Credit Card Fraud Detection

Amara Nasir
Department of Computer science
NUCES,FAST
Lahore,Pakistan
l192426@lhr.nu.edu.pk

Ahmed Meer
Department of Computer science
NUCES,FAST
Lahore,Pakistan
l192463@lhr.nu.edu.pk

Haziqa Sajid
Department of Computer science
NUCES,FAST
Lahore,Pakistan
l202322@lhr.nu.edu.pk

***Abstract*—Credit card fraud has serious economic repercussions on the society. Every year billions of dollars are lost due to credit card fraud. The patterns of fraud have gradually evolved into more sophisticated and intricate manner. The technologies like smartphones, cloud computing, digital transactions have opened opportunities for new fraud vectors. This forces institutions to hunt for better and improved fraud detection systems. Recently, Machine learning techniques have been widely employed to address this problem.**

It is of utter importance to extract the appropriate features from dataset. Keeping track of customer's transactional behavior gives insight into model development. Anything that falls out of the spending behavior serves as a red flag and a potential candidate for fraudulent transaction.

We have done a comparative study by implementing machine learning algorithms and study their performance on credit card dataset.

Keywords—Fraud detection, AI, Machine Learning, Credit card, Xg-boost, SVM, Logistic Regression, Random Forest, Gradient Boosted Machines

1 Introduction

Fraudsters have evolved their strategies to evade detection. Over the course of past few years, the employment of machine learning techniques in credit card fraud detection has been an interesting domain. During constructing credit card fraud detection model, many factors have to be considered like skewedness of data, dimensionality of the search space, features obtained after preprocessing, and cost sensitivity.

Due to the cost sensitive nature of fraud detection problem, the cost incurred after a false negative is relatively different from false positive. Along with it, failing to detect a false negative and false positive can have its impacts on the system as well.

The rest of the paper is organized as follows. Section 2 gives the problem statement followed by Literature Review. Section 4 describes data set and section 5 gives methodology. Section 6 demonstrates the results by drawing conclusion in section 7.

2 Problem Statement

2.1 Lack of real world Datasets

The research in credit card fraud detection is tremendously limited because lack of real world datasets because of security concerns. Even when datasets are made public, their results are masked, which ultimately makes it difficult to make a fair assessment of our work and results. However, few researchers have worked on synthetic datasets to reproduce the real world data.

The other type of obstacle that hinder the development of fraud detection techniques is that there are laws in different countries forbidding data from leaving their borders like the regulations imposed by GDPR i.e. General Data Protection Regulation of EU. That is why large-scale datasets are unavailable for research and cannot be published for a wider community.

2.2 Metrics for Fraud detection

The problem of fraud detection encompasses the determination of potential fraud transactions over non-fraudulent transactions. Our job is to detect common attributes in the fraud vector. There is a pattern of events followed by criminals. The goal is to design model in a way so that it detects common attributes of fraud vectors.

F-score stands out as the metric for assessing fraudulent transactions. It correctly indicates the number of correctly identifies fraudulent transactions over the missed transactions. However, there is also a limitation to it. F-score is biased in case of class imbalance. Therefore, as comparison metric it does not serve the purpose very well in this case.

Confusion matrix is another classification metric used for performance measurement. It classifies the whole problem set into true positive, true negative, false positive, and false negative. Accuracy can be computed with confusion matrix.

2.3 Handling Class imbalance

The credit card fraud detection is a huge class imbalance problem. Because the number of fraud transactions compared to the number of genuine transactions is very little in real world transactional datasets. It generally has an adverse effect on the performance of the model applied. This problem is usually addressed by incorporating dummy data into the dataset. Along with that, Smote is a technique used widely to balance out class imbalance problem.

2.4 Evolving industry technologies

The rapid increase in fraudulent transactions has its merit to the exponential growth in technologies like smartphones, digital payments, e-commerce, e-wallets, data breaches, cloud computing services, micro payments. Crime has migrated towards these technologies very fast. This age of technology driven lifestyle has brought a major shift in the fraud vectors. More sophisticated fraud vectors are emerging day by day.

2.5 Delay in Fraud detection

Another factor that serves as a major hinderance in fraud detection is the latency in the detection of fraudulent transactions. The latency can span over the time period of

days and even months. This means that the data that is to be fed into the training algorithm is already outdated.

3 Literature Review

The methods along with their descriptions employed over the course of many years for fraud detection are summarized in the table below.

Method	Description
Expert systems/ Decision Tree	Known as knowledge based systems. Includes rules, decision trees, case based reasoning (CBR)
Supervised Neural Networks	Deducing a function from data with inputs and auxiliary labelled outputs
Unsupervised Neural Networks & Clustering	Data with similar properties is placed nearby to make meaningful clusters. Unsupervised neural networks has its importance in detecting anomalies in the transactional data.
Bayesian Network	A probabilistic model is generated by deducing conditional dependencies from the data.
Evolutionary Algorithms	Goal is to find optimal functions to classify fraud using a heuristic algorithm. This algorithm copies the behavior of biological natural selection.
Support Vector Machine (SVM)	Creating classifiers from the training data with the use of inputs and relevant outputs. It creates separating hyperplanes between the two classes.
Eclectic & Hybrid	Includes a range of novel methods

In the following sections, overview of the methods along with prominent contributions is presented.

3.1 Expert Systems/ Decision tree

A. Rule Based

The most prominent contribution is from Correia et al. (2015) [1]. They created a set of 14 rules that hunt for specific patterns to indicate fraud vectors. The proposed methodology is implemented by a software tool named PROTON. An event processing network was created which included event processing agents for every rule. The dataset used comprised a real world dataset having 5.6 billion transactions. The results generated by this methodology were very promising. 80% of fraud transactions were identified with an FPR (false positive ratio) value of just 0.02%.

The contribution by Correia et al gives the lowest value for Alert D. it one of the best research studies conducted for fraud transaction detection.

B. Decision tree/ Random Forest

Fadaei Noghani and Moattar at al. (2017) [2] used firstly feature section and afterwards random forest decision tree approach. The three measures used are 1) Chi squared, 2) ReliefF to ascertain volubility, 3) Information gain.

The subset dataset is created by using features generated from each method having highest rank. The feature subset is classified through C4.5 decision tree and the resultant accuracy is monitored. If the feature tends to minimize accuracy then this one is left out and next highest ranking feature is selected. This whole process results in the removal of low ranked features from the set.

From the resultant dataset, a random forest is created. A public dataset was used for this paper. It included almost 29,104 transactions. This paper demonstrated a promising F-score of 0.9996 with a total of 27 trees in the forest. FPR ratio is not provided in the paper which does not render it very useful for real world data.

3.2 Supervised Neural Networks

A. Multi-Layer Pereceptron (MLP)

Rymann Tubb et al. (2016) [3] SOAR extraction method. SOAR stands for Sparse Oracle-based adaptive rule. SOAR is utilized to extricate knowledge through association rules from the neural networks. This paper has utilized a real world dataset for the detection of fraudulent transactions. The dataset has 171 million transactions. The aim of the paper is to demonstrate that the fraud rules can be extricated from a black box classifier. 11 rules with high confidence were extracted from the output of neural network. These extracted rules were able to differentiate most of the non-fraudulent transactions.

B. Convolutional Neural Network

Fu et al. (2016) [4] presented their work based on convolutional neural networks. They started with two premises: 1) before training, important features had to be derived. 2) Fraudulent transactions were detected through convolutional neural network. This paper highlights the emphasis on pre-processing of data. This paper uses trading entropy. Trading entropy is the money spent on each merchant item type like food, appliances, and goods. Then using this, trading entropy fraud transactions are detected. Because the transaction with a number significantly different from the previous recorded transactions stands out as a flag.

This paper has utilized data from a Chinese bank containing 260 million transactions. Sparse rate concept is also used which states the usage behavior of credit card in China.

KNN is used for the clustering of fraud transactions. The non-fraud transactions were under sampled. Experiment was performed with a variety of class balances. The records were transformed into a pattern suitable for convolutional neural network i.e. matrix. The rows contain trading entropy and the columns contain the statistical aggregation of fields over a range of different time periods. The train set contained data for 11 month while the test set constituted one month data.

The results deduced from the efforts were compared to the other algorithms like SVM, Random forest decision tree, and multilayer perceptron. The proposed CNN model outperformed the other approaches in terms of F-score.

3.3 Unsupervised Neural Networks and Clustering

Yigit Kultur et al. (2017) [5] proposed a novel cardholder behavior model (CBM) for the detection of fraud credit card transactions. Clustering is applied to segregate fraud transactions from legal ones. The CBM decides on the basis of past spending behavior whether the new transaction is legit or not. The clustering algorithms chosen this paper are cobweb, DBSCAN, and expectation maximization.

For each cardholder two different profiles are created, one for holidays and other for regular dates. The dataset of 105 card holders constituting 150,957 transactions is used. It is taken from Turkish bank.

The training dataset comprises only 150,277 transactions while the test set was fed only 767 transactions.

The system accesses the performance on the basis of single card vs multi card. Single card and multi card CBM is built to support the transactions of users having multiple credit cards. The transactions with unusual spending behaviors are highlighted.

Another approach employed by the writers is Holiday season spending CBM. It encompasses the transactional behavior in holidays and deals with raising alarm in fraudulent transactions within the holiday season. Dataset from a leading bank in turkey is used in this paper. Single card CBM model detected more fraudulent transactions than multi card CBM. So the Single card CBM has more sensitivity towards outliers. But the precision measures and accuracy of multi-card CBM is better than single card. The Holiday season CBM was able to identify frauds with higher sensitivity than other approaches of CBM.

The work presented by the authors holds its significance because it encompasses the real world behavior of customers.

3.4 SVM

Zanin et al (2018) [6] proposed a method named Parenclictic network analysis. The new features are derived from fields combined with correlation between entities. One network is created as a result which used both the classes. The network proposed in the paper is utilized to create 7 metrics for every transaction. The derived metrics serve as an input to the multiple layer perceptron.

The dataset used in this paper has 180 million transactions. This is a Spanish bank data of 7 million cardholders across a time period of 1 year.

The best results were driven where original fields are used in conjunction with derived network features.

4 DATASET

IEEE Computational Intelligence Society (IEEE-CIS) in partnership with Vesta Corporation provides this dataset for a kaggle competition in order to implement and highly accurate fraud detection system. IEEE-CIS works on a variety of AI and ML problems including fraud detection and implementing algorithms like deep neural networks, evolutionary algorithms etc. Vesta Corporation is a world's leading payment service company who seek the best solutions for fraud detection. The selected dataset comes from Vesta's real-world e-commerce transactions and contain a wide range of features from device type to product features.

4.1 Data Statistics

After acquiring the dataset, the next step was to perform a high-level discovery of contained information. The dataset is broken into two files, identity and transaction, which are joined by an attribute TransactionID. Not all transactions have corresponding identity information. The train and test set were provided separately and the task was to predict the fraudulent transactions in the test set.
Following table shows some important data statistics.

Data Statistics	**Information**
Size	1.25 GB (uncompressed)
Total rows in training file (transactions file)	590540
Total rows in training file (identity file)	144233
Total rows in testing file (transactions file)	506691
Total rows in testing file (identity file)	141907
Total attributes (transactions file)	394
Total attributes (identity file)	41
Attribute type	Mixed (continuous, categorical etc.)
Prediction variable type	Binary
Data contain null values	Yes
Requires pre-processing	Yes
Columns details available	Partially

5 PROPOSED METHODOLOGY

This section discusses in detail the steps undertaken by us from exploratory data analysis (EDA) until model training. We created a machine learning pipeline by starting with data wrangling and performed EDA in detail on the raw data for better statistical analysis of data. We also had to perform pre-processing in detail because the data suffered from majority of the problems described in section 1 like class imbalance, null values. Then the next step undertaken is feature engineering. PCA was also performed for better segregation of principal components. It helped us in determining the features that had most influence on the desired output. After data was prepared, it was fed to machine learning models like SVM, Random Forest, Xg boost etc. A comparative analysis is done to better analyze which algorithm outstands in performance and accuracy on our dataset.

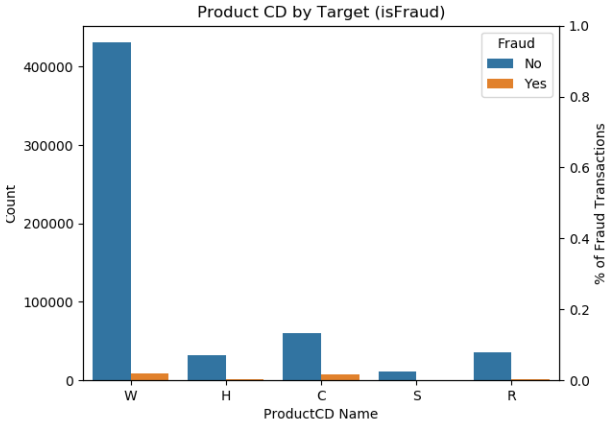
5.1 Exploratory Data Analysis

Exploratory data analysis (EDA) is an approach that encompasses the process of discover patterns, spot hidden anomalies, verify our assumptions with the aid of summary statistics and illustrate data with data visualization techniques. EDA is widely used to draw meaningful insights from the datasets. EDA techniques are employed because it

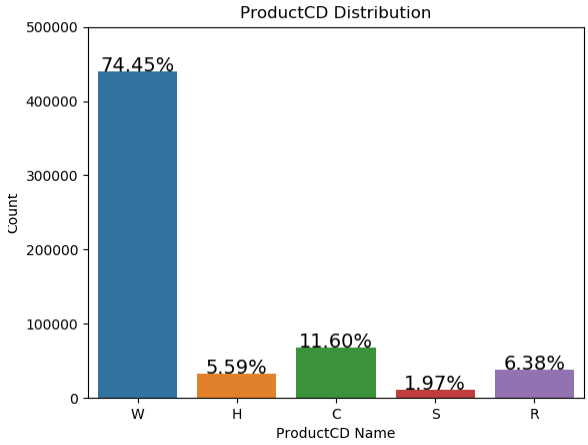
is not intuitive to look at the columns of the dataset and determine the underlying characteristics.

EDA involves a lot steps. The major steps are described here. The first step involves looking for null or missing values in side columns. Second step is to encode categorical features. Numerical variables might have to be normalized and scaled. Another step is the removal of duplicate values inside columns.

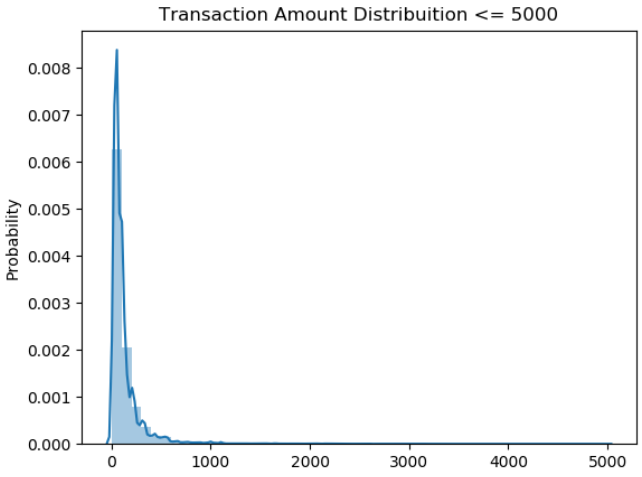
Below is a graph representing the categorical attribute *ProductCD* visualized on the basis of target values i.e. fraud vs non-fraud. The attribute *ProductCD* represents the product code. As we can clearly see, the data suffers from class imbalance problem among target (Fraud, NoFraud) at least in some product codes like 'W'. All the product codes in general, have minimal fraudulent transactions values. The class imbalance problem is dealt with detail in coming sections.



We can also visualize the same product codes without reference to target variable. They are visualized on the basis of their relevant percentages in the dataset. Following graph illustrates it well.

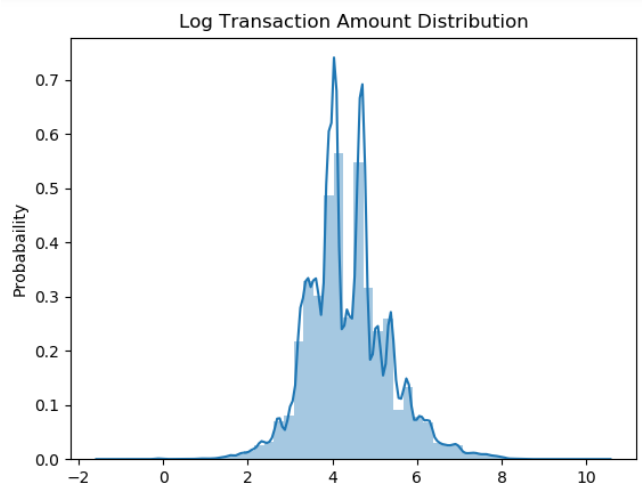


One of the most important features in the dataset is the *TransactionAmt* which refers to the amount of transaction that record is referring to. This is an important feature because the transaction amount has a potential to play an important role in determining whether the transaction is fraudulent or non-fraudulent. We have analyzed this attribute in detail. For example, the mean, max and min values were analyzed and it was found that most of the transactions are over 5000 in the dataset. Hence, we wanted to see how are the transactions that are below 5000 distributed among the dataset in general. Following plot shows the distribution of transaction amounts only for the records where the amount is less than or equal to 5000.



In the above plot, we can clearly see that the records whose transaction amounts are under 5000 show a unique kind of behavior in terms of their data distribution. There is a strong positive skew in the dataset. Most of these records have values ranging from zero to less than a thousand.

Another analysis made on the transaction amounts was to check the overall distribution of the dataset in terms of magnitudes of transaction amounts. For this purpose, we generated a log transformed probability distribution plot of the dataset. This means, the transaction amounts are log transformed here. Following plot shows this distribution.

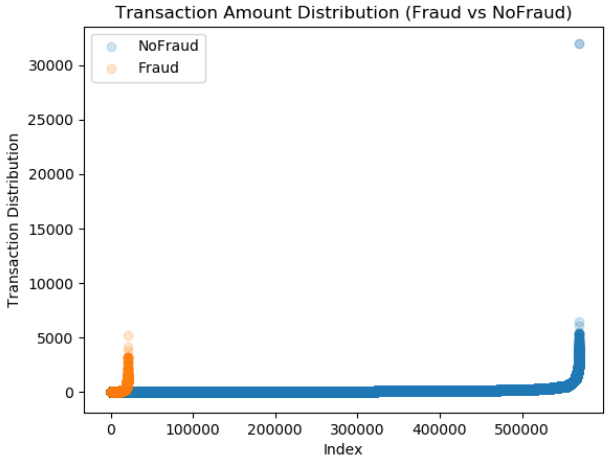


We can clearly see that the log transformed transactoin amount plot shows that the dataset follows a normal distribution when it comes to transaction amounts in general. This is a good factor considering that msot of them training algorithms we are going to use in later section can work well on datasets that follow gaussian distributions on one or more features. From the above plot, we can see that most of the dataset records have transaction amounts (in log

transformed space) within the range of three to five. We can also consider the exponential of these values as the mean transaction amount that we should expect most of the times in these transactions.

Another important feature or column that needs to be analyzed well during the exploratory data analysis phase is the target variable itself. The target variable is called *isFraud* in the dataset. It is a binary categorical variable which can take values zero and one. It is important to note here that the target variable in our dataset is not a continous variable that tells the severity or probability of a transaction to be fraudulent. Instead, the target variable is distinctly telling us which transactions shall we consider fraduelent and which we should not. Therefore, we did a bit of analysis of dataset based on this target variable as well.

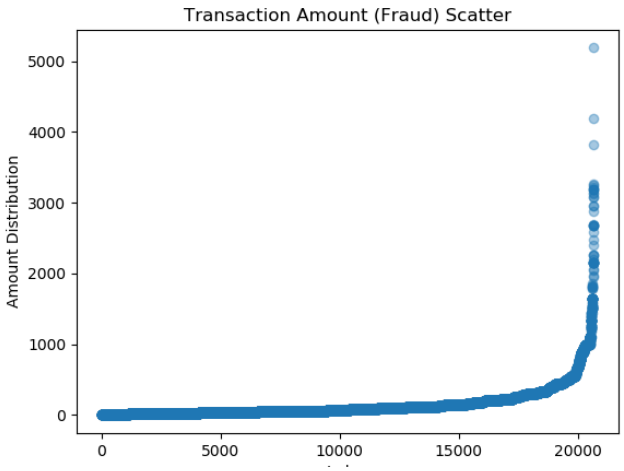
Following plot shows the scatter diagram of all the records in the dataset in terms of fradulent and non-fraudulent transactions.



Above plot clearly shows the distribution of data records in terms of fraudulent and non-fraudulent transactions. Blue points in the scatter plot represent the non-fraudulent transactions while dark orange points in the scatter plot represent the fraudulent transactions. As it was previously explained, almost all the fraud detection datasets face the problem of class imbalance with great severity. From the above plot we can see that our dataset has a similar situation as well. We can clearly see that there are outlier samples in both fraudulent as well as non-fraudulent records.

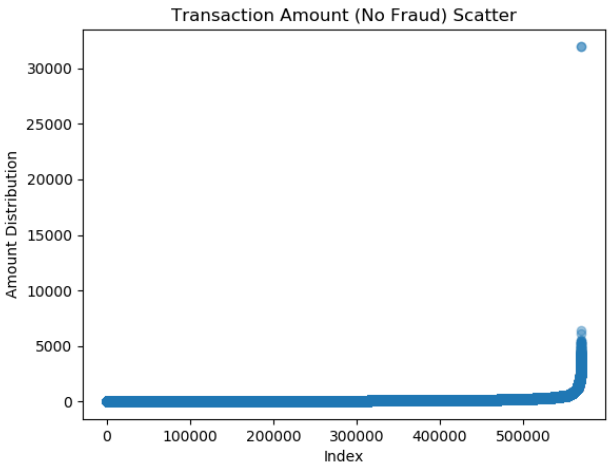
To further elaborate and analyze this, we then plotted both fraudulent and non-fraudulent points in separate scatter plots so that their outliers can become more visibly prominent. There are many reasons for performing detailed analysis by plotting both types of records separately. For example, one advantage of plotting them in this manner is that we can clearly identify the outlier records and inspect them visually. In this way, we can see and determine the rough threshold value on the y axis of the plot to get a cut off value when we are planning to remove these outliers. When we think of removing the outliers, it is also important to note that the we should not be removing outliers that we generally see in the combined plot of scattered points. Reason behind this is that a record that might look like an outlier in combined plot might not actually be an outlier and it could be a fraudulent transaction instead, which we will never want to remove from our data since this is what we are trying to predict in this problem statement. This is another reason that we plot both kind of transactions separately.

Following plot shows the scatter plot of fraudulent transactions.



In the above plot, we can clearly see that among roughly twenty thousand records of fraudulent transactions, there are very few but still outlier records present. These are the records whose distribution lies roughly above three thousand on the y axis.

Following plot shows the scatter plot of non-fraudluent transacions.



Again, in the above plot, we can cleary spot the outliers among these non-fraudulent transactions. The non-fraudulebnt transactions roughly count up to five hundred thousand in number and very few of them are outliers but they are still present. We can spot them roughly above twenty five thousand value on the y axis in above plot. These values are then used to define a threshold when

removing outliers from both fraudulent and non-frauduelnt transactions. Before using this dataset for modelling purpose, we remove these outliers based on the strategy and threshold concept as we just explained.

5.2 Data Pre-Processing and Data Wrangling

In the field of data science, most of the time is spent on cleaning and formatting the data. Data preprocessing and data wrangling is used to transform the raw data set into a clean dataset to make it usable for data analyses or visualization. In real world, data is collected from different sources which are often messy, noisy, inconsistent, missing or ill-formatted. In order to achieve better results for our machine learning models, data has to be in proper format like there must not be null/nan values, missing values or garbage. Data should be clean and properly formatted before giving it to any model. The basic purpose of data preprocessing is to bring that data in such format that it can be directly used in machine learning and deep learning models.

There are total 394 attributes/columns in which 21 attributes are categorical and 373 attributes are numerical. List of all categorical attributes is as follows:

- Product CD
- Email domain
- Card1 – Card6
- Addr1, Addr2
- P_email domain
- R_email domain
- M1 – M9

A. Handle Missing Values

Missing value problem was handled in three following steps.

First of all, dataset was checked if it had any missing or null/nan values and removed all the columns which had more than 50% missing values. The rationale behind this is that if some attribute does not have ay valid value or data or more then 50% of the entire dataset, it will be almost useless to impute that attribute regardless of the fact whichever imputing technique is used. Imputation techniques tend to fill in the missing values using the patterns that exist in the already existing records, which do not have missing values in that attribute. But in case of attributes that are filled for only 50% of the dataset, it will not depict a true picture if we use them to impute for the rest of 50% of dataset. Hence, we simply remove these attributes from our data. It should also be noted here that by dropping these features, we are not losing any important underlying information or pattern from our data. This is because the dataset is rich when it comes to attributes.

After performing this step, only 220 columns were left. This is still a good count in terms of number of features that we can use in any machine learning model. To proceed with data imputation, our next step was to check for missing values in remaining data columns.

For the remaining columns/attributes, we imputed each attribute with a different strategy depending upon the data type of each attribute. These were broadly dealt in tow main categories, which are categorical and numerical attributes.

The categorical attributes were first checked for distinct values and all missing values were spotted in their columns. These values were then replaced with most occurring/mode values against each categorical attribute.

Then, our next step was to deal with numeric attributes and impute their values as well. When it comes to numeric attributes, it is important to note that these attributes can be imputed in much more brilliant ways than we impute simple categorical attributes. In case of categorical attributes, we usually do not have a lot of flexibility, so most of the times, we simply use the mode value to replace. In some cases, median values can also be used to replace them.

For numeric attributes, the simplest approach would be to impute them using mean just like we used mode or median in case of categorical attributes. But there are more sophisticated methods that can be utilized to impute such numeric features. One of them is to use a regression model that trains on rest of the dataset where this attribute had valid values and then predict for the records where the values of this feature are missing. We also used a similar strategy with a small change that we explain as follows.

If we simply use a regression model and train it on the records that have valid values for say a feature A, and then use this model to predict values of feature A for records where it has missing values, this would work as fine as it seems. But, there is a problem that arises when we have more than one numeric features that have missing values in them. Suppose there is another feature B, and it also has some missing values that need to be imputed. Now, if we use the same strategy initially for feature A as we just explained, we will actually not be able to do that. The reason for this is that when we try to train the model on records that have valid values for feature A, that training set might have some records that have missing values for feature B and hence the model will throw error during training. That means, we need to find some strategy such that we fill up all other features before we impute values for any feature.

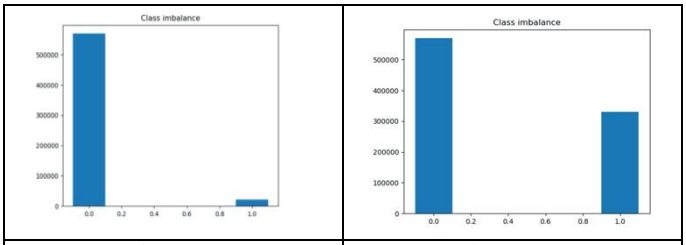
To deal with this problem, we came up with a method that we are going to explain now. We firstly impute all the missing values for all numeric features using any randomly seen value for that feature. While doing so, we remember the indices or row numbers against each feature such that we have a record of which rows were imputed for which feature and we store them in some variable say a python dictionary. Once we are done with all features, now we repeat to impute all the features again, but this time using regression. Now the regression model will not throw any training time error since it will never see any missing values for any feature while training. Hence, we repeat this one by one for all features and eventually all the features are imputed using regression model. This is an intelligent and robust technique to impute missing values for numeric features in case when there are more than one numeric features that have missing values.

So after performing all these steps, missing values were finally handled in dataset. Moreover, no missing values were left anymore.

B. *Class Imbalance*

This dataset was highly imbalanced as there were around 569877 instances for class 0 and only 20663 instances for class 1. In order to handle the problem of class imbalance, SMOTE (Synthetic Minority Over Sampling Technique) was applied. SMOTE is most commonly used technique to oversample the data in minority class in order to handle class imbalance problems. It works by drawing a line between the examples in the feature space, and then by putting new samples along that line, new synthetic samples are created. It actually selects a random point from minority class and by using k-nearest neighbor algorithm, nearest examples of minority class are selected. Then, it draws a line between these examples and by putting synthetic points along these lines, new samples of minority class are created.

Therefore, after applying SMOTE, data was balanced and class imbalance was resolved eventually. There were around 569877 instances for class 0 and 330608 instances for class 1 then as shown in following figures.



a) Before SMOTE b) After SMOTE

C. *One Hot Encoding*

Many machine learning algorithms do not work on categorical data and they need the input variables/features to be in numeric form. This means the categorical data must be converted to numerical data. One of the most popular ways to do this is One Hot Encoding. In one hot encoding, binary value is added against each unique value of attribute/features. It creates one binary variable against each category and which category value is present that becomes '1' and the other one becomes '0'.

One hot encoding was also applied to categorical columns and total number of columns were increased from 220 to 294 after applying one hot encoding. Now all the attributes are in numeric form. It should be noted that after applying one-hot encoding on categorical columns, the original columns were dropped, as they were not needed anymore.

D. *Data Normalization/Standardization*

Data normalization is the process of changing the values of numeric columns to a common scale without affecting the differences in the original range of values. Normalization is usually required when attributes have different ranges of values. Then it is a good practice to bring data down on same scale for machine learning algorithms. For example, if one of the variables is in the range of 1000's and the other variable is in the range of 0.1 then first variable will dominate the other variable. Hence, in order to reduce the difference, values of range of 1000 are converted into 0 to 1 range. Therefore, when the model is sensitive to magnitude and have different scale then it is a good practice to normalize data first.

In our data set, because of large dimensions of the data, we had to apply PCA, which has been explained in next section. PCA is also affected by scale so we had to normalize the data before applying PCA.

E. *Dimensionality Reduction*

Dimensionality reduction is the process of transforming high dimensional data into low dimensional data by obtaining a set of principle variables. It reduces the dimensions of the data such that it retains some meaningful properties of the original data. Because the higher the input variables/features of the data, the harder it becomes for machine learning and deep learning models to learn and work on it. This is where the dimensionality reduction algorithms have to play their role. PCA is one of the most famous dimensionality reduction algorithms that works on the condition that data in a higher dimensional space is mapped to lower dimensional space by keeping maximum variance. In addition, when data is reduced, then storage space is reduced. And it also reduces computation time and redundant features in data.

In dimensionality reduction part, first of all transaction id was dropped as it had no role in model development. Therefore, after removing we had total 293 columns. Next step was to apply PCA in order to reduce dimensions of the data.

PCA was firstly applied with 50 components and the results showed that maximum variance was reduced after four or five components. So again, it was applied with 10 components and it was enough to capture data variability. Therefore, after applying PCA, only ten attributes were left and data was ready to feed into any machine learning model. Therefore, this is the completion of pre-processing part and now, the dataset was completely transformed to be trained for any machine learning model.

5.3 Data Modelling

After performing all preprocessing and feature engineering steps on the dataset, the last and most important step is to train a supervised learning algorithm on the processed dataset. There are several algorithms in the literature that researchers and professionals have used to build fraud detection systems.

A. *Data Split*

Data was split into train, validation and test sets in 60%, 20% and 20% ratio respectively. There are many strategies for effective data splits. For example, some researchers use small dataset and apply only k-fold cross validation by splitting the dataset into k-folds. However, we have used this strategy of train, validation and test sets. The purpose of this is to train the machine learning algorithm on train set, and then evaluate its performance on the validation set. Based on the metric observed on the validation set, the hyper-parameters of the algorithm are adjusted and another model is trained and evaluated on the validation step. This keeps happening in multiple iterations until we observe a decent level of accuracy on the validation set. Once that

stage is reached, the hyper-parameters are considered to be final. After that, inference is made on the test set and metric are again calculated on this set now. The purpose of re-evaluating the algorithm on test set is to make sure the model's accuracy or other evaluation metrics reported are totally unbiased, since the algorithm has never seen the test set before in any training cycle. Hence, we use this strategy of three splits of data.

B. Machine Learning Algorithms

Generally, researchers have used many algorithms for the problem of fraud detection or other anomaly detection problems. These algorithms broadly include linear and non-linear models like Support Vector Machines, Logistic Regression, Random Forests etc. as well as Artificial Neural Networks and some of the advanced deep learning architectures like auto-encoders that are specifically setup and tuned for their dataset patterns.

While choosing between multiple options of training algorithms, we have kept a criteria that we are aiming to produce generic solution that does not only work well with this dataset, instead it can work with almost all the similar kind of fraud detection or even more generic anomaly detection datasets. Opting for highly tuned (dataset based) deep learning architectures are a good option when one is interested to develop a stand-alone solution for a particular client or customer. However, in case one wants to produce a product that can be deployed for various customers that follow a certain criteria in terms of their transactions and sales, such generic solutions prove to be more useful. Another advantage of using these algorithms is that they do not required very high training times as well as dataset specific tuning. These can be easily deployed to other datasets by minor tweaking in the model hyper-parameters.

Considering above criteria, we chose following models and trained each one of them on our data. All of these models were fine-tuned in multiple runs and eventually the best performing among those runs were finally taken.

- Support Vector Machines
- Logistic Regression
- Random Forests
- Xtreme Gradient Boosted (XGBoost)
- Gradient Boosted Machines

C. Hyperparameter Tuning

Hyperparameter tuning is one of the most important steps in any supervised learning algorithm. It refers to the process of adjusting and fine-tuning the values of model parameters that need to be set beforehand. That means, before the algorithm looks at the training data, these parameters are required to be set. These are mostly related to assumptions about dataset and some other model behaviors like how aggressively the algorithm should move in the direction of negative gradient in order to reach the global extremum of the function that it is trying to estimate form the provided dataset. Although these hyper-parameters are required to set before the training process, still there are methods in which we can iteratively adjust their value by looking at model's performance with each set of hyper-parameters.

In our case, we have used sci-kit learn's method called *GridSearchCV* which performs the same task of tuning hyoer-parameter values. We will explain a little bit about how this method works as follows.

The method takes an estimator as parameter. The estimator needs to be one of the classifier or a supervised learning method. In addition to this, a python dictionary is provided as another parameter. This dictionary contains key-value pairs of *hyper-parameter:[range of possible values]*. There are some other parameters of this method as well like the value of number of folds when using cross validation. This method implements a cross validation technique, which we explain in the next heading. When these required parameters are provided to the method, finally its *fit* function is called. This puts the algorithm in running phase. What this method does is, that it creates a grid of hyper-parameters based on the ranges of individual hyper-parameters of estimator provided and runs a copy of estimator on each possible combination that lie within the ranges of hyper-parameters you provided. It further uses cross validation technique to evaluate the performance of each estimator. You can run these training jobs in parallel by setting the method's *n_jobs* parameter to the number of virtual cpu cores you want it to use in parallel.

Once the methods is completely run, it then provides the best estimator found as an attribute. It also provides the metrics calculated by the best estimator on validation set like accuracy etc.

D. K-fold Cross Validation

Cross validation is one the most efficient techniques used to evaluate the performance of a trained model and adjusting its hyper-parameters accordingly. It takes a parameter for total number of folds. It then splits the entire dataset into folds number of sections. It keeps one of those sets for testing purpose, and uses the rest of sets to train the model. When the training is completed, it generates evaluation metrics on the hold out test set. The same procedure is repeated each time a separate set is kept as test set. The data is randomly shuffled before generating these sets. The number of folds chosen is called *k*.

In our case, we have used 5-fold cross validation to evaluate the performance of model. In this approach, data is randomly split into training and test set. After that, model training is performed on training set and test data set is used for validation. Firstly, model is fit on k-1 folds, and then model is validated using the remaining kth fold and score is noted down. Then, this process is repeated for k times and an average of all k scores is calculated and used as an overall performance measure of the model.

5.4 Evaluation

As explained in the previous section, we have used *GridSearchCV* method from python's sci-kit learn module to train our model and perform its hyper-parameter tuning. This was done for all the machine learning models that we had planned to use for this dataset. Once the process of grid search and hyper-parameter tuning was completed on all the

algorithms, the next step was to perform evaluation of these algorithms in order to see how well they perform on the validation and test datasets and hence find the better performing algorithm in general.

Usually there are many evaluation metrics that can be used to evaluate the performance of any machine learning algorithm. These metric differ for supervised and unsupervised algorithms, classification and regression algorithms, as well as for traditional and deep learning architectures. In our case, our problem was categorized as a supervised learning problem. Then within that, it belongs to the class of classification problems. Classification problems are the ones where the target variable that needs to be predicted has unique or distinct values instead of continuous or real-valued numbers. If we go bit further into detail, we can say that our problem is a *Binary Classification Problem*. There are different evaluation metrics for binary and multi-class classification problems. Among those available for the binary classification problems, we have used the ones which are most widely used evaluate the performance of binary machine learning algorithms. These evaluation metrics are listed below.

- Accuracy score
- Precision
- Recall
- F1-score
- ROC AUC score
- Average Precision score
- ROC Curve
- Confusion Matrix

We have calculated above metrics on all five machine learning algorithms that we have trained on our dataset. Following part of the report explains them in detail along with their visuals.

A. Accuracy

Accuracy simply means the fraction of samples that were correctly classified among total number of samples that were used in inference. Following table shows the accuracies reported by different models on unseen test sets.

Model	Accuracy
SVM	60%
Logistic Regression	60%
Random Forest	70%
Xgboost	72%
Gradient Boosted	71%

B. Precision

While accuracy tells you the overall fraction of correctly classified samples, it does not consider any special criteria to distinguish between accuracy of positive class samples that were correctly predicted and negative class samples that were correctly predicted. To account for that, precision comes into play. Precision refers to the number of true positives among the total number of true positives and false positives. That means the fraction of correct classifications among all the positively predicted samples. Hence, it focusses more on the sample predicted as class 1 (i.e. Fraud). It is also called the positive predictive rate. Following table shows the precision values of all models for the positive class i.e. Fraud.

Model	Precision (Fraud)
SVM	57%
Logistic Regression	57%
Random Forest	74%
Xgboost	72%
Gradient Boosted	72%

Following table shows the precision scores of all models for the negative class i.e. NoFraud.

Model	Precision (NoFraud)
SVM	69%
Logistic Regression	66%
Random Forest	67%
Xgboost	71%
Gradient Boosted	69%

C. Recall

Recall refers to the number of true positives among the number of true positives and false negatives. It is also called the true positive rate. Following table shows the recall values for positive class i.e. Fraud

Model	Recall (Fraud)
SVM	83%
Logistic Regression	79%
Random Forest	62%
Xgboost	70%
Gradient Boosted	66%

Following table shows the recall scores of all models for the negative class i.e. NoFraud.

Model	Recall (NoFraud)
SVM	38%
Logistic Regression	41%
Random Forest	79%
Xgboost	74%
Gradient Boosted	75%

D. F1-score

F1-score shows the weighted average of precision and recall. The best value of f1-score is 1 and worst value is 0. Following table shows the f1-scores of positive class i.e. Fraud.

Model	F1-score (Fraud)
SVM	68%
Logistic Regression	66%
Random Forest	67%
Xgboost	71%
Gradient Boosted	69%

Following table shows the f1-scores of all models for the negative class i.e. NoFraud.

Model	F1-score (NoFraud)
SVM	49%
Logistic Regression	50%
Random Forest	73%
Xgboost	72%
Gradient Boosted	72%

E. ROC AUC Score

This score calculates the area under the Receiver Operating Curve (ROC) from prediction scores. It should be noted that this score does not utilized the actual predicted labels, instead it uses the prediction scores. Prediction scores refer to the predicted probabilities of each sample of being the positive class. Following table shows the ROC AUC scores of all models.

Model	ROC AUC
SVM	69%
Logistic Regression	65%
Random Forest	76%
Xgboost	79%
Gradient Boosted	76%

F. Average Precision Score

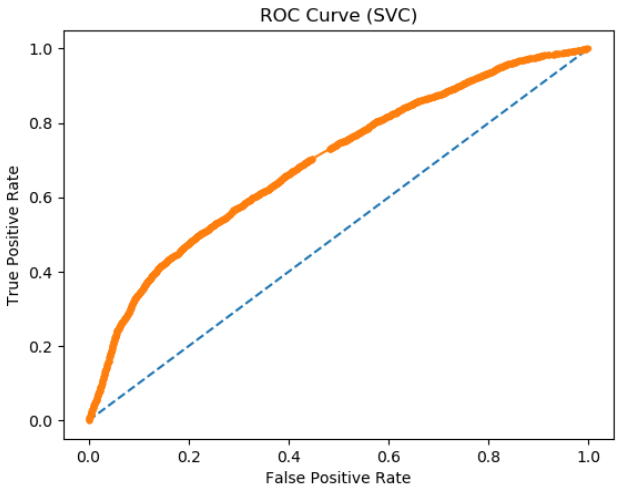
This score summarizes the precision-recall curve as the weighted mean of precision achieved at each threshold, with the increase in recall form the previous threshold used as the weight. Following table shows the average precision scores for all models. We can clearly see in the table below that the highest value of Average Precision Score is exhibited by Xgboost which is 79% where as Random Forest has the second highest value which is 76%. These values can be see in the table below.

Model	Average Precision
SVM	68%
Logistic Regression	63%
Random Forest	76%
Xgboost	79%
Gradient Boosted	75%

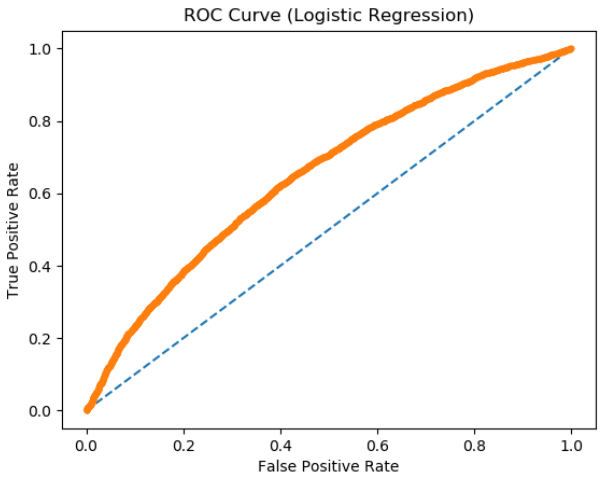
G. ROC Curve

Receiver Operating Characteristic (ROC) curve is a graphical plot that shows the diagnostic ability of a binary classifier as its discriminant threshold is varied. It is one of the most important evaluation metrics for checking performance of any classification model. Following are the plots of ROC curve for each model.

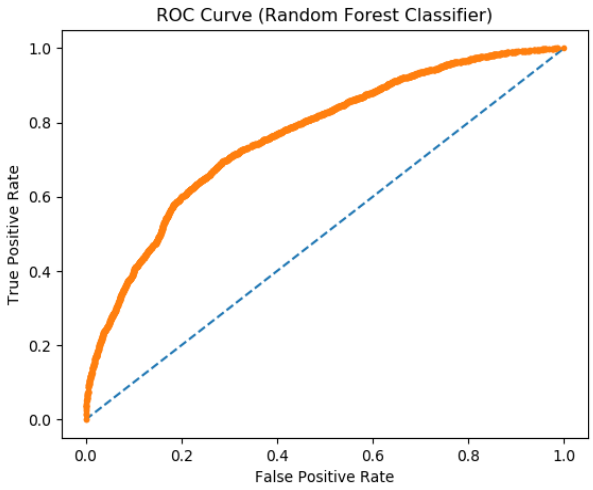
Support Vector Machines



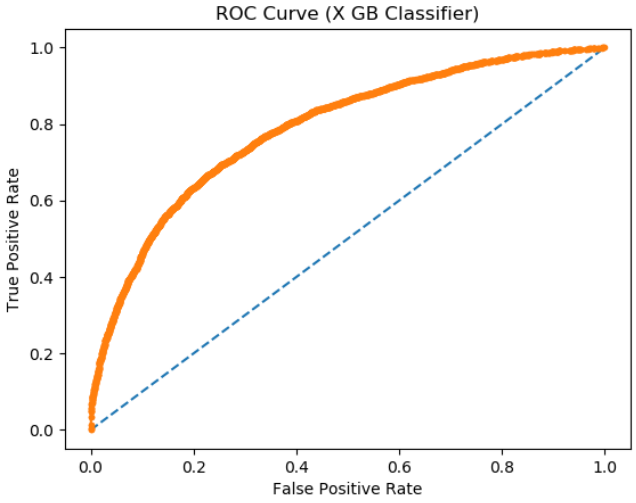
Logistic Regression



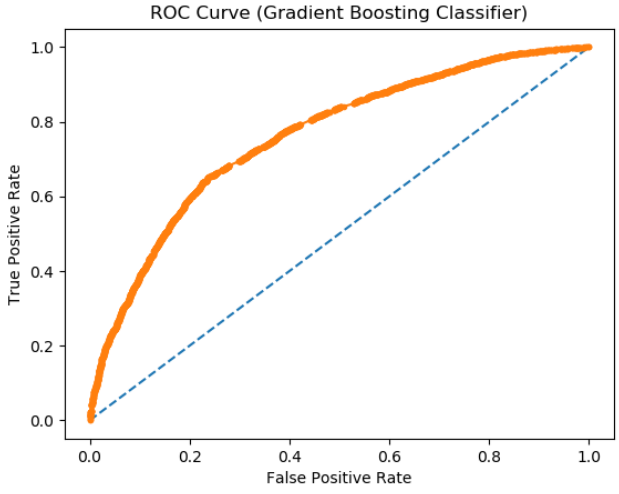
Random Forest



XGBoost



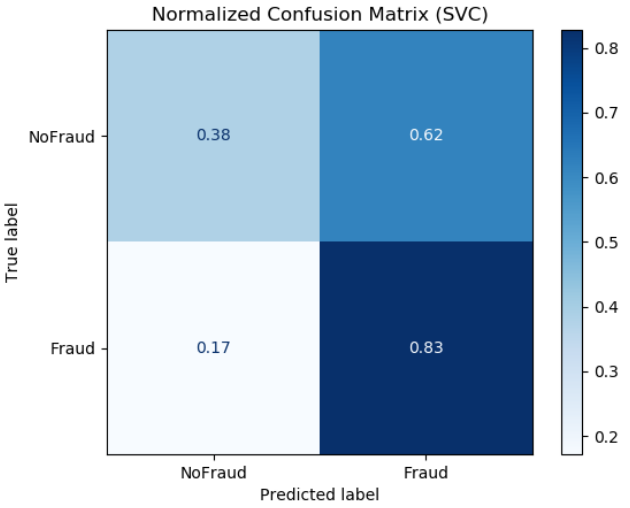
Gradient Boosted Machines



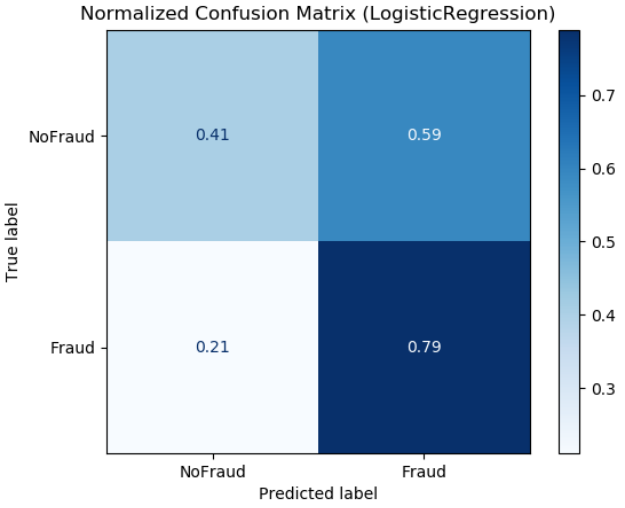
H. Confusion Matrix

A confusion matrix is often used to evaluate the performance of a classification model. It states the counts of true positives against false positives and true negatives and vice versa. We have generated normalized confusion matrices for all models. Following are confusion matrix plots for all models run.

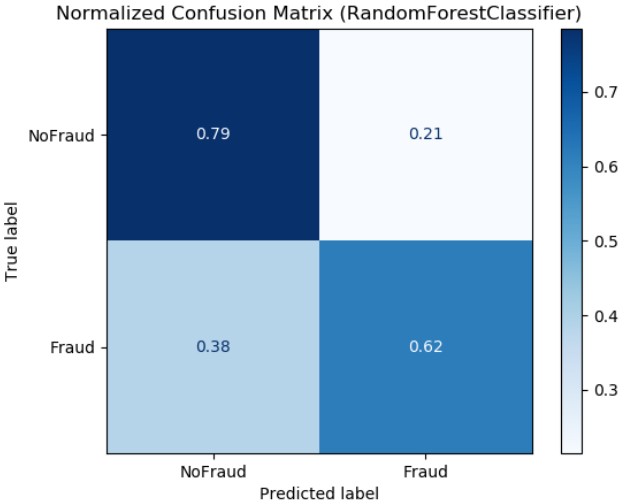
Support Vector Machines



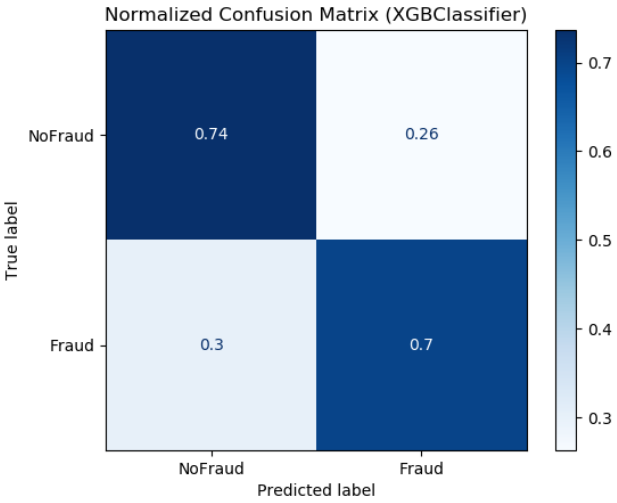
Logistic Regression



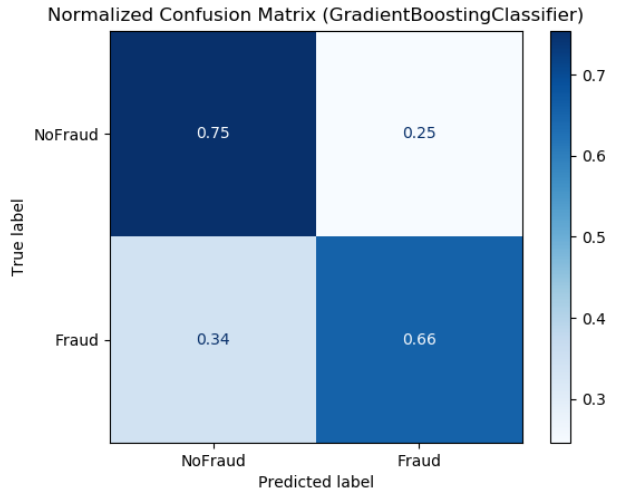
Random Forest



Xgboost



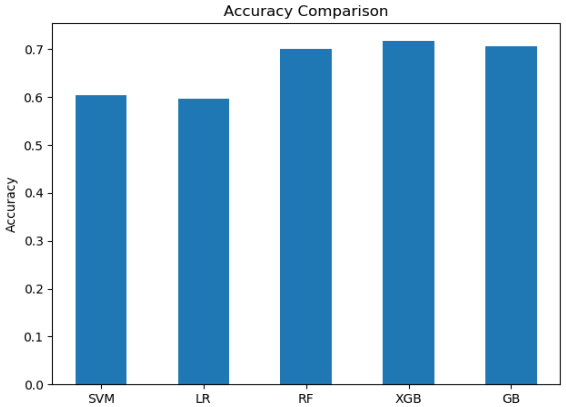
Gradient Boosted Machines



5.5 Comparative Analysis

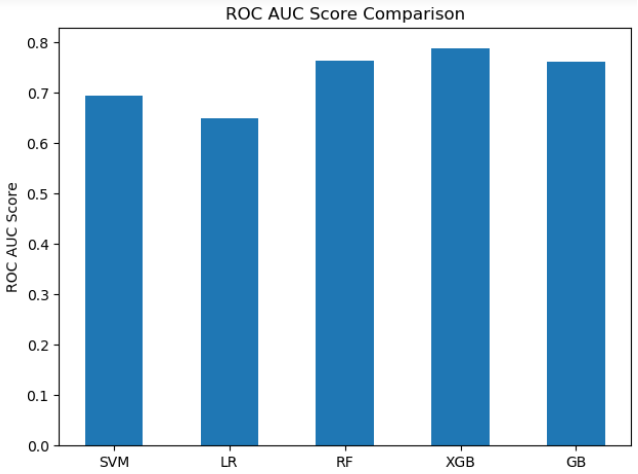
Now we compare the performances of all models in terms of accuracy, precision, recall, f1-score and other metrics like ROC curve etc. We go through each metric one by one and comment about which algorithm depicts better performance at that evaluation metric. In the following part, we will go through these evaluation metrics one by one and comment on the comparative performance of all these models on every single metric thereby conforming to the visual plots.

A. Accuracy



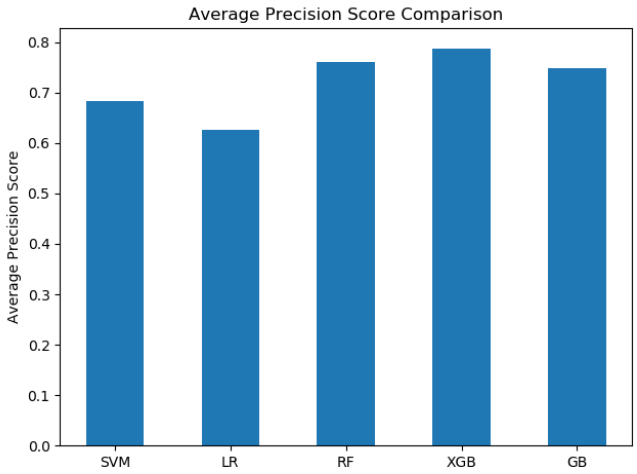
It is clearly visible that Xgboost has the best accuracy among all models.

B. ROC AUC Score



We can see in above plot that, again, Xgboost outperforms other models in terms of ROC AUC score as well.

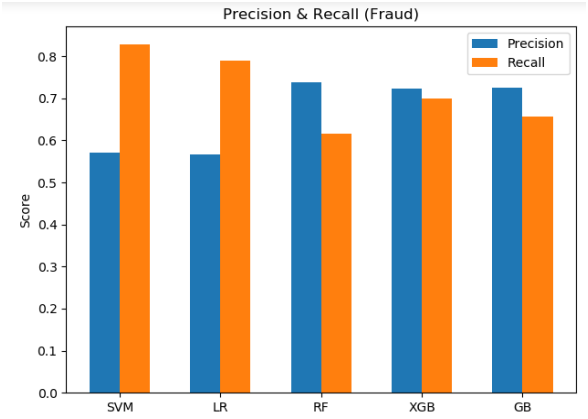
C. Average Precision Score

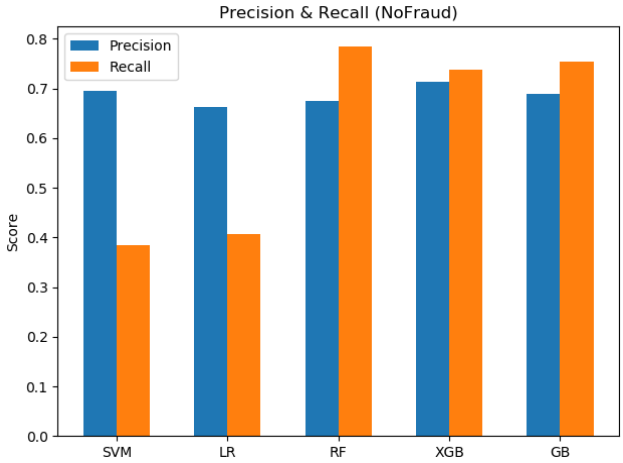


Once more time, Xgboost is performing best among all in terms of average precision score as well.

D. Precision & Recall

To compare precision & recall, we plot them together against all models. We have plotted for positive and negative classes separately. Following two plots show this.

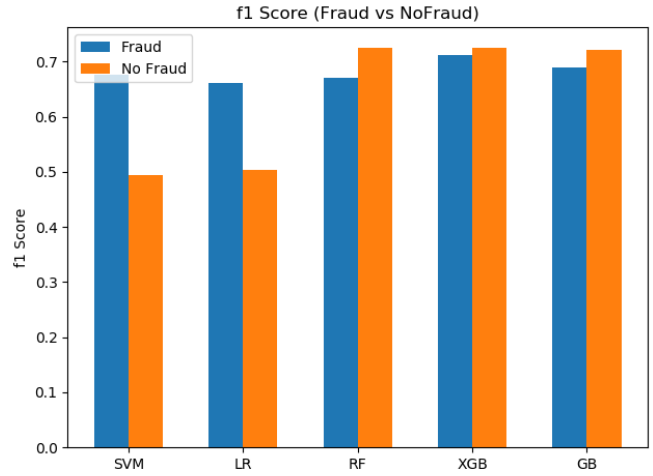




For positive class, SVM has highest recall and Random Forest has highest precision. For negative class, Random Forest has highest recall and Xgboost has highest precision.

E. F1-Score

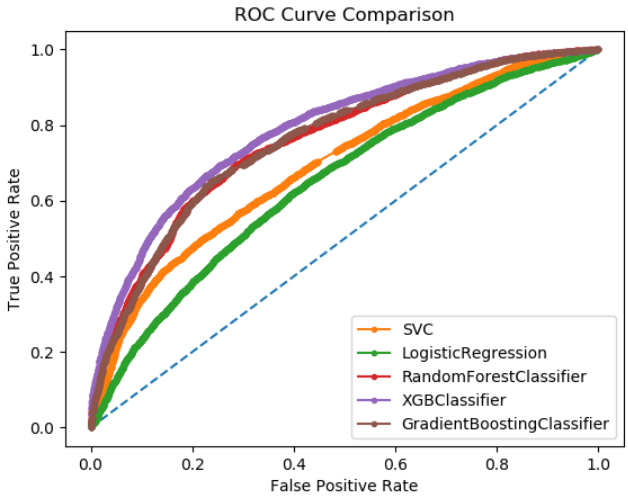
Following plot shows the f1-score for both classes combined in a single plot.



We can see that Xgboost has the highest f1-score in both cases i.e. positive and negative classes.

F. ROC Curve

Following plot shows the ROC curves of all models in a single plot for comparison purposes.



6 RESULTS

In the previous sections, we have analyzed in detail, all the evaluation metrics on individual as well as comparative levels in order to make a visual and analytical comparison between all the models that were trained and fine-tuned using hyper-parameter tuning techniques.

To justify which model performs best, it is not enough to look at these evaluation metrics only, since each one of the evaluation metric may be good for one model but not as good for the other as we have seen in the previous sections. For example, if Xgboost is showing best values for precision and f1-scores, it might not be showing a very good ROC curve wherewas Random Forest might be outperforming Xgboost in terms of ROC curve. Therefore, there is not a single gold standard which can determine which algorithm is performing the best among all.

In our case, we are more concerned about the problem statement or the type of problem we are going to solve. In case of fraud detection problems, it is very important to accurately pick and detect the fraudulent transactions. This means that f1-score value is an important metric for our case of problem. Now, if we look at the f1-score values, Xgboost has the highest value for positive class, whereas Random Forest has the highest value for the negative class. But if we look at the value of f1-score of Xgboost for negative class, it is almost equal to that of Random Forest model. Hence, we can say that in our case, Xgboost classifier is outperforming the other models in terms of our concerns of use case. The raw accuracy of Xgboost is also the highest amongst all other models. Here we would again like to mention, that this is not a standard procedure to determine the best performing model, it depends from one use case to the other.

Other than Xgboost model, the next best performing model in this case would be Random Forest. This is inferred based on the comparison charts of precision, recall and f1-score. Random Forest is an ensemble model of Decision Trees which assigns random samples from the training dataset to each of the Decision Trees to train themselves on. These Decision Trees are then evaluated and the better performing ones are kept while the ones that do not perform well are strengthened in next iterations of the training loop. This technique makes Random Forest one of the top performing classifiers for any general classification and even regression tasks. However, as we have stated earlier, for our problem use case, we will consider the output from Xgboost to be the best performing on our fraud detection dataset.

Another perspective to look and evaluate the performance of the classifiers is to look at their corresponding ROC curves. We have already shown the ROC curves of each individual model as well as a combined plot where the ROC curves of all the models are plotted together. This is a very important and reliable technique to compare between the performances of these models. Before we comment on the best performing classifier based on this, let us first state what an ROC curve shows in terms of model

stability and performance. ROC curve shows visually a trade-off between the model sensitivity and specificity. Model sensitivity refers to the True Positive Rate of the model. Similarly, model specificity shows the False Positive Rate of model subtracted from 1. If we take a random classifier that generates random predictions, its ROC curve will look like a diagonal along the plot. Along this curve, the model True Positive Rate and False Positive Rate would be equal. We have plotted this line as the blue dotted line that can be seen in the combined ROC curve plot. Along with this, several other ROC curves are plotted that correspond to their respective classifiers. As a rule of thumb, an ROC curve would show a good quality classifier the more its curve tends to move from the random classifier's curve towards to the top left corner of the quadrant. Similarly, an ROC curve that tends to move from the random classifier's curve towards the lower right corner of the quadrant shows relatively bad performance of the classifier.

Based on above discussion, if we take a deeper look into the combined ROC curve plot, we can clearly see that the curve of Xgboost classifier tends to be the most outward toward the top left corner. In addition, the ROC curve of Logistic Regression is the most inward moving curve among all the models. Hence, the combined ROC curve plot also verifies our initial conclusion that Xgboost is by far the best performing classifier in our fraud detection problem.

7 CONCLUSION

Based on the previous discussion and results, we conclude that fraud detection, which is a critical problem for credit card transactions, can be effectively handled by using some state of the art machine learning techniques. No doubt, the development of such system requires a detailed and thorough analysis and exploration of the dataset at the initial stage. This will be then followed by a detailed pre-processing phase that must be done in order to make the data useful for feeding into machine learning algorithms. Finally, the choice of machine learning algorithm is also a very important step especially when dealing with the fraud detection problem. All the algorithms that we have used in this project are widely used for such space of problems. There are other algorithms as well which we can find in literature that are also used for fraud detection systems. But, the choice of algorithm would mainly depend on the dataset statistics like data row count, number of features as well as the underlying objective of the problem that one is trying to achieve in terms of precision and accuracy.

8 REFERENCES

[1] Correia, I., Fournier, F., & Skarbovsky, I. (2015, June). The uncertain case of credit card fraud detection. In *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems* (pp. 181-192). J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] Fadaei Noghani, F., & Moattar, M. (2017). Ensemble classification and extended feature selection for credit card fraud detection. *Journal of AI and Data Mining*, *5*(2), 235-243.K. Elissa, "Title of paper if known," unpublished.

[3] Ryman-Tubb, N. F. (2016). *Understanding payment card fraud through knowledge extraction from neural networks using large-scale datasets.* University of Surrey (United Kingdom).Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[4] Fu, K., Cheng, D., Tu, Y., & Zhang, L. (2016, October). Credit card fraud detection using convolutional neural networks. In *International Conference on Neural Information Processing* (pp. 483-490). Springer, Cham.

[5] Kültür, Yiğit, and Mehmet Ufuk Çağlayan. "A novel cardholder behavior model for detecting credit card fraud." *Intelligent Automation & Soft Computing* (2017): 1-11.

[6] Zanin, M., Romance, M., Moral, S., & Criado, R. (2018). Credit card fraud detection through parenclitic network analysis. *Complexity, 2018.*