Project Report

On

Credit Card Fraud Detection

By

Pallavi Bolar (2020510010)

Aniruddha Deshmukh (2020510015)

Under the guidance of
Internal Supervisor

Prof. Prachi Dalvi

Department of Master of Computer Applications
Sardar Patel Institute of Technology
Autonomous Institute Affiliated to Mumbai University
2021-22

CERTIFICATE OF APPROVAL

This is to certify that the following students

Pallavi Bolar (2020510010)

Aniruddha Deshmukh (2020510015)

Have satisfactorily carried out work on the project entitled

“Credit Card Fraud Detection”

Towards the fulfilment of project, as laid down by
Sardar Patel Institute of Technology during year
2021-22.

Project Guide
(Prof. Prachi Dalvi

PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

Pallavi Bolar (2020510010)

Aniruddha Deshmukh (2020510015)

Have successfully completed the Project report on

“Credit Card Fraud Detection”,

which is found to be satisfactory and is approved at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,
ANDHERI (W), MUMBAI.

INTERNAL EXAMINER

EXTERNAL EXAMINER

Head of Department

(Dr. Pooja Raundale)

Principal

(Dr. B.N.Chaudhari)

TABLE OF CONTENTS

ABSTRACT

The use of smart cards like credit cards has increased as technology advances. Simultaneously abuse and extortion in utilizing Visas have additionally come to light. As a result, fraud can cost the user money. One needs to be aware that their credit card is completely safe. The goal of our project is to find credit card scams. Any malicious use of our card can be identified, allowing for the detection of fraud.

A good number of people prefer to use credit cards whenever they make a purchase online. Even if we don't have any savings at the time, having a high credit limit on our credit cards sometimes allows us to make costly purchases. However, cybercriminals misuse these features in other ways.

A system that can classify such transactions based on their characteristics and flag them as fraudulent for bank officials to act upon must be developed to address this issue.

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Problem Definition

There is an increase in credit card fraud. Credit card fraud can occur in both online and offline transactions. When engaging in illegal or fraudulent activity, virtual cards are required for online transactions, while physical cards are used for offline transactions. Many fraudulent transactions can occur without the perpetrators' knowledge because of these credit card fraud activities.

Fraudsters seek confidential information like credit card numbers, bank account numbers, and other user data to conduct transactions. For online transactions, fraudsters must steal the user's identification as well as online data to complete the transaction, whereas, for offline transactions, fraudsters must steal the user's credit card. Consequently, credit card fraud has emerged as a major issue in today's technological environment, affecting bank transactions in a significant way.

Sensitive data is lost because of numerous fraudulent transactions that are difficult for both the customer and the banking authority to identify. Based on transaction behavior, there are several models for identifying fraudulent transactions. These models fall into two categories: algorithms for supervised and unsupervised learning in the current system, they determined the accuracy of the fraudulent activities by employing techniques like Cluster Analysis, Support Vector Machine, and Nave Bayer's Classification. The purpose of this paper is to determine the accuracy of fraudulent transactions by using the Random Forest Algorithm.

1.2 Objectives and Scope

The objective of this project is to detect credit card fraud accurately. Machine learning helps us to detect these types of fraud activities that occur in credit card transactions in an accurate manner. We have included the issues that are responsible for and the activities that lead to credit card fraud in this project.

Several machine learning algorithms, such as logistic regression and random forest employing ensemble classifiers on an unbalanced dataset, are constructed by applying boosting techniques to it. The random forest classifier and the ADA boost algorithm were utilized in this system. Accuracy, precision, and the area under the curve score are the foundations of both algorithms.

We choose the algorithm with the highest area under the curve score, precision, and accuracy after comparing their outputs. We choose the best algorithm for detecting credit card fraud based on this. This study's conclusion demonstrates how to train and evaluate the best classifier using supervised methods, resulting in a more precise solution.

1.3 System Requirements

1.3.1 Hardware Requirements:

Processor	Dual-Core i3 or above
Ram	8 GB or above
Storage	20 GB Hard-disk space and above

1.3.2 Software Requirements

Operating System	OS Independent
Software	Jupyter Notebook
Packages	Pandas, Numpy, Matplotlib, Seaborn, Plotly, Sklearn, LightGBM, XGBoost, Catboost

2. LITERATURE SURVEY

Aleskerov et al. [6] present CARDWATCH, a Neural Network based database mining system used for credit card fraud detection. The system has an interface to a variety of commercial databases and a graphical user interface.

Jianyun et al. [7] suggested a framework for detecting fraudulent transactions in an online system. That paper describes an FP tree-based method to dynamically create user profile for the purpose of fraud detection. But this technique doesn’t consider unusual patterns i.e., short term behavioral changes of genuine card holders.

Wen-Fang et al. [8] have proposed research on credit card fraud detection model which is based on outlier detection mining on distance sum, which shows that it can detect credit card fraud better than anomaly detection based on clustering.

3. SOFTWARE REQUIREMENT SPECIFICATION (SRS) AND DESIGN

3.1 Purpose

The purpose of this project is to detect credit card fraud accurately model which helps the company in determining the transactions that are fraudulent or not to take actions on it. This model will help companies to detect fraud faster to take measures to protect their customers from being framed.

3.2 Definition

To build a credit card fraud detection model which takes a dataset containing fraudulent and non-fraudulent transactions to determine whether a new transaction is fraudulent or not.

3.3 Abbreviations

1. ML - Machine Learning
2. CAT Boost - Categorical Boosting
3. ADA Boost- Adaptive Boosting
4. XG Boost - Extreme Gradient Boosting
5. Light GBM- Light Gradient Boosting Machine
6. AUC - Area Under the Curve
7. ROC - Receiver Operating Characteristic curve
8. CSV - Comma Separated File

3.4 Overall Description

3.4.1 Product Functions

The product function includes a jupyter notebook which contains the required libraries, model building, visualization packages and finally evaluation metrics for credit card fraud detection.

4. PROJECT ANALYSIS AND DESIGN

4.1 Methodologies Adapted

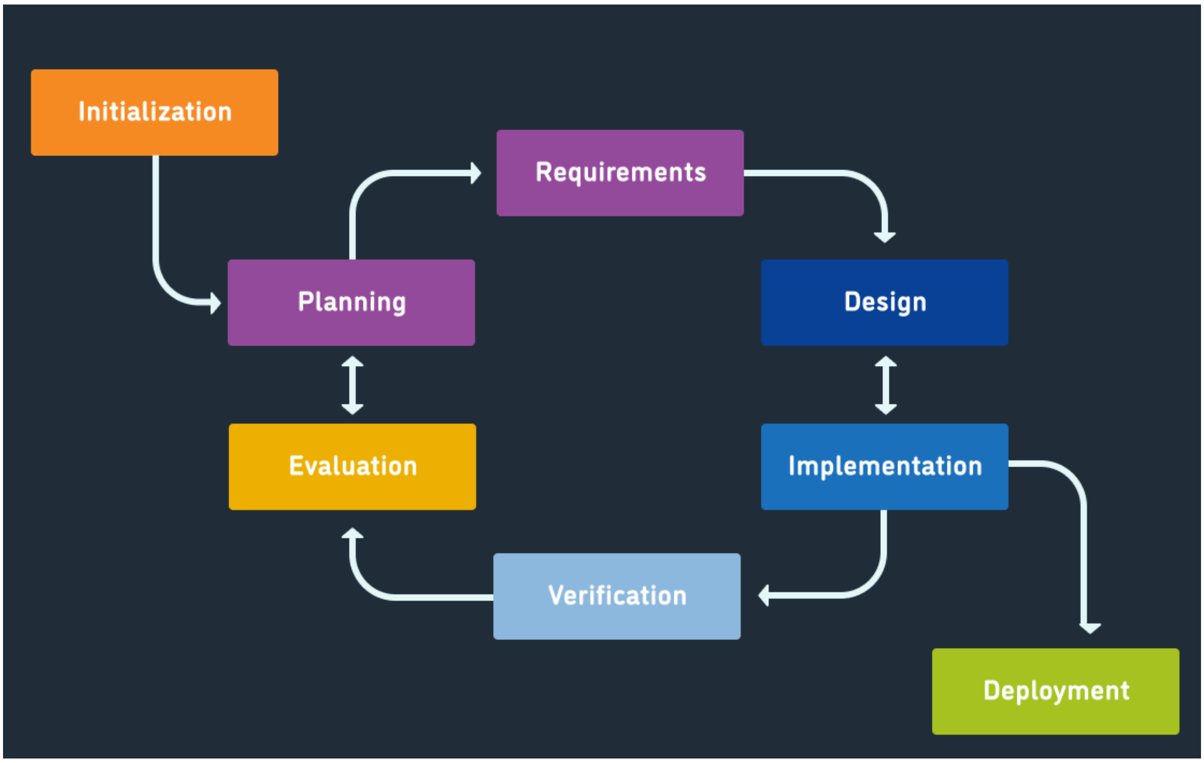
Methodology involves dividing software development work into distinct stages and coming upwith tasks or activities aimed at achieving better planning and time management. It is considered a trivial part of the systems development life cycle.

Iterative Model:

The iterative model is a particular implementation of a software development life cycle (SDLC) that focuses on an initial, simplified implementation, which then progressively gainsmore complexity and a broader feature set until the final system is complete.

In this Model, you can start with some of the software specifications and develop the firstversion of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.

The Iterative Model allows the accessing earlier phases, in which the variations maderespectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.



When to use the Iterative Model?

1. When requirements are defined clearly and easy to understand.
2. When the software application is large.
3. When there is a requirement of changes in future.

Advantages of Iterative Model:

1. Testing and debugging during smaller iteration are easy.
2. A Parallel development can plan.
3. It is easily acceptable to ever-changing needs of the project.
4. Risks are identified and resolved during iteration.
5. Limited time spent on documentation and extra time on designing.

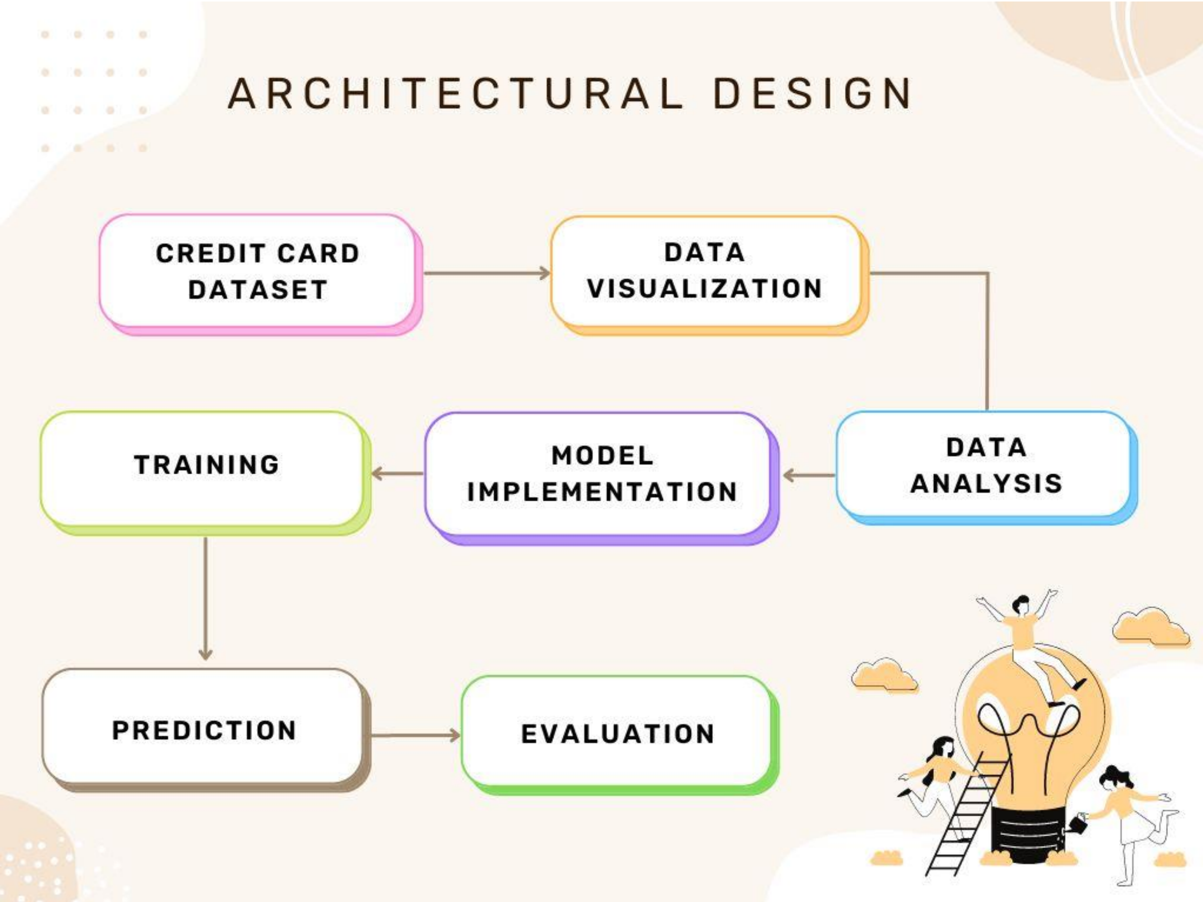
4.2 Proposed Technique

The proposed techniques for detecting fraud in credit card systems are used in this paper. We evaluate each algorithm's accuracy, precision, and AUC scores to determine which one is the most accurate and can be used by credit card merchants to identify fraudulent transactions. The architectural diagram used to select the optimal algorithm for the system's framework is depicted in the image below.

Steps For Finding Best Algorithm:

1: Import the dataset
2: Convert the data into data frames format
3: Do random sampling
4: Decide the amount of data for training data and testing data
5: Give 70% data for training and remaining data for testing
6: Assign train datasets to the models
7: Apply the algorithm among 3 different algorithms and create the model
8: Make predictions for test datasets for each algorithm
9: Calculate accuracy of each algorithm
10: Apply confusion matrix for each variable.
11: Compare the algorithms for all the variables and find out the best algorithm.

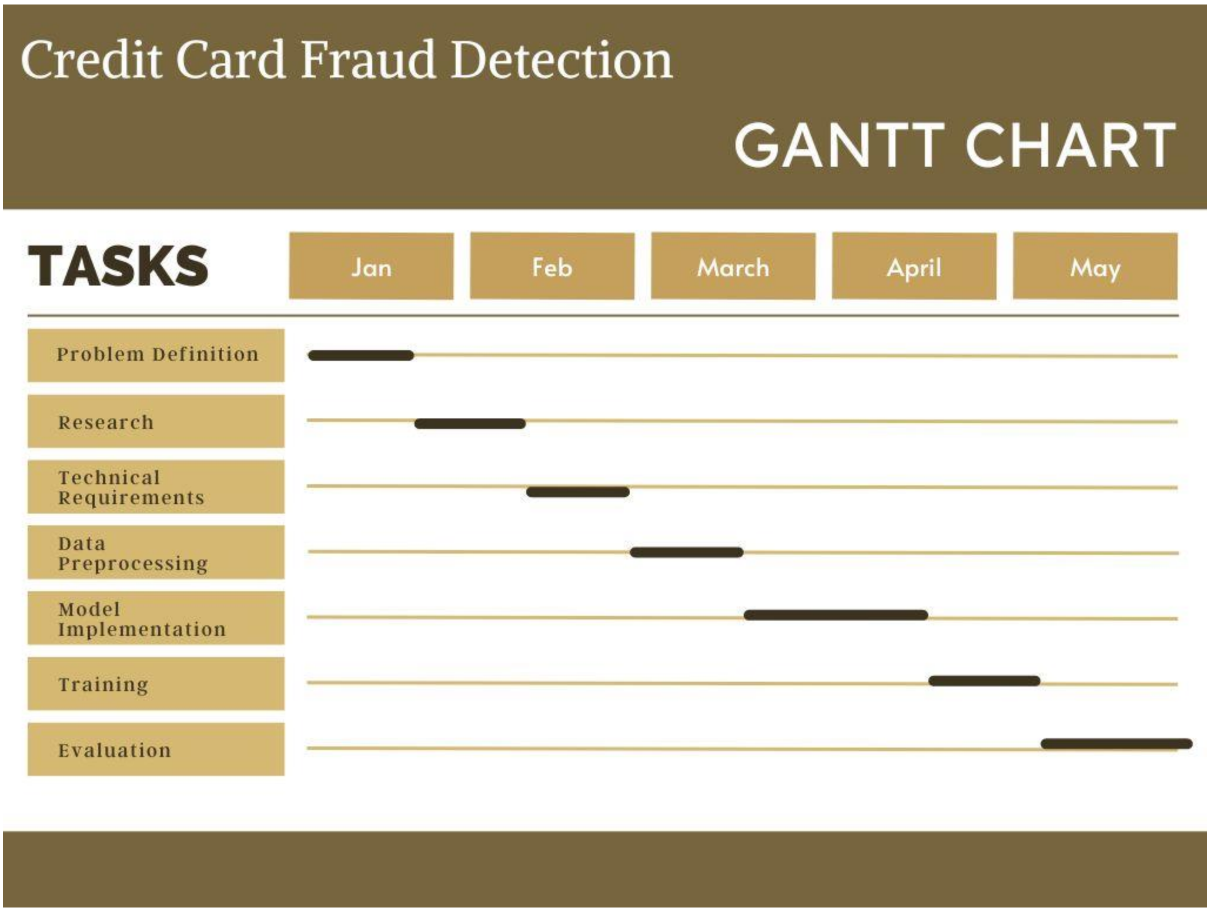
4.3 Architectural Design



5. PROJECT IMPLEMENTATION AND TESTING

5.1 Gantt Chart

Gantt chart is a type of a bar chart that is used for illustrating project schedules. Gantt charts can be used in any projects that involve effort, resources, milestones, and deliveries. Gantt charts allow project managers to track the progress of the entire project. Through Gantt charts, the project manager can keep a track of the individual tasks as well as of the overall project progression. Gantt charts can be successfully used in projects of any scale. When using Gantt charts for large projects, there can be an increased complexity when tracking the tasks.



5.2 Code

Credit Card Fraud Detection Predictive Models

Load packages

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import plotly.graph_objs as go
import plotly.figure_factory as ff
from plotly import tools
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
init_notebook_mode(connected=True)

import gc
from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import roc_auc_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from catboost import CatBoostClassifier
import lightgbm as lgb
from lightgbm import LGBMClassifier
import xgboost as xgb

pd.set_option('display.max_columns',100)

RFC_METRIC = 'gini' #metric used for RandomForestClassifier
NUM_ESTIMATORS = 100 #number of estimators used for RandomForestClassifier
NO_JOBS = 4 #number of parallel jobs used for RandomForestClassifier
RANDOM_STATE = 2018
```

```
MAX_ROUNDS = 1000 #lgb iterations
EARLY_STOP = 50 #lgb early stop
OPT_ROUNDS = 1000 #To be adjusted based on best validation rounds
VERBOSE_EVAL = 50 #print out metric result


import os

PATH ="C://Users/Pallavi PC/Documents/Sem3Project/CreditCard"
```

Read The Data

In [2]:
```
data_df = pd.read_csv("C:/Users/Pallavi PC/Documents/Sem3Project/CreditCard/creditcard.csv")
```

Check the data

In [3]:
```
print("Credit Card Fraud Detection data - rows:",data_df.shape[0]," columns:",data_df.shape[1])
```

```
Credit Card Fraud Detection data - rows: 284807  columns: 31
```

In [4]:
```
data_df.head()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670

Check missing data

In [5]:
```
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()
```

Out[5]:

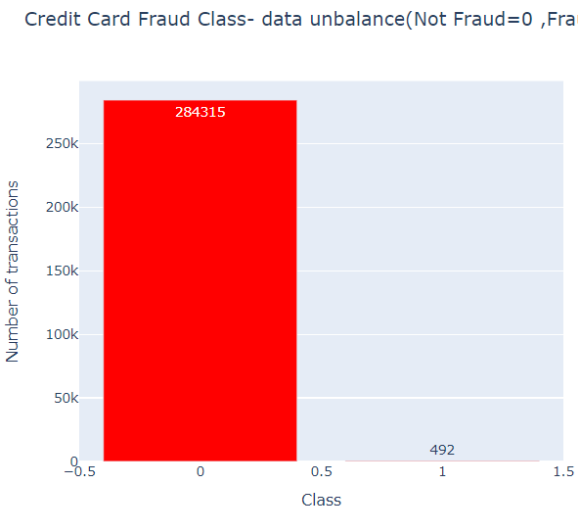
	Time	V16	Amount	V28	V27	V26	V25	V24	V23	V22	V21	V20	V19	V18	V17	V15	V1	V14	V13	V12	V11	V10	V9	V8	V7	V6	V5	V4
Total	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Percent	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

There is no missing data in the entire dataset.

Data Unbalance

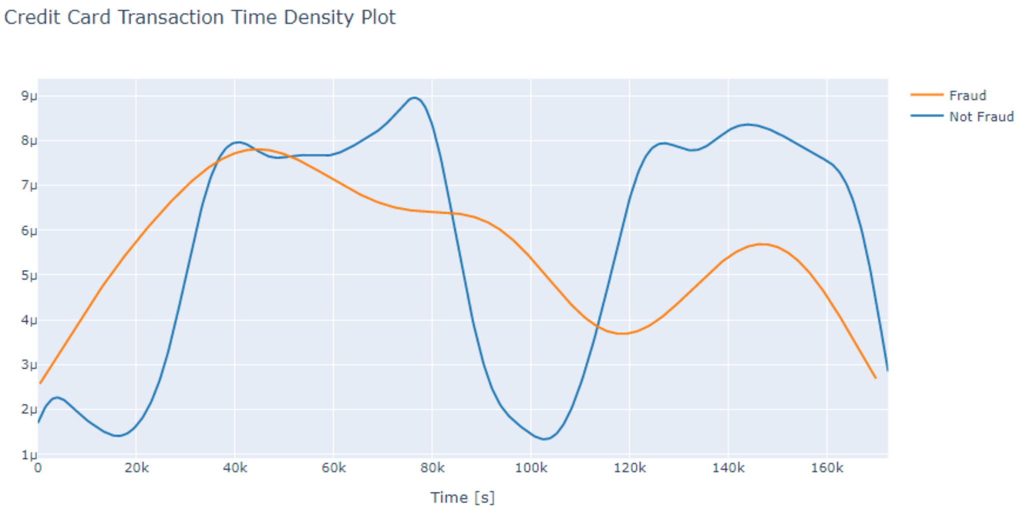
In [6]:
```
temp=data_df["Class"].value_counts()
df=pd.DataFrame({'Class':temp.index,'values':temp.values})


trace=go.Bar(
    x=df['Class'],y=df['values'],
    name="Credit Card Fraud Class- data unbalance(Not Fraud=0 ,Fraud = 1)",
    marker=dict(color="Red"),
    text=df['values']
)
data=[trace]
layout=dict(title='Credit Card Fraud Class- data unbalance(Not Fraud=0 ,Fraud = 1)',
            xaxis=dict(title='Class',showticklabels=True),
            yaxis=dict(title='Number of transactions'),
            hovermode='closest',width=600
)
```

```
fig=dict(data=data,layout=layout)
iplot(fig,filename='class')
```



Data Exploration

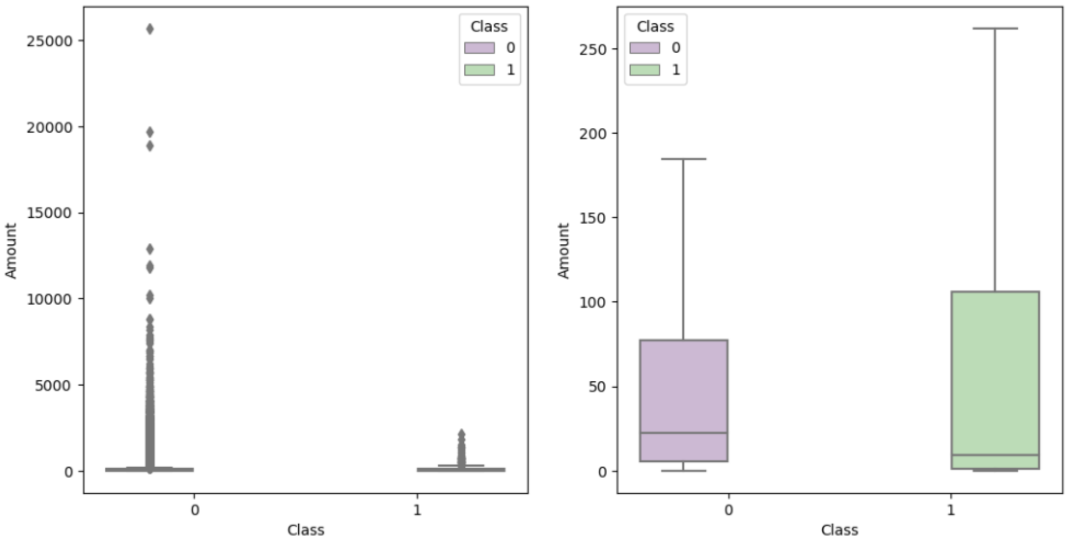
In [7]:
```
class_0=data_df.loc[data_df['Class']==0]["Time"]
class_1=data_df.loc[data_df['Class']==1]["Time"]

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 Transaction Time Density Plot',xaxis=dict(title='Time [s]'))
iplot(fig,filename='dist_only')
```



Fraudulent transactions have a distribution even more than valid transactions -Equally distributed in time

#Transactions amount

In [8]:
```
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();
```



In [9]:
```
tmp=data_df[['Amount','Class']].copy()
class_0=tmp.loc[tmp['Class']==0]['Amount']
class_1=tmp.loc[tmp['Class']==1]['Amount']
class_0.describe()
```

Out[9]:
```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

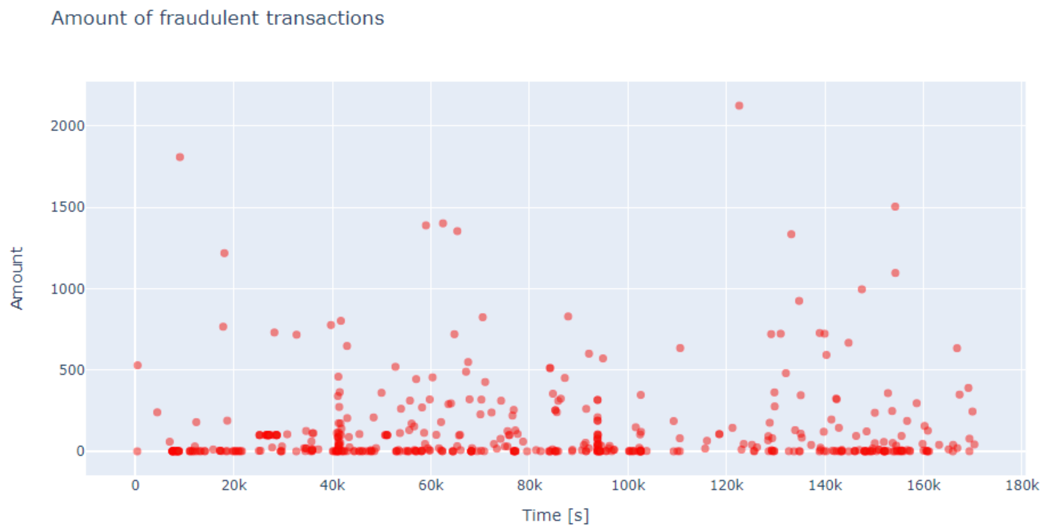
In [10]:
```
class_1.describe()
```

Out[10]:
```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

The real transaction have a larger mean value, larger Q1, smaller Q3 and larger outliers. Fraudulent transactions have a smaller Q1 and mean and smaller outliers.

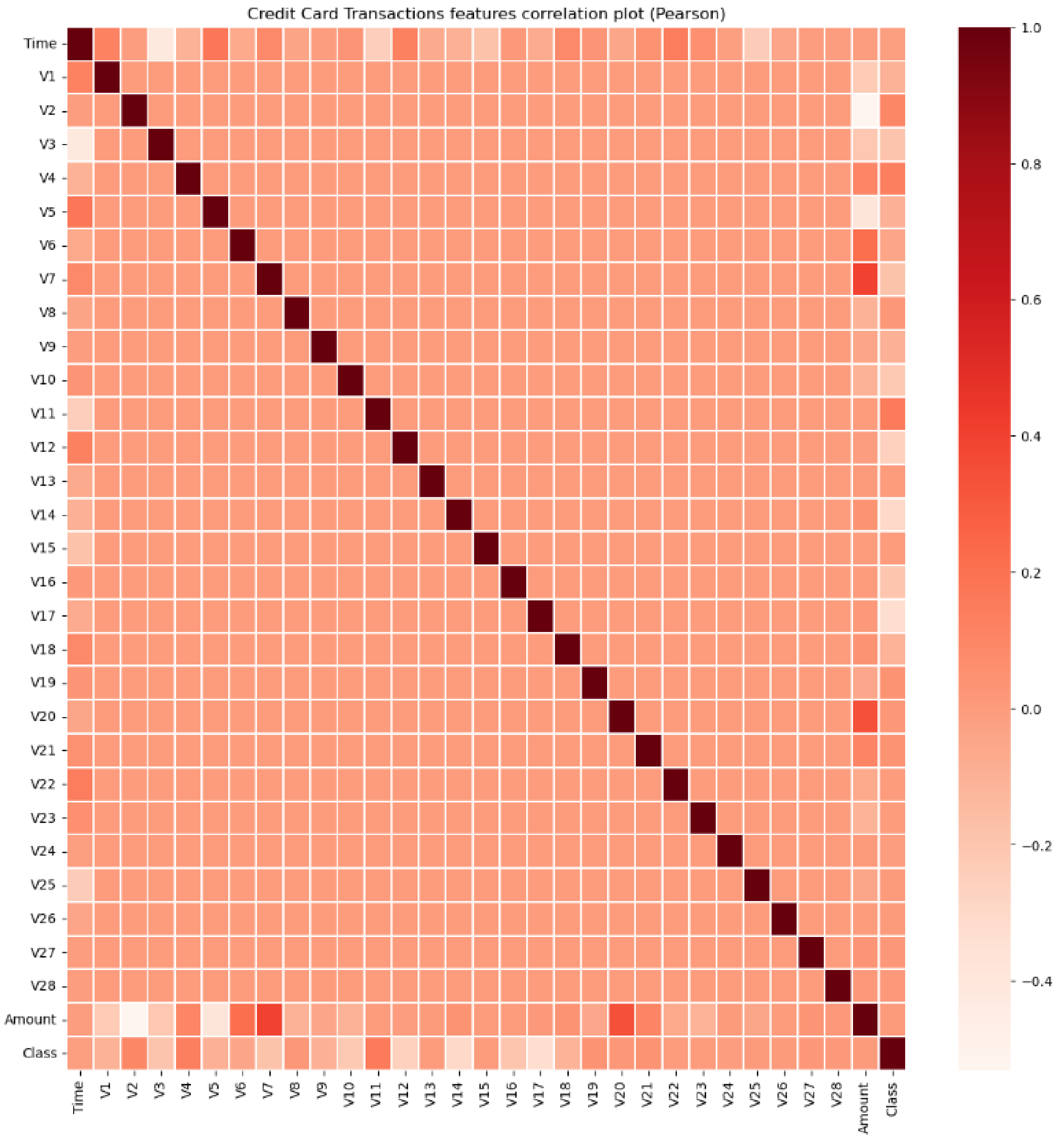
Lets plot the fraudulent transactions (amount) against time. The time is shown in seconds.

In [11]:
```
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')
```



Features Correlation

In [12]:
```
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()
```

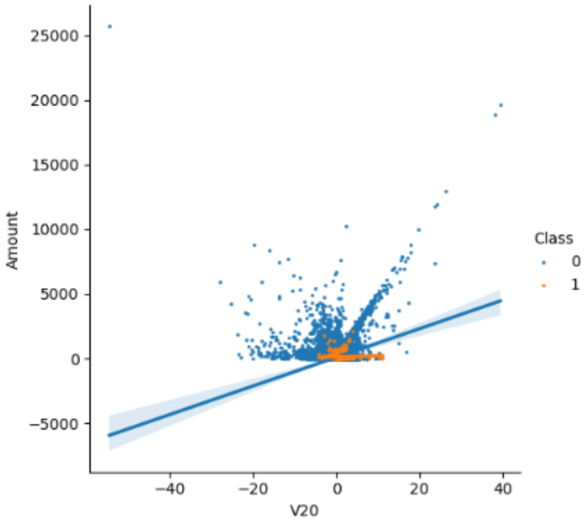


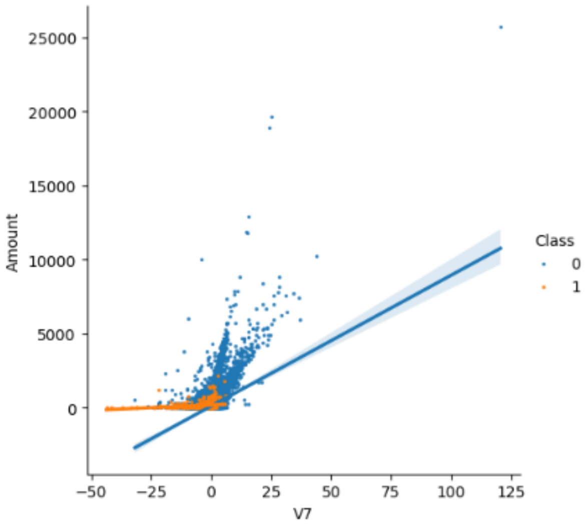
As expected there is no notable correlation between features V1-V28. There are certain correlations between some of these features and Time (inverse correlation with V3) and Amount(direct correlation with V7 and V20, inverse correlation with V2 and V5).

Let's plot the correlated and inverse correlated values on the same graph.

Let's start with the direct correlated values (V20;Amount) and (V7;Amount)

In [13]:
```
s=sns.lmplot(x='V20',y='Amount',data=data_df,hue='Class',fit_reg=True,scatter_kws={'s':2})
s=sns.lmplot(x='V7',y='Amount',data=data_df,hue='Class',fit_reg=True,scatter_kws={'s':2})
plt.show()
```

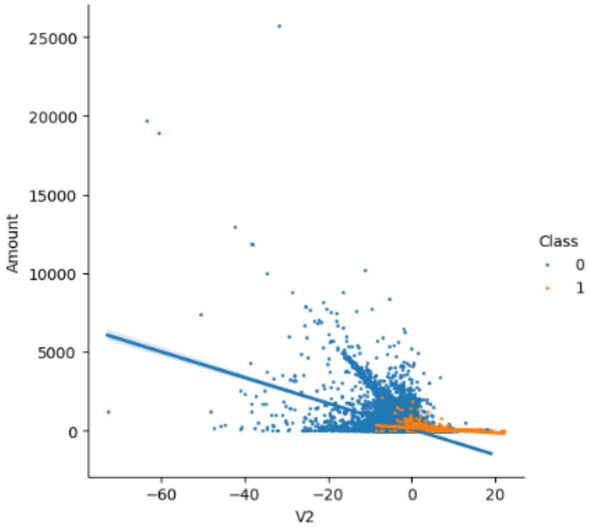


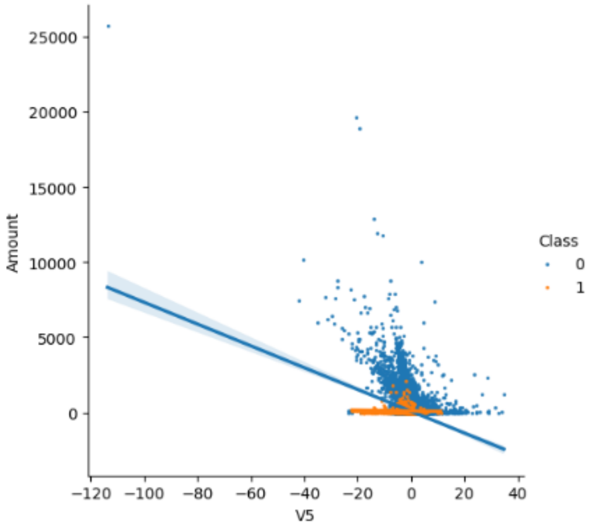


We can confirm that the two couples of features are correlated(the regression lines for Class=0 have a positive slope whilst the regression lines for Class=1 have a smaller positive slope).

Let's plot the inverse correlated values now.

In [14]:
```
s=sns.lmplot(x='V2',y='Amount',data=data_df,hue='Class',fit_reg=True,scatter_kws={'s':2})
s=sns.lmplot(x='V5',y='Amount',data=data_df,hue='Class',fit_reg=True,scatter_kws={'s':2})
plt.show()
```





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).

Predictive models

Let's define predictor features and the target features.

In [15]:
```
target='Class'
predictors= ['Time','V1','V2','V3','V4','V5','V6','V7','V8','V9','V10',\
            'V11','V12','V13','V14','V15','V16','V17','V18','V19',\
            'V20','V21','V22','V23','V24','V25','V26','V27','V28',\
            'Amount']
```

Split data in train, test and validation set

Lets define train, validation and test sets

In [16]:
```
train_df, test_df=train_test_split(data_df,random_state=RANDOM_STATE,shuffle=True)
train_df, valid_df=train_test_split(train_df,random_state=RANDOM_STATE,shuffle=True)
```

Lets start with a RandomForestClassifier model

RandomForestClassifier

Define model parameters

Lets set the parameters for the model

In [17]:
```
clf=RandomForestClassifier(n_jobs=NO_JOBS,
                           random_state=RANDOM_STATE,
                           criterion=RFC_METRIC,
                           n_estimators=NUM_ESTIMATORS,
                           verbose=False)
```

Lets train the RandomForestClassifier using the train_df data and fit function.

In [18]:
```
clf.fit(train_df[predictors],train_df[target].values)
```

Out[18]:
```
RandomForestClassifier(n_jobs=4, random_state=2018, verbose=False)
```

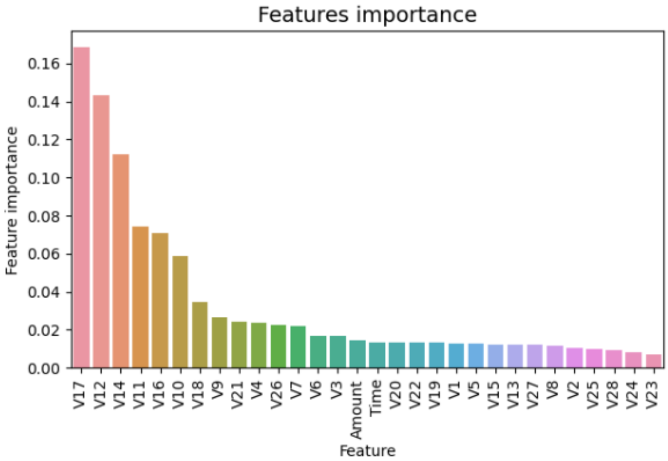
Lets now predict the target values for the valid_df data using predict function.

In [19]:
```
preds= clf.predict(valid_df[predictors])
```

Lets visualize the features importance.

Features importance

In [20]:
```
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()
```

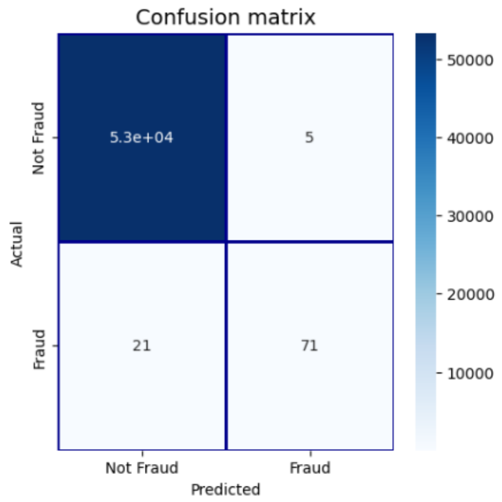


The most important features are V17,V12,V14,V11,V16,V10.

Confusion matrix

Lets show a confusion matrix for the results we obtained

In [21]:
```
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()
```



In [22]:
```
roc_auc_score(valid_df[target].values,preds)
```

Out[22]: 0.8858226697006027

The ROC-AUC score obtained with RandomForestClassifier is 0.88

AdaBoostClassifier

Prepare the model

In [23]:
```
clf=AdaBoostClassifier(random_state=RANDOM_STATE,
                       algorithm='SAMME.R',
                       learning_rate=0.8,
                       n_estimators=NUM_ESTIMATORS)
```

Fit the model

In [24]:
```
clf.fit(train_df[predictors],train_df[target].values)
```

Out[24]: AdaBoostClassifier(learning_rate=0.8, n_estimators=100, random_state=2018)

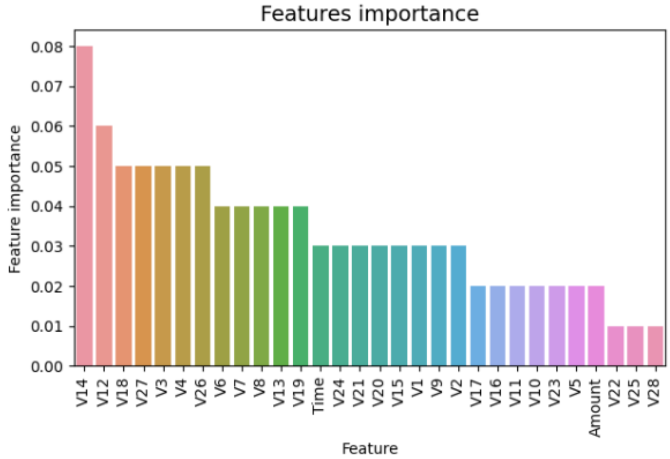
Predict the target values

Lets now predict the target values for the valid_df data using predict function

In [25]:
```
preds=clf.predict(valid_df[predictors])
```

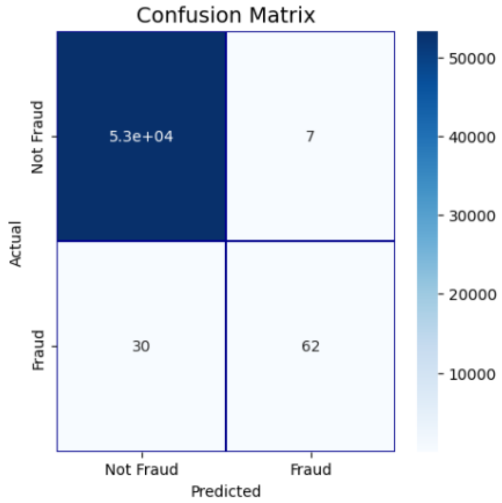
Features importance

In [26]:
```
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

In [27]:
```
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()
```



Area under curve

In [28]:
```
roc_auc_score(valid_df[target].values,preds)
```

Out[28]: 0.8368908680156263

The ROC-AUC score obtained with AdaBoostClassifier is 0.83

CatBoostClassifier

Prepare the model

In [29]:
```
clf=CatBoostClassifier(iterations=500,
                      learning_rate=0.02,
                      depth=12,
                      eval_metric='AUC',
                      random_seed=RANDOM_STATE,
                      bagging_temperature=0.2,
                       od_type='Iter',
                       metric_period=VERBOSE_EVAL,
                       od_wait=100)
```

In [30]:
```
clf.fit(train_df[predictors],train_df[target].values)
```

```
0:      total: 804ms    remaining: 6m 41s
50:     total: 28.7s    remaining: 4m 12s
100:    total: 59.4s    remaining: 3m 54s
150:    total: 1m 29s   remaining: 3m 25s
200:    total: 1m 59s   remaining: 2m 57s
250:    total: 2m 29s   remaining: 2m 28s
300:    total: 2m 59s   remaining: 1m 58s
350:    total: 3m 29s   remaining: 1m 28s
400:    total: 3m 57s   remaining: 58.8s
450:    total: 4m 29s   remaining: 29.3s
499:    total: 4m 59s   remaining: 0us
```

Out[30]: <catboost.core.CatBoostClassifier at 0x267244a6400>

Predict the target values

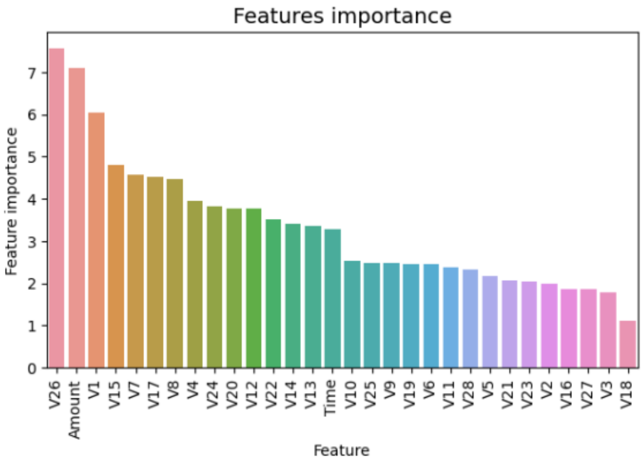
Lets now predict the target values for the valid_df data using predict function

In [31]:
```
preds=clf.predict(valid_df[predictors])
```

Features importance

Lets visualize the features importance

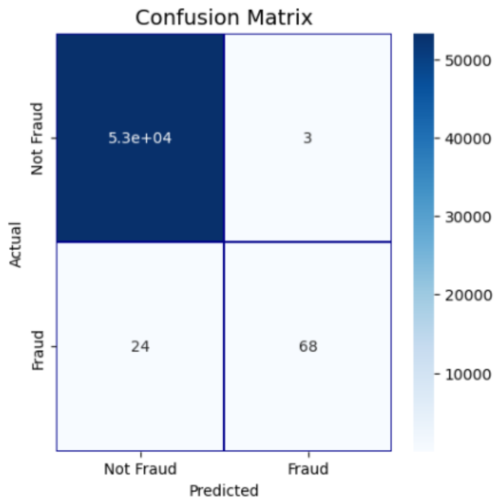
In [32]:
```
tmp=pd.DataFrame({'Feature':predictors,'Feature importance':clf.feature_importances_})
tmp=tmp.sort_values(by='Feature importance',ascending=False)
plt.figure(figsize=(7,4))
plt.title('Features importance',fontsize=14)
s= sns.barplot(x='Feature',y='Feature importance',data=tmp)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
plt.show()
```



Confusion matrix

Lets visualize the confusion matrix

In [33]:
```
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()
```



Area under curve

In [34]:
```
roc_auc_score(valid_df[target].values,preds)
```

Out[34]: 0.8695370800812312

The ROC-AUC score obtained with CatBoostClassifier is 0.83

XGBoost

Prepare the model

In [35]:
```
#Prepare the train and valid datasets
dtrain= xgb.DMatrix(train_df[predictors],train_df[target].values)
dvalid= xgb.DMatrix(valid_df[predictors],valid_df[target].values)
dtest= xgb.DMatrix(test_df[predictors],test_df[target].values)

#what to monitor(in this case-"train" and "valid")
watchlist=[(dtrain,'train'),(dvalid,'valid')]

#set xgboost parameters
params={}
params['ojective']='binary:logistic'
params['eta']=0.039
params['silent']=True
params['max_depth']=2
params['subsample']=0.8
params['colsample_bytree']=0.9
params['eval_metric']='auc'
params['random_state']=RANDOM_STATE
```

Train the model

In [36]:
```
model=xgb.train(params,
               dtrain,
               MAX_ROUNDS,
               watchlist,
               early_stopping_rounds=EARLY_STOP,
               maximize=True,
               verbose_eval=VERBOSE_EVAL)
```

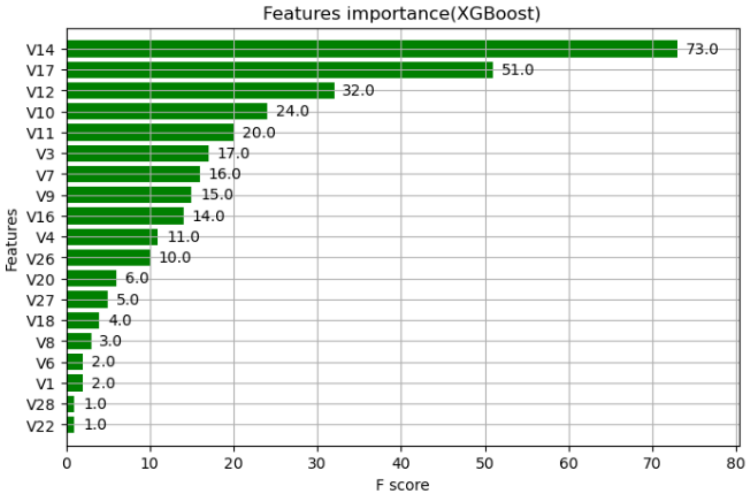
```
C:\Users\Pallavi PC\Anaconda\lib\site-packages\xgboost\core.py:617: FutureWarning:

Pass `evals` as keyword args.
```

```
[16:51:09] WARNING: C:/buildkite-agent/builds/buildkite-windows-cpu-autoscaling-group-i-03de431ba26204c4d-1/xgboost/xgboost-ci-windows/src/learner.cc:767:
Parameters: { "ojective", "silent" } are not used.

[0]	train-auc:0.88432	valid-auc:0.86945
[50]	train-auc:0.92204	valid-auc:0.92355
[100]	train-auc:0.92375	valid-auc:0.92346
[105]	train-auc:0.92374	valid-auc:0.92347
```

The best validation score(ROC-AUC) was 0.923 for round 55

Plot variable importance

In [37]:
```
fig, (ax)=plt.subplots(ncols=1,figsize=(8,5))
xgb.plot_importance(model,height=0.8,title="Features importance(XGBoost)",ax=ax,color="green")
plt.show()
```



Predict test set

We used the train and validation sets for training and validation. We will use the trained model now to predict the target value.

In [38]:
```
preds=model.predict(dtest)
```

Area under curve

In [39]:
```
roc_auc_score(test_df[target].values,preds)
```

Out[39]:
```
0.9222889177357566
```

The ROC-AUC score obtained with XGBoostClassifier is 0.92

LightGBM

Define model parameters

In [40]:
```
params= {
            'boosting_type':'gbdt',
            'objective':'binary',
            'metric':'auc',
            'learning_rate':0.05,
            'num_leaves': 7,
            'maxdepth': 4,
            'min_child_samples':100,
            'max_bin':100,
            'subsample':0.9,
            'subsample_freq':1,
            'colsample_bytree0':0.7,
            'min_child_weight':0,
            'min_split_gain':0,
            'nthread':8,
            'verbose':0,
            'scale_pos_weight':150,
}
```

Prepare the model

In [41]:
```
dtrain= lgb.Dataset(train_df[predictors].values,
                   label=train_df[target].values,
                   feature_name=predictors)
dvalid=lgb.Dataset(valid_df[predictors].values,
                  label=valid_df[target].values,
                  feature_name=predictors)
```

Run the model

Lets run the model using train function

In [42]:
```
evals_results={}

model=lgb.train(params,
               dtrain,
               valid_sets=[dtrain,dvalid],
               valid_names=['train','valid'],
               evals_result=evals_results,
               num_boost_round=MAX_ROUNDS,
               early_stopping_rounds=2*EARLY_STOP,
               verbose_eval=VERBOSE_EVAL,
               feval=None)
```

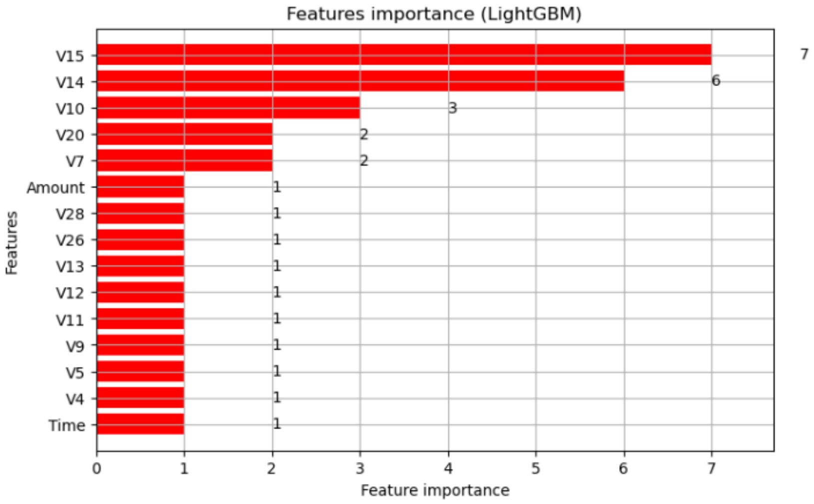
```
C:\Users\Pallavi PC\Anaconda\lib\site-packages\lightgbm\engine.py:181: UserWarning:

'early_stopping_rounds' argument is deprecated and will be removed in a future release of LightGBM. Pass 'early_stopping()' cal
lback via 'callbacks' argument instead.

C:\Users\Pallavi PC\Anaconda\lib\site-packages\lightgbm\engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of LightGBM. Pass 'log_evaluation()' callback via
'callbacks' argument instead.

C:\Users\Pallavi PC\Anaconda\lib\site-packages\lightgbm\engine.py:260: UserWarning:

'evals_result' argument is deprecated and will be removed in a future release of LightGBM. Pass 'record_evaluation()' callback
via 'callbacks' argument instead.
```

```
[LightGBM] [Warning] Unknown parameter: maxdepth
[LightGBM] [Warning] Unknown parameter: colsample_bytree0
[LightGBM] [Warning] Unknown parameter: maxdepth
[LightGBM] [Warning] Unknown parameter: colsample_bytree0
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.014125 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Warning] Unknown parameter: maxdepth
[LightGBM] [Warning] Unknown parameter: colsample_bytree0
Training until validation scores don't improve for 100 rounds
[50]    train's auc: 0.987391   valid's auc: 0.901302
[100]   train's auc: 0.991777   valid's auc: 0.90235
Early stopping, best iteration is:
[5]     train's auc: 0.946079   valid's auc: 0.954642
```

Best validation score(ROC-AUC) was obtained for round 16 o which AUC-0.962

Plot variable importance

In [43]:
```
fig,(ax)= plt.subplots(ncols=1,figsize=(8,5))
lgb.plot_importance(model,height=0.8,title="Features importance (LightGBM)",ax=ax,color="red")
plt.show()
```



Predict test data

In [44]:
```
preds=model.predict(test_df[predictors])
```

Area under curve

Lets calculate the ROC-AUC score for the prediction

In [45]:
```
roc_auc_score(test_df[target].values,preds)
```

Out[45]: 0.9152625551072966

The ROC-AUC score obtained for the test set is 0.91

5.3 Results

	Random Forest	ADA Boost	CAT Boost	XG Boost	Light GBM
AUC Score	0.88	0.83	0.86	0.92	0.96

6. FUTURE SCOPE

In this instance, the comparative analysis shows that the Random Forest with Boosting method clearly outperforms the other credit card fraud detection methods. However, one of the paper's flaws is that we can't use machine learning to distinguish the names of fraud and non-fraud transactions for the supplied dataset when using the three approaches. We can overcome this obstacle for the project's future development using a variety of methods.

7. CONCLUSION

We have learned about machine learning applications like Random Forest, ADA Boost, CAT Boost, XG Boost, and the Light GBM model in this paper. If these algorithms are incorporated into a bank's credit card fraud detection system, it will be possible to anticipate the likelihood of fraudulent transactions shortly after they occur. In addition, banks can minimize risks and avoid large losses by employing a variety of antifraud strategies.

In contrast to other classification issues, the study's objective was approached with a variable penalty for misclassification. Precision, f1-score, and accuracy are used to evaluate the performance of the proposed system. We examined the data, identifying data imbalances, displaying the features, and determining their relationships. Precision, f1-score, and accuracy are used to evaluate the performance of the proposed system.

With the Random Forest Classifier, we were able to predict the target for the test set with an AUC score of 0.88. With the ADA Boost Classifier, we were able to predict the target for the test set with an AUC score of 0.83, with the CAT Boost Classifier, we were able to predict the target for the test set with an AUC score of 0.86. With the XG Boost Classifier, we were able to predict the target for the test set with an AUC score of 0.92. With the Light GBM Classifier, we were able to predict the target for the test set with an AUC score of 0.96. Using cross-validation, we were able to achieve a test prediction AUC value of 0.91.

8. BIBLIOGRAPHY

[1] Credit Card Fraud Detection | Kaggle

[2] Credit Card Fraud Detection: Everything You Need to Know (inscribe.ai)

[3] https://towardsdatascience.com/credit-card-fraud-detection-9bc8db79b956?gi=13ce411a06d2

[4] (PDF) Credit card fraud and detection techniques: A review (researchgate.net)

[5] (PDF) Credit Card Fraud Detection using Machine Learning and Data Science (researchgate.net)

[6] CARDWATCH: a neural network based database mining system for credit card fraud detection | IEEE Conference Publication | IEEE Xplore

[7] [PDF] Behaviour Mining for Fraud Detection | Semantic Scholar

[8] Research on Credit Card Fraud Detection Model Based on Distance Sum | IEEE Conference Publication | IEEE Xplore