
# Table of Contents

# 1. Introduction

Most of us must have faced the situation where our card transaction gets denied, even though we know that we have enough funds in our bank for that transaction. And soon after the transaction denial, we get a message from our bank saying 'Please reply 'Y' if you just tried this transaction or else reply 'N'. Perhaps it is often embarrassing at that moment for us but this fraud prevention system is actually saving consumers millions of dollars per year.

Customers in such a situation where the transaction failure was falsely notified, either immediately get in touch with their banks or few might go ahead and try a different card. But, there are instances where customers blame the merchant they are buying from and hence they don't follow through with their purchase and select some other merchants.

It is really important to improve this fraud detection as the financial impact of this is vast, because it's not just that one transaction you are saying 'no' to, it is potentially the lifetime value of a very profitable customer.

The common approach to fighting fraud is to tighten security parameters for transactions. It makes sense on the surface. Better security = Less fraud. While it may successfully reduce instances of fraud, it may stop legitimate transactions too, leaving substantial revenue on the table.

For one financial institution, it is important to identify false transactions, but also maintain quality customer service. A vigilant fraud detection effort cannot be intrusive to the customer by flagging and declining legitimate transactions. This financial institution needs a fraud detection system that strikes a balance between oversight and customer service.

Using the correct approach, the financial institution could correctly identify transitions that had been erroneously identified as fraud and also be able to identify additional fraud that had gone undetected. Besides dramatically improving the company's ability to detect fraud, the precise approach increases customer satisfaction as well, reducing the friction between the company and its customers by significantly improving the transaction approval process while increasing the effectiveness of fraud detection.

# 2. Data Loading and Description

We acquire data for this study from the given link where data is present in .csv file format. The total data to be used for this project is captured in two parts. One of the files (Transaction file) contains details about the customer transactions such as the card used, billing region and country of purchaser, time difference between subsequent transactions, product code, transaction amount, etc. The second file (Identity file) consists of the identity information about the transactions. This file has data such as device information, device type used for the transaction, browser details, etc. The Identity file does not have data for all the transactions in the main Transaction file. Also, few of the columns have a lot of missing data.

## 2.1. Transaction Data Details

It contains money transfer and also other gifting goods and service, like you booked a ticket for others, etc.

a) TransactionDT : timedelta from a given reference datetime (not an actual timestamp).It corresponds to the difference in time for subsequent transactions. The overall data spans for 6 months.
b) TransactionAMT : transaction payment amount in USD
c) ProductCD : Product code, the product for each transaction. It mainly consists of code values like ['W', 'H', 'C', 'S', 'R']
d) card1 - card6 : Payment card information, such as card4 specifies card types such as :'discover', 'mastercard', 'visa', 'american express',etc. While card6 column specifies whether the card is a debit or credit card.
e) addr1 : address as billing region of purchaser.
f) addr2 : address as billing country of purchaser.
g) dist: distances between billing address, mailing address, etc.
h) P_ and R_emaildomain: purchaser and recipient email domain
i) C1-C14: counting, such as how many addresses are found to be associated with the payment card, etc. The actual meaning is masked.
j) D1-D15: timedelta, such as days between previous transactions, etc.
k) M1-M9: match, such as names on card and address, etc.
l) Vxxx: Vesta engineered rich features, including ranking, counting, and other entity relations. For example, how many times the payment card associated with an IP and email or address appeared in 24 hours time range, etc. All Vesta features were derived as numerical, some of them are count of orders within a clustering, a time-period or condition, so the value is finite and has ordering (or ranking).

## 2.2. Identity Data Details

Variables in this table are identity information – network connection information (IP, ISP, Proxy, etc) and digital signature (UA/browser/os/version, etc) associated with transactions. They're collected by Vesta’s fraud protection system and digital security partners.

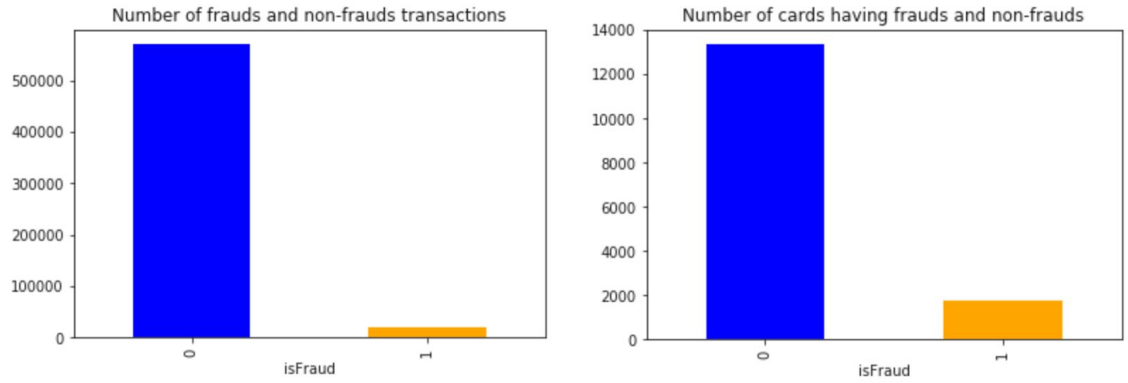
a) DeviceType : It mentions whether the device used for transaction is mobile or desktop device.
b) DeviceInfo: It gives details about the device whether it is ‘Samsung’, 'iOS Device', 'Windows', etc.
c) id01 to id20 : These are numerical features for identity, which are collected by Vesta and security partners such as device rating, ip_domain rating, proxy rating, etc. Also it records behavioral fingerprints like account login times/failed to login times, how long an account stayed on the page, etc.
d) id23: It specifies if any IP proxy is used for transaction.
e) id30-id31: These identify OS and browser details whether Android, Linux, iOS, etc. was used for transaction using any of Chrome, Safari, etc. browsers.

## 3. Data Exploration

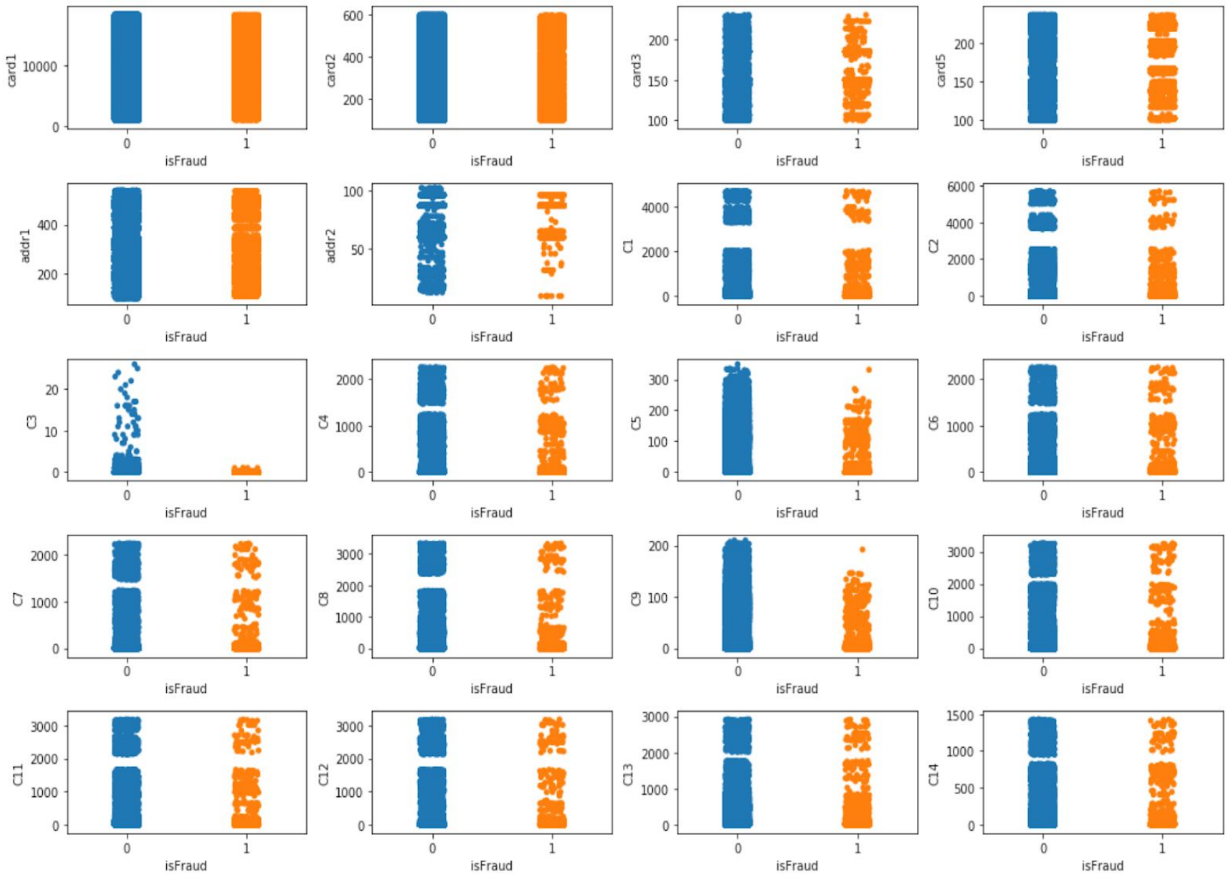
- As data in the Transaction file is more as compared to Identity file, for merging we then perform ‘left’ merge to get a single dataset. The shape of the dataset is (590540 rows, 435 columns).
- Most of the Categorical columns in the dataset have datatype as Object. Hence, we then convert these object columns to Category datatype and assign proper values to the category.
- The column ‘DeviceType’ has few NaN values for respective ‘DeviceInfo’ values. But, since this ‘DeviceInfo’ column has some valid ‘DeviceType’ values in other rows, based on this combination the missing NaN values are then filled.
- Similarly, NaN values for column ‘card4’ are populated based on the common values for column ‘card1’.
- The column ‘card6’ mainly has two categorical values i.e. ‘debit’ and ‘credit’. But, there are only around 3 to 4 rows having values for this column as ‘charge card’ and ‘debit or credit’. On further exploration, it seems that the ‘charge card’ is equivalent to ‘credit’ and ‘debit or credit’ is equivalent to ‘debit’. Hence, the values are updated accordingly in the dataset.
- The ‘TransactionDT’ column has values like 86400, 86506, etc. This is basically the number of seconds per day. 24*60*60 = 86400. That means considering the first transaction is done on day 1, we can evaluate the day difference of subsequent transactions. As the maximum value of this column is 15811131 which corresponds to day 183, we can conclude that total dataset spans for 6months. Hence, from the ‘TransactionDT’ column we can create new column ‘TransactionDay’ which gives the number of days for that transaction from subsequent transactions.
- There are around 74 columns which have data less than 90000 which is less than 15% of total data. Hence, we can remove such columns from the dataset as it will not provide accuracy for further modelling due to maximum missing values.

## 4. Data Story and Visualization

- Out of total 590540 transactions, 569877 transactions are not fraud while 20663 transactions are fraud. This is determined by the column ‘isFraud’ in the dataset. If we consider unique cards for these transactions, then 13350 cards have non-fraud transactions while 1740 cards have fraud transactions. Number of fraud transactions are very less as compared to non-fraud transactions, this infers that the data is highly imbalanced and in our modelling we need to use some balancing logic for such data. This imbalance in data is mostly observed in many Binary Classification problems, hence before feeding the data to model we need to perform either under sampling or over sampling of data. The details about handling of such imbalanced data in this project is later mentioned in ‘Data Pre-processing’ section of modelling.

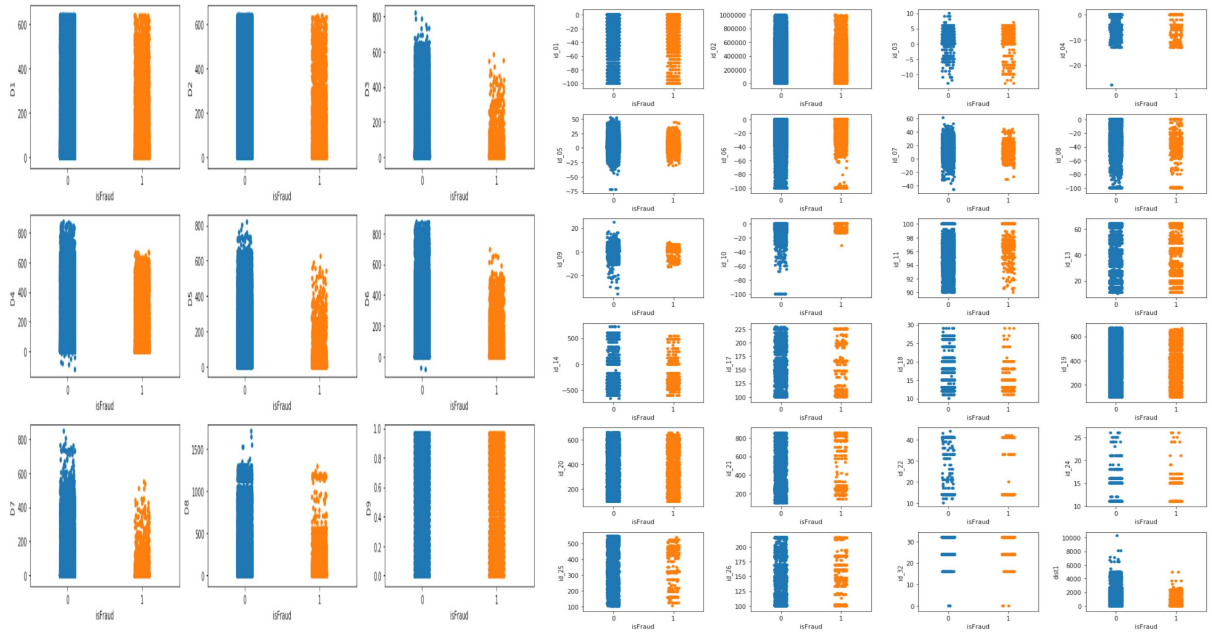



- Different visualizations are plotted to see the relationship between the frauds and non-frauds category. As per below shown plots, there is no specific pattern observed in fraud and non-fraud transactions for most of the variables. as the data visuals are mostly similar for both fraud and non-fraud transactions. Only for variable C3, fraud transactions have very low value and can be visualized from below plot.

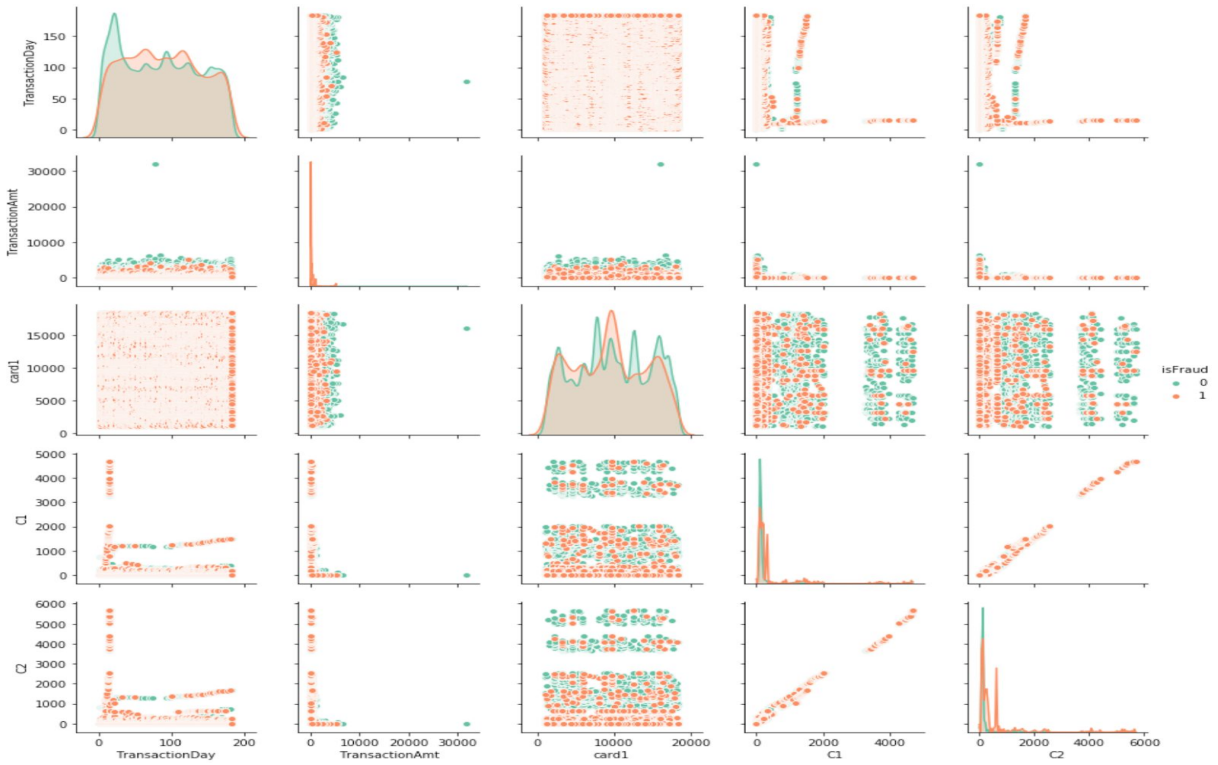



- Columns ‘D1-D9’, ‘C1-C14’ and a few ‘id’ columns have all continuous variables. But, there is no distinct difference in the values between frauds and non-frauds. As shown in

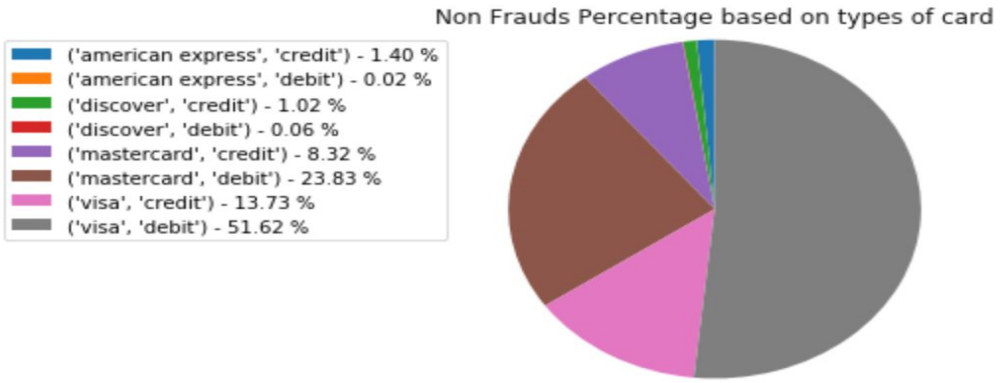
the plot, for both fraud and non-fraud type of transactions these fields have almost similar values.

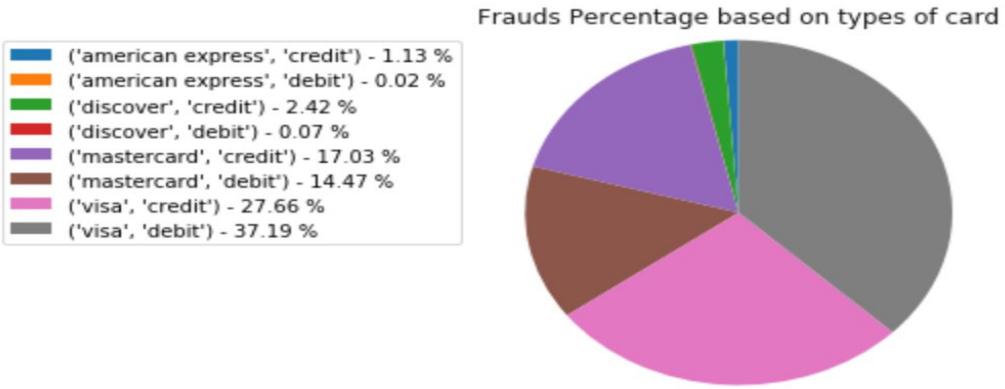


- Below figure shows one pair plot for a few fields which have continuous values. This is to identify if there is some relation between these continuous variables in terms of frauds and non-frauds. There is only a direct relationship between ‘C1’ and ‘C2’. These two fields have a positive linear relation. But, even this relation does not vary or differentiate any fraud and non-frauds transaction.

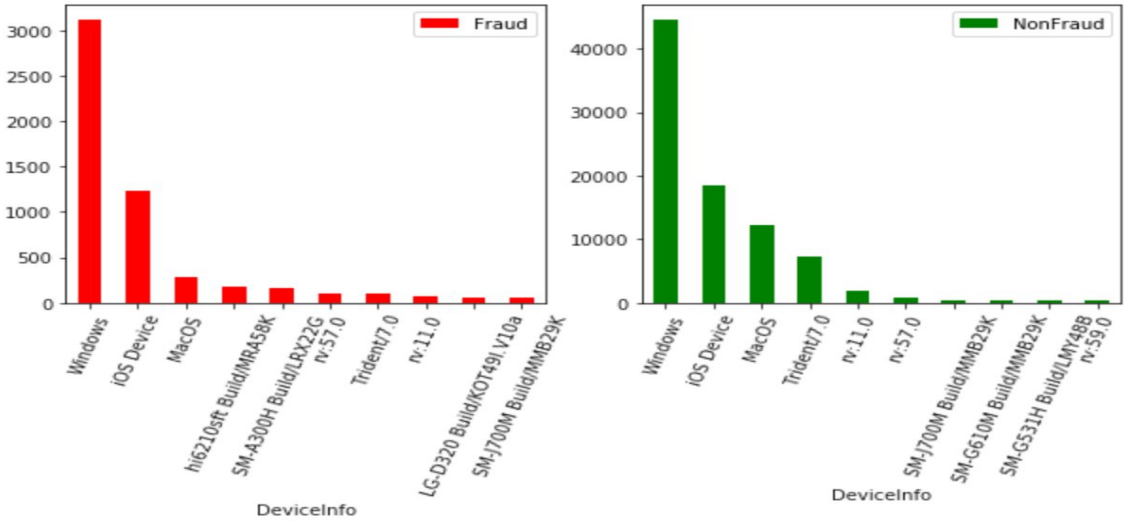



- As per pie plots shown below, ‘Visa Debit’ cards have a maximum percentage of both frauds and non-frauds. So, even based on type of cards also there seems to be no distinguishable behaviour for fraud and non-fraud transactions.




- Also, from the plot below, it is seen that 'Windows' devices have a maximum number of frauds and non-frauds. This can be also because maximum people use 'Windows' devices more than any other type of devices. And, we can also infer that more customer purchases are done from desktops or laptops as compared to mobile devices.




- The details about the mentioned data story and wrangling can be found in this [IPython Notebook].

# 5. Modelling

Based on previous data exploration, the variables did not have any linear correlations. So, we tried to find if there is any proportion of the variance or fluctuation of one variable that is predictable from the other variable. In this project, the dependent variable is 'isFraud' variable as we are interested in identifying how this variable is affected by other independent variables or pairs of independent variables.

Hence, we calculated the coefficient of determination i.e R-squared to identify the correlations in this IPython Notebook. The coefficient of determination is the ratio of the explained variation to the total variation. The coefficient of determination is such that $0 < R^2 < 1$, and denotes the strength of the linear association between x and y. The coefficient of determination represents the percent of the data that is closest to the line of best fit. Using Linear Regression model to calculate the coefficient of determination value, we get to know this value comes out to be very small for any pair of the independent variables. Hence, linear model is not an option for this dataset as we have also concluded from the data visualization.

Below are the details about different model selection techniques which can identify the frauds from the customer transactions in the most efficient way. And, all the detailed coding about modelling can be found in this IPython Notebook.

## 5.1. Data Pre-Processing

Before providing the data to the machine learning algorithms, we need to perform a few data processing steps. The steps which we followed are related to categorical fields and handling of missing values throughout the dataset. It also involves splitting data to properly train and test the model.

a) Label Encoder: In the dataset, there are some variables with numerical values, binary values and few variables have categorical values. This label encoding needs to be applied on those categorical variables.

b) Simple Imputer: In the dataset, the variables having numerical values consists of few missing values. In order to handle these missing values, we can apply some imputation techniques. Here, we are applying Simple Imputer technique to replace the missing values with the ‘median’ value of the respective variables.

c) Data Splitting: Next step involves splitting the dataset into training and testing data. In this project, we split the data into 70-30% i.e. 70% training data and 30% testing data. Also, we split the data in such fashion that the same fraction of both fraud and non-fraud transactions are present in train and test data.

d) Handling Imbalanced Dataset: As mentioned earlier in the Data Exploration section, the overall dataset is highly imbalanced. So, in order to balance this imbalanced dataset, we need to perform some data sampling. In this project, we have used NearMiss algorithm for under-sampling and SMOTE technique for oversampling.

NearMiss is an under-sampling technique which aims to balance class distribution by randomly eliminating majority class examples. When instances of

two different classes are very close to each other, we remove the instances of the majority class to increase the spaces between the two classes. This helps in the classification process. To prevent problems of information loss in most under-sampling techniques, near-neighbor methods are widely used.

SMOTE (Synthetic Minority Oversampling Technique) synthesises new minority instances between existing minority instances. It generates the virtual training records by linear interpolation for the minority class. These synthetic training records are generated by randomly selecting one or more of the k-nearest neighbors for each example in the minority class.

One major point to adhere and always remember is that sampling before cross validation is wrong because it could cause data leakage i.e. making some copies of the same points ending up in both the training set and the test set. This would make the machine learning classifier cheat because the classifier will see some data points in the test set being identical to some other data points in the training set, and as such, will lead to overfitting, which will eventually cause inaccurate results. Data sampling is advised to always be done during splitting of data and/or cross validation. In this project, we will do data sampling during cross validation. This is to reduce bias and overfitting.

### 5.2. Performance Metric

Once data preprocessing is done, we can feed the dataset to the machine learning algorithms and create an efficient model. In this project, we need to classify the fraud and non-fraud transactions. The very simple metric to measure classification is basic accuracy i.e. ratio of correct predictions to the total number of samples in the dataset. However, in the case of imbalanced classes this metric can be misleading, as high metrics doesn't show prediction capacity for the minority class. We may get 99% accuracy but still lousy prediction capacity on the class we are truly interested in.

The Receiver operating characteristic (ROC) curve is the typical tool for assessing the performance of machine learning algorithms, but it actually does not measure well for imbalanced data. The ROC curves plot FPR (False Positive Rate) vs. TPR (True Positive Rate). Because TPR only depends on positives, ROC curves do not measure the effects of negatives. The area under the ROC curve (AUC) assesses overall classification performance. AUC does not place more emphasis on one class over the other, so it does not reflect the minority class well. The Precision-Recall (PR) curves are more informative than ROC when dealing with highly skewed datasets. The PR curves plot precision vs. recall (FPR). Because Precision is directly influenced by class imbalance so the Precision-recall curves are better to highlight differences between models for highly imbalanced data sets. When we compare different models with imbalanced settings, the area under the Precision-Recall curve will be more sensitive than the area under the ROC curve.

### 5.3. Logistic Regression

Logistic Regression is the go-to method for binary classification problems. For Logistic Regression, we first use all the above mentioned preprocessing steps except the step for handling the imbalanced dataset. Here, we get very high accuracy i.e. 0.96, but this is due to the imbalanced classes in the dataset. The table below shows the metrics calculation output for different algorithms. From the table, we can see that when just Logistic Regression is performed then the Precision and Recall value for non-fraud transactions is very high. But, this model fails to identify any of the fraud transactions, which is our main aim in this project.

| Model Name | Accuracy | Precision | | Recall | |
|---|---|---|---|---|---|
| | | Non-Fraud | Fraud | Non-Fraud | Fraud |
| Logistic Regression | 0.96 | 0.97 | 0.20 | 1.00 | 0.00 |
| Logistic Regression with PCA | 0.97 | 0.97 | 0.00 | 1.00 | 0.00 |
| Logistic Regression with NearMiss sampling | 0.73 | 0.99 | 0.09 | 0.74 | 0.70 |
| Logistic Regression with SMOTE sampling | 0.63 | 0.97 | 0.04 | 0.64 | 0.44 |

Also, we can see that similar metrics output is received when we perform Logistic Regression with PCA (Principal Component Analysis). After this from the table we can see when we handle the imbalanced data with either NearMiss under sampling or SMOTE oversampling, we get better Recall but Precision remains very small for fraud class.

### 5.4. Decision Tree Classifier

Decision Tree Classifier are a popular supervised learning method for a variety of reasons. Benefits of decision trees include that they are easy to interpret and they don't require feature scaling. They have several flaws including being prone to overfitting. For Decision Tree Classifier, we follow similar steps as we did above for Logistic Regression. The decision tree algorithm is effective for balanced classification, although it does not perform well on imbalanced datasets. We can observe the same from the below metrics table. Even Decision Tree Classifier fails to identify fraud transactions efficiently.

| Model Name | Accuracy | Precision | | Recall | |
|---|---|---|---|---|---|
| | | Non Fraud | Fraud | Non Fraud | Fraud |
| Decision Tree Classifier with PCA | 0.94 | 0.97 | 0.16 | 0.97 | 0.16 |
| Decision Tree Classifier with NearMiss sampling | 0.18 | 0.98 | 0.04 | 0.15 | 0.92 |
| Decision Tree Classifier with SMOTE sampling | 0.81 | 0.97 | 0.07 | 0.83 | 0.38 |

### 5.5. Random Forest Classifier

Random Forest Classifier is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Similar steps are followed here for Random Forest Classifier models. Here, we can observe that this classifier with SMOTE sampling has very better metrics values as compared to any of the previous models. The Precision and Recall value are better for both fraud and non-fraud transactions with great accuracy as well.

| Model Name | Accuracy | Precision | | Recall | |
|---|---|---|---|---|---|
| | | Non Fraud | Fraud | Non Fraud | Fraud |
| Random Forest Classifier with PCA | 0.96 | 0.97 | 0.33 | 1.00 | 0.04 |
| Random Forest Classifier with NearMiss sampling | 0.68 | 0.99 | 0.09 | 0.68 | 0.83 |
| Random Forest Classifier with SMOTE sampling | 0.98 | 0.98 | 0.89 | 1.00 | 0.42 |

### 5.6. XGBoost

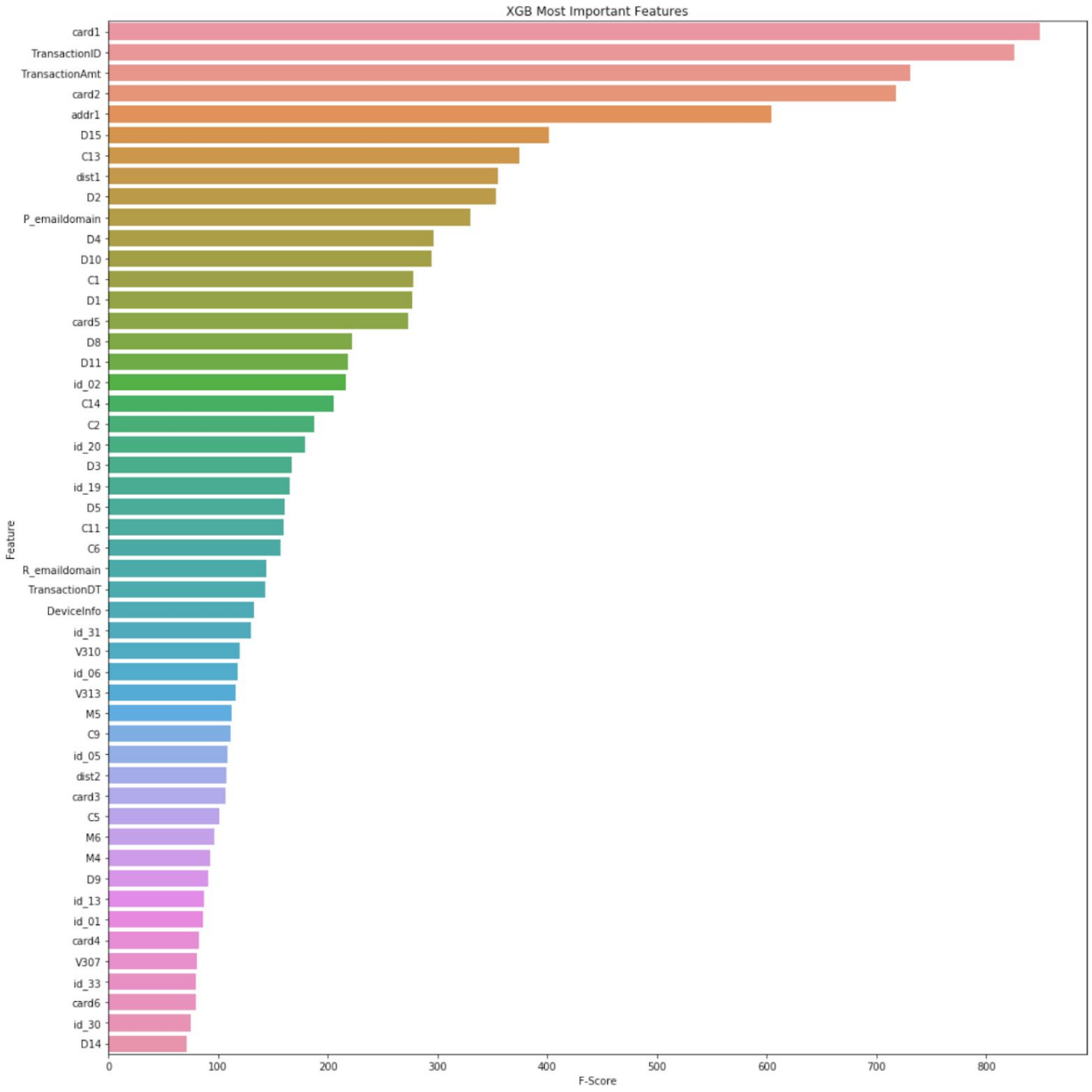
XGBoost (eXtreme Gradient Boosting) is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way.

XGBoost handles missing values internally, hence we don't need any imputation here. Also, handling of imbalanced dataset class can be done by using one of the parameters to be passed to the model which is '*scale_pos_weight*' [default=1]. This parameter controls the balance of positive and negative weights and a typical value to be considered is equal to sum(negative instances) / sum(positive instances).

Based on the table below, it seems that XGBoost provides better Recall, Precision and Accuracy. And these values are nearly similar to what we got from Random Forest Classifier with SMOTE sampling.

| Model Name | Accuracy | Precision | | Recall | |
|---|---|---|---|---|---|
| | | Non Fraud | Fraud | Non Fraud | Fraud |
| XGBoost Classifier | 0.94 | 0.99 | 0.34 | 0.94 | 0.83 |

A benefit of using ensemble methods like XGBoost is that it can automatically provide estimates of feature importance from a trained predictive model. In the plot shown below we can see the top 50 important features.

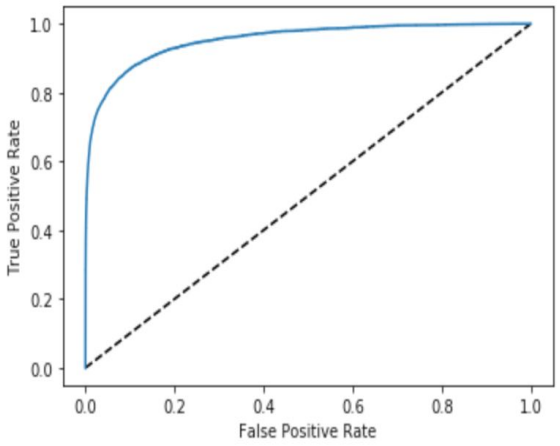
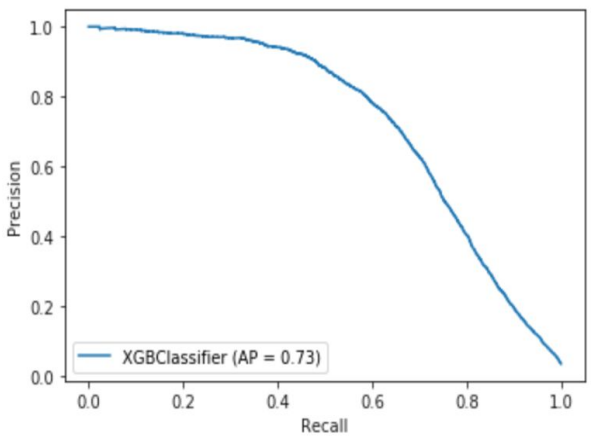



## 6. Model Recommendation

The table below shows the two main metric parameters based on which we can conclude which is the best model for fraud detection from the customer transactions. So, as shown the Average Precision Recall score for most of the models is very low. Only RandomForestClassifier with SMOTE sampling model and XGBoost Classifier model has good Average Precision Recall score as well as ROC-AUC score. The best model to select from all the models which we have used in the project is the XGBoost Classifier.

| Model Name | ROC-AUC score | Average Precision Recall Score |
|---|---|---|
| Logistic Regression | 0.50 | 0.07 |
| Logistic Regression with PCA | 0.50 | 0.06 |
| Logistic Regression with NearMiss sampling | 0.72 | 0.10 |
| Logistic Regression with SMOTE sampling | 0.54 | 0.06 |
| Decision Tree Classifier with PCA | 0.57 | 0.06 |
| Decision Tree Classifier with NearMiss sampling | 0.53 | 0.04 |
| Decision Tree Classifier with SMOTE sampling | 0.60 | 0.05 |
| Random Forest Classifier with PCA | 0.52 | 0.12 |
| Random Forest Classifier with NearMiss sampling | 0.75 | 0.14 |
| Random Forest Classifier with SMOTE sampling | 0.71 | 0.64 |
| XGBoost Classifier | 0.95 | 0.73 |

The ROC curve and PR curve for the recommended XGBoost Classifier model is shown in the below plot. The ROC curve and PR curve plots for remaining models can be visualized in the IPython Notebook.

| XGBoost Classifier ROC Curve (0.95) | XGBoost Classifier PR Curve (0.73) |
|---|---|
|  |  |

## 7. Future Work

Our best model identified here, which is XGBoost Classifier model can be further made more efficient by tuning its parameters. We can perform GridSearch for fine tuning of hyper-parameters of XGBoost Classifier. Also, we can execute all previous models by considering only the most important features extracted from XGBoost Classifier. This can further improve performance of algorithms.