**INTRODUCTION:**

# Fraud Detection on Bank Payments:

# Fraud and detecting it:

Fraudulent behaviour can be seen across many different fields such as e-commerce, healthcare, payment and banking systems. Fraud is a billion-dollar business and it is increasing every year. The PwC global economic crime survey of 2018 [1] found that half (49 percent) of the 7,200 companies they surveyed had experienced fraud of some kind.

Even if fraud seems to be scary for businesses it can be detected using intelligent systems such as rules engines or machine learning. Most people here in Kaggle are familiar with machine learning but for rule engines here is a quick information. A rules engine is a software system that executes one or more business rules in a runtime production environment. These rules are generally written by domain experts for transferring the knowledge of the problem to the rules engine and from there to production. Two rules examples for fraud detection would be limiting the number of transactions in a time period (velocity rules), denying the transactions which come from previously known fraudulent IP's and/or domains.

Rules are great for detecting some type of frauds but they can fire a lot of false positives or false negatives in some cases because they have predefined threshold values. For example let's think of a rule for denying a transaction which has an amount that is bigger than 10000 dollars for a specific user. If this user is an experienced fraudster, he/she may be aware of the fact that the system would have a threshold and he/she can just make a transaction just below the threshold value (9999 dollars).

For these type of problems ML comes for help and reduce the risk of frauds and the risk of business to lose money. With the combination of rules and machine learning, detection of the fraud would be more precise and confident.

**MOTIVATION:**

**Purpose**: In the wake of big corporate houses failing to evade corporate frauds, a study was needed to identify primary factors leading to fraud. The main purpose of this paper is to identify the major factors and motivations for fraud in the corporate sector.

**Design/Methodology/Approach**: There is a big pool of literature regarding the motivation of fraud. This paper aims to consolidate and conduct a literature review to identify the most prominent motivation factors resulting into fraud in the corporate sector. The data for the study was collected from secondary sources such as books, journals, reports etc.

**Findings**: The study formulated a new fraud motivation model, where the integrity of the manager or the individual is given more importance than other factors. The study also identified various individual and business pressures that could motivate the managers or individuals that could lead to irrational behaviour and fraud. The study identifies integrity as the most important factor for that motivates an individual to commit fraud.

**Research Limitation/Implications**: The study is based on secondary source of data. The model prepared is yet to be applied.

**Originality/Value**: A new fraud motivation model based on integrity, pressure, opportunity, capability and rationalization was structured.

**TOOLS:**

* **Anaconda**
* **Jupyter Notebook**
* **Banksim datasets**
* **K-Neighbours**

**DATA MINING:**

Definition simple words, data mining is defined as a process used to extract usable data from a larger set of any raw data. It implies analysing data patterns in large batches of data using one or more software. Data mining has applications in multiple fields, like science and research. As an application of data mining, businesses can learn more about their customers and develop more effective strategies related to various business functions and in turn leverage resources in a more optimal and insightful manner. This helps businesses be closer to their objective and make better decisions. Data mining involves effective data collection and warehousing as well as computer processing. For segmenting the data and evaluating the probability of future events, data mining uses sophisticated mathematical algorithms. Data mining is also known as Knowledge Discovery in Data (KDD).

**Algorithms:**

* **C4.5 Algorithm**
* **K-mean Algorithm**
* **KNN Algorithm[nearest neighbour]**
* **Apriori algorithm**
* **FP-Growth Algorithm**
* **Scalable Clustering Algorithm**

## **DATA MINING FUNCTIONALITIES**:

There are a number of data mining functionalities that the organized and scientific methods offer. Let us look at a few major ones.

1. Classification
2. [Association Analysis](https://www.jigsawacademy.com/blogs/data-science/data-mining-functionalities#Association-Analysis)
3. [Cluster Analysis](https://www.jigsawacademy.com/blogs/data-science/data-mining-functionalities#Cluster-Analysis)
4. [Class/Concept Description: Characterization and Discrimination](https://www.jigsawacademy.com/blogs/data-science/data-mining-functionalities#Class-Concept-Description-Characterization-and-Discrimination)
5. [Prediction](https://www.jigsawacademy.com/blogs/data-science/data-mining-functionalities#Prediction)
6. [Outlier Analysis](https://www.jigsawacademy.com/blogs/data-science/data-mining-functionalities#Outlier-Analysis)
7. [Evolution & Deviation Analysi](https://www.jigsawacademy.com/blogs/data-science/data-mining-functionalities#Evolution-&-Deviation-Analysis)[s](https://www.jigsawacademy.com/blogs/data-science/data-mining-functionalities#Evolution-&-Deviation-Analysis)

1. Classification:

Classification is a data mining technique that categorizes items in a collection, based on some predefined properties. A training set containing items whose properties are known and is used to train the system to predict the category of items from an unknown collection of items. It uses methods like if-then, decision trees or neural networks to predict a class or essentially classify a collection of items.

### 2. **Association Analysis**:

Also known as Market Basket Analysis for its wide use in retail sales, Association analysis aims to discover associations between items occurring together frequently. Association analysis is based on rules having 2 parts:

1. antecedent (if)
2. consequent(then)

### 3. **Cluster Analysis**:

Cluster Analysis, fundamentally similar to classification, where similar data are grouped together with the difference being that a class label is not known. Clustering algorithms group data based on similar features and dissimilarities. Used in image processing, pattern recognition and bioinformatics, clustering is a popular functionality of data mining.

### **4**. **Class/Concept Description: Characterization and Discrimination:**

A class or concept implies there is a data set or set of features that define the class or a concept. A class can be a category of items on a shop floor, and concept could be the abstract idea on which data may be categorized like products to be put on clearance sale and non-sale products. There are two concepts here, one that helps with grouping and the other that helps in differentiating.

* Data Characterization:

Characterization involves summarization of general features of the data, resulting in specific rules that define a target class. A data analysis technique called Attribute-oriented Induction is employed on the data set for achieving characterization.

* Data Discrimination:

Discrimination is used to separate distinct sets of data, based on the disparity in attribute values. It is a comparison of features of a class with features of one or more contrasting classes.

### 5. **Prediction**:

### In data mining, there are primarily two types of predictions, numeric predictions and class predictions. Numeric predictions are made by creating a linear regression model that is based on historical data. Prediction of numeric values helps businesses ramp up for a future event that might impact business in a positive or a negative way. Class predictions are uses to fill in missing class information for products using a training data set where the class for products is known.

### 6. **Outlier Analysis:**

Outlier analysis is important to understand the quality of data. If there are too many outliers, you cannot trust the data or draw patterns out of it. An outlier analysis of the data that cannot be grouped into any classes by the algorithms is pulled up.

**TOOLS:**

**Jupyter Notebook:**

This article provides a high-level overview of [Project Jupyter](https://jupyter.org/) and the widely popular Jupyter notebook technology. The overarching message I’d like to convey is why you should be using Jupyter for your data science projects.

Jupyter is a nonprofit organization created to “develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.” Spun-off from IPython in 2014 by co-founder Fernando Pérez, Project Jupyter supports execution environments in several dozen languages.

The name “Jupyter” was chosen to bring to mind the ideas and traditions of science and the scientific method. Additionally, the core programming languages supported by Jupyter are Julia, Python, and R. While the name Jupyter is not a direct acronym for these languages (Julia (Ju), Python (Py) and R), it does establish a firm alignment with them.

**Anaconda:**

* It gives you all the standard packages used in scientific computing in a convenient package without having to worry about installing them all individually with their dependencies.
* If you don't plan on using typical scientific computing packages (numpy, matplotlib, scipy, pandas, etc.) or any of the packaged software (jupyter notebooks, spyder IDE), then the only downside is that you're downloading software that you might not need.
* Regardless if you go with a distribution like Anaconda or just a fresh python environment, it's useful to learn about environment management and package installation with pip and venv or conda.
* Besides the convenience, there's not going to be a major difference between using anaconda vs setting up your own environment. It's all the same python underneath the hood.

### Anaconda Navigator:

Anaconda Navigator is a desktop [graphical user interface (GUI)](https://en.wikipedia.org/wiki/Graphical_user_interface) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using [command-line commands](https://en.wikipedia.org/wiki/Command-line_interface). Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux).

The following applications are available by default in Navigator:[[18]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-18)

* [**JupyterLab**](https://en.wikipedia.org/wiki/Project_Jupyter#JupyterLab)
* [**Jupyter Notebook**](https://en.wikipedia.org/wiki/Project_Jupyter#Jupyter_Notebook)
* **QtConsole****[[19]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-qtconsole-19)**
* [**Spyder**](https://en.wikipedia.org/wiki/Spyder_(software))

**Banksim dataset:**

We detect the fraudulent transactions from the Banksim dataset. This synthetically generated periods and with different amounts. For more information on the dataset you can check the Kaggle page for this dataset which also has the link to the original paper.

Here what we’ll do in this kernel:

* **Exploratory Data Analysis (EDA)**
* **Data Preprocessing**
* **Oversampling with SMOTE**
* **K-Neighbours Classifier**
* **Random Forest Classifier**
* **XGBoost Classifier**

**CODE:**

**In [1]:**

*# Necessary imports*

*## Data loading, processing and for more*

importpandasaspd

importnumpyasnp

fromimblearn.over\_samplingimportSMOTE

*## Visualization*

importseabornassns

importmatplotlib.pyplotasplt

*# set seaborn style because it prettier*

sns.set()

*## Metrics*

fromsklearn.model\_selectionimporttrain\_test\_split

fromsklearn.metricsimportconfusion\_matrix,classification\_report

fromsklearn.metricsimportroc\_curve,auc

*## Models*

importxgboostasxgb

fromsklearn.neighborsimportKNeighborsClassifier

fromsklearn.ensembleimportRandomForestClassifier

fromsklearn.ensembleimportVotingClassifier

Using TensorFlow backend.

**out[1]**

oUsingTbackend.

Using TensorFlow backend.

***In[2]:***

*#read the data and show first 5 rows*

data=pd.read\_csv("../input/bs140513\_032310.csv")

data.head(5)

**In[3]**

Data.info()

In[3]:

<class ‘pandas.core.frame.DataFrame’>

RangeIndex: 594643 entries, 0 to 594642

**Out[3]**

Data columns (total 10 columns):

Step 594643 non-null int64

Customer 594643 non-null object

Age 594643 non-null object

Gender 594643 non-null object

zipcodeOri 594643 non-null object

merchant 594643 non-null object

zipMerchant 594643 non-null object

category 594643 non-null object

amount 594643 non-null float64

fraud 594643 non-null int64

dtypes: float64(1), int64(2), object(7)

memory usage: 45.4+ MB

**In[4]**

*# Create two dataframes with fraud and non-fraud data*

df\_fraud=data.loc[data.fraud==1]

df\_non\_fraud=data.loc[data.fraud==0]

sns.countplot(x="fraud",data=data)

plt.title("Count of Fraudulent Payments")

plt.show()

print("Number of normal examples: ",df\_non\_fraud.fraud.count())

print("Number of fradulent examples: ",df\_fraud.fraud.count())

*#print(data.fraud.value\_counts()) # does the same thing above*

**Out[4]**

Number of normal examples: 587443

Number of fradulent examples: 7200

**In[5]**

Print(“Mean feature values per category”,data.groupby(‘category’)[‘amount’,’fraud’].mean())

Mean feature values per category amount fraud

Category

**Out[5]**

‘es\_barsandrestaurants’ 43.461014 0.018829

‘es\_contents’ 44.547571 0.000000

‘es\_fashion’ 65.666642 0.017973

‘es\_food’ 37.070405 0.000000

‘es\_health’ 135.621367 0.105126

‘es\_home’ 165.670846 0.152064

‘es\_hotelservices’ 205.614249 0.314220

‘es\_hyper’ 45.970421 0.045917

‘es\_leisure’ 288.911303 0.949900

‘es\_otherservices’ 135.881524 0.250000

‘es\_sportsandtoys’ 215.715280 0.495252

‘es\_tech’ 120.947937 0.066667

‘es\_transportation’ 26.958187 0.000000

‘es\_travel’ 2250.409190 0.793956

‘es\_wellnessandbeauty’ 65.511221 0.04759

I**n[6]**

# Create two dataframes with fraud and non-fraud data

Pd.concat([df\_fraud.groupby(‘category’)[‘amount’].mean(),df\_non\_fraud.groupby(‘category’)[‘amount’].mean(),\

Data.groupby(‘category’)[‘fraud’].mean()\*100],keys=[“Fraudulent”,”Non-Fraudulent”,”Percent(%)”],axis=1,\

Sort=False).sort\_values(by=[‘Non-Fraudulent’])

**Out[6]**

**In[7]**

Plot histograms of the amounts in fraud and non-fraud data

#Plt.figure(figsize=(30,10))

Sns.boxplot(x=data.category,y=data.amount)

Plt.title(“Boxplot for the Amount spend in category”)

Plt.ylim(0,4000)

Plt.legend()

Plt.show()

**Out[7]**

No handles with labels found to put in legend.

**In[8]**

# Plot histograms of the amounts in fraud and non-fraud data

Plt.figure(figsize=(30,10))

Sns.boxplot(x=data.category,y=data.amount)

Plt.title(“Boxplot for the Amount spend in category”)

Plt.ylim(0,4000)

Plt.legend()

Plt.show()

**Out[8]**

**In[9]**

Print((data.groupby(‘age’)[‘fraud’].mean()\*100).reset\_index().rename(columns={‘age’:’Age’,’fraud’ : ‘Fraud Percent’}).sort\_values(by=’Fraud Percent’))

**Out[9]**

Age Fraud Percent

7 ‘U’ 0.594228

6 ‘6’ 0.974826

5 ‘5’ 1.095112

1 ‘1’ 1.185254

3 ‘3’ 1.192815

2 ‘2’ 1.251401

4 ‘4’ 1.293281

0 ‘0’ 1.957586

**Data Preprocessing:**

**In[10]**

Print(“Unique zipCodeOri values: “,data.zipcodeOri.nunique())

Print(“Unique zipMerchant values: “,data.zipMerchant.nunique())

# dropping zipcodeori and zipMerchant since they have only one unique value

Data\_reduced = data.drop([‘zipcodeOri’,’zipMerchant’],axis=1)

**Out[10]**

Unique zipCodeOri values: 1

Unique zipMerchant values: 1

**Checking the data after dropping.**

**In[11]**

Data\_reduced.columns

**Out[11]**

Index([‘step’, ‘customer’, ‘age’, ‘gender’, ‘merchant’, ‘category’, ‘amount’,

‘fraud’],

Dtype=’object’)

Data\_reduced.loc[:,[‘customer’,’merchant’,’category’]].astype(‘category’) data\_dum = pd.get\_dummies(data\_reduced.loc[:,[‘customer’,’merchant’,’category’,’gender’]],drop\_first=True)

#Dummies print(data\_dum.info())

**In [12]:**

# turning object columns type to categorical for easing the transformation process

Col\_categorical = data\_reduced.select\_dtypes(include= [‘object’]).columns

For col in col\_categorical:

Data\_reduced[col] = data\_reduced[col].astype(‘category’)

#categorical values  numeric values

Data\_reduced[col\_categorical] = data\_reduced[col\_categorical].apply(lambda x: x.cat.codes)

Data\_reduced.head(5)

**Out[12]**

**In[13]**

X = data\_reduced.drop([‘fraud’],axis=1)

Y = data[‘fraud’]

Print(X.head(),”\n”)

Print(y.head())

**Out[13]**

Step customer age gender merchant category amount

0 0 210 4 2 30 12 4.55

1 0 2753 2 2 30 12 39.68

2 0 2285 4 1 18 12 26.89

3 0 1650 3 2 30 12 17.25

4 0 3585 5 2 30 12 35.72

0

1. 0

2 0

1. 0

4 0

Name: fraud, dtype: int64

**In [14]:**

Y[y==1].count()

**Out[14]**

7200

**Oversampling with SMOTE:**

**In [15]:**

Sm = SMOTE(random\_state=42)

X res, y\_res = sm.fit\_resample(X, y)

Y res = pd.DataFrame(y\_res)

Print(y\_res[0].value\_counts())

**Out[15]**

1. 587443
2. 587443

Name: 0, dtype: int64

**In [16]:**

# I won’t do cross validation since we have a lot of instances

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_res,y\_res,test\_size=0.3,random\_state=42,shuffle=True,stratify=y\_res)

**In [17]:**

# %% Function for plotting ROC\_AUC curve

Def plot\_roc\_auc(y\_test, preds):

Fpr, tpr, threshold = roc\_curve(y\_test, preds)

Roc\_auc = auc(fpr, tpr)

Plt.title(‘Receiver Operating Characteristic’)

Plt.plot(fpr, tpr, ‘b’, label = ‘AUC = %0.2f’ % roc\_auc)

Plt.legend(loc = ‘lower right’)

Plt.plot([0, 1], [0, 1],’r—‘)

Plt.xlim([0, 1])

plt.ylim([0, 1])

Plt.ylabel(‘True Positive Rate’

Plt.xlabel(‘False Positive Rate’)

Plt.show()

**In[18]**

# The base score should be better than predicting always non-fraduelent

Print(“Base accuracy score we must beat is: “,

Df\_non\_fraud.fraud.count()/ np.add(df\_non\_fraud.fraud.count(),df\_fraud.fraud.count()) \* 100)

**Out[18]**

Base accuracy score we must beat is: 98.7891894800746

**K-Neighbours Classifier:**

**In[19]**

# %% K-elloNeigbors

Knn = KNeighborsClassifier(n\_neighbors=5,p=1)

Knn.fit(X\_train,y\_train)

Y\_pred = knn.predict(X\_test)

Print(“Classification Report for K-Nearest Neighbours: \n”, classification\_report(y\_test, y\_pred))

Print(“Confusion Matrix of K-Nearest Neigbours: \n”, confusion\_matrix(y\_test,y\_pred))

Plot\_roc\_auc(y\_test, knn.predict\_proba(X\_test)[:,1])

**Out[19]**

/opt/conda/lib/python3.6/site-packages/ipykernel\_launcher.py:5: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

Classification Report for K-Nearest Neighbours:

Precision recall f1-score support

0 1.00 0.98 0.99 176233

1 0.98 1.00 0.99 176233

Micro avg 0.99 0.99 0.99 352466

Macro avg 0.99 0.99 0.99 352466

Weighted avg 0.99 0.99 0.99 352466

Confusion Matrix of K-Nearest Neigbours:

[[172041 4192]

[ 376 175857]]

**Random Forest Classifier:**

**In[20]**

# %% Random Forest Classifier

Rf\_clf = RandomForestClassifier(n\_estimators=100,max\_depth=8,random\_state=42,

Verbose=1,class\_weight=”balanced”)

Rf\_clf.fit(X\_train,y\_train)

Y\_pred = rf\_clf.predict(X\_test)

Print(“Classification Report for Random Forest Classifier: \n”, classification\_report(y\_test, y\_pred))

Print(“Confusion Matrix of Random Forest Classifier: \n”, confusion\_matrix(y\_test,y\_pred)) Plot\_roc\_auc(y\_test, rf\_clf.predict\_proba(X\_test)[:,1])

**Out[20]**

/opt/conda/lib/python3.6/site-packages/ipykernel\_launcher.py:6: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples,), for example using ravel().

[Parallel(n\_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[Parallel(n\_jobs=1)]: Done 100 out of 100 | elapsed: 2.7min finished

[Parallel(n\_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[Parallel(n\_jobs=1)]: Done 100 out of 100 | elapsed: 2.9s finished

Classification Report for Random Forest Classifier:

Precision recall f1-score support

0 1.00 0.97 0.98 176233

1 0.97 1.00 0.98 176233

micro avg 0.98 0.98 0.98 3524

macro avg 0.99 0.98 0.98 352466 Weighted avg 0.99 0.98 0.98 352466

**XGBoost Classifier:**

**In[21]**

XGBoost\_CLF = xgb.XGBClassifier(max\_depth=6, learning\_rate=0.05, n\_estimators=400,

Objective=”binary:hinge”, booster=’gbtree’,

N\_jobs=-1, nthread=None, gamma=0, min\_child\_weight=1, max\_delta\_step=0,

Subsample=1, colsample\_bytree=1, colsample\_bylevel=1, reg\_alpha=0, reg\_lambda=1,

Scale\_pos\_weight=1, base\_score=0.5, random\_state=42, verbosity=True)

XGBoost\_CLF.fit(X\_train,y\_train)

Y\_pred = XGBoost\_CLF.predict(X\_test)

Print(“Classification Report for XGBoost: \n”, classification\_report(y\_test, y\_pred))

Print(“Confusion Matrix of XGBoost: \n”, confusion\_matrix(y\_test,y\_pred))

Plot\_roc\_auc(y\_test, XGBoost\_CLF.predict\_proba(X\_test)[:,1])

**Out[21]**

/opt/conda/lib/python3.6/site-packages/sklearn/preprocessing/label.py:219: using ravel().

Y = column\_or\_1d(y, warn=True)

/opt/conda/lib/python3.6/site-packages/sklearn/preprocessing/label.py:252: Y = column\_or\_1d(y, warn=True)

Classification Report for XGBoost:

Precision recall f1-score support

0 1.00 1.00 1.00 176233

1 1.00 1.00 1.00 176233

Micro avg 1.00 1.00 1.00 352466

Macro avg 1.00 1.00 1.00 352466

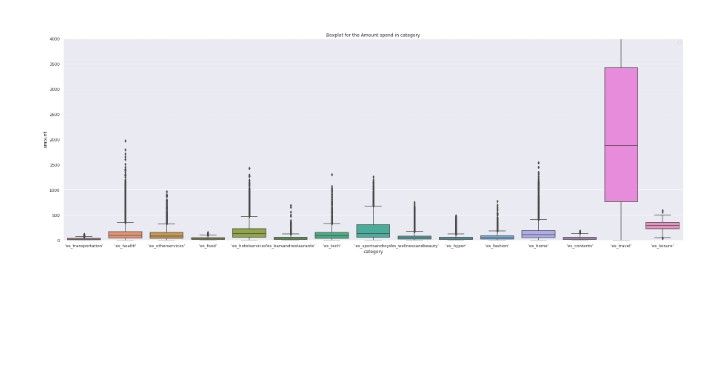
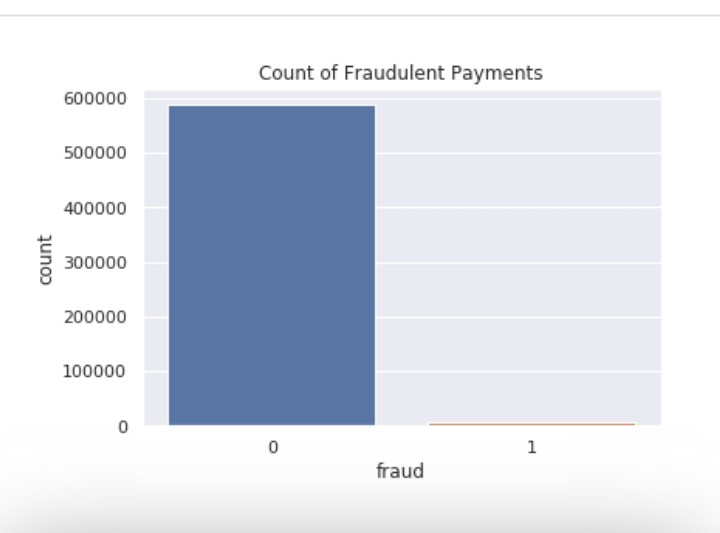
Weighted avg 1.00 1.00 1.00 352466

Confusion Matrix of XGBoost:

[[175727 506]

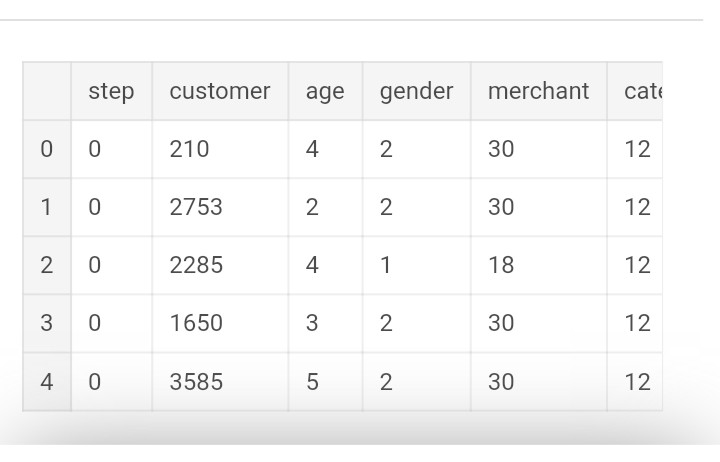
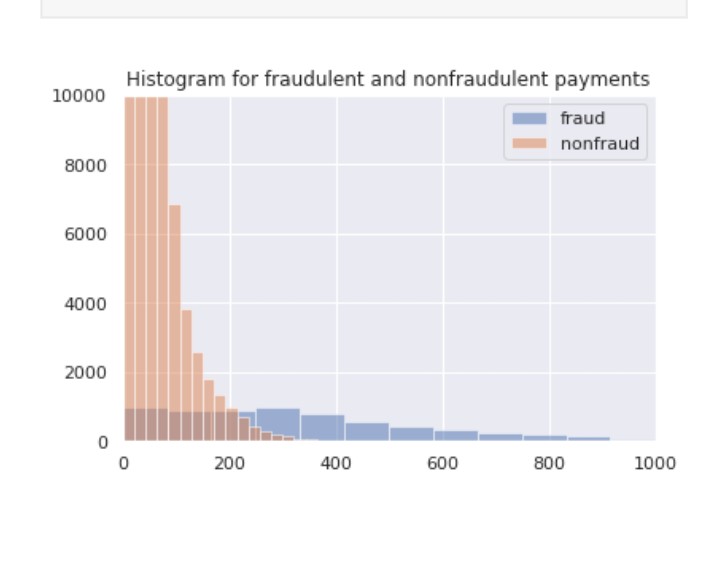
[ 310 175923]]

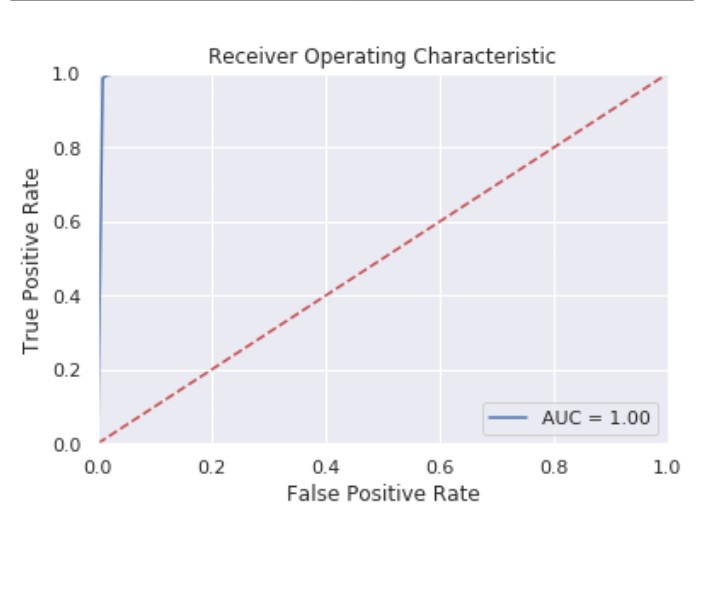
**RESULT:**

Out[7]

Out[ 8]

Out [12]

Out [20]



Out[21]