Credit card fraud detection using machine learning

Krunal Thummar

The University of Winnipeg, Winnipeg MB R3B2E9, Canada
thummar-k@webmail.uwinnipeg.ca

Abstract. Fraud analysis has been an emerging research topic in recent years. With the growth of shopping needs, payment methods are also becoming advanced to provide ease of purchase. credit card is becoming preferred payment method day by day. The scope of this work will be to utilize and analyze data available from the previous transactions of customers and analysis transactions to examine whether it is a fraud or non-fraud transaction. My major focus will be finding the best suitable techniques to recognize fraudulent activities. In this research, I used the already available dataset to perform my experiments. I will use different machine learning algorithms like the random forest, decision tree, logistic regression, knn, XGBoost to compare and find the best solution with a high AUC score. Additionally, I will consider not only the accuracy of the algorithm but also the F1-score to determine the best results because of an unbalanced dataset.

Keywords: Machine Learning · Logistic regression · Rendom forest · Decision tree · XGBoost · SMOTE · Recursive Feature Elimination

1 Introduction

Financial institutions and Credit cards play a vital role in everyone's life. Credit cards are widely used around the globe. It becomes an unavoidable part of personal, commercial, and global activities. Though credit card uses provides more and more beneficial, it is crucial to consider the strong security aspect of credit card services. With the increasing use of credit cards, security becomes a critical aspect. Many techniques have been proposed to confront fraudulent activities that occur in finance and the banking system. However, the purpose of providing different solutions against credit card fraud is the same; each one has its own characteristics, advantages, and drawbacks. With all researches and comparisons, we can analyze the best solution to detect fraudulent transactions. Credit card-related fraud is escalates drastically alongside development of advanced methodologies resulting in the loss of plenty of money each year in the whole world. Statistics from the ICC (Internet Crime Complaint) Center show that fraudulent activities are rising drastically in the last years [5]. Monetary losses caused due to online fraud only in the U.S. were reported $3.4 billion in 2011 and increasing significantly as shown in Figure 1. Fraud analysis and detection involves identifying numerous fraud related activities alongside various genuine transactions as quickly as possible. A various methods to detect financial fraud methods are emerging rapidly in order to adapt to new advanced activities that are introducing related to fraud detection across the world. However, this explores and headway of approaching systems for the discovery of cheats become more obstacle because of the extreme limit of the thoughts trade-in misrepresentation location.
In this project, I examined many classification algorithms using credit card uses dataset available on Kaggle. Before starting the analysis it is important to know the structure of the available dataset. In this study, I used scikit learn in jupyter notebook to perform all experiments.
In this study, The dataset evaluation is also important. After evaluation, I found that the dataset is highly imbalanced.

The report is organized as follows. In section 2, I described the related work of this study which is completed by other researchers. In section 3, I show the description of a theory that is used for the analysis and visualization of data. In section 4, I describe algorithms for predicting whether the transaction is fraudulent or non-fraud. In section 5, I show the results of the analysis, providing a conclusion with references in section 6.

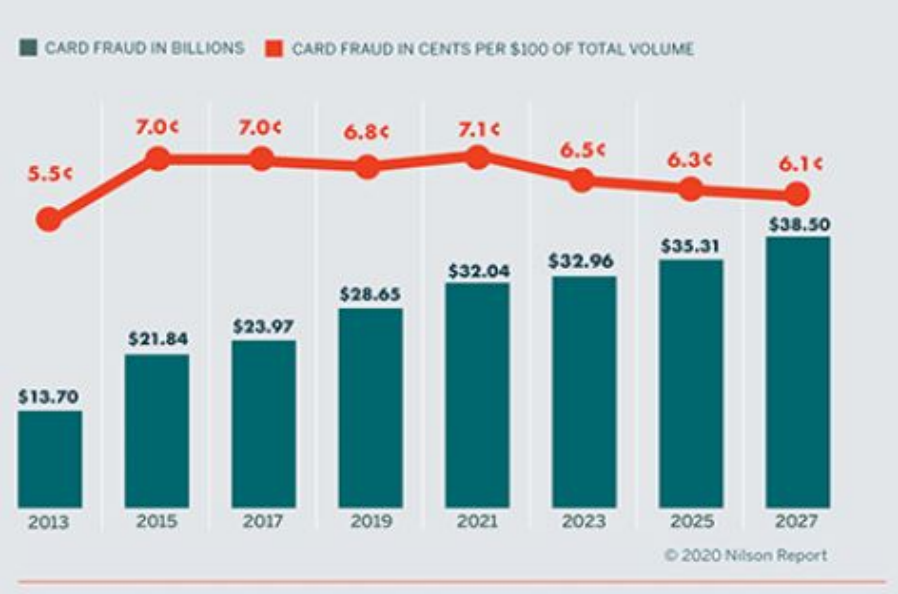

Fig. 1. Statistics of credit card fraud

2 Literary search

Vaishnavi et al.,(2019)[1] developed a novel method for the detection of fraud. Researchers used clustering and sliding window techniques to differentiate cardholders and aggregate transactions into groups respectively. Researchers used smote operation and one class classifiers to deal with imbalanced dataset. they used different classifiers to train the model and obtained the results from the classifier with the highest rating score. They found out that with smote classifiers were performing better than the original dataset. In their experiments, Logistic regression, Random forest, and decision tree gave better results.

The paper [2] introduced a methodology for credit card fraud detection using big data analytics. Researchers used multiple machine learning algorithms to predict the fraud and to pre-process the dataset, they designed a framework using Hadoop network. Moreover, they used a confusion matrix to evaluate the accuracy of the proposed analytical model. The confusion matrix tells how the tuples in training and testing models are correctly classified. After experimenting with different models, they found that random forest and decision tree performs the best in terms of accuracy, precision, and recall. However, researchers found random forest a bit problematic to the memory in term of handling large dataset.

Yong et al.,2019 [3] compared three machine learning algorithms Random forest, gradient boosting method, light gradient boosting method, and compared their result to prove the efficiency of the algorithms. Researchers used a dataset available from Kaggle which contained over 410,000 records of credit card transactions. They used SMOTE method to handle imbalanced dataset. Moreover, they used the one-hot encoding technique to deal with certain feature values and make data numeric and continuous. Their Experiment showed that LightGBM performed well in terms of time and accuracy.

3 Theoretical Framework

To detect the fraud transaction in real-time I used an available dataset from Kaggle to perform training and testing on them and train our model to detect the transaction is fraud or a normal one. To use the dataset available on Kaggle which is highly imbalanced we need to take care of data by doing some pre-processing work on them. To do so, I applied Synthetic Minority Over-Sampling Technique(in short, SMOTE) to the dataset. The motivation behind this technique is to balance class distribution by randomly increasing minority class samples by replicating them. After applying some pre-processing steps, I applied different classifiers to them to check the accuracy of results and compare the difference between all applied algorithms. The purpose of comparing the results is to determine which algorithm is giving the better result for our problem. Further, I evaluated which classification algorithm is accurate by comparing their precision, Recall, and F1-score values after applying them in our original dataset and SMOTE dataset.

3.1 Classifiers

To build a successful system I used many different machine learning algorithms. First of all, the main goal is to train the system based on current data available and then perform different machine learning algorithms to compare results. Many algorithms like Logistic regretion, random forest, XGBoost, K-nearest neighbor, Decision tree were used and compared.

The Logistic Regression is a classification algorithm that is used to predict the probability of a categorical dependent variable. It is a supervised machine learning algorithm. Logistic regression is used to predict the probability of a categorical dependent variable that whether it will belong to fraud class or Non-fraud class. In our case, it will categorize the transaction to make a decision of fraudulent or normal transaction.

Random Forest Classifier is an ensemble learning method that creates multiple decision trees from the randomly selected training set data. later it generates and combines votes from all different decision trees to decide the final class of the test object.

XGBoost is a distributed gradient boosting library which is optimized, highly efficient, flexible, and portable. There is a hyperparameter in XGBoost called scale_pos_weight. By default, the value of this hyperparameter is set to 1. As a result, it weighs the balance of positive examples, relative to negative examples during the decision tree boosting process. Gradients are used to correct errors made by the existing state of the ensemble of decision trees. Additionally, it is used as the basis for fitting subsequent trees added to boost. scale_pos_weight value is determined to scale the positive class gradient. As a result, its effects on the scaling errors generated by the model at a time of training on the positive class. Further, it also encourages the model to over-correct them. As a result, this can help the model achieve better performance when making predictions on the positive class. A sensible default value to set for the scale_pos_weight hyperparameter is the inverse of the class distribution.

K-nearest neighbor(KNN, in short) is a supervised lazy learning algorithm used to solve classification as well as regression problems. It determines the parameter k, which is the nearest data

point to consider. Then calculates the distance between the new point and all the training samples. Later it determines the closest neighbors based on the k number of minimum distances and uses the majority from nearest neighbors to make a decision. nearest neighbors are considered based on uniform weight and distance weight. In distance weight, closer the points are, more weight is given to them at the time of considering the results.

Decision trees are also used to solve classification problems and regression problems. It generates the tree that can be updated step-by-step by splitting the dataset into smaller datasets (numerical and categorical. In decision tree method, the results are represented in the leaf nodes. A decision tree is a supervised learning method. I used sqrt and $\log_2$ for a number of features selection. Also, gini impurity and entropy for split quality.

the problem with the imbalanced dataset us that there are too few examples of minority class for a model to effectively learn the decision boundary. Oversample and Undersample the examples in the minority class and majority class respectively are the possible solutions for imbalanced datasets. However, To handle an imbalanced dataset, I used the method called the Synthetic minority oversampling technique.

3.2 SMOTE

SMOTE synthesizes new examples to deal with minority class. Algorithm of SMOTE works as follows:-

Step 1: It first sets the minority class set A, for each $x \in A$, the k number of nearest neighbors are generated by calculating the Euclidean distance between x and all other sample in set A.

Step 2: Rate of the sampling N is set according to the proportion of the imbalance data. For each $x \in A$, N examples (i.e x1, x2, ... xn) are randomly selected from its k-nearest neighbors. Later, They construct the set A_1.

Step 3: For each example $x_k \in A_1$ (k=1, 2, 3...N), formula to generate new samples is given below:

$$y' = y + rand(0, 1) * | y - y_k | \tag{1}$$

Where:
rand(0, 1) = the random number between 0 and 1.

3.3 skewness

Skewness defines a measure of the asymmetry of a dataset. The value of skewness can be positive, negative, or undefined. For example, a dataset has a skew if there are values far away from the mean on one side means if it is above or below. The distribution which is perfectly symmetrical, the mean, the median, and the mode will all have the same value. In terms of negative skew, it occurs when the dataset contains values that are much less than the mean but far lesser values much greater than the mean. it can be defined by equation (2).

$$Skewness = \frac{\sum_i^N Y_i - \bar{Y}}{(N-1) * (sd)^3} \tag{2}$$

Where:
N = total number of variables contained by distribution
Y_i = random variable
$\bar{Y}$ = mean of the distribution
sd = standard deviation

3.4 Robust scaler

Standardization of dataset features is a basic requirement for many machine learning techniques. Some algorithms like logistic regression, nearest neighbors perform better when features are on relatively identical scale or near to normal distribution.
A Robust Scaler is used when outliers are present in the dataset and we want to reduce their influence. RobustScaler standardizes features by removing the median and dividing each feature by the interquartile range. features with large values are converted to a similar scale to the other features. For each feature X, we calculate the median and two quantiles $X_{0.25}$ and $X_{0.75}$. It can be calculated with equation (3)

$$NewY_i = \frac{Y_i - Y_m d}{Y_{0.75} - Y_{0.25}} \tag{3}$$

Where:
($Y_{0.75}$-$Y_{0.25}$) is an interquartile range.

4 Implementation

In this section, I show how the approach illustrated in previous sections has been implemented to get results. The Execution steps are shown in Figure 2. For the Implementation of this project, I used Python Programming Language version 3 in jupyter notebook, 64-bit system with windows 10 Operating system, 6 GB RAM, and 2.7 GHz Intel Core i-7 Processor.

4.1 Data Collection

In this project, to detect fraud and non-fraud transaction, I used a dataset available in the kaggle [7] site in the comma-separated values(CSV). transactions described in the dataset are made by credit card users in September 2013 in the region of Europe. It has total of 284,807 transactions from which only 492 are frauds. The Details of dataset can be found in Table 1.

Table 1. Credit card Dataset Information

Column	Description
Time	It represent the time elapsed between each transaction and the first transaction in seconds
Amount	It shows amount of the transactions
V1 to V28	These are features obtained with principle component analysis.
Class	It is the decision variable. It takes value 1 for fraud and value 0 for the rests.

4.2 Data analysis

To analyze the dataset features, I firstly checked for any null values. There are no null values in any features. Secondly, I visualized the distribution of the Amount and Time features. the visualization of time and amount features is shown in Figure 3 and figure 4 respectively. After finding the skewness of the amount and time features, I found that Amount is highly positively skewed and Time is negatively skewed.

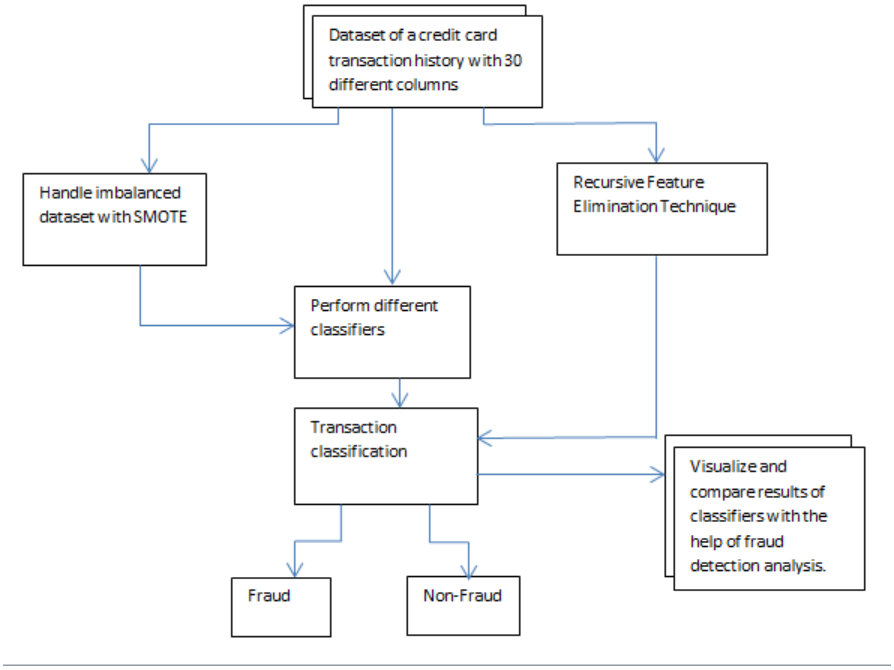

Fig. 2. Project architecture

4.3 data preprocessing

To overcome outliers present in the time and amount features, I used one scaler method called a robust scaler. It scaled features value and removed outliers. This step was necessary because some classifiers may perform better when there are fewer outliers present in the data. I only needed to scale the time and amount features as rest V1 to V28 features are already the result of Principle Component Analysis technique. PCA is a reduction of dimensions technique that is generally used to reduce the dimensionality of a huge data sets. It works by transforming a larger set of variables into smaller one without losing important information in huge set.Hence no need to check outliers or any uncertainty presented in those 28 features. As discussed in section 3.2, I applied SMOTE to overcome the imbalance dataset issue. Figure 5 and Figure 6 depict the illustration of class distribution in the original dataset and class distribution after applying SMOTE respectively.

5 Experiments and Results

In this section, I show how the approach illustrated in previous sections to get results. The python application that I developed is working based on the scikit-learn library to import and use different classifiers. It has predefined algorithms like Logistic regression, XGBoost,etc. to use for our binary classification. The results of this classification is shown in Figure 7. where I considered classification algorithms like Logistic regression, Random forest, XGBoost, K nearest neighbors and Decision tree, and evaluated them using a confusion matrix. I evaluated the train and test F1_score measures for fraud case detection. Since there are plenty of more examples available for non-fraud

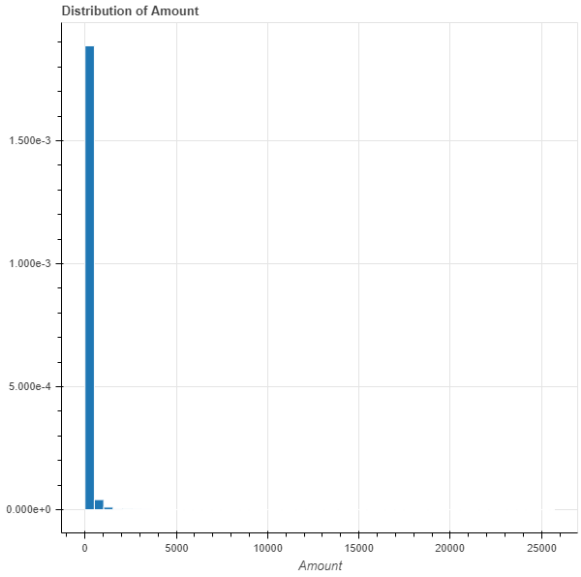

Fig. 3. Distribution of Amount feature

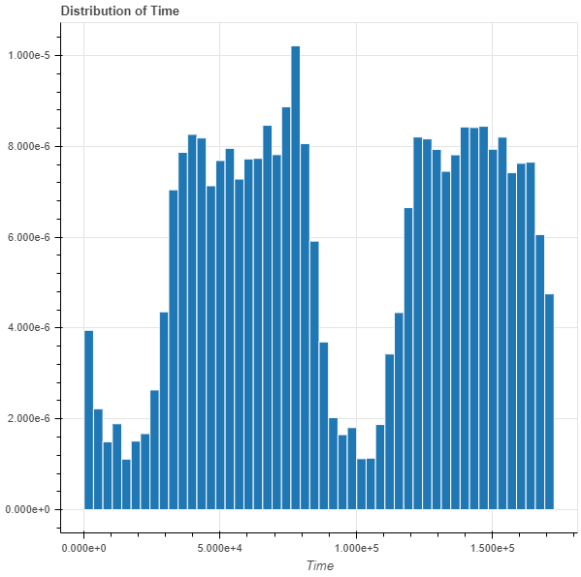

Fig. 4. Distribution of time feature

cases, it always evaluated 100% accuracy in most classifiers. So, accuracy is misleading in terms of evaluation in this project. So, I considered other evaluation measures to finalize my result.

5.1 Evaluation Measures

To evaluate classification results, I used the precision, Recall, and F1-score parameters. Precision is the ratio of the number of correctly predicted samples to all predicted samples. Recall gives an idea of defining the number of samples which are predicted correctly out of all actual sample. F1-score combines precision and recall. The number of fraud transactions classified as fraud is called True Positive but if the number of Non-fraud which are classified as fraud is called false positive. So, the formula for the Precision is defined in Equation (4).

$$\text{Precision} = \frac{\text{True\_Positive}}{\text{True\_Positive} + \text{False\_Positive}} \tag{4}$$

Recall is the ratio of True_Positive to the number of True_Positive and False_Positive. It is defined in Equation (5)

$$\text{Recall} = \frac{\mathit{True\_Positive}}{\mathit{True\_Positive} + \mathit{False\_Negative}} \tag{5}$$

The F-measure or balanced F-score (F1 score) is the harmonic mean of precision and recall defined in equation (6).

$$\text{F}_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

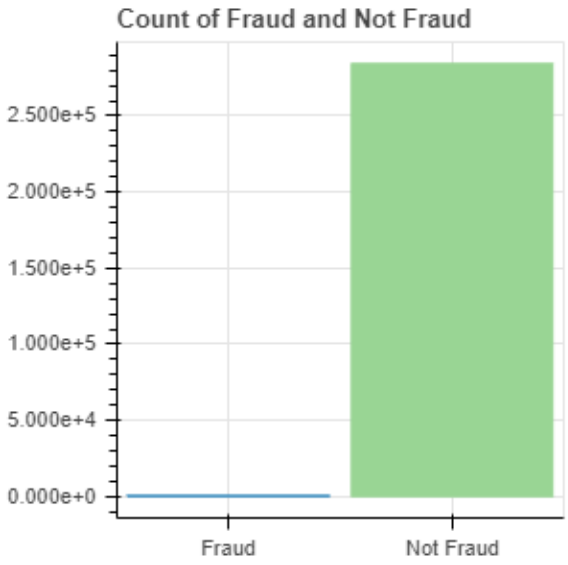

Fig. 5. class distribution of original data

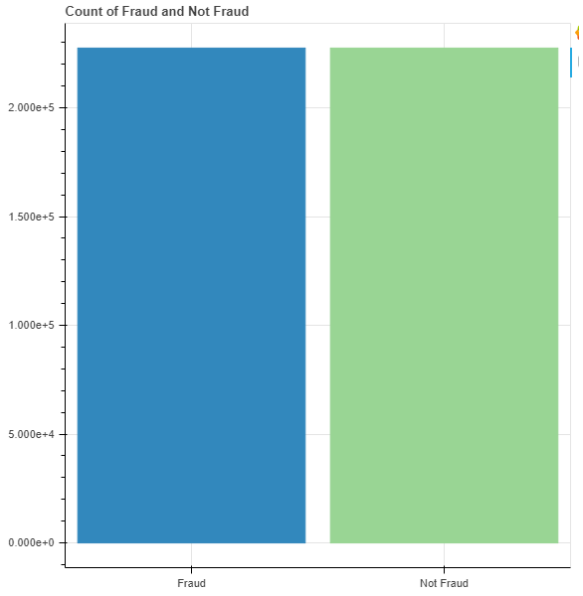

Fig. 6. class distribution of data after SMOTE

In this paper, I used Pandas Dataframe to compare the Actually labeled column with the predicted classified column values and got a binary class confusion matrix. Different classifiers were tested and compared to make a decision about the best performance classification. Table 2 illustrates the precision, recall, and F1-score values from the confusion matrix generated by different classifiers. F1-Score is considered to get results because it is important to classify fraud transactions however not to classify the genuine transactions as fraud as it results in customer dissatisfaction and not suitable for an institution. Figure 8 shows the graph comparison of the F1-score of different classifiers.

Table 2. Result score comparision

Classifier	Precision	Recall	F1-score
Logistic regression	85%	64%	73%
Logistic regression(With Smote)	6%	89%	11%
Random forest	95%	72%	82%
Random forest(With Smote)	92%	75%	85%
XGBoost	94%	80%	86%
KNN	96%	70%	81%
Decision tree	91%	68%	78%

5.2 Result evaluation

As we can see in Table 2, XGBoost outperformed with an F1-score of 86%, Random forest applied in a SMOTE dataset scored slightly less with am F1-score of 85%. However, Logistic regression

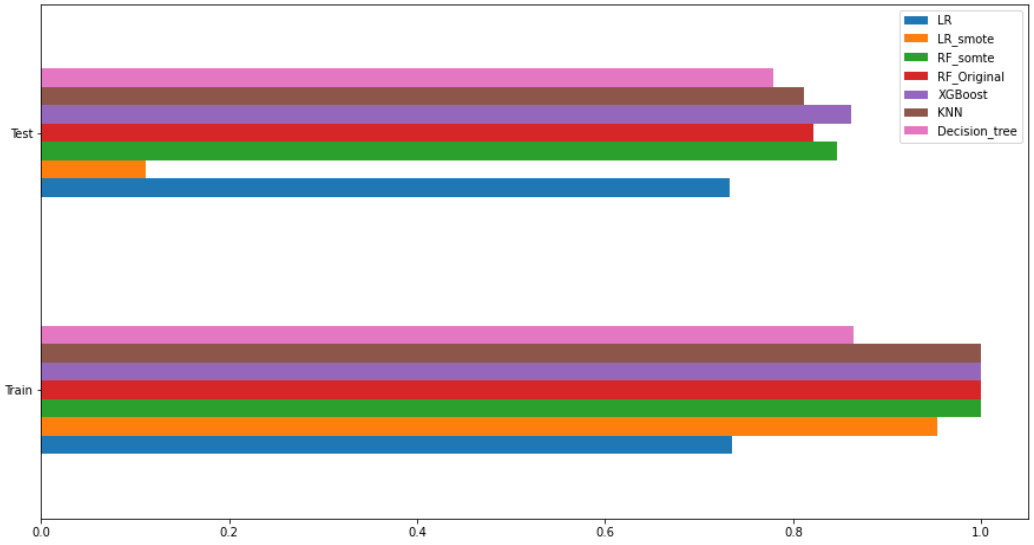

Fig. 7. Comparison of train and test F1-score evaluation

applied on SMOTE dataset performed poorly by 11% of F1-score only. When the class balance in the test dataset is different from the class balance in the training set, some machine learning does not perform well in that case. Here, we had smote dataset applied for the training set so it only increases minority samples in the training dataset, not on testing. K-nearest neighbor was performed with two different leaf sizes of 10 and 30 with kd-tree and brute force algorithms. Also, uniform and distance weights were considered. It resulted as kd-tree with 10 leaf size and distance weight were the best combination. It gave 96% precision, which is the best among all other classifiers. However, the recall value is the only 70%. As per the results, XGBoost is the best model to use for these types of classification problems because it is fast and reliable to use. Additionally, we do not need to handle imbalanced data separately as the model itself handles that with its hyperparameter. So, it results in less time-consuming with higher performance for our binary classification problem. Considering the F1-score for dataset training, most classifiers performed very well with the F1-score of 100% except Logistic regression and decision tree performed on original data.

Here to explain how the calculation is considered, I added the XGBoost confusion matrix shown in Figure9.

The value of precision for Fraud class will be $\frac{78}{78+5}=0.9397$.
The precision for Non Fraud class will be $\frac{56859}{56859+20}=0.9990$.
So, The classification precision for the Fraud is 93.97% and for Non-Fraud is 99.90%. Which is very good result for fraud detection technique.
Recall for Fraud class will be $\frac{78}{78+20}=0.7959$.
recall for Non-Fraud class will be $\frac{56859}{56859+5}=0.9999$.
We can compute F1-score by using precision and recall values. $2 \cdot \frac{0.9397 \cdot 0.7959}{0.9397+0.7959}=0.8579$.

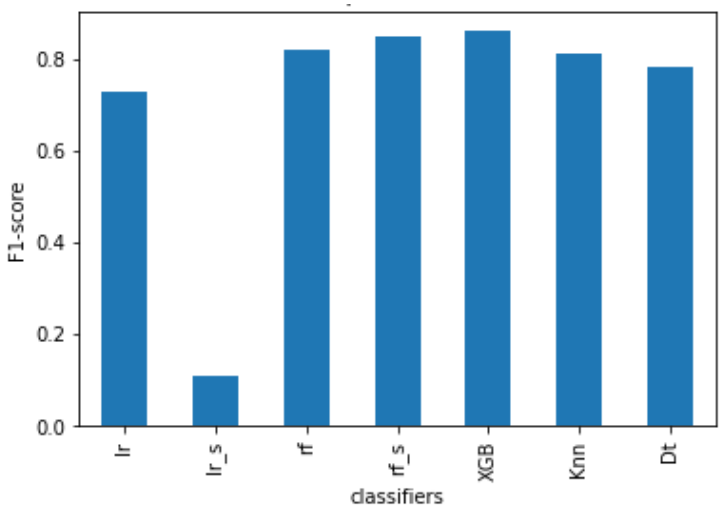

Fig. 8. Evaluation measures comparison

5.3 Receiver Operating Characteristic Curve

As per the precision section, I generated s confusion Matrix for all classifiers. The confusion matrix provides information about precision, Recall, True Positive Rate, and False positive rate. The efficiency of fraud detection classification at different values of threshold determined by Receiver Operating Characteristic(ROC) curve. The ROC curve can be determined by True Positive rate(TPR) and False Positive Rate(FPR). The TPR and FPR can be determined from confusion matrix. The formula for the TPR and FPR in Equation (7) and (8).

$$\text{True\_Positive\_Rate(TPR)} = \frac{\text{True\_Positive}}{\text{True\_Positive} + \text{False\_Negative}} \tag{7}$$

$$\text{False\_Positive\_Rate(FPR)} = \frac{\text{False\_Positive}}{\text{False\_Positive} + \text{True\_Negative}} \tag{8}$$

The values of TPR and FPR are required for the ROC curve. The TPR is required to set on the y-axis and the FPR is on the x-axis. The ROC curve determines the results of all classification thresholds and can help to get the best threshold value out of all possible values. The ROC curve is used for binary classification. The area under curve score(AUC score) measures the total area underneath the ROC curve. AUC provides a comprehensive measure of performance relative to all possible classification thresholds. Figure 10 shows the resulted AUC score of different classifiers. Here, I show the calculation of TPR and FPR values based on the confusion matrix shown in fig 9. The value of true_positive_rate will be $\frac{78}{78+20}$=0.7959 which is also called recall value as we calculated in the previous section. The value of the false_positive_rate will be $\frac{5}{5+56859}$=0.000.

The goal of the ROC curve is to maximize the TPR and minimize the FPR as much as possible to get the optimum threshold value. The Area Under the ROC Curve(AUC) measures a two-dimensional area underneath the entire ROC curve from (0,0) to (1,1). AUC has a range of values starting from 0 to at most 1. Any model with 100% wrong prediction results in an AUC of 0.0. One

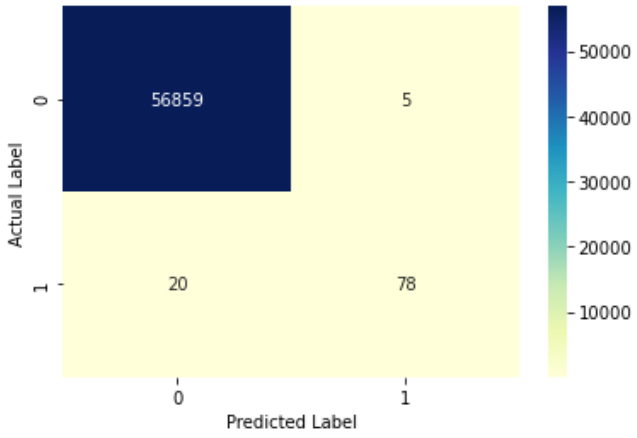

Fig. 9. Confusion matrix of XGBoost

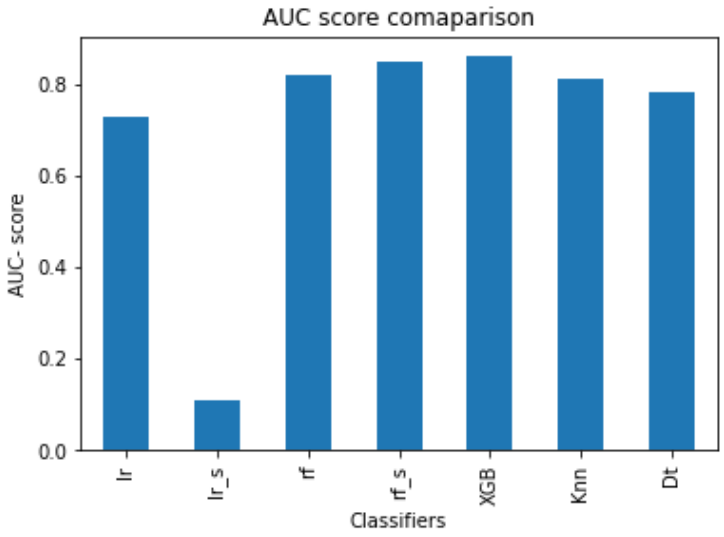

Fig. 10. AUC score comparison

with the 100% correct predictions has an AUC of 1.0. It has two benefits. AUC is scale-invariant. Meaning that It measures the predictions are ranked correctly, rather than their absolute values and It is classification-threshold-invariant means It measures the quality of the model's predictions irrespective of what classification threshold is chosen [6]. Based on my experiments, XGBoost gave the highest AUC score of 0.86 following random-forest classifier using smote with 0.85 of AUC score.

5.4 Recursive features elimination

Recursive Feature Elimination(RFE) is a popular feature selection algorithm. REF is popular because it is easy to configure and with effective feature selection in the dataset at training time which are more or most relevant to make a decision about the target variable. The first estimator is trained on the initial set of features and the importance of each feature is obtained through any specific attribute. Later features with the least important to make a decision about target variable are pruned from the current set of features.
In my experiments, the dataset contains all features which are transformed by PCA and only numeric values are present. Though there might be some features we can eliminate to get better or at least the same result. Feature elimination results in faster execution of classifiers and not to collect some unnecessary data while adding values in the dataset. Also, there might be some features that are decreasing the evaluation metrics score. So after applying REF, we can eliminate the least important features.
In my experiments, first I performed RFE to select 15 relevant features then I performed the same to select 25 features. The result of the elimination of that features is shown in Figure 11. By this technique, I could not get higher performance but for 25 features selection, I got the same result as XGBoost of F1-score 86% and AUC 0.897. The 25 selected features are Time, V1, V2, V4, V5, V7, V8, V9, V10, V11, V12, V13, V14, V15, V17, V18, V19, V20, V21, V22, V23, V24, V26, V27, V28.

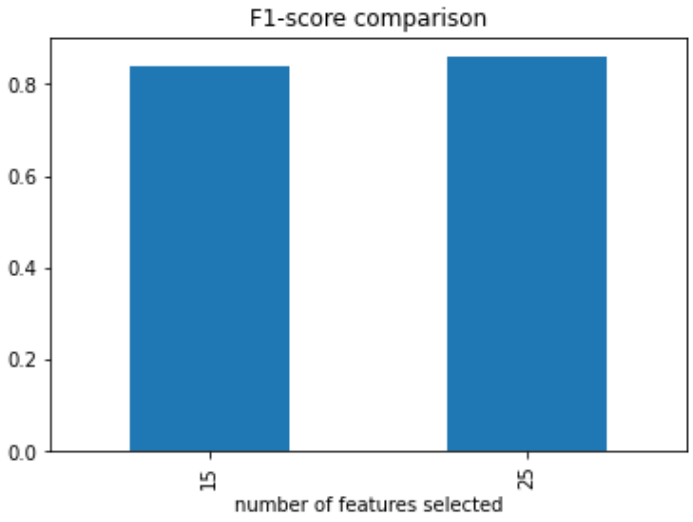

Fig. 11. Feature elimination result

6 Conclusion

In this Report, I processed the imbalanced dataset by applying SMOTE and After comparing different classifiers, I found out that the classifiers were performing better than before. However, a classifier like Logistic regression poorly performed on balanced dataset. Also, accuracy is misleading in the case of the large highly imbalanced dataset. So, it is better to use other evaluation parameters like Precision, Recall, and F1-score to come up with a good solution. I finally observed that random forest using smote dataset, Knn and XGBoost are the algorithms that gave better results. Although, Feature elimination techniques were not able to provide the best solution, I was able to eliminate 5 features with the same performance.

References

1. Dornadula, Vaishnavi Nath, and S. Geetha. "Credit card fraud detection using machine learning algorithms." Procedia Computer Science 165 (2019): 631-641.
2. Suraj Patil, Varsha Nemade, Piyush Kumar Soni,Predictive Modelling For Credit Card Fraud Detection Using Data Analytics,Procedia Computer Science,Volume 132,2018,Pages 385-395, ISSN 1877-0509.
3. Fang, Yong, Yunyun Zhang, and Cheng Huang. "Credit card fraud detection based on machine learning." Comput. Mater. Continua 1000.61 (2019)
4. Samaneh et al.[Zahra, Reza et al.], A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective (2016)
5. https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf
6. https://developers.google.com/machine-learning/crash-course
7. https://www.kaggle.com/mlg-ulb/creditcardfraud
8. https://nilsonreport.com/
9. https://machinelearningmastery.com/xgboost-for-imbalanced-classification/
10. https://scikit-learn.org/stable/

Acknowledgment

Special thanks to Chanakya Vivek Kapoor for his insights and observations on machine learning algorithms.