all four confusion matrix groups (TP, FN, TN, and FP), proportionally to the size of positive and negative instances in the dataset.

From the above comparison, it can be seen that Stacked Classifier performed the best as compared to the other. Extra Trees Classifier also performed very well but had lower MCC and Cohen Kappa Score as compared to the Stacking based Classifier. Also, in terms of Accuracy, Precision, F1, Sensitivity, and Specificity, the Stacked Classifier performed better as compared to the other classifiers.

Except for the Adaboost and Gradient Boosting Classifier, the different bagging methods we used such as Radom Forest and Extra Trees Classifier as well as boosting methods including XGBOOST performed very well. We also used Cohen Kappa to rate the models which is a statistical measure used to compare the reliability of two or models in machine learning identifying how frequently the models agree. The Cohen Kappa score was highest for the Staked Classifier model confirming that the Stacked Classifier model is performing the best. As a result, Stacked Classifier, Extra Trees Classifier, and Random Forest Classifier can be considered as the top 3 models.

[Figure 13] shows the feature importance of the Variables using Permutation Importance. V14, V17, and V10 are the top three features which affect the prediction of the model and hence fraudulent and non-fraudulent transactions.

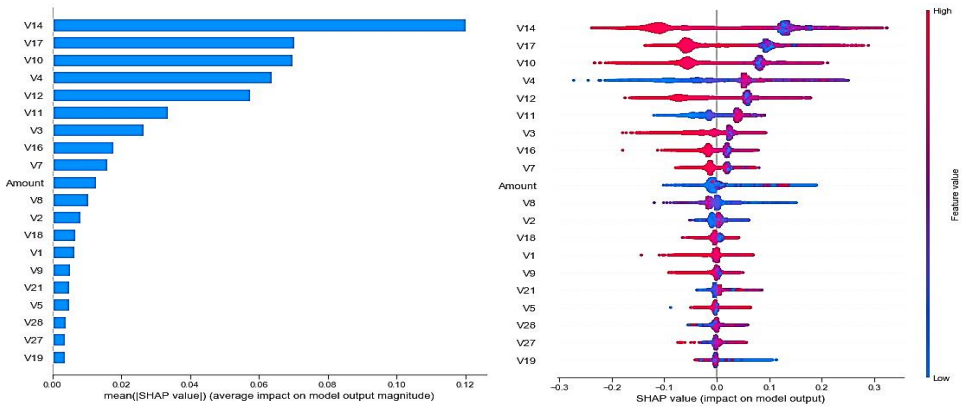

Figure 13. Feature importance of the variables using permutation importance and SHAP

10. Conclusion

We used several ensemble models which were Bagging based ensemble (Random Forest and Extra Trees Classifier), Boosting based ensemble (AdaBoost, Gradient Boosting Machine and XGBOOS, stacking based ensemble for traditional and Deep Learning-based), and Simple as well as weight-based Voting ensembles and compared them with various evaluation metrics. The evaluation metrics we used were Accuracy Score, Confusion Matrix, Precision, Recall, F1-score, ROC Curve, AUC score, Matthews's correlation coefficient, and Cohen Kappa. It can be concluded that stacking based on powerful machine learning algorithms resulted in higher performance than any individual machine learning model. We also interpreted the second-best performing algorithm which was Extra Trees and Random Forest Classifier.

Acknowledgment

This submission is part of the first author COMP5800 MS Project course supervised by Dr. Sabah Mohammed. The source code is published in Github: https://github.com/psatyadileep/ensemble_techniques_for_credit_fraud

References

[1] S. Bagga, A. Goyal, N. Gupta, and A. Goyal, "Credit card fraud detection using pipelining and ensemble learning," Procedia Computer Science, vol.173, pp.104-112, **(2020)**, DOI: 10.1016/j.procs.2020.06.014

[2] I. Sohony, R. Pratap, and U. Nambiar, "Ensemble learning for credit card fraud detection," pp.289-294, **(2018)** DOI: 10.1145/3152494.3156815

[3] M. Zareapoor and P. Shamsolmoali, "Application of credit card fraud detection: Based on bagging ensemble classifier," Procedia Computer Science, vol.48, pp.679-685, **(2015)**

[4] A. Barahim, A. Alhajri, N. Alasaibia, N. Altamimi, N. Aslam, and I. Khan, "Enhancing the credit card fraud detection through ensemble techniques," Journal of Computational and Theoretical Nanoscience," vol.16, pp.4461-4468, **(2019)**, DOI: 10.1166/jctn.2019.8619

[5] D. Singh, N. Keerthana, and S. Bhandari, "Performance evaluation of class balancing techniques for credit card fraud detection," IEEE CPCSI-2017

[6] S. Altyeb, "An intelligent approach to credit card fraud detection using an optimized light gradient," IEEE Boosting Machine, **(2020)**

[7] G. Douzas, F. Bacao, and F. Last "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE," Elsevier

[8] Y. Dong and X. Wang, "New over-sampling approach: Random-SMOTE for learning from imbalanced data sets," Springler **(2011)**

[9] M. K. Randhawa, "Credit card fraud detection using AdaBoost and majority voting," IEEE, **(2018)**

[10] Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE, Elsevier **(2018)**

[11] D Almhaithawi, A. Jafar, and M Aljnidi, "Example-dependent cost-sensitive credit cards fraud detection using SMOTE and Bayes minimum risk," August, © Springer Nature Switzerland AG, **(2020)**

[12] H. Wang, P. Zhu; X. Zou; S. Qin, "An ensemble learning framework for credit card fraud detection based on training set partitioning and clustering," IEEE smart world, **(2018)**

[13] S. Dhankhad, E. Mohammed, B. Far, "Supervised machine learning algorithms for credit card fraudulent transaction detection: A comparative study," IEEE IRI, **(2018)**

[14] D. Prusti, S. K. Rath, "Fraudulent transaction detection in credit card by applying ensemble machine learning techniques," IEE ICCCNT, **(2019)**

[15] F. Carcilloa “Combining unsupervised and supervised learning in credit card fraud detection,”

[16] S. Khatri, A. Arora, A. P. Agrawal, “Supervised machine learning algorithms for credit card fraud detection: A Comparison,” IEEE, **(2020)**

[17] J. V. V. S. Sasank, G. Sahith, K. Abhinav, M. Belwa, “Credit card fraud detection using various classification and sampling techniques: A comparative study,” IEEE ICCES, **(2019)**

[18] J. O. Awoyemi, A. O. Adetunmbi, S. A. Oluwadare, “Credit card fraud detection using machine learning techniques: A comparative analysis,” IEEE ICCNI, **(2017)**

[19] L. Li, “An extensive review on recent deep learning applications,” Asia-pacific Journal of Convergent Research Interchange, SoCoRI, ISSN: 2508-9080 (Print), 2671-5325 (Online), vol.5, no.3, pp.221-231, September **(2019)**, DOI: 10.21742/apjcri.2019.09.22