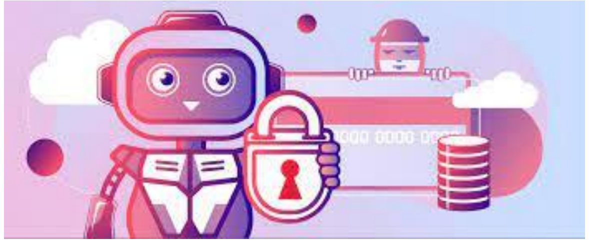


Credit Card Fraud Detection With ML Model

-Presented by
Saikat Chowdhury

Table of Contents

- Abstruct
- Introduction
- Problem Statement
  - Fraud Detection
  - Fraud Detection Process
- Data Understanding
- Data Preparation
- Exploratory Data Analysis
- Train/Test Data Splitting
- Model Selection and Model Building
- Model Evaluation
- Inference

Abstruct

- Credit card fraud detection is presently the most frequently occurring problem in the present world.
- This is due to the rise in both online transactions and e-commerce platforms.
- Credit card fraud generally happens when the card was stolen for any of the unauthorized purposes or even when the fraudster uses the credit card information for his use.
- In the present world, we are facing a lot of credit card problems. To detect the fraudulent activities the credit card fraud detection system was introduced.

Introduction

- A credit card fraud is perpetrated whenever any person intentionally and unsympathetically used the information on the card for dishonest purpose, impersonation and other egocentric and dubious intentions.

Problem Statement

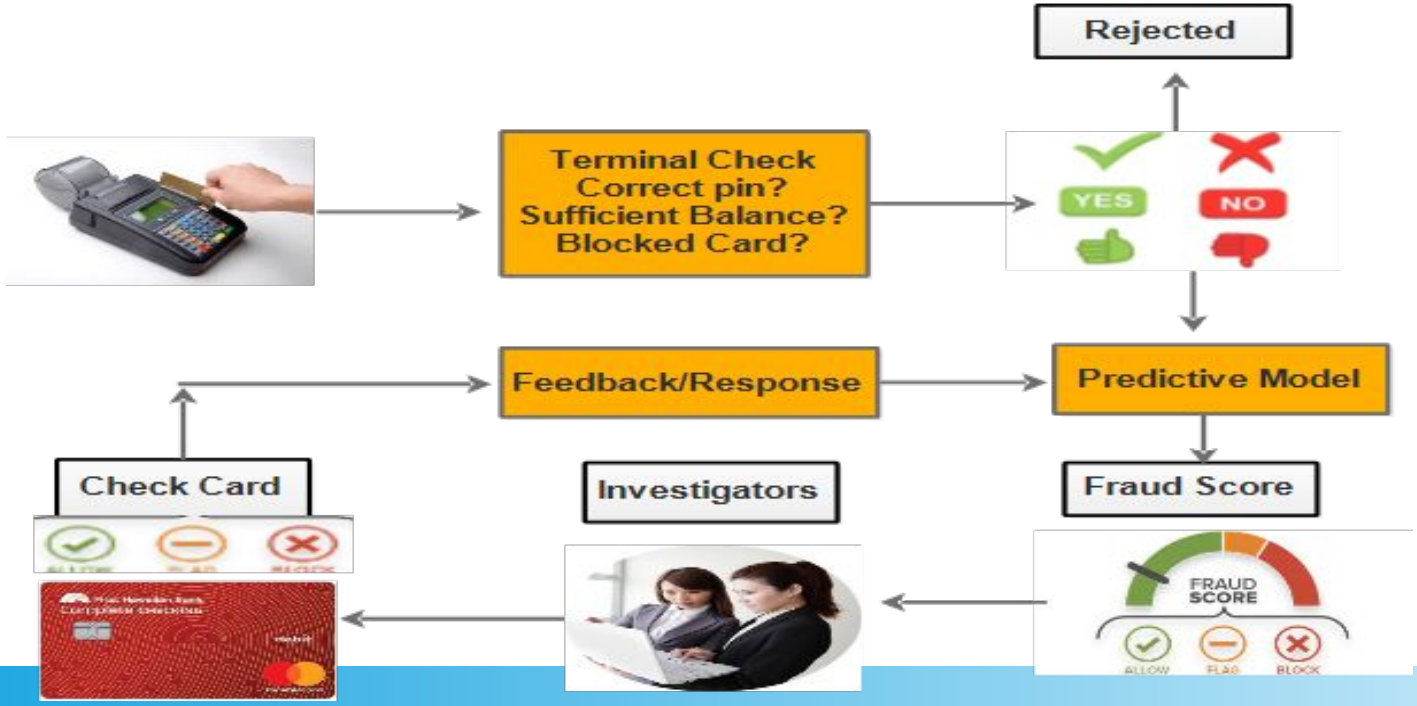
- Fraudulent activities have increased a lot in the recent year.
- Due to this steep increase in banking frauds, it is important to detect these fraudulent transactions in time in order to help consumers as well as banks, who are losing their credit worth each day.
- Every fraudulent credit card transaction that occur is a direct financial loss to the bank as the bank is responsible for the fraud transactions as well it also affects the overall customer satisfaction adversely.
- The Credit Card Fraud Detection Problem includes modelling past credit card transactions with the data of the ones that turned out to be fraud.
- The aim of this project is to identify and predict fraudulent credit card transactions using Machine Lerning models.

Fraud Detection

- Credit card fraud means using a person's card without one's knowledge with the intention of withdrawing funds and purchasing of goods.
- The most common ways of card theft are done by stealing the card before it is received by the customer, getting the card details from the owner via phone calls etc.

Fraud Detection Process

- Credit card fraud detection is the process of knowing whether or not a set of credit card transactions is in the category of fraudulent or legitimate instances of buying or selling something.
- Some desirable characteristics of a Fraud Detection System (FDS) include efficient detection of fraud, and high effectiveness or productivity in relation to its cost in transaction checking.
-

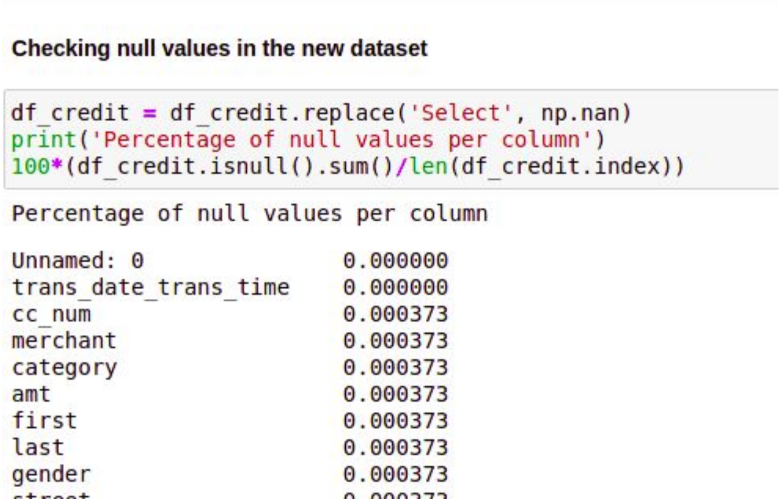


Data Understanding and Data Preparation

Steps follows for Data understanding and Data Preparation:-

- First, we looked at two csv files and we merged the both data sets to make a single data set.
- After merged there are total 1852394 rows and, 23 columns, and There are total 1850 fraud transactions and 234831 normal transactions in the single data set after cleaning the data set.
- Next, we checked for null values, duplicates, outliers.
- We have also performed univariate and bivariate analysis on this data set.
- After cleaning the data set, we have removed the categorical features and replaced them with dummy variables which had numerical values.
- Then we have splitted the entire set into validation and train data set (Train: 70% and Test:30%).
- We have applied certain models on the data like KNN Model, Decision Tree Model, Random Forest Model using the approach of Repeated k-Fold and Stratified K-Fold.
- We have noted the reading of the various parameters which shall help us to understand which machine learning model is good for dealing with such problems.

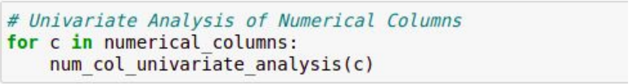
Data Understanding and Data Preparation



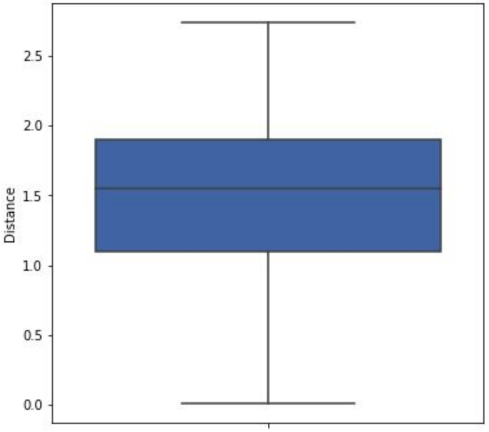
Checking null values in the new dataset

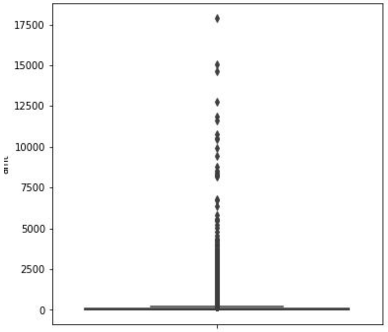
```
df_credit = df_credit.replace('Select', np.nan)
print('Percentage of null values per column')
100*(df_credit.isnull().sum()/len(df_credit.index))
```

```
Percentage of null values per column

Unnamed: 0               0.000000
trans_date_trans_time    0.000000
cc_num                   0.000373
merchant                 0.000373
category                 0.000373
amt                      0.000373
first                    0.000373
last                     0.000373
gender                   0.000373
street                   0.000373
```

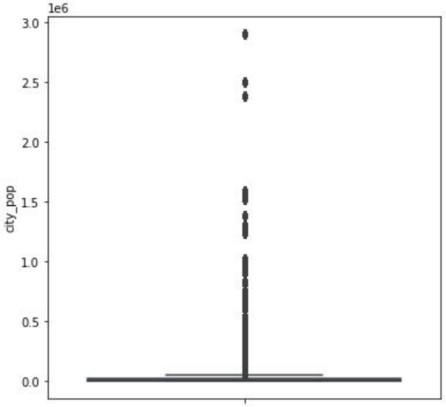


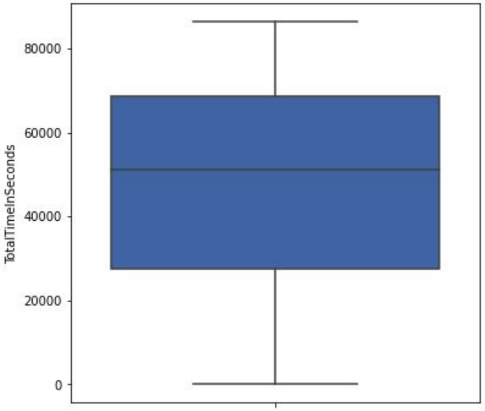
```
# Univariate Analysis of Numerical Columns
for c in numerical_columns:
    num_col_univariate_analysis(c)
```





- **Inference:-** Null values are less and negligible in number so we can decide to drop these variable with null values





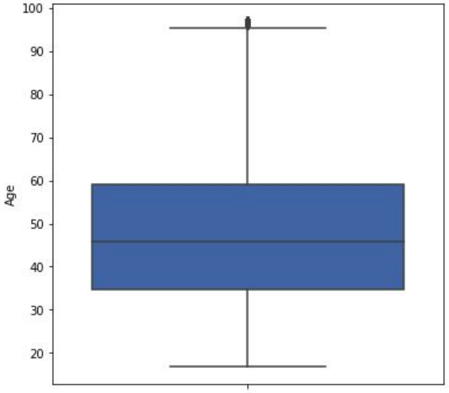
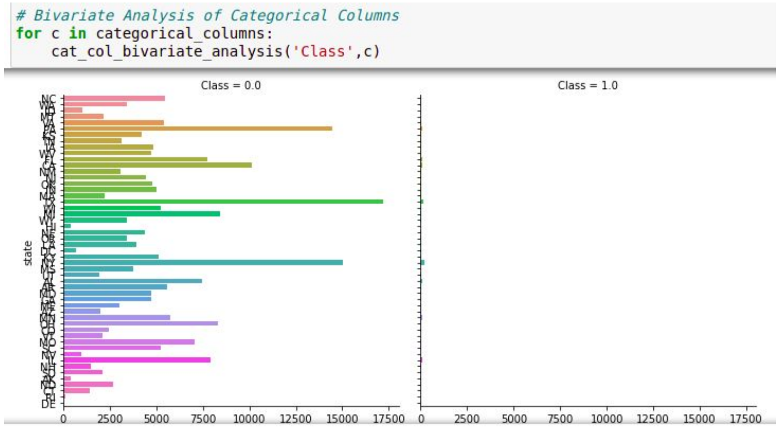


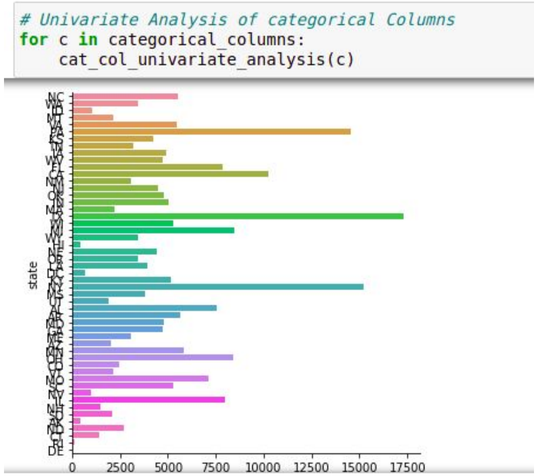
Fig: Univariate Analysis of Numerical Columns

Data Understanding and Data Preparation



```
# Bivariate Analysis of Categorical Columns
for c in categorical_columns:
    cat_col_bivariate_analysis('Class',c)
```

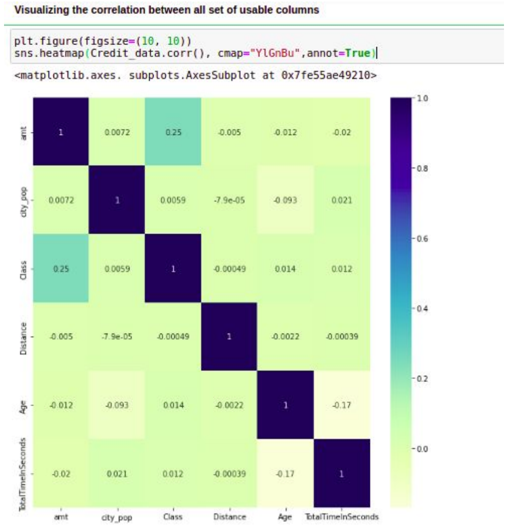
Fig: Bivariate Analysis of Categorical Columns



```
# Univariate Analysis of categorical Columns
for c in categorical_columns:
    cat_col_univariate_analysis(c)
```

Fig: Univariate Analysis of categorical Columns

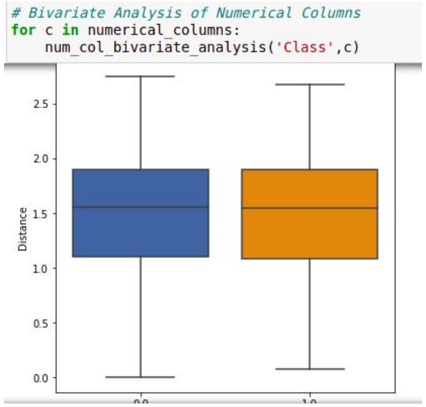
Fig: Visualizing the correlation between all set of usable columns



Visualizing the correlation between all set of usable columns

```
plt.figure(figsize=(10, 10))
sns.heatmap(Credit_data.corr(), cmap="YlGnBu",annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe55ae49210>
```



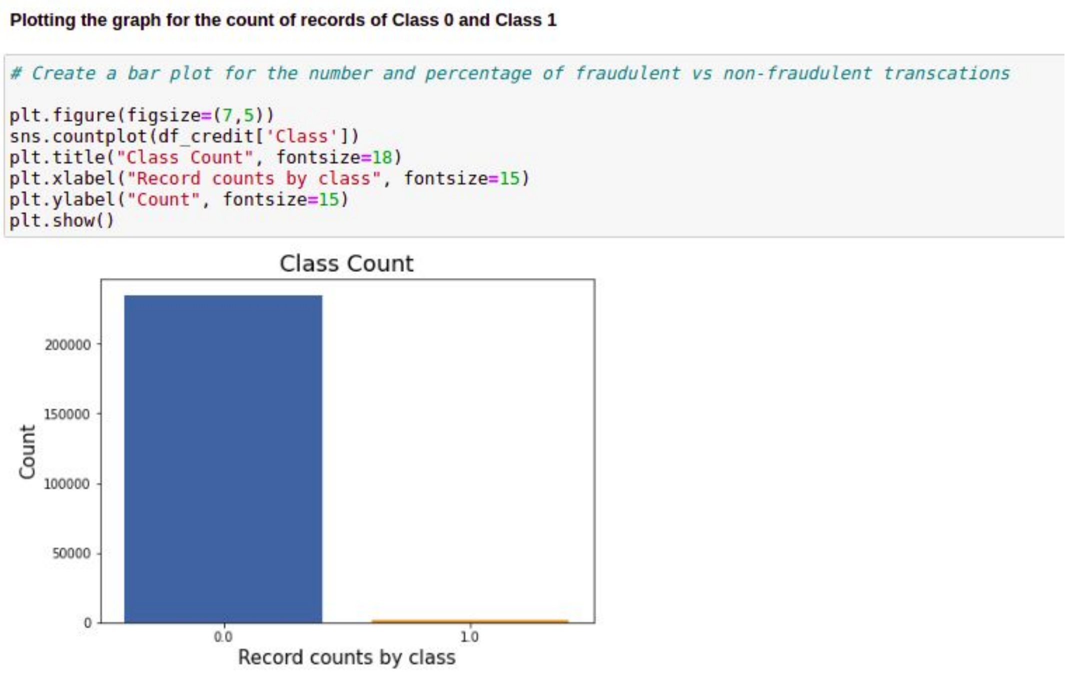
```
# Bivariate Analysis of Numerical Columns
for c in numerical_columns:
    num_col_bivariate_analysis('Class',c)
```

Fig: Bivariate Analysis of Numerical Columns

Exploratory Data Analysis

- **Record Count by class**:-

- Starting with the count of fraud and normal transaction-
- There are total 1850 fraud transactions and 234831 normal transactions in the data set.
- This comes out to be 99.2 % normal transactions and 0.8% fraud transactions.



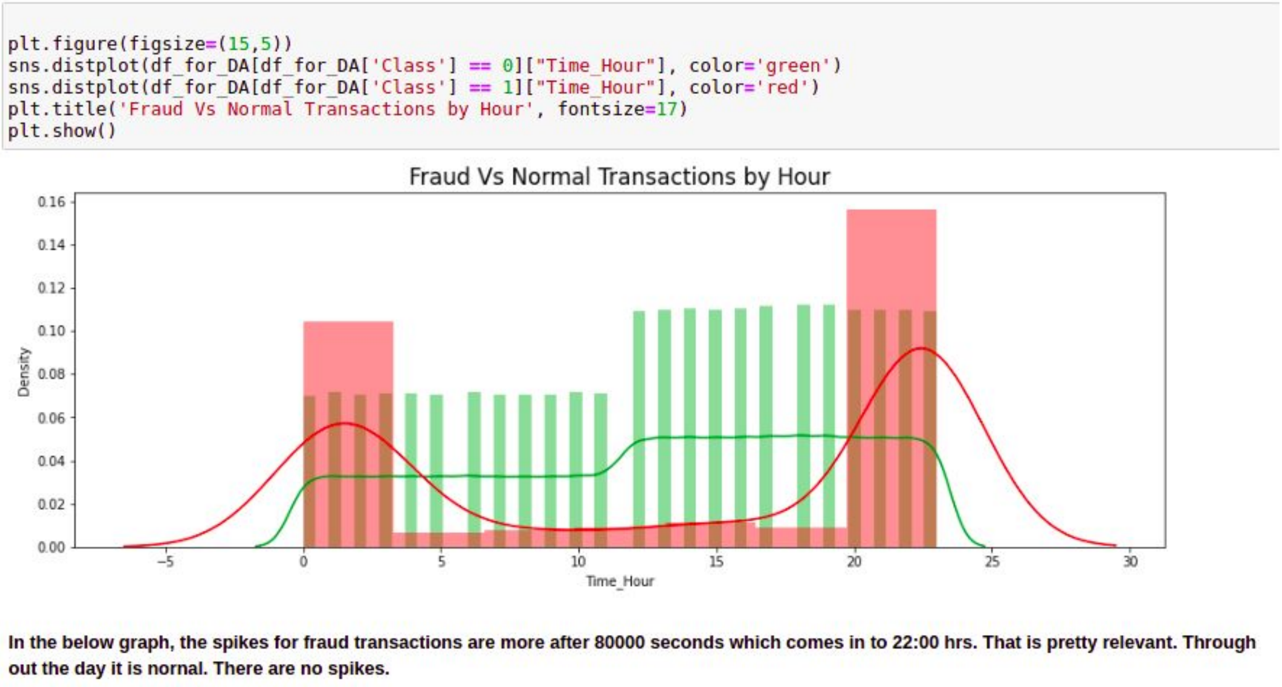
Plotting the graph for the count of records of Class 0 and Class 1

```
# Create a bar plot for the number and percentage of fraudulent vs non-fraudulent transcations

plt.figure(figsize=(7,5))
sns.countplot(df_credit['Class'])
plt.title("Class Count", fontsize=18)
plt.xlabel("Record counts by class", fontsize=15)
plt.ylabel("Count", fontsize=15)
plt.show()
```

Exploratory Data Analysis

Time range within which fraud transactions happened

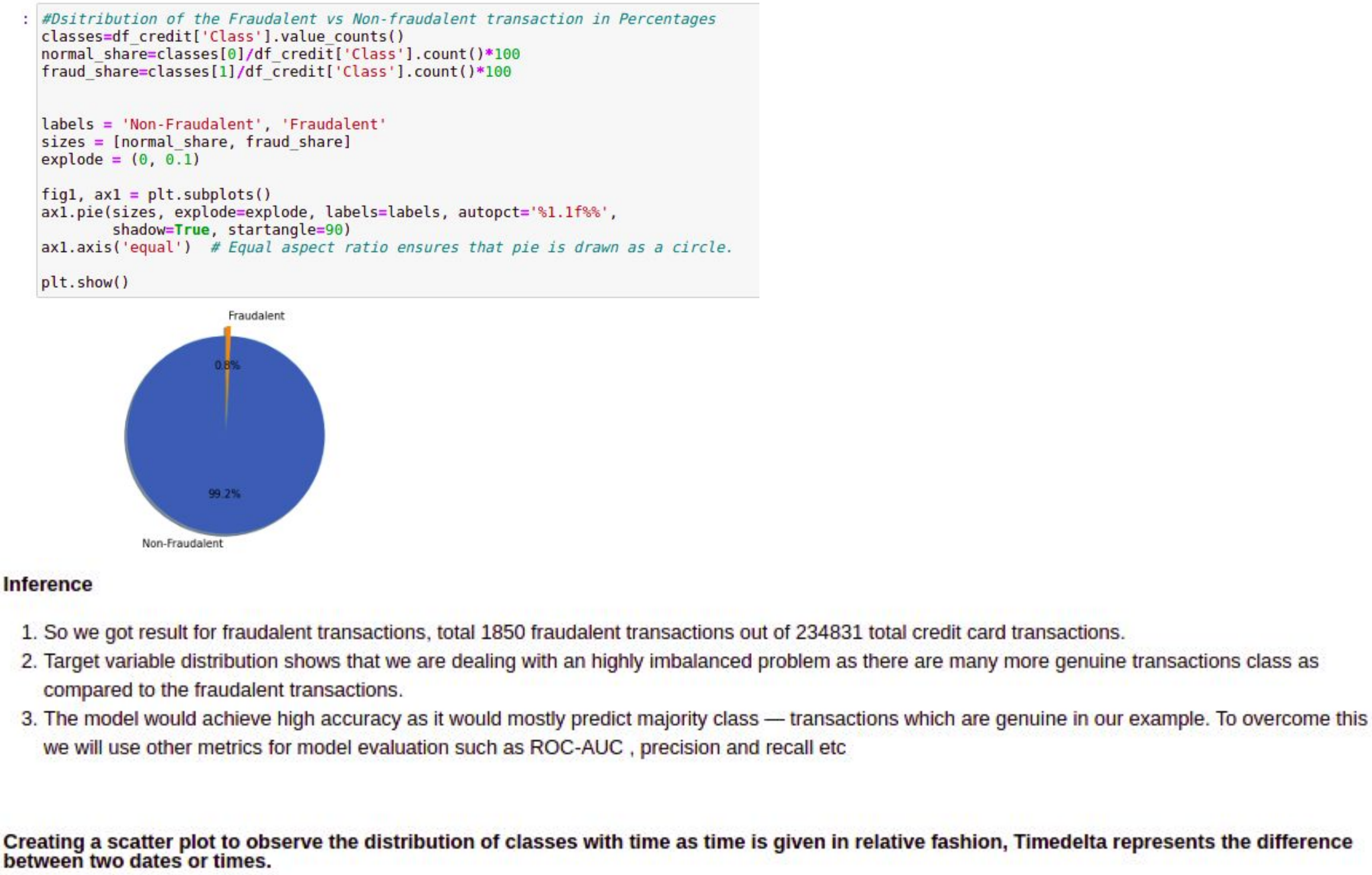
- In the below graph, the spikes for fraud transactions are more after 80000 seconds which comes out to 22:00 hrs. That is pretty good.
- All the day there are no variations in transactions.
- There are no spikes. But as the time gradually increases to early mid night,
- The frequency false transactions increase.



```
plt.figure(figsize=(15,5))
sns.distplot(df_for_DA[df_for_DA['Class'] == 0]["Time_Hour"], color='green')
sns.distplot(df_for_DA[df_for_DA['Class'] == 1]["Time_Hour"], color='red')
plt.title('Fraud Vs Normal Transactions by Hour', fontsize=17)
plt.show()
```

In the below graph, the spikes for fraud transactions are more after 80000 seconds which comes in to 22:00 hrs. That is pretty relevant. Through out the day it is nornal. There are no spikes.

Exploratory Data Analysis



```
#Dsitribution of the Fraudalent vs Non-fraudalent transaction in Percentages
classes=df_credit['Class'].value_counts()
normal_share=classes[0]/df_credit['Class'].count()*100
fraud_share=classes[1]/df_credit['Class'].count()*100


labels = 'Non-Fraudalent', 'Fraudalent'
sizes = [normal_share, fraud_share]
explode = (0, 0.1)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

Inference

1. So we got result for fraudalent transactions, total 1850 fraudalent transactions out of 234831 total credit card transactions.
2. Target variable distribution shows that we are dealing with an highly imbalanced problem as there are many more genuine transactions class as compared to the fraudalent transactions.
3. The model would achieve high accuracy as it would mostly predict majority class — transactions which are genuine in our example. To overcome this we will use other metrics for model evaluation such as ROC-AUC , precision and recall etc

Creating a scatter plot to observe the distribution of classes with time as time is given in relative fashion, Timedelta represents the difference between two dates or times.

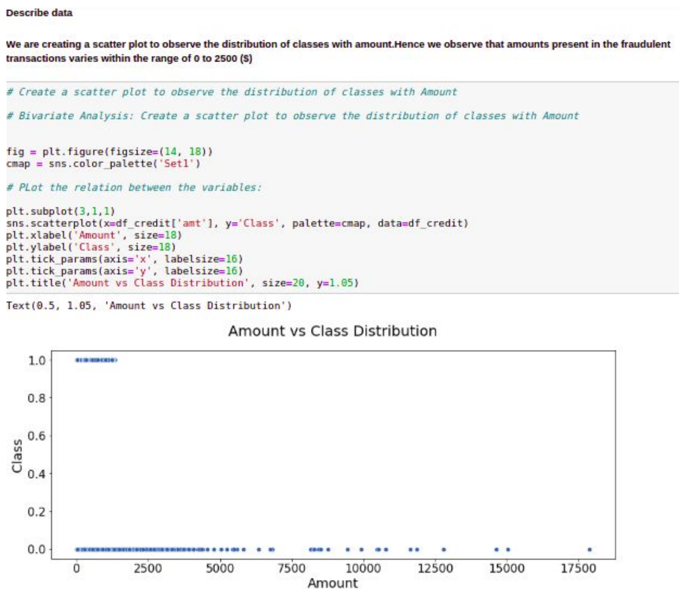
Exploratory Data Analysis

Amount versus Class Distribution:

All the fraud transactions that have happened in the data set ranges from $0 to $2500. The frauds have crossed the limit of $2500 in the plot.

Important features which helps in prediction:

The most important variables which help us to predict the model are given in the screenshot aside.

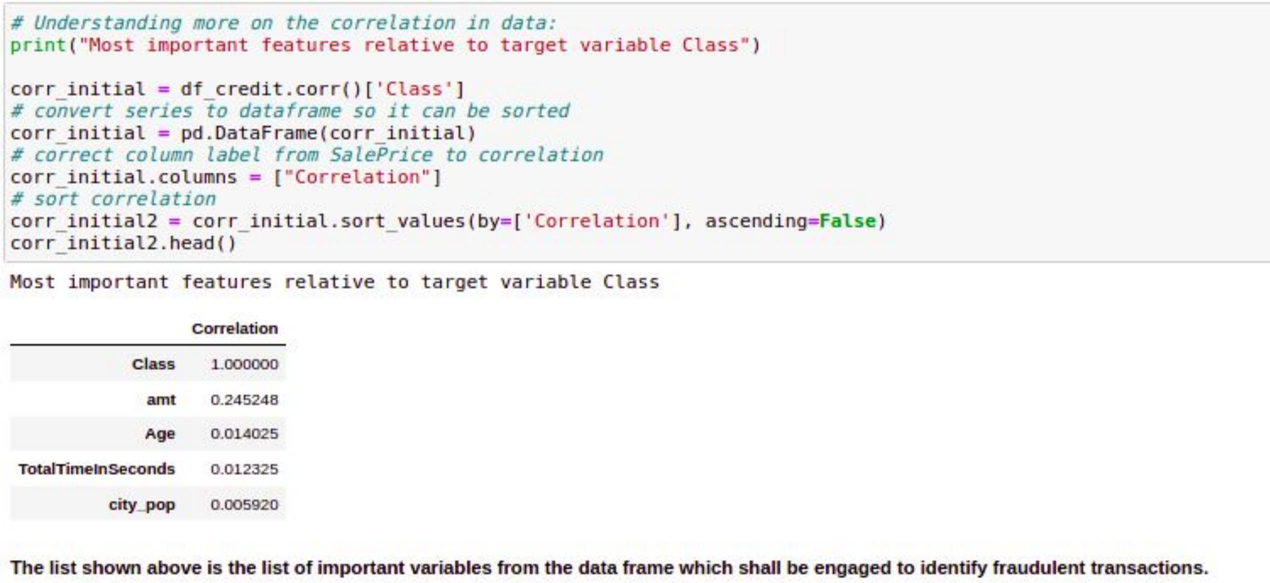


Describe data

We are creating a scatter plot to observe the distribution of classes with amount.Hence we observe that amounts present in the fraudulent transactions varies within the range of 0 to 2500 ($)

```
# Create a scatter plot to observe the distribution of classes with Amount

# Bivariate Analysis: Create a scatter plot to observe the distribution of classes with Amount


fig = plt.figure(figsize=(14, 18))
cmap = sns.color_palette('Set1')

# PLot the relation between the variables:

plt.subplot(3,1,1)
sns.scatterplot(x=df_credit['amt'], y='Class', palette=cmap, data=df_credit)
plt.xlabel('Amount', size=18)
plt.ylabel('Class', size=18)
plt.tick_params(axis='x', labelsize=16)
plt.tick_params(axis='y', labelsize=16)
plt.title('Amount vs Class Distribution', size=20, y=1.05)
```

```
Text(0.5, 1.05, 'Amount vs Class Distribution')
```



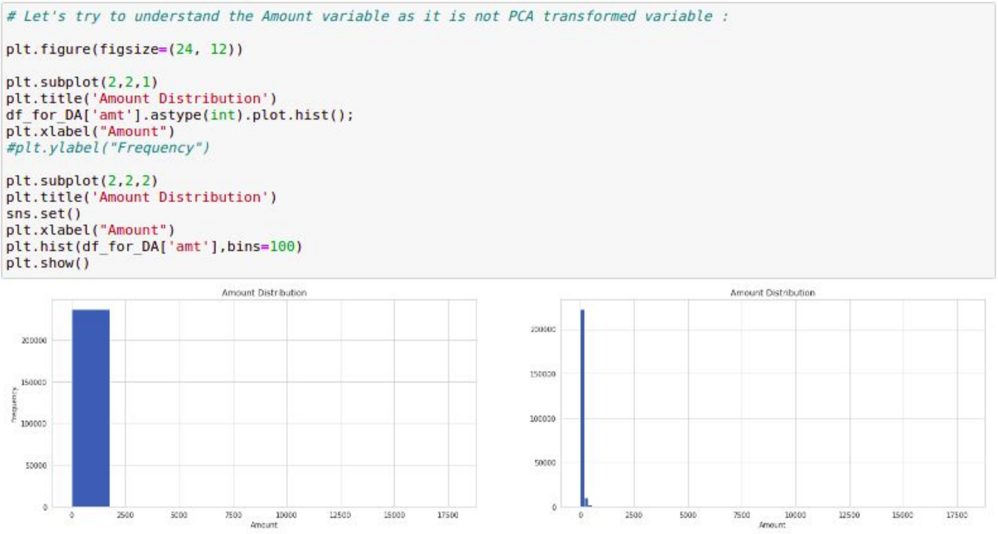
```
# Understanding more on the correlation in data:
print("Most important features relative to target variable Class")

corr_initial = df_credit.corr()['Class']
# convert series to dataframe so it can be sorted
corr_initial = pd.DataFrame(corr_initial)
# correct column label from SalePrice to correlation
corr_initial.columns = ["Correlation"]
# sort correlation
corr_initial2 = corr_initial.sort_values(by=['Correlation'], ascending=False)
corr_initial2.head()
```

```
Most important features relative to target variable Class
```

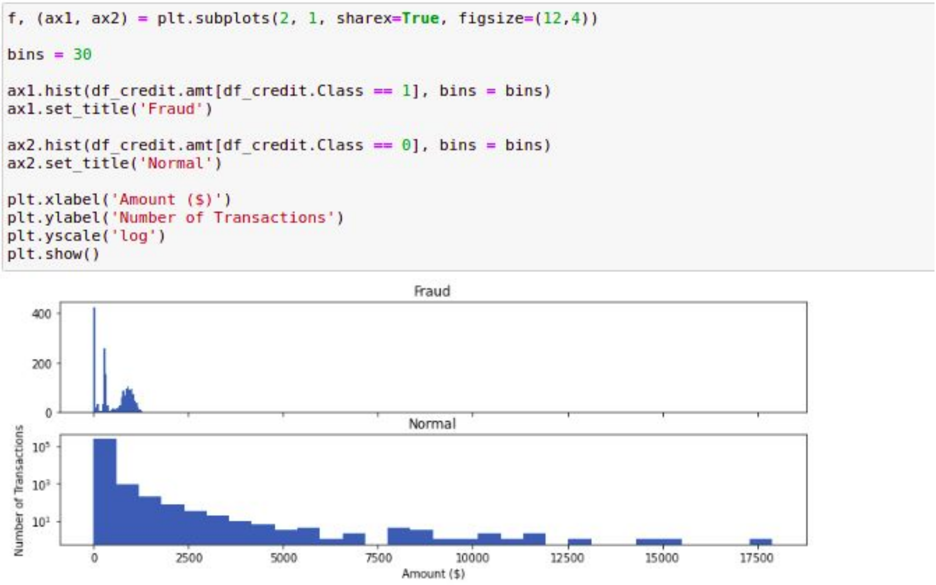
	Correlation
Class	1.000000
amt	0.245248
Age	0.014025
TotalTimeInSeconds	0.012325
city_pop	0.005920

The list shown above is the list of important variables from the data frame which shall be engaged to identify fraudulent transactions.

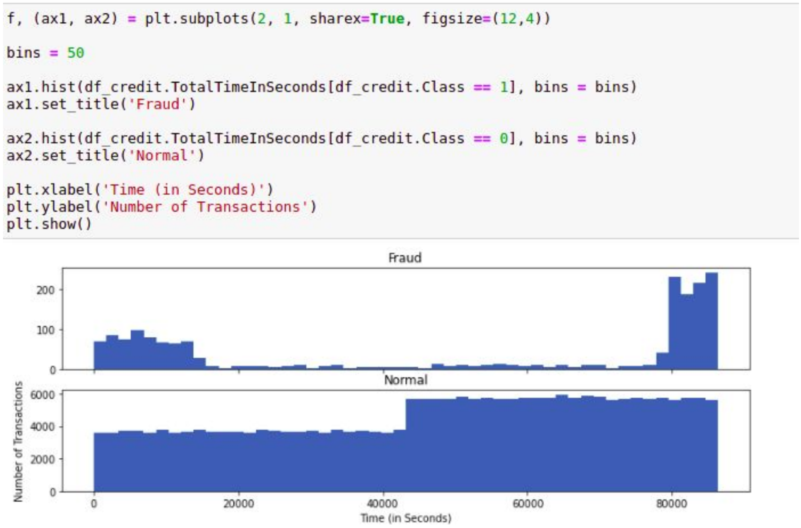
Exploratory Data Analysis



```
# Let's try to understand the Amount variable as it is not PCA transformed variable :

plt.figure(figsize=(24, 12))

plt.subplot(2,2,1)
plt.title('Amount Distribution')
df_for_DA['amt'].astype(int).plot.hist();
plt.xlabel("Amount")
#plt.ylabel("Frequency")

plt.subplot(2,2,2)
plt.title('Amount Distribution')
sns.set()
plt.xlabel("Amount")
plt.hist(df_for_DA['amt'],bins=100)
plt.show()
```



```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(12,4))

bins = 30

ax1.hist(df_credit.amt[df_credit.Class == 1], bins = bins)
ax1.set_title('Fraud')

ax2.hist(df_credit.amt[df_credit.Class == 0], bins = bins)
ax2.set_title('Normal')

plt.xlabel('Amount ($)')
plt.ylabel('Number of Transactions')
plt.yscale('log')
plt.show()
```



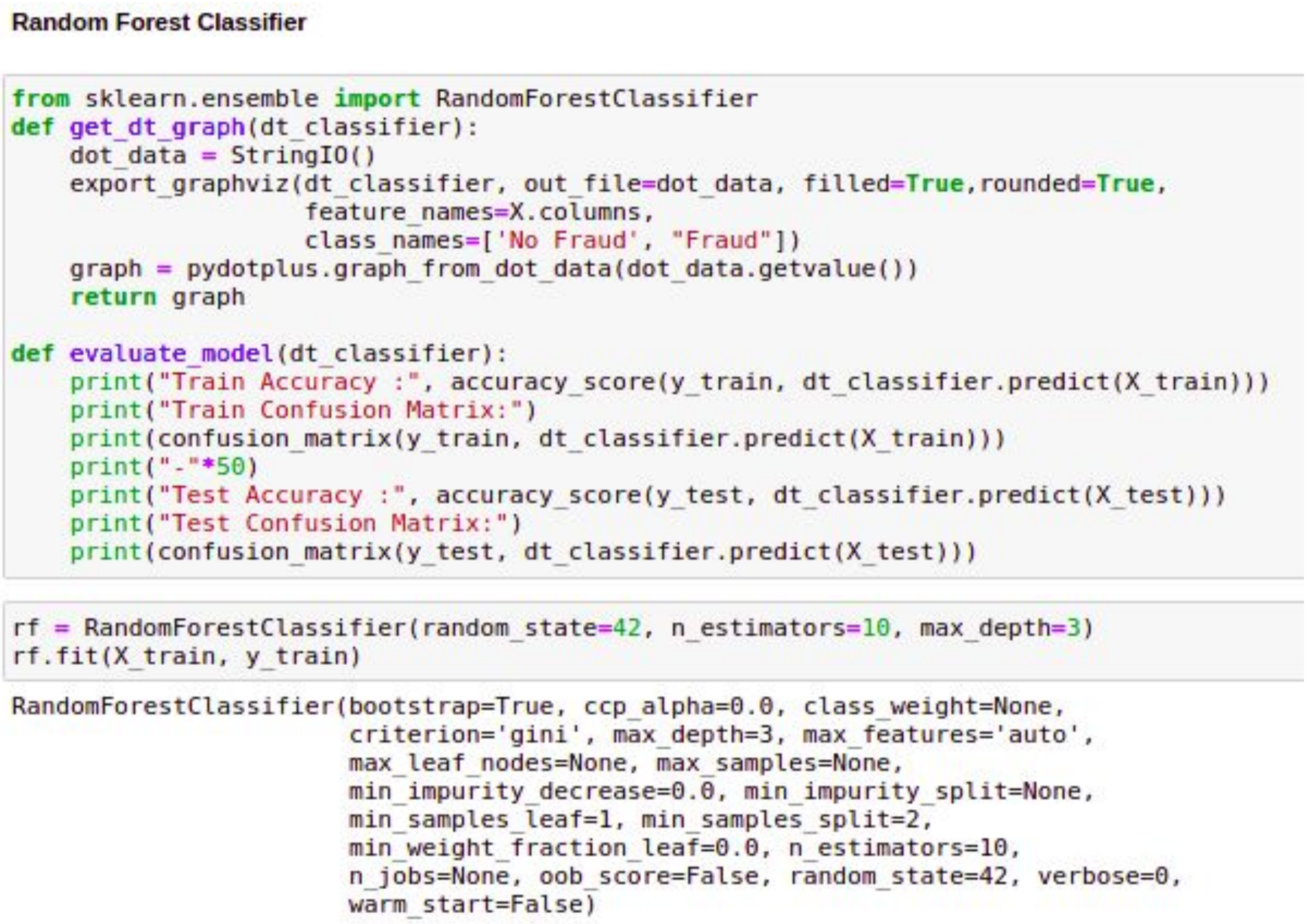
```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(12,4))

bins = 50

ax1.hist(df_credit.TotalTimeInSeconds[df_credit.Class == 1], bins = bins)
ax1.set_title('Fraud')

ax2.hist(df_credit.TotalTimeInSeconds[df_credit.Class == 0], bins = bins)
ax2.set_title('Normal')

plt.xlabel('Time (in Seconds)')
plt.ylabel('Number of Transactions')
plt.show()
```

- **Visualizations for Number of All transactions**

Train/Test Data Splitting

- The train-test split is a technique for evaluating the performance of a machine learning algorithm.
- It can be used for classification or regression problems and can be used for any supervised learning algorithm.
- The procedure involves taking a dataset and dividing it into two subsets.
- The simplest way to split the modelling dataset into training and testing sets is to assign 2/3 data points to the former and the remaining one-third to the latter.
- Therefore, we train the model using the training set and then apply the model to the test set. In this way, we can evaluate the performance of our model.

Train/Test Data Splitting



Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
def get_dt_graph(dt_classifier):
    dot_data = StringIO()
    export_graphviz(dt_classifier, out_file=dot_data, filled=True,rounded=True,
                    feature_names=X.columns,
                    class_names=['No Fraud', "Fraud"])
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    return graph

def evaluate_model(dt_classifier):
    print("Train Accuracy :", accuracy_score(y_train, dt_classifier.predict(X_train)))
    print("Train Confusion Matrix:")
    print(confusion_matrix(y_train, dt_classifier.predict(X_train)))
    print("-"*50)
    print("Test Accuracy :", accuracy_score(y_test, dt_classifier.predict(X_test)))
    print("Test Confusion Matrix:")
    print(confusion_matrix(y_test, dt_classifier.predict(X_test)))
```

```
rf = RandomForestClassifier(random_state=42, n_estimators=10, max_depth=3)
rf.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=3, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=42, verbose=0,
                       warm_start=False)
```

Model Selection

- Before building any model, we had to clean the data and perform the PCA so that models can be applied on the data set.

Steps followed to build the model:-

- Cleaned the data set and performed the PCA.
- We have divided the data into train set and test/validation set. (Train set:70% and Test set:30%)
- Then we moved ahead creating trees for Decision tree model and Random Forest model.
- We have used the approach of Repeated K –Fold and Stratified K-Fold to implement models like KNN Model Classifier, Decision Tree Model, Random Forest Model.

Model Selection

KNN Model:-

- The abbreviation KNN stands for "K-Nearest Neighbour". It is a supervised machine learning algorithm. The algorithm can be used to solve both classification and regression problem statements. The number of nearest neighbours to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'.

Decision Tree Model:

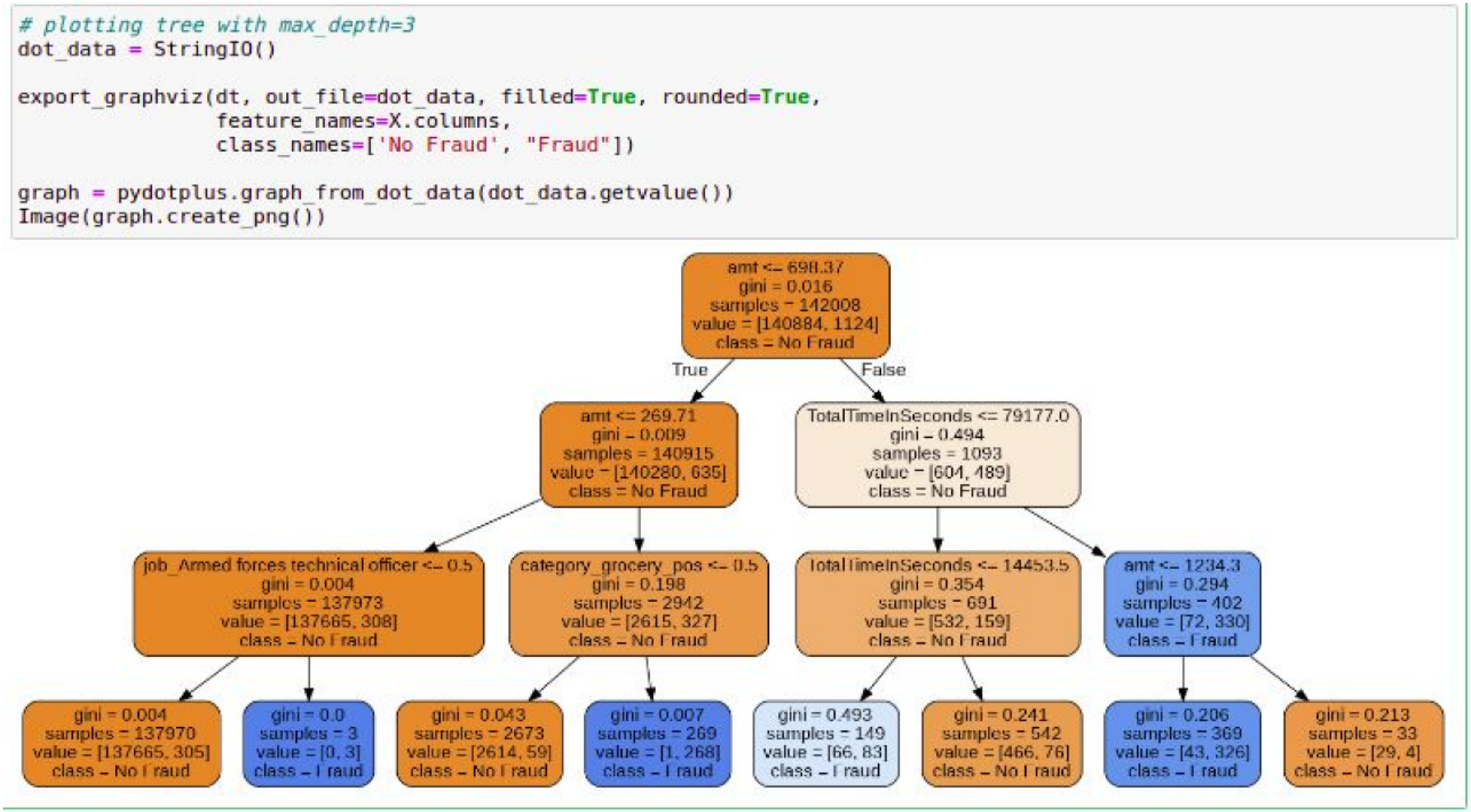
- A decision tree is a flowchart-like structure in which each internal node represents a test on a feature, each leaf node represents a class label and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules.

Random Forest Model:

- A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

Model Selection and Model Building

Decision Tree Model on the current data set:

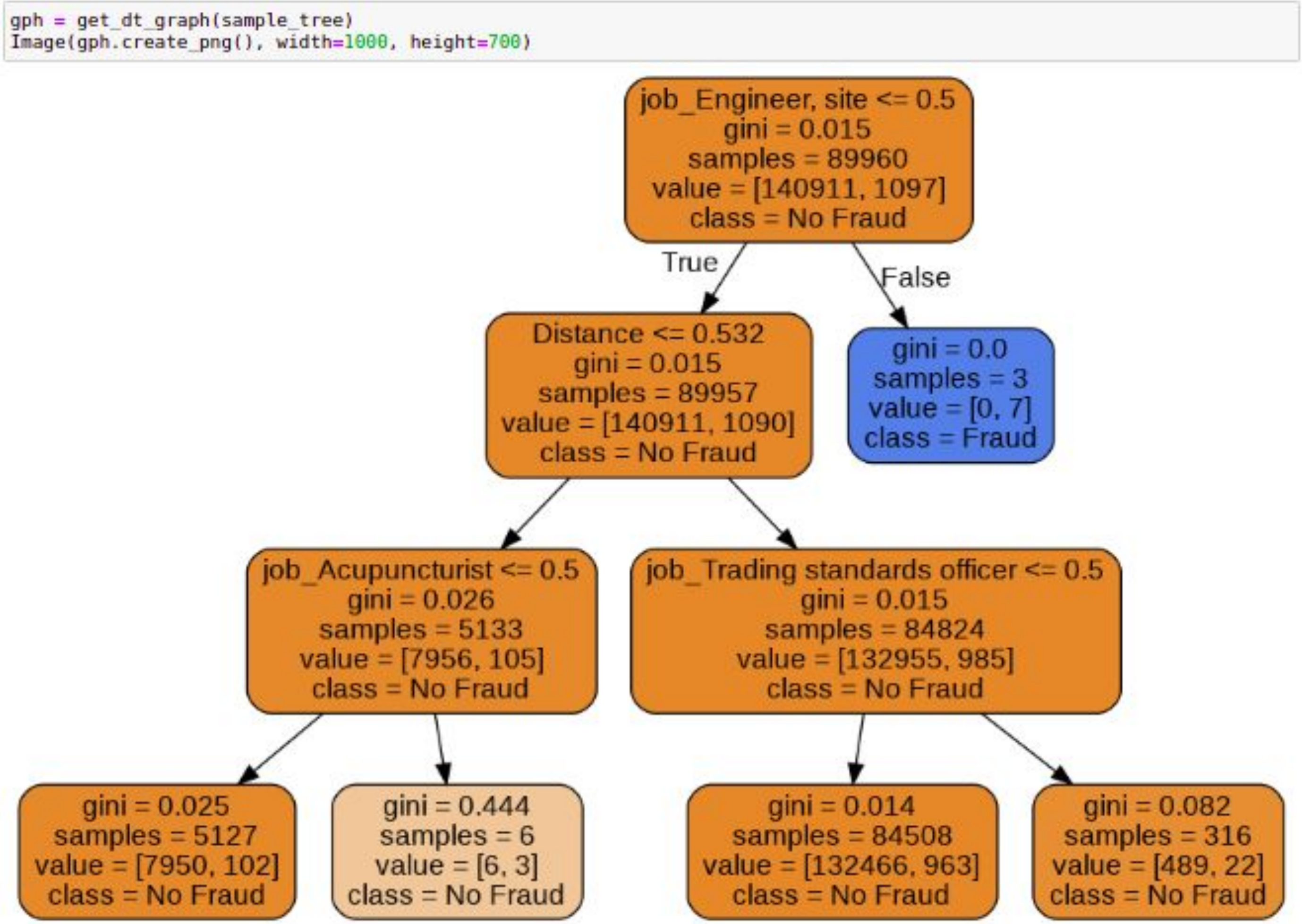


```
# plotting tree with max_depth=3
dot_data = StringIO()

export_graphviz(dt, out_file=dot_data, filled=True, rounded=True,
                feature_names=X.columns,
                class_names=['No Fraud', "Fraud"])

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

In the above chart shown, we can see the various conditions and criterias applied to the data to declare the record/transaction as fraud.

Model Selection and Model Building

Random Forest Model on the current data set:



```
gph = get_dt_graph(sample_tree)
Image(gph.create_png(), width=1000, height=700)
```

job_Engineer, site <= 0.5
gini = 0.015
samples = 89960
value = [140911, 1097]
class = No Fraud

True

False

Distance <= 0.532
gini = 0.015
samples = 89957
value = [140911, 1090]
class = No Fraud

gini = 0.0
samples = 3
value = [0, 7]
class = Fraud

job_Acupuncturist <= 0.5
gini = 0.026
samples = 5133
value = [7956, 105]
class = No Fraud

job_Trading standards officer <= 0.5
gini = 0.015
samples = 84824
value = [132955, 985]
class = No Fraud

gini = 0.025
samples = 5127
value = [7950, 102]
class = No Fraud

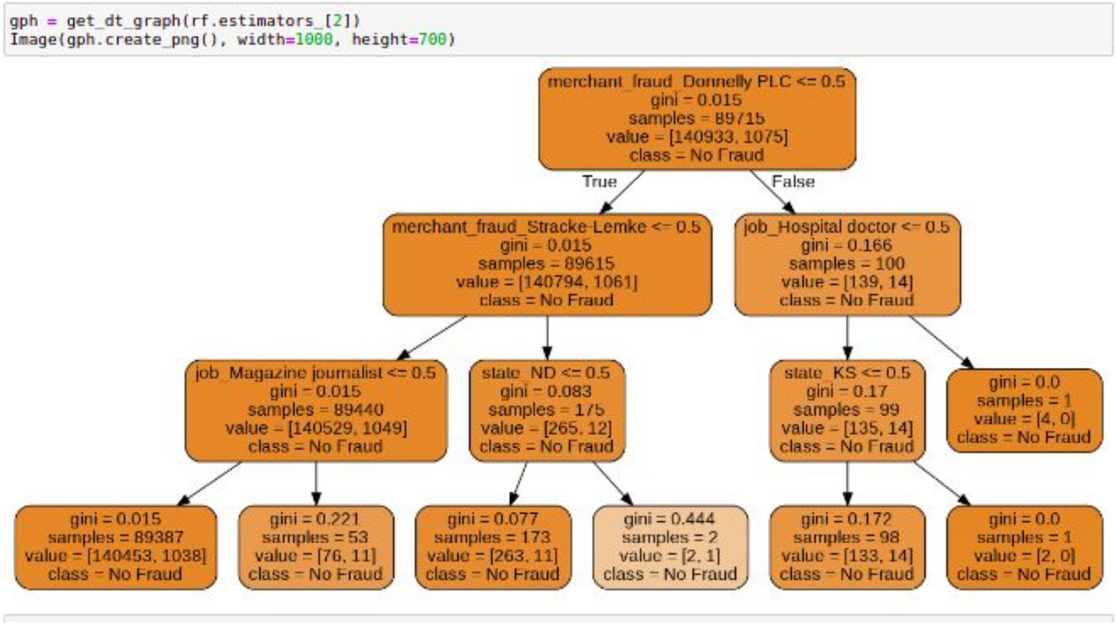
gini = 0.444
samples = 6
value = [6, 3]
class = No Fraud

gini = 0.014
samples = 84508
value = [132466, 963]
class = No Fraud

gini = 0.082
samples = 316
value = [489, 22]
class = No Fraud

Model Selection and Model Building

Random Forest Model on the current data set:

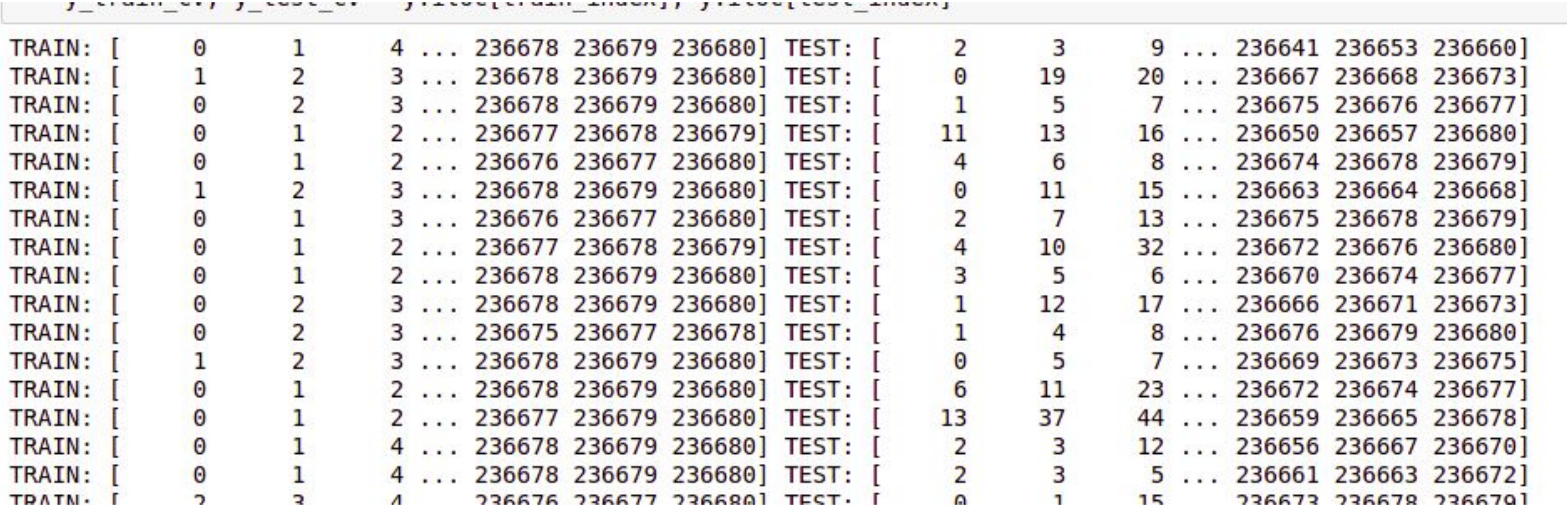


```
gph = get_dt_graph(rf.estimators_[2])
Image(gph.create_png(), width=1000, height=700)
```

In the above random forest tree, we can see few nodes which are frauds (blue in color). But this model does not seems to be effective enough to catch fraud transactions. So we are going to use Repetitive K-Fold and Stratified K-Fold approach and implement both the models again and capture the results in further slides.

Model Selection and Model Building

- **Repetitive K Fold approach applied on current data set :**
- Repetitive K Fold applied to the model:



```
    y_train_cv, y_test_cv = y.iloc[train_index], y.iloc[test_index]

TRAIN: [     0      1      4 ... 236678 236679 236680] TEST: [     2      3      9 ... 236641 236653 236660]
TRAIN: [     1      2      3 ... 236678 236679 236680] TEST: [     0     19     20 ... 236667 236668 236673]
TRAIN: [     0      2      3 ... 236678 236679 236680] TEST: [     1      5      7 ... 236675 236676 236677]
TRAIN: [     0      1      2 ... 236677 236678 236679] TEST: [    11     13     16 ... 236650 236657 236680]
TRAIN: [     0      1      2 ... 236676 236677 236680] TEST: [     4      6      8 ... 236674 236678 236679]
TRAIN: [     1      2      3 ... 236678 236679 236680] TEST: [     0     11     15 ... 236663 236664 236668]
TRAIN: [     0      1      3 ... 236676 236677 236680] TEST: [     2      7     13 ... 236675 236678 236679]
TRAIN: [     0      1      2 ... 236677 236678 236679] TEST: [     4     10     32 ... 236672 236676 236680]
TRAIN: [     0      1      2 ... 236678 236679 236680] TEST: [     3      5      6 ... 236670 236674 236677]
TRAIN: [     0      2      3 ... 236678 236679 236680] TEST: [     1     12     17 ... 236666 236671 236673]
TRAIN: [     0      2      3 ... 236675 236677 236678] TEST: [     1      4      8 ... 236676 236679 236680]
TRAIN: [     1      2      3 ... 236678 236679 236680] TEST: [     0      5      7 ... 236669 236673 236675]
TRAIN: [     0      1      2 ... 236678 236679 236680] TEST: [     6     11     23 ... 236672 236674 236677]
TRAIN: [     0      1      2 ... 236677 236679 236680] TEST: [    13     37     44 ... 236659 236665 236678]
TRAIN: [     0      1      4 ... 236678 236679 236680] TEST: [     2      3     12 ... 236656 236667 236670]
TRAIN: [     0      1      4 ... 236678 236679 236680] TEST: [     2      3      5 ... 236661 236663 236672]
TRAIN: [     2      3      4 ... 236676 236677 236680] TEST: [     0      1     15 ... 236673 236678 236679]
```

Model Selection and Model Building

Inference

Perform cross validation with Repeated K-Fold:-

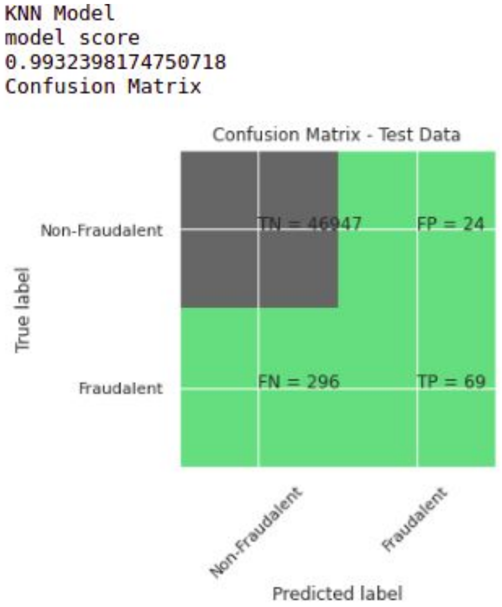
- The k-fold cross-validation procedure is a standard method for estimating the performance of a ML algorithm or configuration on a dataset.A single run of the k-fold cross-validation procedure may result in a noisy estimate of model performance. Different splits of the data may result in very different results.

- Repeated k-fold cross-validation provides a way to improve the estimated performance of a machine learning model.This involves simply repeating the cross-validation procedure multiple times and reporting the mean result across all folds from all runs.

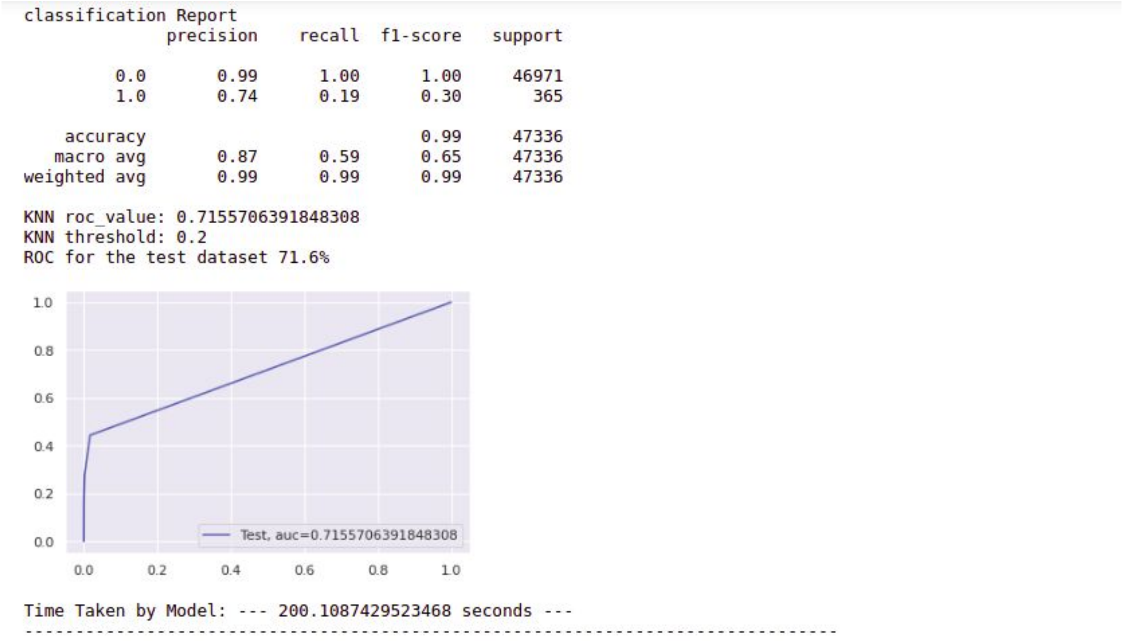
- This mean result is expected to be a more accurate estimate of the true unknown underlying mean performance of the model on the dataset, as calculated using the standard error.

Model Selection and Model Building

KNN Model on the current data set using Repetitive K Fold approach:-



```
KNN Model
model score
0.9932398174750718
Confusion Matrix
```



```
classification Report
              precision    recall  f1-score   support

         0.0       0.99      1.00      1.00     46971
         1.0       0.74      0.19      0.30       365

    accuracy                           0.99     47336
   macro avg       0.87      0.59      0.65     47336
weighted avg       0.99      0.99      0.99     47336

KNN roc_value: 0.7155706391848308
KNN threshold: 0.2
ROC for the test dataset 71.6%
```

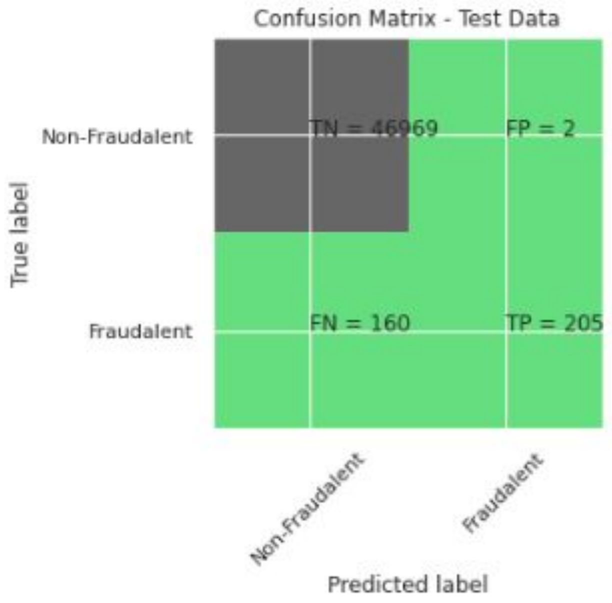
```
Time Taken by Model: --- 200.1087429523468 seconds ---
--------------------------------------------------------------------------------
```

The area under the curve for this model is 71.55% which is not as good as we are expecting. We are expecting a higher value for AUC.

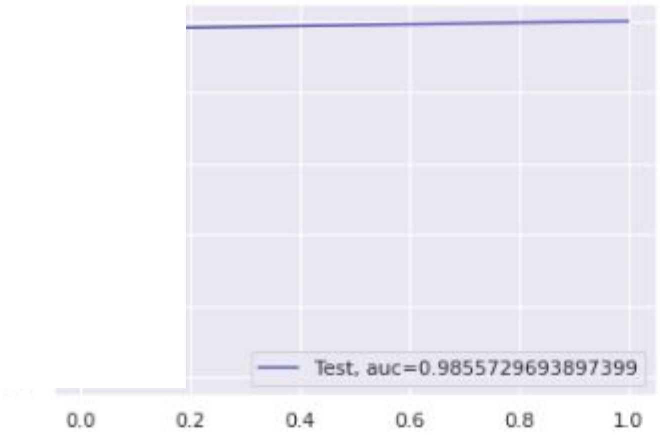
Model Selection and Model Building

- **Random Model on the current data set using Repetitive K Fold approach**:-

```
Random Forest Model
Model Accuracy: 0.9965776575967551
Confusion Matrix
```



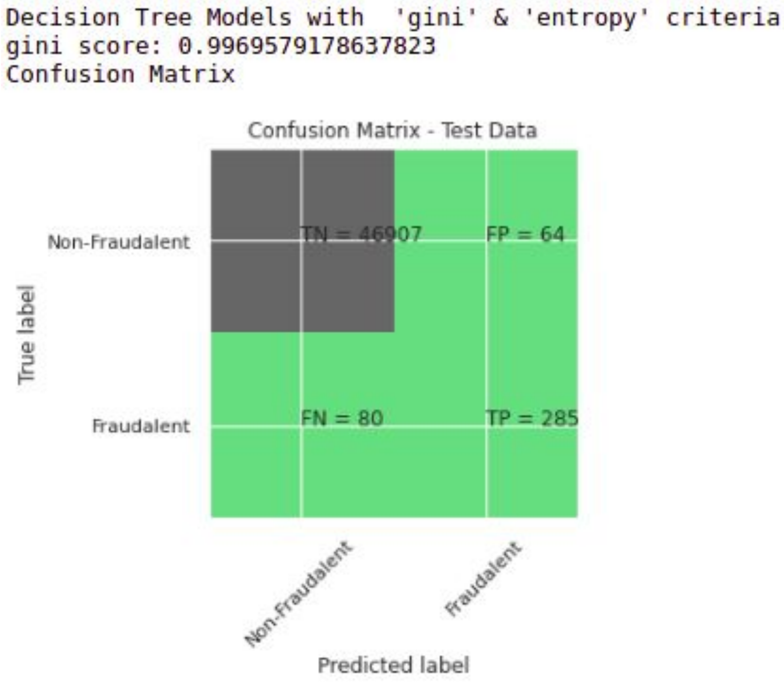
```
classification Report
             precision    recall  f1-score   support

                  1.00      1.00      1.00     46971
                  0.99      0.56      0.72       365

                                      1.00     47336
                  0.99      0.78      0.86     47336
                  1.00      1.00      1.00     47336

: roc_value: 0.9855729693897399
: threshold: 0.04
:est dataset 98.6%
```



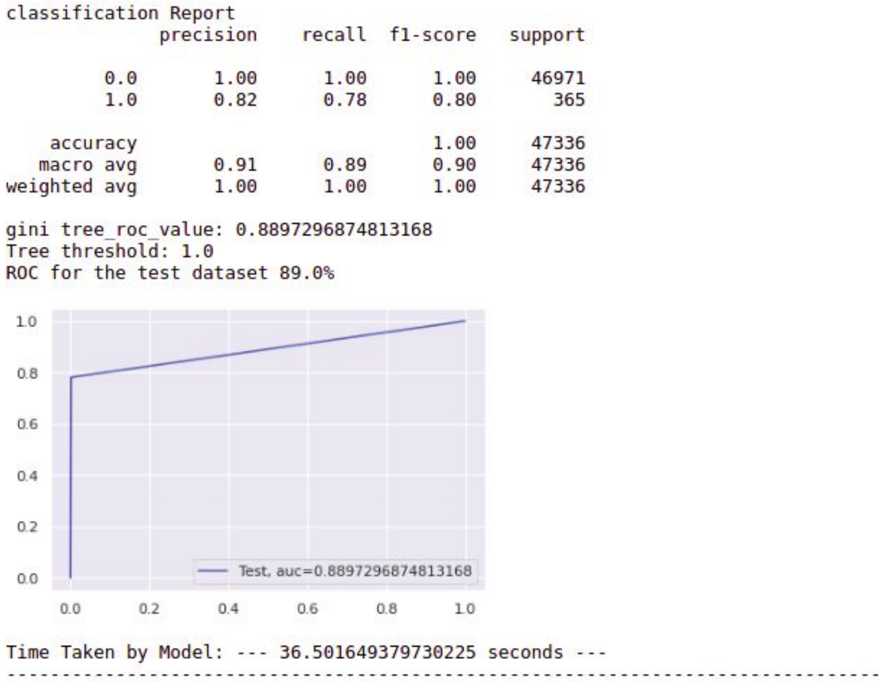
```
Time Taken by Model: --- 203.47433519363403 seconds ---
--------------------------------------------------------------------------------
```

Model Selection and Model Building

- **Decision Tree Model on the current data set using Repetitive K Fold approach:-**



```
Decision Tree Models with  'gini' & 'entropy' criteria
gini score: 0.9969579178637823
Confusion Matrix
```

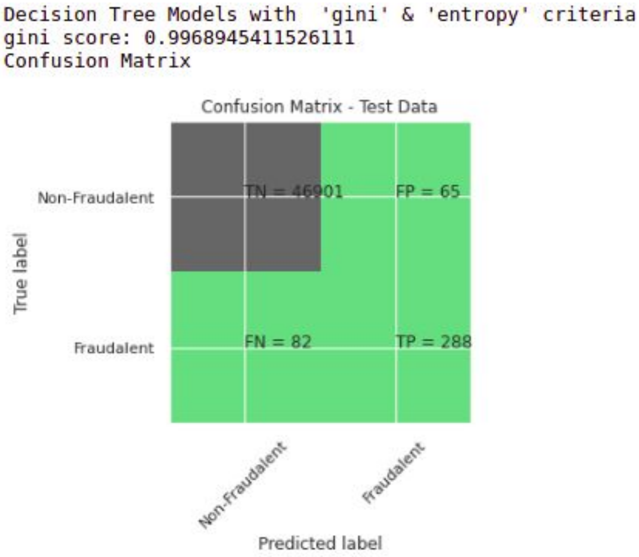


```
classification Report
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     46971
         1.0       0.82      0.78      0.80       365

    accuracy                           1.00     47336
   macro avg       0.91      0.89      0.90     47336
weighted avg       1.00      1.00      1.00     47336

gini tree_roc_value: 0.8897296874813168
Tree threshold: 1.0
ROC for the test dataset 89.0%
```

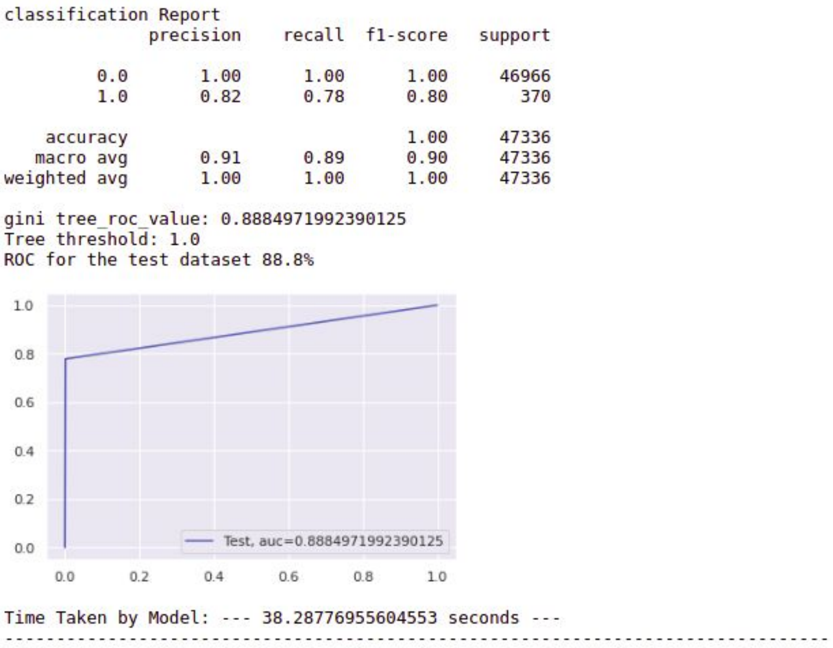
```
Time Taken by Model: --- 36.501649379730225 seconds ---
--------------------------------------------------------------------------------
```

Model Selection and Model Building

Decision Tree Model on the current data set using Repetitive K Fold approach:-



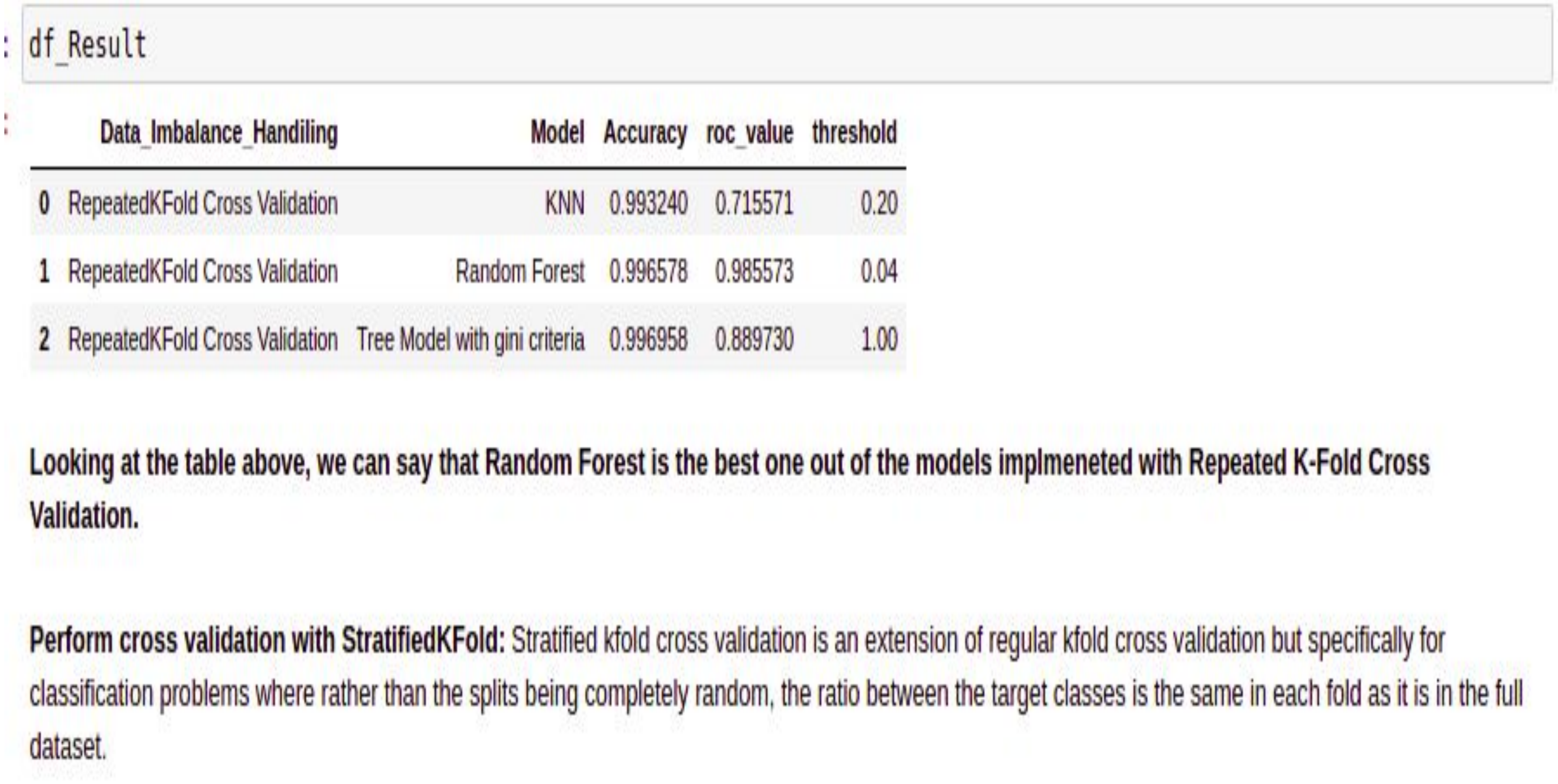
```
Decision Tree Models with  'gini' & 'entropy' criteria
gini score: 0.9968945411526111
Confusion Matrix
```



```
classification Report
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     46966
         1.0       0.82      0.78      0.80       370

    accuracy                           1.00     47336
   macro avg       0.91      0.89      0.90     47336
weighted avg       1.00      1.00      1.00     47336

gini tree_roc_value: 0.8884971992390125
Tree threshold: 1.0
ROC for the test dataset 88.8%
```

```
Time Taken by Model: --- 38.28776955604553 seconds ---
--------------------------------------------------------------------------------
```

Model Evaluation



```
df_Result
```

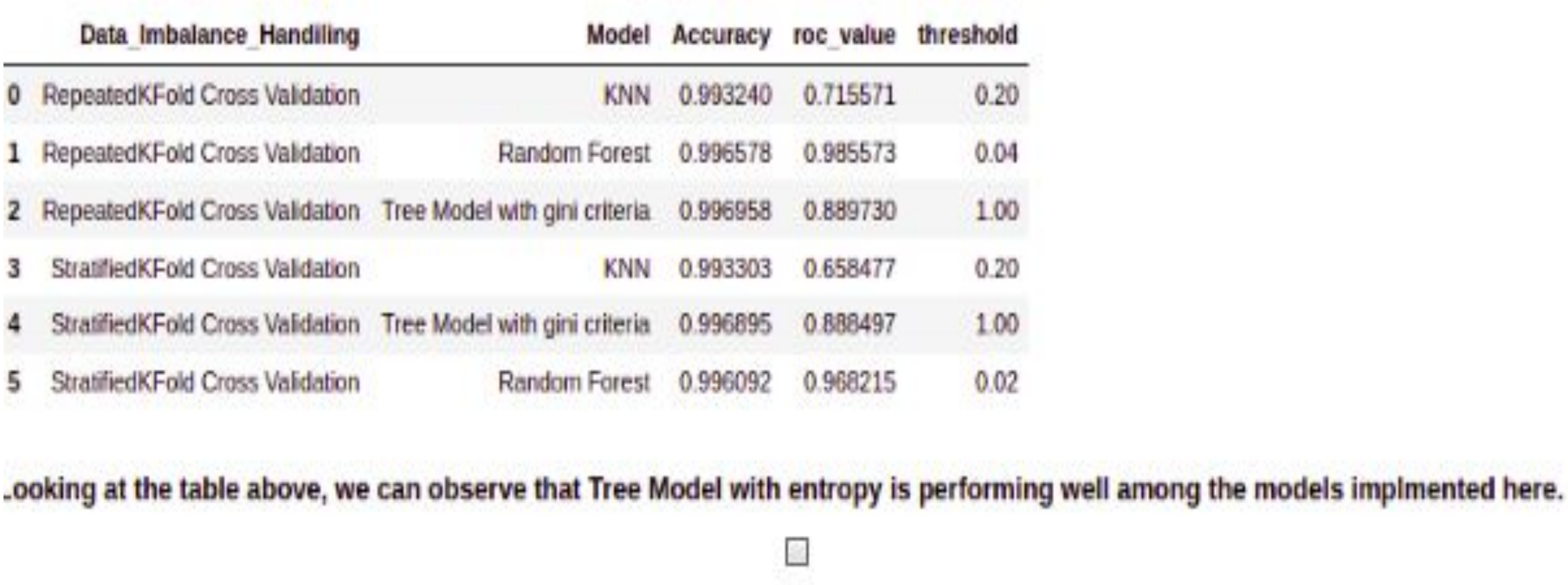
	Data_Imbalance_Handiling	Model	Accuracy	roc_value	threshold
0	RepeatedKFold Cross Validation	KNN	0.993240	0.715571	0.20
1	RepeatedKFold Cross Validation	Random Forest	0.996578	0.985573	0.04
2	RepeatedKFold Cross Validation	Tree Model with gini criteria	0.996958	0.889730	1.00

Looking at the table above, we can say that Random Forest is the best one out of the models implmeneted with Repeated K-Fold Cross Validation.

Perform cross validation with StratifiedKFold: Stratified kfold cross validation is an extension of regular kfold cross validation but specifically for classification problems where rather than the splits being completely random, the ratio between the target classes is the same in each fold as it is in the full dataset.

Model Evaluation

- **Conclusion on the Stratified K-Fold approach**:-



	Data_Imbalance_Handiling	Model	Accuracy	roc_value	threshold
0	RepeatedKFold Cross Validation	KNN	0.993240	0.715571	0.20
1	RepeatedKFold Cross Validation	Random Forest	0.996578	0.985573	0.04
2	RepeatedKFold Cross Validation	Tree Model with gini criteria	0.996958	0.889730	1.00
3	StratifiedKFold Cross Validation	KNN	0.993303	0.658477	0.20
4	StratifiedKFold Cross Validation	Tree Model with gini criteria	0.996895	0.888497	1.00
5	StratifiedKFold Cross Validation	Random Forest	0.996092	0.968215	0.02

.ooking at the table above, we can observe that Tree Model with entropy is performing well among the models implmented here.

Inference

- Here, we see that KNN model is performing poorly with Stratified K-Fold cross validation method.Hence, we cannot consider this model as the appropriate one. The AOC percentage is 65.847% which is not good at all.
- Machine learning model tends to become more stable for the data set with the increase in the AUC score.
- Gini Index, also known as Gini impurity, calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly. If all the elements are linked with a single class then it can be called pure.
- Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information.

Inference

- Load two dataset and merged these into one and cleaned it by using Exploratory data Analysis.
- Using univariate and bivariate analysis for EDA.
- Then Divided the cleaned data set into validation data set and train data set. then Build few models using the data set and also using various approaches(Repeated K-Fold & Stratified K-Fold).
- After these steps, we understand and decided for conclusion using different approaches, that most suitable of Machine learning model depends upon the AUC score.
- Now Decided that Using Repeated k-Fold, Random Forest Classifier proves to be the suitable model among all the others models implemented here.
- Using Stratified k-Fold, Random Forest Classifier proves to be the suitable model among all the others models implemented here.

Conclusion

- Credit card fraud is without a doubt an act of criminal dishonesty.
- In this scenario, after handling On a high level fraud data, these models can help the bank to identify the frauds and notify the customer about it.
- Fraud detection is a complex issue that requires a substantial amount of planning before throwing machine learning algorithms at it. Nonetheless, it is also an application of data science and machine learning for the good, which makes sure that the customer's money is safe and not easily tampered with.
- There are certain advanced models which can also be used to idenity frauds in Credit Card transaction scenarios.