**CREDIT CARD FRAUD DETECTION USING PREDICTIVE MODELS**

**ABSTRACT**

With the emergence of internet and e-commerce, the use of credit card is an unavoidable one. The credit cards are used for purchasing goods and services. We can make both online and offline payment easily with the help of credit cards. Credit card fraud presents more and more threat that has serious consequences in the financial sector. As a result, financial institutions are forced to continually improve their fraud detection systems. In recent years, several studies have used machine learning and data mining techniques to provide solutions to this problem. Bankruptcy fraud is the type of fraud, it is very challenging to identify fraud. The credit card's failure of a debtor to pay their debt which is also called as Insolvent gives rise to Bankruptcy fraud. Sometimes, banks are required to cover their losses itself. This can be forbidden, by passing the required details to credit bureau. This is one of the ways wherein it helps to identify the past history related to its transactions details of its corresponding customers. Depending upon the historical details, further suitable action can be taken by Banks to avoid this type of fraud. For banks it has become very difficult for detecting the fraud in credit card system. Machine learning plays a vital role for detecting the credit card fraud in the transactions. For predicting these transactions banks make use of various machine learning methodologies, past data has been collected and new features are been used for enhancing the predictive power. The performance of fraud detecting in credit card transactions is greatly affected by the sampling approach on data-set, selection of variables and detection techniques used. In this project we are focusing on implementation of the Adaboost, Catboost, LightGBM, XGBoost and Random forest classifiers. Based on the acquired accuracy scores of these Machine Learning Algorithms, we can identify the Algorithm which can detect the fraudulent transactions more accurately by using the k-fold cross validation technique. For this a dataset is collected on credit card transactions. For this dataset we are using these Machine Learning Algorithms. Dataset of credit card transactions contains a total of 284808 credit card transactions of a European bank data set. It considers fraud transactions as the “positive class” and genuine ones as the “negative class” .The data set is highly imbalanced, it has about 0.172% of fraud transactions and the rest are genuine transactions.

**Acknowledgement**

I wish to express our heartfelt gratitude to **Dr.G.Viswanathan**, Chancellor, VIT, Vellore, for providing facilities for the Industrial Internship. I am highly grateful to our Vice President, **Dr.G. Sekar Viswanathan**, Vice chancellor **Dr. Anand A. Samuel,** and Pro-Vice Chancellor **Dr.S.Narayanan,** for providing the necessary resources.

My sincere gratitude to **Dr. Balakrushna Tripathy**, Dean, School of Information Technology and Engineering, for giving me the opportunity to undertake the project.

I wish to express my sincere gratitude to **Dr. S. Sree Dharinya**, Head of the Department, Software and Systems Engineering, **Prof. P.Ushapreethi & Prof. Ramaprabha KP**, Industrial Internship Coordinators, M.Tech (Software Engineering), School of Information Technology and Engineering for providing me continuous support to do my project work**.**

I would like to express my special gratitude and thanks to my external guide **Mr Brijesh.** Manager, Dafn Tech Private Limited and internal guide **Prof. Jayaram Reddy.** Assistant Professor, School of Information Technology and Engineering for their esteemed guidance, immense support and encouragement to complete the internship successfully.

I thank the management of VIT, Vellore for permitting me to use the library resources. I also thank all the faculty members of VIT, Vellore for giving me the courage and strength I needed to complete my goals. This acknowledgement would be incomplete without expressing my whole hearted thanks to my family and friends who motivated me during the course of the work.

1. **INTRODUCTION**
   1. **Problem Statement:**

The Credit Card Fraud Detection Problem includes modelling past credit card transactions with the knowledge of the ones that turned out to be fraud. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

* 1. **Motivation:**

There are many fraud detection systems but still there are some disadvantages like the system could not find fraudulent transactions accurately, this is due to the technology involved in the fraud detection systems. But by using Machine Learning algorithms we can find the fraudulent transactions more accurately.

* 1. **Objective:**

The objective of credit card fraud detection is to reduce losses due to payment fraud for both merchants and issuing banks and increase revenue opportunities for merchants and also to reduce the overall frauds through credit card transactions.

* + 1. **Proposed system:**

In the proposed system, to identify the fraudulent credit card transaction with 100% accuracy based on the previous data Machine Learning algorithms are used.

* + 1. **Advantages of proposed system:**

Due to Behaviour and location analysis approach, there is a drastic reduction in the number of False Positives transactions identified as malicious by an FDS although they are actually genuine. The system stores previous transaction patterns for each user. Based upon previous data of that user the system recognizes unusual patterns in the procedure in future. Through this process the Machine Learning algorithms used in this project helps in identifying the fraudulent transactions more effectively for new data by the use of previous data.

1. **SYSTEM DESIGN**
   1. **System Architecture:**

Data cleaning and visualization

Read the dataset

Dividing the data in to train, test and validate sets

Feature selection

XGBoost classifier

lightGBM classifier

Catboost classifier

Adaboost classifier

Random forest classifier

K fold cross validation selection

Performance analysis of the classifier

Classifier selection

Performance and accuracy results

* 1. **Module Description**

1. **Data collection**

The datasets contains transactions made by credit cards in September 2013 by european cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. It contains only numerical input variables which are the result of a PCA transformation. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

1. **EDA**

EDA refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

1. **Data preparation**

Data preparation is the process of transforming raw data so that data scientists and analysts can run it through machine learning algorithms to uncover insights or make predictions. The data preparation process can be complicated by issues such as: Missing or incomplete records.

1. **Creating train and test data**

Dividing the data collected in to train and test data. Train data set should be larger than test data set i.e, 75% train data should be taken.

Training Dataset: The sample of data used to fit the model.

Validation Dataset: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

Test Dataset: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

1. **Outlier detection and removal**

In this process outlier detection helps in detecting and subsequently excluding outliers from a given set of data. An outlier may be defined as a piece of data or observation that deviates drastically from the given norm or average of the data set.

1. **Classification Algorithms**

The classification algorithms used in this project are: Random Forest, Adaboost, Catboost, LightGBM, XGboost. Classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation.

Random Forest works on mainly 4 steps:

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

“CatBoost” name comes from two words “Category” and “Boosting”. It yields state-of-the-art results without extensive data training typically required by other machine learning methods, and Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems.

1. **K-fold cross validation**

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation. Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
   1. Take the group as a hold out or test data set
   2. Take the remaining groups as a training data set
   3. Fit a model on the training set and evaluate it on the test set
   4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores.
   1. **System Specification**
      1. **Software Requirements**

Anaconda distribution

Jupyter notebook

Python-3.1

* + 1. **Hardware Requirements**

Processor - Intel

RAM - 4 Gb

Hard Disk - 260 GB

Key Board - Standard Windows Keyboard

Mouse - Two or Three Button Mouse

* 1. **Detailed Design**
     1. **Use case Diagram**

****

* + 1. **Sequence Diagram**

****

* + 1. **Class Diagram**

****

* + 1. **Dataflow Diagram**

Customer profile database

Credit card transactions

**TA**

**Deviated sequence**

Normal

**PS**

Anomaly

**DS**

Detects fraud

FHD

* + 1. **Activity Diagram**

****

1. **IMPLEMENTATION**
   1. **Implementation Details**
2. **Data collection:**

The datasets contains transactions made by credit cards in September 2013 by european cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

1. **Reading the data:**

data\_df = pd.read\_csv(r"C:\Users\DELL\Desktop\intern\_submission\creditcard.csv")

1. **check the data:**

print("Credit Card Fraud Detection data - rows:",data\_df.shape[0]," columns:", data\_df.shape[1])

It gives number of rows and columns present in the dataset.

1. **Glimpse the data:**

data\_df.head()

It displays the first 5 rows of the dataset.

1. **Check missing values:**

To check if there are any missing values in the dataset.

**Code:**

total = data\_df.isnull().sum().sort\_values(ascending = False)

percent = (data\_df.isnull().sum()/data\_df.isnull().count()\*100).sort\_values(ascending = False)

pd.concat([total, percent], axis=1, keys=['Total', 'Percent']).transpose()

1. **Transaction in time:**

Displays the occurrence of fraud and non-fraud transactions with respect to the time. From the plot we can observe that Fraudulent transactions have a distribution more even than valid transactions - are equally distributed in time, including the low real transaction times, during night in Europe time zone.

**Code:**

class\_0 = data\_df.loc[data\_df['Class'] == 0]["Time"]

class\_1 = data\_df.loc[data\_df['Class'] == 1]["Time"]

*#plt.figure(figsize = (14,4))*

*#plt.title('Credit Card Transactions Time Density Plot')*

*#sns.set\_color\_codes("pastel")*

*#sns.distplot(class\_0,kde=True,bins=480)*

*#sns.distplot(class\_1,kde=True,bins=480)*

*#plt.show()*

hist\_data = [class\_0, class\_1]

group\_labels = ['Not Fraud', 'Fraud']

fig = ff.create\_distplot(hist\_data, group\_labels, show\_hist=False, show\_rug=False)

fig['layout'].update(title='Credit Card Transactions Time Density Plot', xaxis=dict(title='Time [s]'))

iplot(fig, filename='dist\_only')

1. **Transactions amount:**

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12,6))

s = sns.boxplot(ax = ax1, x="Class", y="Amount", hue="Class",data=data\_df, palette="PRGn",showfliers=True)

s = sns.boxplot(ax = ax2, x="Class", y="Amount", hue="Class",data=data\_df, palette="PRGn",showfliers=False)

plt.show();

1. **Fraudulent transaction:**

Plotting of the fraudulent transactions (amount) against time. The time is shown is seconds from the start of the time period (totally 48h, over 2 days).

**Code:**

fraud = data\_df.loc[data\_df['Class'] == 1]

trace = go.Scatter(

x = fraud['Time'],y = fraud['Amount'],

name="Amount",

marker=dict(

color='rgb(238,23,11)',

line=dict(

color='red',

width=1),

opacity=0.5,

),

text= fraud['Amount'],

mode = "markers"

)

data = [trace]

layout = dict(title = 'Amount of fraudulent transactions',

xaxis = dict(title = 'Time [s]', showticklabels=True),

yaxis = dict(title = 'Amount'),

hovermode='closest'

)

fig = dict(data=data, layout=layout)

iplot(fig, filename='fraud-amount')

1. **Outlier detection and removal:**

f, axes = plt.subplots(nrows=2, ncols=4, figsize=(26,16))

f.suptitle('Features With High Negative Correlation', size=35)

sns.boxplot(x="Class", y="V3", data=subsample, ax=axes[0,0])

sns.boxplot(x="Class", y="V9", data=subsample, ax=axes[0,1])

sns.boxplot(x="Class", y="V10", data=subsample, ax=axes[0,2])

sns.boxplot(x="Class", y="V12", data=subsample, ax=axes[0,3])

sns.boxplot(x="Class", y="V14", data=subsample, ax=axes[1,0])

sns.boxplot(x="Class", y="V16", data=subsample, ax=axes[1,1])

sns.boxplot(x="Class", y="V17", data=subsample, ax=axes[1,2])

f.delaxes(axes[1,3])

1. **Features correlation:**

There is no notable correlation between features V1-V28. There are certain correlations between some of these features and Time (inverse correlation with V3) and Amount (direct correlation with V7 and V20, inverse correlation with V1 and V5).

**Code:**

plt.figure(figsize = (14,14))

plt.title('Credit Card Transactions features correlation plot (Pearson)')

corr = data\_df.corr()

sns.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns,linewidths=.1,cmap="Reds")

plt.show()

1. **Split the data in train, test and validate:**

**Code:**

train\_df, test\_df = train\_test\_split(data\_df, test\_size=TEST\_SIZE, random\_state=RANDOM\_STATE, shuffle=True )

train\_df, valid\_df = train\_test\_split(train\_df, test\_size=VALID\_SIZE, random\_state=RANDOM\_STATE, shuffle=True )

1. **MODEL BUILDING:**

**Define model parameters:**

Run the model using the training set for training. Then, we will use the validation set for validation.

We will use as validation criterion GINI, which formula is

GINI = 2 \* (AUC) - 1,

where AUC is the Receiver Operating Characteristic - Area Under Curve (ROC-AUC). Number of estimators is set to 100 and number of parallel jobs is set to 4.

We start by initializing the RandomForestClassifier.

**Code:**

clf = RandomForestClassifier(n\_jobs=NO\_JOBS,

random\_state=RANDOM\_STATE,

criterion=RFC\_METRIC,

n\_estimators=NUM\_ESTIMATORS,

verbose=False)

**Train the classifier:**

Let's train the RandonForestClassifier using the train\_df data and fit function. And, predict the target values for the valid\_df data, using predict function.

**Code:**

clf.fit(train\_df[predictors], train\_df[target].values)

preds = clf.predict(valid\_df[predictors])

**Feature importance:**

Visualize the features importance

**Code:**

tmp = pd.DataFrame({'Feature': predictors, 'Feature importance': clf.feature\_importances\_})

tmp = tmp.sort\_values(by='Feature importance',ascending=False)

plt.figure(figsize = (7,4))

plt.title('Features importance',fontsize=14)

s = sns.barplot(x='Feature',y='Feature importance',data=tmp)

s.set\_xticklabels(s.get\_xticklabels(),rotation=90)

plt.show()

**Confusion matrix:**

**Code:**

cm = pd.crosstab(valid\_df[target].values, preds, rownames=['Actual'], colnames=['Predicted'])

fig, (ax1) = plt.subplots(ncols=1, figsize=(5,5))

sns.heatmap(cm,

xticklabels=['Not Fraud', 'Fraud'],

yticklabels=['Not Fraud', 'Fraud'],

annot=True,ax=ax1,

linewidths=.2,linecolor="Darkblue", cmap="Blues")

plt.title('Confusion Matrix', fontsize=14)

plt.show()

**ROC-AUC curve:**

**Code:**

roc\_auc\_score(valid\_df[target].values, preds)

plt.title('Confusion Matrix', fontsize=14)

plt.show()

1. **Model evaluation:**

kf = KFold(n\_splits = NUMBER\_KFOLDS, random\_state = RANDOM\_STATE, shuffle = True)

*# Create arrays and dataframes to store results*

oof\_preds = np.zeros(train\_df.shape[0])

test\_preds = np.zeros(test\_df.shape[0])

feature\_importance\_df = pd.DataFrame()

n\_fold = 0

for train\_idx, valid\_idx in kf.split(train\_df):

train\_x, train\_y = train\_df[predictors].iloc[train\_idx],train\_df[target].iloc[train\_idx]

valid\_x, valid\_y = train\_df[predictors].iloc[valid\_idx],train\_df[target].iloc[valid\_idx]

evals\_results = {}

model = LGBMClassifier(

nthread=-1,

n\_estimators=2000,

learning\_rate=0.01,

num\_leaves=80,

colsample\_bytree=0.98,

subsample=0.78,

reg\_alpha=0.04,

reg\_lambda=0.073,

subsample\_for\_bin=50,

boosting\_type='gbdt',

is\_unbalance=False,

min\_split\_gain=0.025,

min\_child\_weight=40,

min\_child\_samples=510,

objective='binary',

metric='auc',

silent=-1,

verbose=-1,

feval=None)

model.fit(train\_x, train\_y, eval\_set=[(train\_x, train\_y), (valid\_x, valid\_y)],

eval\_metric= 'auc', verbose= VERBOSE\_EVAL, early\_stopping\_rounds= EARLY\_STOP)

oof\_preds[valid\_idx] = model.predict\_proba(valid\_x, num\_iteration=model.best\_iteration\_)[:, 1]

test\_preds += model.predict\_proba(test\_df[predictors], num\_iteration=model.best\_iteration\_)[:, 1] / kf.n\_splits

fold\_importance\_df = pd.DataFrame()

fold\_importance\_df["feature"] = predictors

fold\_importance\_df["importance"] = clf.feature\_importances\_

fold\_importance\_df["fold"] = n\_fold + 1

feature\_importance\_df = pd.concat([feature\_importance\_df, fold\_importance\_df], axis=0)

print('Fold %2d AUC : %.6f' % (n\_fold + 1, roc\_auc\_score(valid\_y, oof\_preds[valid\_idx])))

del model, train\_x, train\_y, valid\_x, valid\_y

gc.collect()

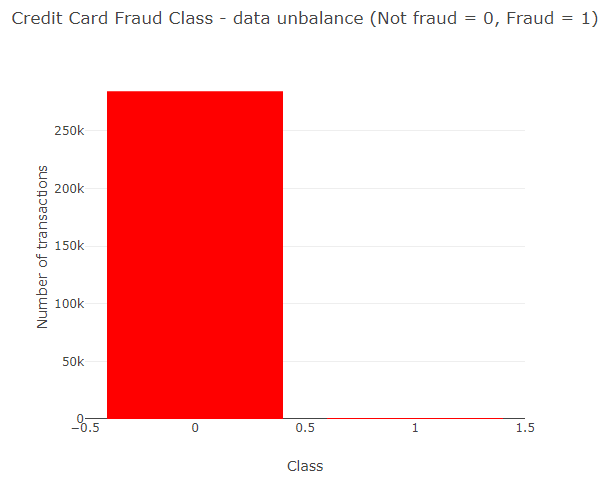
n\_fold = n\_fold + 1

train\_auc\_score = roc\_auc\_score(train\_df[target], oof\_preds)

print('Full AUC score %.6f' % train\_auc\_score)

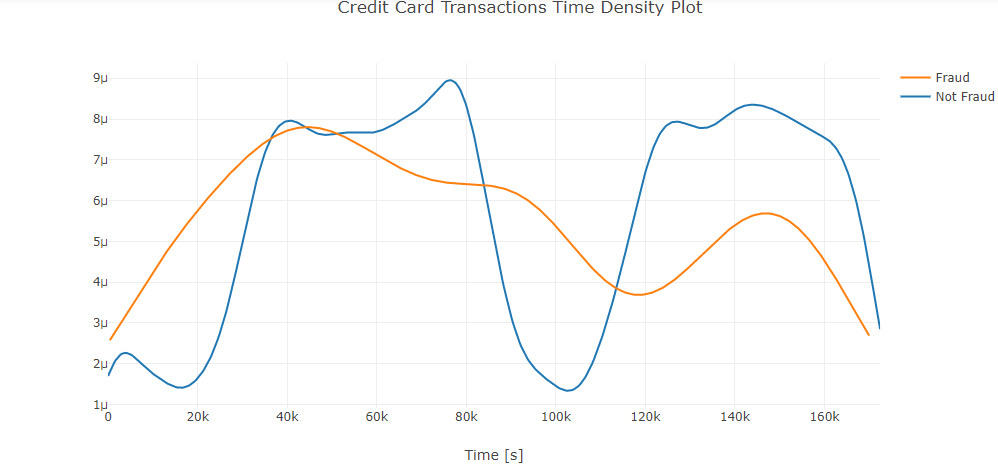
1. **RESULTS AND DISCUSSIONS**

**Checking of data unbalance wrt target value “class”:**



**Data exploration:**

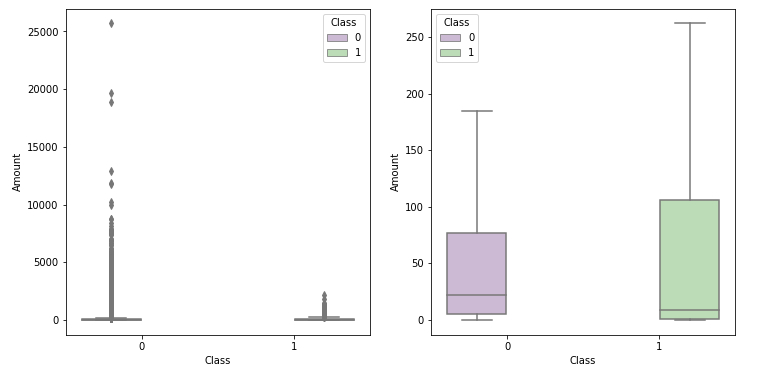
* + 1. **Transaction in time:**

****

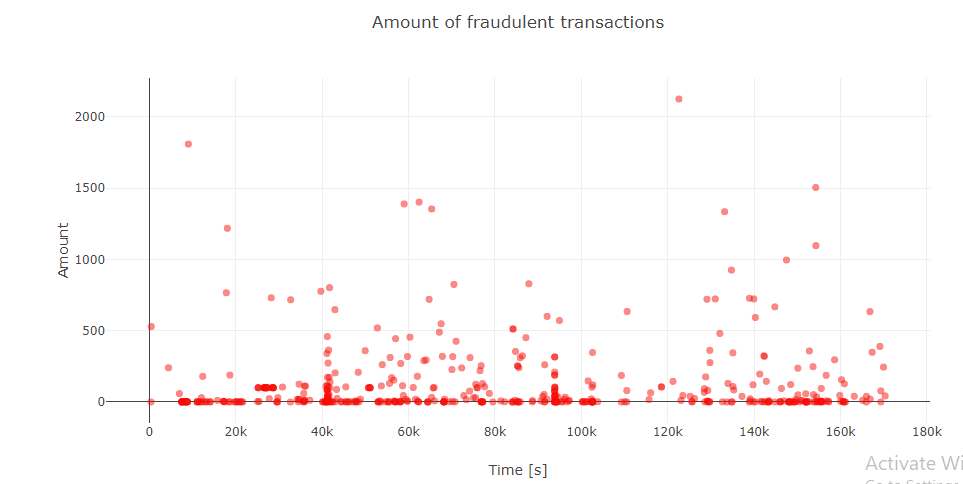
From the transaction in time graph-

Fraudulent transactions have a distribution more even than valid transactions - are equally distributed in time, including the low real transaction times, during night in Europe time zone.

* + 1. **Transactions amount:**

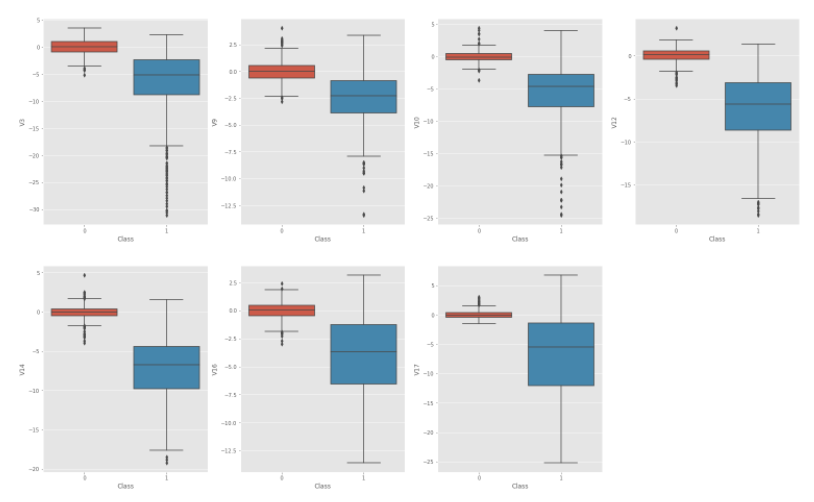


**Amount of fraudulent Transactions:**

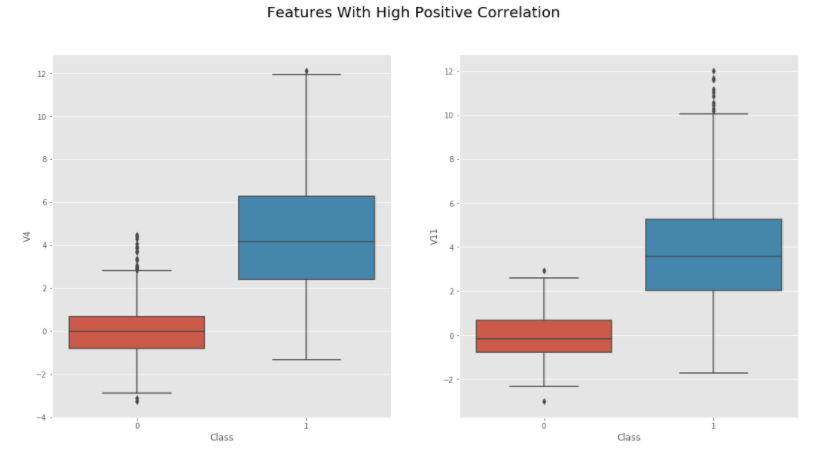


**Outlier detection and removal:**

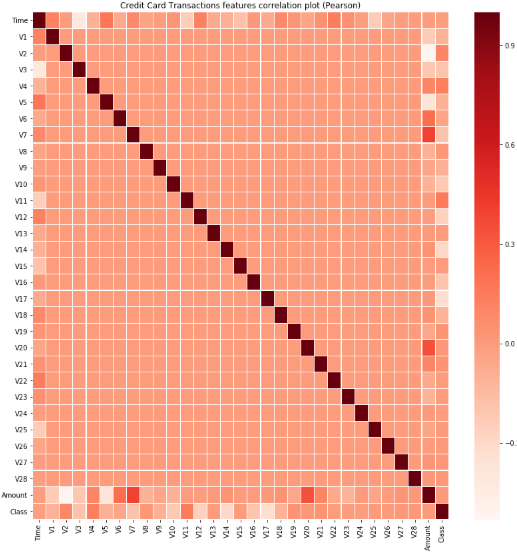
Features with high negative correlation:



Features with high positive correlation:



**Features correlation:**



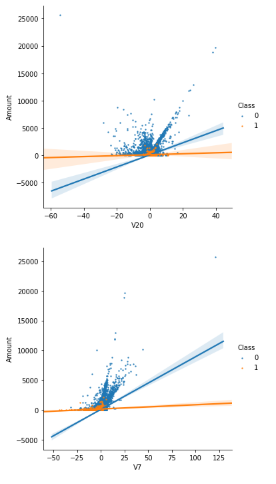
From the feature correlation graph:

There is no notable correlation between features **V1**-**V28**.

There are certain correlations between some of these features and **Time** (inverse correlation with **V3**) and **Amount** (direct correlation with **V7** and **V20**, inverse correlation with **V1** and **V5**).

* The correlated and inverse correlated values on the same graph.
* The direct correlated values: {V20;Amount} and {V7;Amount}.

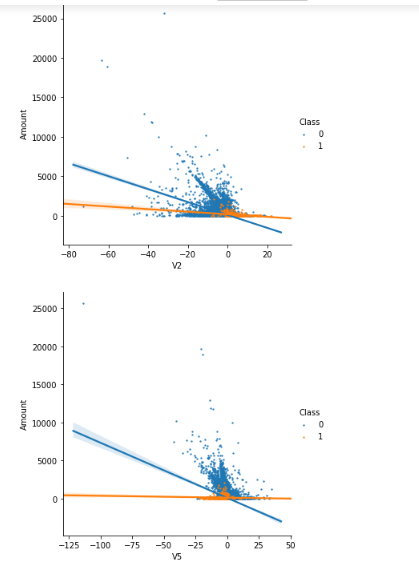
**Plotting of direct correlated values:**



From the plotting of direct correlated values:

We can confirm that the two couples of features are correlated (the regression lines for Class = 0 have a positive slope, whilst the regression line for Class = 1 have a smaller positive slope).

**Plotting of inverse correlated values:**

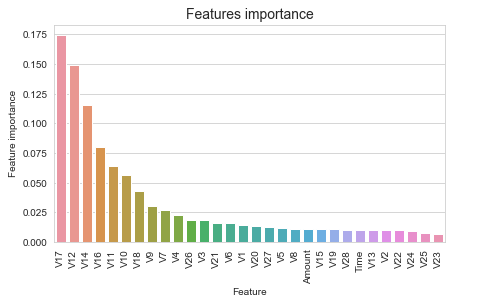
****

From the inverse correlation plot:

* We can confirm that the two couples of features are inverse correlated (the regression lines for Class = 0 have a negative slope while the regression lines for Class = 1 have a very small negative slope).

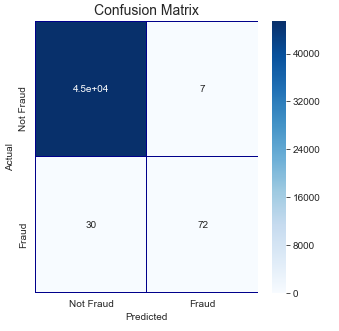
**Random forest classifier:**

1. **Feature importance:**

****

The most important features are V17, V12, V14, V10, V11, V16

1. **Confusion matrix:**

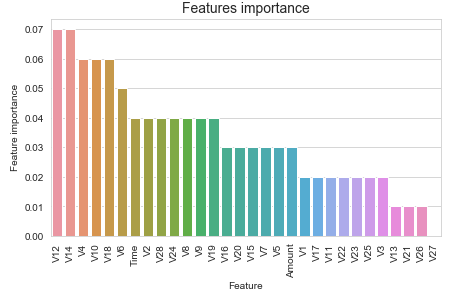
****

1. **ROC-AUC Score:**

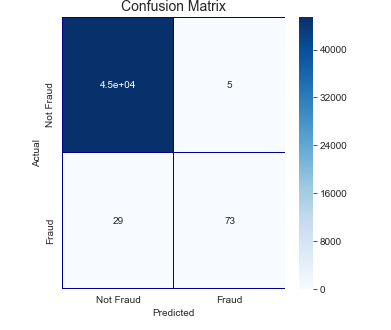
The ROC-AUC score obtained with RandomForrestClassifier is 0.85.

**Adaboost classifier:**

1. **Feature importance:**

****

1. **Confusion matrix:**

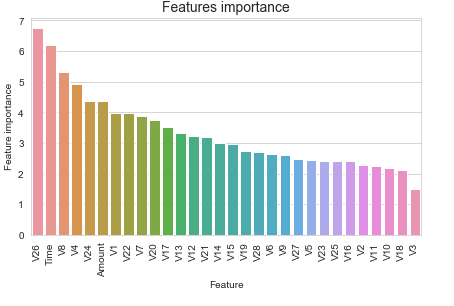
****

1. **ROC-AUC Score:**

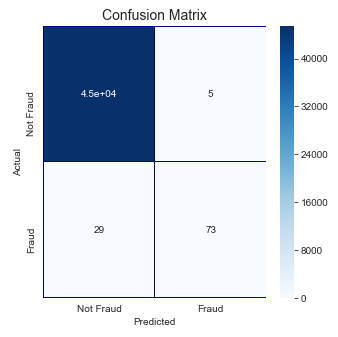
**The ROC-AUC score obtained with AdaBoostClassifier is 0.83.**

**Catboost classifier:**

1. **Feature importance:**

****

1. **Confusion matrix:**

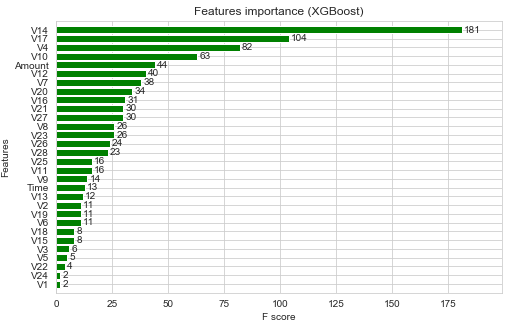
****

1. **ROC-AUC Score:**

The ROC-AUC score obtained with CatBoostClassifier is 0.86.

**XGboost classifier:**

1. **Feature importance:**

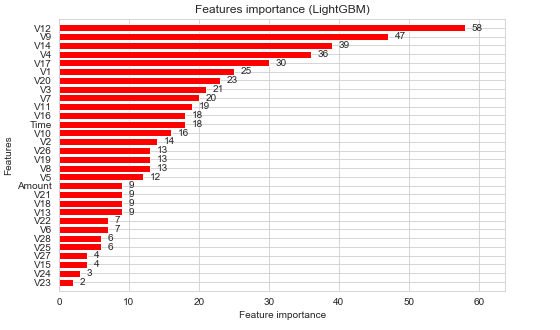


1. **ROC-AUC score:**

The ROC-AUC score obtained with XGBoost Classifier is 0.97.

**LightGBM classifier:**

1. **Feature importance:**



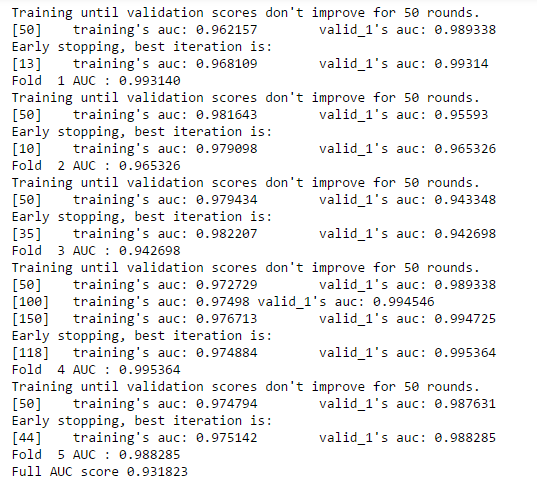
1. **ROC-AUC Score:**

The ROC-AUC score obtained with LightGBM Classifier is 0.94.

**Training and validation using cross-validation:**

We will use cross-validation (KFolds) with 5 folds. Data is divided in 5 folds and, by rotation, we are training using 4 folds (n-1) and validate using the 5th (nth) fold.

Test set is calculated as an average of the predictions



The AUC score for the prediction from the test data was 0.93.

We prepare the test prediction, from the averaged predictions for test over the 5 folds.

We investigated the data, checking for data unbalancing, visualizing the features and understanding the relationship between different features. We then investigated two predictive models. The data was split in 3 parts, a train set, a validation set and a test set. For the first three models, we only used the train and test set.

We started with **RandomForrestClassifier**, for which we obtained an AUC scode of **0.85** when predicting the target for the test set.

We followed with an **AdaBoostClassifier** model, with lower AUC score (**0.83**) for prediction of the test set target values.

We then followed with an **CatBoostClassifier**, with the AUC score after training 500 iterations **0.86**.

We then experimented with a **XGBoost** model. In this case, se used the validation set for validation of the training model. The best validation score obtained was **0.984**. Then we used the model with the best training step, to predict target value from the test data; the AUC score obtained was **0.974**.

We then presented the data to a **LightGBM** model. We used both train-validation split and cross-validation to evaluate the model effectiveness to predict 'Class' value, i.e. detecting if a transaction was fraudulent. With the first method we obtained values of AUC for the validation set around **0.974**. For the test set, the score obtained was **0.946**.

With the cross-validation, we obtained an AUC score for the test prediction of **0.93**.

1. **CONCLUSION AND FUTURE WORK**
   1. **Conclusion**

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.

* 1. **Future work**

Future work will include a comprehensive tuning of the Random Forest algorithm. Having a data set with non-anonymized features would make this particularly interesting as outputting the feature importance would enable one to see what specific factors are most important for detecting fraudulent transactions.

1. **REFERENCE**

[1] L.J.P. van der Maaten and G.E. Hinton, “Visualizing High-Dimensional Data Using t-SNE”, Journal of Machine Learning Research, Vol.9 2579-2605, May 2008.

[2] Awoyemi, John O., Adebayo O. Adetunmbi, and Samuel A. Oluwadare. "Credit card fraud detection using machine learning techniques: A comparative analysis." In 2017 International Conference on Computing Networking and Informatics (ICCNI),Chicago, pp. 1-9. IEEE, 2017.

[3] Raj, S. Benson Edwin, and A. Annie Portia. "Analysis on credit card fraud detection methods." In 2011 International Conference on Computer, Communication and Electrical Technology (ICCCET), Chicago,pp. 152-156. IEEE, 2011.

[4] Adewumi, Aderemi O., and Andronicus A. Akinyelu. "A survey of machine-learning and nature-inspired based credit card fraud detection techniques.",International Journal of System Assurance Engineering and Management,Vol.8, no. 2 (2017): 937-953, November 2017.

[5]John O. Awoyemi, et.al., “Credit card fraud detection using Machine Learning Techniques: A Comparative Analysis” IEEE , 2017

[6] Sahil Dhankhad, et.al., “Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study” , IEEE International Conference on Information Reuse and Integration for Data Science, pp 122-125, 2018

[7] Zahra Kazemi and Houman Zarrabi, “Using deep networks for fraud detection in the credit card transactions”, IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), pp 0630 – 0633, 2017

[8] Samaneh Sorournejad, et.al., “A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective”, pp 1 - 26