About Project

In this project we will predict fraudulent activity in credit card transactions with the help of Machine learning models.

About Data

The data set includes credit card transactions made by European cardholders over a period of two days in September 2013. Out of a total of 2,84,807 transactions, 492 were fraudulent. This data set is highly unbalanced, with the positive class (frauds) accounting for 0.172% of the total transactions. The data set has also been modified with principal component analysis (PCA) to maintain confidentiality. Apart from 'time' and 'amount', all the other features (V1, V2, V3, up to V28) are the principal components obtained using PCA. The feature 'time' contains the seconds elapsed between the first transaction in the data set and the subsequent transactions. The feature 'amount' is the transaction amount. The feature 'class' represents class labelling, and it takes the value of 1 in cases of fraud and 0 in others.

Scope & Objective

For many banks, retaining high profitable customers is the number one business goal. Banking fraud, however, poses a significant threat to this goal for different banks. In terms of substantial financial losses, trust and credibility, this is a concerning issue to both banks and customers alike.

In the banking industry, credit card fraud detection using machine learning is not only a trend but a necessity for them to put proactive monitoring and fraud prevention mechanisms in place. Machine learning is helping these institutions to reduce time-consuming manual reviews, costly chargebacks and fees as well as denials of legitimate transactions.

Business Problem Statement

The problem statement chosen for this project is to predict fraudulent credit card transactions with the help of machine learning models. In this project, we will analyse customer-level data that has been collected and analyzed during a research collaboration of World line and the Machine Learning Group.

It has been estimated by Nilson Report that by 2020, banking frauds would account for $30 billion worldwide. With the rise in digital payment channels, the number of fraudulent transactions is also increasing in new and different ways.

Data dictionary

The data set includes credit card transactions made by European cardholders over a period of two days in September 2013. Out of a total of 2,84,807 transactions, 492 were fraudulent. This data set is highly unbalanced, with the positive class (frauds) accounting for 0.172% of the total transactions. The data set has also been modified with principal component analysis (PCA) to maintain confidentiality. Apart from 'time' and 'amount', all the other features (V1, V2, V3, up to V28) are the principal components obtained using PCA. The feature 'time' contains the seconds elapsed between the first transaction in the data set and the subsequent transactions. The feature 'amount' is the transaction amount. The feature 'class' represents class labeling, and it takes the value of 1 in cases of fraud and 0 in others.

Preprocessing the Data

1.Importing the requred

```
import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.preprocessing import  RobustScaler
from sklearn.preprocessing import PowerTransformer

from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV

#pip install xgboost

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

#pip install imblearn

#pip install -U imbalanced-learn

```
#pip install imblearn # if you dont have imblearn install it
import math
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler

from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import f1_score, classification_report
```

2.Importing the table

```
df=pd.read_csv("creditcard.csv")
print(df.head())
```

```
print("------------------------------------------------------------------
------------------------------")
print("Columns names")
print(df.columns)
```

```
   Time        V1        V2        V3        V4        V5        V6
V7  \
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388
0.239599
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -
0.078803
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499
0.791461
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203
0.237609
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921
0.592941

         V8        V9  ...       V21       V22       V23       V24
V25  \
0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928
0.128539
1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846
0.167170
2  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -
0.327642
3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575
0.647376
4 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -
0.206010

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0

[5 rows x 31 columns]
----------------------------------------------------------------------
-------------------------
Columns names
Index(['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',
       'Class'],
      dtype='object')
```

3.Describing data

```
print("Data Description")
print(df.describe())
print("---------------------------------------------------------------------------------------------")
print("Number of rows and columns")
print(df.shape)
print("---------------------------------------------------------------------------------------------")
print("data information")
print(df.info())
print("---------------------------------------------------------------------------------------------")
```

```
Data Description
                Time            V1            V2            V3  
V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  
2.848070e+05   
mean    94813.859575  3.918649e-15  5.682686e-16 -8.761736e-15  
2.811118e-15   
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  
1.415869e+00   
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -
5.683171e+00   
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -
8.486401e-01   
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -
1.984653e-02   
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  
7.433413e-01   
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  
1.687534e+01   

                 V5            V6            V7            V8  
V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  
2.848070e+05   
mean  -1.552103e-15  2.040130e-15 -1.698953e-15 -1.893285e-16 -
3.147640e-15   
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  
1.098632e+00   
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -
1.343407e+01   
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -
6.430976e-01   
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -
5.142873e-02   
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  
5.971390e-01   
```

```
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01
1.559499e+01

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ...  1.473120e-16  8.042109e-16  5.282512e-16  4.456271e-15
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min    ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%    ... -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%    ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%    ...  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max    ...  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00

                V25           V26           V27           V28
Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
284807.000000
mean   1.426896e-15  1.701640e-15 -3.662252e-16 -1.217809e-16
88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01
250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01
0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02
5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02
22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02
77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01
25691.160000

               Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
----------------------------------------------------------------------
-------------------------
Number of rows and columns
(284807, 31)
----------------------------------------------------------------------
-------------------------
data information
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
None
----------------------------------------------------------------------
-------------------------
```

```
print("Data percentaile")
print(df.describe(percentiles=[0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08
,0.09,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]))
print("------------
----------------------------------------------------------------------
-------------")
print("Checking for null")
print(df.isnull().sum())
```

```
#after varyfing the data we found that there is no missing of data
print("there is no null or missing data in the selected sourse")
```

```
Data percentaile
                Time            V1            V2            V3
V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05
2.848070e+05
mean    94813.859575  3.918649e-15  5.682686e-16 -8.761736e-15
2.811118e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00
1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -
5.683171e+00
1%       2422.000000 -6.563199e+00 -4.960300e+00 -3.978377e+00 -
3.122987e+00
2%       5964.120000 -4.757888e+00 -3.419452e+00 -3.192457e+00 -
2.742465e+00
3%      11490.180000 -3.864514e+00 -2.677873e+00 -2.809154e+00 -
2.528801e+00
4%      19880.720000 -3.298547e+00 -2.236980e+00 -2.572682e+00 -
2.362446e+00
5%      25297.600000 -2.899147e+00 -1.971975e+00 -2.389740e+00 -
2.195683e+00
6%      28426.360000 -2.604921e+00 -1.787717e+00 -2.232574e+00 -
2.041926e+00
7%      30660.840000 -2.381382e+00 -1.649294e+00 -2.098802e+00 -
1.913953e+00
8%      32432.000000 -2.183695e+00 -1.536374e+00 -1.985171e+00 -
1.816284e+00
9%      33699.540000 -2.025807e+00 -1.443029e+00 -1.885165e+00 -
1.732360e+00
10%     35027.000000 -1.893272e+00 -1.359862e+00 -1.802587e+00 -
1.656329e+00
20%     47694.200000 -1.134663e+00 -7.908142e-01 -1.169050e+00 -
1.066085e+00
30%     60776.000000 -7.472943e-01 -4.363931e-01 -6.168060e-01 -
6.962262e-01
40%     73261.400000 -4.267559e-01 -1.538049e-01 -1.990610e-01 -
3.941610e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -
1.984653e-02
60%    120396.000000  1.035107e+00  3.027378e-01  4.970396e-01
2.848562e-01
70%    132929.000000  1.224825e+00  6.368450e-01  8.435405e-01
5.588663e-01
80%    145247.800000  1.694936e+00  9.573136e-01  1.215700e+00
9.861875e-01
90%    157640.400000  2.015409e+00  1.326635e+00  1.676173e+00
1.482807e+00
```

```
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00
1.687534e+01

                 V5            V6            V7            V8
V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
2.848070e+05
mean  -1.552103e-15  2.040130e-15 -1.698953e-15 -1.893285e-16 -
3.147640e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00
1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -
1.343407e+01
1%    -3.060396e+00 -2.124023e+00 -3.012847e+00 -4.033899e+00 -
2.455288e+00
2%    -2.280545e+00 -1.777548e+00 -2.060577e+00 -2.482355e+00 -
2.208978e+00
3%    -1.999164e+00 -1.607532e+00 -1.756126e+00 -1.696369e+00 -
2.023964e+00
4%    -1.827092e+00 -1.488528e+00 -1.569482e+00 -1.111268e+00 -
1.889022e+00
5%    -1.702021e+00 -1.406757e+00 -1.434423e+00 -8.421469e-01 -
1.758426e+00
6%    -1.601647e+00 -1.343659e+00 -1.330192e+00 -6.975611e-01 -
1.646555e+00
7%    -1.510612e+00 -1.288168e+00 -1.246063e+00 -6.089100e-01 -
1.541464e+00
8%    -1.431076e+00 -1.242475e+00 -1.183942e+00 -5.444541e-01 -
1.446596e+00
9%    -1.364763e+00 -1.203030e+00 -1.129334e+00 -4.951725e-01 -
1.363815e+00
10%   -1.302171e+00 -1.167450e+00 -1.078148e+00 -4.589454e-01 -
1.294047e+00
20%   -8.595815e-01 -8.781527e-01 -6.833164e-01 -2.662999e-01 -
8.088755e-01
30%   -5.444552e-01 -6.678449e-01 -4.260203e-01 -1.581584e-01 -
5.021651e-01
40%   -2.901118e-01 -4.794725e-01 -1.593408e-01 -6.605715e-02 -
2.577302e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -
5.142873e-02
60%    1.866249e-01 -4.819964e-02  2.135123e-01  1.204889e-01
1.761695e-01
70%    4.570287e-01  2.232447e-01  4.461749e-01  2.434003e-01
4.459420e-01
80%    8.022349e-01  6.289952e-01  6.983318e-01  4.355289e-01
7.915832e-01
90%    1.407893e+00  1.509365e+00  1.039387e+00  7.693811e-01
1.301671e+00
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01
```

```
1.559499e+01

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ...  1.473120e-16  8.042109e-16  5.282512e-16  4.456271e-15
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min    ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
1%     ... -1.469679e+00 -1.654625e+00 -1.193417e+00 -1.657308e+00
2%     ... -9.512384e-01 -1.406782e+00 -7.635010e-01 -1.455924e+00
3%     ... -6.829981e-01 -1.270531e+00 -6.127155e-01 -1.361242e+00
4%     ... -5.651468e-01 -1.164705e+00 -5.302926e-01 -1.288145e+00
5%     ... -5.046735e-01 -1.081892e+00 -4.722464e-01 -1.143662e+00
6%     ... -4.659488e-01 -1.016797e+00 -4.299106e-01 -1.069152e+00
7%     ... -4.336716e-01 -9.706882e-01 -3.963218e-01 -1.023114e+00
8%     ... -4.060592e-01 -9.292533e-01 -3.703223e-01 -9.803386e-01
9%     ... -3.848837e-01 -8.975203e-01 -3.484155e-01 -9.326570e-01
10%    ... -3.674467e-01 -8.674884e-01 -3.286342e-01 -8.703584e-01
20%    ... -2.663929e-01 -6.416046e-01 -2.037425e-01 -4.442769e-01
30%    ... -1.878180e-01 -4.344265e-01 -1.261571e-01 -2.635025e-01
40%    ... -1.064266e-01 -2.018541e-01 -6.512670e-02 -4.580667e-02
50%    ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
60%    ...  5.038722e-02  2.005957e-01  4.484023e-02  1.664115e-01
70%    ...  1.379046e-01  4.127470e-01  1.092714e-01  3.742132e-01
80%    ...  2.354311e-01  6.485158e-01  1.942589e-01  5.323465e-01
90%    ...  3.761555e-01  9.148826e-01  3.392860e-01  7.054036e-01
max    ...  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00

                V25           V26           V27           V28
Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
284807.000000
mean   1.426896e-15  1.701640e-15 -3.662252e-16 -1.217809e-16
88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01
250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01
0.000000
1%    -1.420859e+00 -1.009384e+00 -1.247746e+00 -8.762654e-01
0.120000
2%    -1.159746e+00 -8.797258e-01 -8.162523e-01 -5.951801e-01
0.760000
3%    -1.009502e+00 -8.017020e-01 -6.149432e-01 -4.615097e-01
0.770000
4%    -9.029997e-01 -7.411072e-01 -4.979870e-01 -3.800435e-01
0.890000
5%    -8.250264e-01 -6.973483e-01 -4.152460e-01 -3.178432e-01
0.920000
6%    -7.668064e-01 -6.658172e-01 -3.544441e-01 -2.699225e-01
1.000000
7%    -7.189672e-01 -6.343339e-01 -3.081682e-01 -2.311972e-01
```

```
1.000000
8%    -6.768952e-01 -6.038855e-01 -2.698054e-01 -2.000641e-01
1.000000
9%    -6.405033e-01 -5.739041e-01 -2.378368e-01 -1.750250e-01
1.000000
10%   -6.061010e-01 -5.480343e-01 -2.114656e-01 -1.556155e-01
1.000000
20%   -3.917190e-01 -3.895481e-01 -9.111428e-02 -6.544115e-02
3.570000
30%   -2.583651e-01 -2.744373e-01 -5.423468e-02 -4.026708e-02
8.910000
40%   -1.288661e-01 -1.663338e-01 -2.436683e-02 -1.195983e-02
13.000000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02
22.000000
60%    1.586211e-01  8.611185e-02  2.922621e-02  2.692801e-02
37.000000
70%    2.849178e-01  1.751137e-01  6.315093e-02  5.350104e-02
59.800000
80%    4.175428e-01  3.602736e-01  1.352785e-01  1.013742e-01
100.000000
90%    6.009027e-01  6.889469e-01  2.653679e-01  1.799362e-01
203.000000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01
25691.160000

               Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
1%          0.000000
2%          0.000000
3%          0.000000
4%          0.000000
5%          0.000000
6%          0.000000
7%          0.000000
8%          0.000000
9%          0.000000
10%         0.000000
20%         0.000000
30%         0.000000
40%         0.000000
50%         0.000000
60%         0.000000
70%         0.000000
80%         0.000000
90%         0.000000
max         1.000000
```

```
[23 rows x 31 columns]
------------
----------------------------------------------------------------------
-------------
Checking for null
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
there is no null or missing data in the selected sourse
```

4.Checking for Null data and duplicates and removing them

```
print(df.isnull().any())

Time      False
V1        False
V2        False
V3        False
V4        False
V5        False
V6        False
```

```
V7        False
V8        False
V9        False
V10       False
V11       False
V12       False
V13       False
V14       False
V15       False
V16       False
V17       False
V18       False
V19       False
V20       False
V21       False
V22       False
V23       False
V24       False
V25       False
V26       False
V27       False
V28       False
Amount    False
Class     False
dtype: bool
```

```
print(df.duplicated().any())
print(df[df.duplicated()])
```

```
True
            Time        V1        V2        V3        V4        V5
V6  \
33          26.0 -0.529912  0.873892  1.347247  0.145457  0.414209
0.100223
35          26.0 -0.535388  0.865268  1.351076  0.147575  0.433680
0.086983
113         74.0  1.038370  0.127486  0.184456  1.109950  0.441699
0.945283
114         74.0  1.038370  0.127486  0.184456  1.109950  0.441699
0.945283
115         74.0  1.038370  0.127486  0.184456  1.109950  0.441699
0.945283
...          ...       ...       ...       ...       ...       ...
...
282987  171288.0  1.912550 -0.455240 -1.750654  0.454324  2.089130
4.160019
283483  171627.0 -1.464380  1.368119  0.815992 -0.601282 -0.689115 -
0.487154
283485  171627.0 -1.457978  1.378203  0.811515 -0.603760 -0.711883 -
0.471672
284191  172233.0 -2.667936  3.160505 -3.355984  1.007845 -0.377397 -
```

```
0.109730
284193  172233.0 -2.691642  3.123168 -3.339407  1.017018 -0.293095 -
0.167054

              V7        V8        V9  ...       V21       V22
V23  \
33      0.711206  0.176066 -0.286717  ...  0.046949  0.208105 -
0.185548
35      0.693039  0.179742 -0.285642  ...  0.049526  0.206537 -
0.187108
113    -0.036715  0.350995  0.118950  ...  0.102520  0.605089
0.023092
114    -0.036715  0.350995  0.118950  ...  0.102520  0.605089
0.023092
115    -0.036715  0.350995  0.118950  ...  0.102520  0.605089
0.023092
...          ...       ...       ...  ...       ...       ...       ..
.
282987 -0.881302  1.081750  1.022928  ... -0.524067 -1.337510
0.473943
283483 -0.303778  0.884953  0.054065  ...  0.287217  0.947825 -
0.218773
283485 -0.282535  0.880654  0.052808  ...  0.284205  0.949659 -
0.216949
284191 -0.667233  2.309700 -1.639306  ...  0.391483  0.266536 -
0.079853
284193 -0.745886  2.325616 -1.634651  ...  0.402639  0.259746 -
0.086606

             V24       V25       V26       V27       V28  Amount
Class
33      0.001031  0.098816 -0.552904 -0.073288  0.023307    6.14
0
35      0.000753  0.098117 -0.553471 -0.078306  0.025427    1.77
0
113    -0.626463  0.479120 -0.166937  0.081247  0.001192    1.18
0
114    -0.626463  0.479120 -0.166937  0.081247  0.001192    1.18
0
115    -0.626463  0.479120 -0.166937  0.081247  0.001192    1.18
0
...          ...       ...       ...       ...       ...     ...     ..
.
282987  0.616683 -0.283548 -1.084843  0.073133 -0.036020   11.99
0
283483  0.082926  0.044127  0.639270  0.213565  0.119251    6.82
0
283485  0.083250  0.044944  0.639933  0.219432  0.116772   11.93
0
284191 -0.096395  0.086719 -0.451128 -1.183743 -0.222200   55.66
```

```
0
284193 -0.097597  0.083693 -0.453584 -1.205466 -0.213020   36.74
0

[1081 rows x 31 columns]
```

There are some duplicates present in this Dataset

4.1 In this we have 1081 duplicate data so we are removing it

```
print(f"Duplicated {len(df[df.duplicated()])} rows on idx:
{list(df[df.duplicated()].index)}")
```

```
Duplicated 1081 rows on idx: [33, 35, 113, 114, 115, 221, 223, 1178,
1180, 1382, 1384, 1684, 1686, 2004, 2005, 2006, 2728, 2729, 2731,
2732, 2734, 2735, 2784, 2786, 2998, 3000, 3175, 3177, 3316, 3318,
3321, 3323, 4900, 4902, 5925, 5927, 6411, 6412, 6413, 9027, 9028,
9029, 11132, 11134, 12393, 12394, 12395, 13563, 13564, 13565, 13882,
13883, 13884, 16391, 16393, 17949, 17950, 17951, 18051, 18052, 18053,
18263, 18265, 19617, 19619, 19636, 19638, 19797, 19799, 20418, 20420,
21252, 21254, 21403, 21405, 21676, 21677, 21678, 21683, 21684, 21685,
21966, 21967, 21968, 22476, 22478, 22789, 22791, 23891, 23892, 23893,
24965, 24967, 24992, 24994, 25460, 25461, 25462, 25692, 25694, 26943,
26944, 26946, 26947, 26949, 26950, 27402, 27404, 27776, 27777, 27778,
28543, 28544, 28546, 28547, 28549, 28550, 28644, 28646, 29595, 29597,
30136, 30138, 30671, 30673, 30680, 30682, 31637, 31639, 31848, 31850,
31854, 31856, 31983, 31985, 32955, 32957, 34893, 34929, 34931, 35905,
35907, 36011, 36013, 36209, 36210, 36211, 37309, 37311, 37448, 37450,
37563, 37565, 37742, 37744, 37832, 37834, 38154, 38156, 38433, 38435,
38455, 38457, 38536, 38538, 39370, 39372, 39587, 39588, 39589, 39591,
39592, 39593, 39595, 39596, 39597, 39599, 39600, 39601, 40779, 40781,
43541, 43543, 44393, 44394, 44396, 44397, 44399, 44400, 44927, 44929,
44934, 44936, 45349, 45350, 45351, 45688, 45690, 46011, 46013, 47260,
47261, 47262, 47263, 47264, 47265, 47266, 47267, 47881, 47883, 48272,
48274, 48555, 48557, 48568, 48570, 49029, 49031, 49952, 49954, 50829,
50831, 50919, 50921, 50928, 50930, 51750, 51752, 52021, 52023, 52482,
52484, 53427, 53429, 54312, 54313, 54314, 56056, 56058, 58290, 58291,
58293, 58294, 58296, 58297, 59293, 59295, 60912, 60914, 61671, 61673,
62262, 62264, 62299, 62301, 62356, 62358, 63207, 63208, 63209, 66601,
66603, 68899, 68901, 69431, 69433, 70025, 70027, 70032, 70034, 70713,
70715, 70719, 70721, 70928, 70930, 71548, 71549, 71550, 72322, 72323,
72324, 73495, 73497, 73661, 73663, 74126, 74128, 74683, 74684, 74685,
74903, 74905, 75172, 75174, 75183, 75185, 75852, 75854, 76204, 76206,
76690, 76692, 79109, 79111, 82319, 82321, 82891, 82893, 84608, 84609,
84610, 84833, 84835, 85425, 85481, 85483, 86362, 86364, 89103, 89105,
92203, 92204, 92206, 92207, 92209, 92210, 94543, 94545, 94576, 94577,
94578, 94579, 94581, 94582, 94583, 94584, 94586, 94587, 94588, 94589,
94591, 94592, 94593, 94594, 94596, 94597, 94598, 94599, 94765, 94767,
95129, 95131, 95135, 95136, 95138, 95139, 95141, 95142, 95719, 95721,
95736, 95737, 95739, 95740, 95742, 95743, 95878, 95880, 97449, 97451,
97610, 97612, 98721, 98723, 98774, 98776, 99590, 99592, 99853, 99855,
```

102442, 102443, 102444, 102445, 102446, 103303, 103305, 103736,
103737, 103738, 103739, 103741, 103742, 103743, 103744, 103746,
103747, 103748, 103749, 103751, 103752, 103753, 103754, 103756,
103757, 103758, 103759, 103793, 103795, 104108, 104110, 104218,
104219, 104221, 104222, 104224, 104225, 104748, 104750, 106837,
107447, 107449, 107802, 107804, 107995, 107996, 107997, 108918,
109648, 109899, 109900, 109901, 110527, 110529, 111819, 111821,
111902, 111904, 112324, 112325, 112326, 112409, 112410, 112412,
112413, 112415, 112416, 113068, 113069, 113070, 113386, 113388,
113781, 113783, 114956, 114958, 115158, 115160, 115313, 115315,
115851, 115853, 117323, 117325, 117491, 117493, 118787, 118789,
121328, 121330, 122628, 122630, 123664, 123666, 124584, 124586,
125311, 125313, 126239, 126241, 127031, 127033, 127175, 127177,
127432, 127434, 128630, 128632, 129756, 129758, 130915, 130917,
132002, 132004, 133196, 133198, 133682, 133684, 133854, 133855,
133857, 133858, 133860, 133861, 133866, 133868, 136201, 136203,
137259, 137261, 137338, 137340, 138441, 138442, 138444, 138445,
138447, 138448, 138462, 138464, 138701, 138703, 139104, 139106,
139635, 139637, 140496, 140498, 140841, 140843, 141038, 141039,
141040, 141258, 141260, 142288, 142290, 142327, 142328, 142329,
143334, 143336, 143390, 143392, 143707, 143709, 144734, 144735,
144736, 144858, 144860, 145569, 145571, 145813, 145815, 146782,
146784, 146890, 146959, 146961, 147622, 147623, 147624, 148322,
148324, 148743, 148745, 149852, 149854, 149865, 149867, 150212,
150214, 150619, 150621, 150661, 150663, 150667, 150669, 150678,
150680, 150808, 150809, 150810, 150882, 150883, 150884, 151007,
151008, 151009, 153537, 153539, 154233, 154712, 154714, 154821,
154823, 157312, 157313, 157315, 157316, 157318, 157319, 157331,
157333, 157413, 158353, 159445, 159447, 159811, 159813, 160059,
160192, 162372, 162374, 162923, 162924, 162925, 163841, 163843,
165272, 165273, 165274, 165588, 165589, 165590, 166740, 166742,
167704, 167706, 168431, 168433, 169338, 169340, 169342, 169344,
170002, 170003, 170004, 171582, 171584, 171993, 171995, 172808,
172810, 172906, 172911, 172913, 172954, 172956, 172979, 172981,
173236, 173238, 173812, 173814, 174038, 174040, 174242, 174244,
174801, 174803, 175769, 175771, 176211, 176213, 177175, 177177,
180220, 180222, 185547, 185549, 187391, 187393, 187587, 187588,
187589, 187999, 188001, 189177, 189179, 190125, 190127, 190499,
190500, 190501, 190503, 190504, 190505, 190507, 190508, 190509,
190511, 190512, 190513, 190866, 190867, 190868, 191829, 191831,
192168, 192170, 192311, 192313, 195346, 195348, 196086, 196088,
197877, 197879, 197913, 198893, 198895, 199075, 199077, 200921,
200923, 201632, 201634, 201771, 201773, 202259, 202260, 202261,
203433, 203435, 203444, 203446, 204165, 204166, 204167, 205176,
205178, 206698, 206700, 206857, 206859, 209631, 209633, 209660,
209661, 209662, 210268, 210270, 210533, 210535, 211001, 211003,
212916, 212918, 213168, 213170, 214083, 214085, 214363, 214365,
216005, 216007, 216105, 216106, 216107, 216109, 216110, 216111,
216113, 216114, 216115, 216117, 216118, 216119, 217923, 217925,
218687, 218689, 219245, 219246, 219247, 219456, 219457, 219459,

```
219460, 219462, 219463, 220554, 220556, 221276, 221277, 221278,
221319, 221321, 221474, 221475, 221477, 221478, 221480, 221481,
221487, 221489, 222021, 222022, 222024, 222025, 222027, 222028,
222038, 222040, 222687, 222688, 222689, 222691, 222692, 222693,
222695, 222696, 222697, 222699, 222700, 222701, 222705, 222707,
223041, 223042, 223044, 223045, 223047, 223048, 224612, 224614,
224983, 224984, 224985, 225322, 225324, 225750, 225751, 225752,
226022, 226024, 226611, 228339, 228340, 228341, 228395, 228397,
229342, 229344, 229749, 229751, 230268, 230270, 230415, 230417,
231112, 231114, 231511, 231513, 231972, 231973, 231974, 232538,
232540, 233069, 233071, 233129, 233131, 233166, 233168, 233377,
233379, 234633, 234676, 234677, 234678, 236964, 236966, 236986,
236988, 237648, 237649, 237651, 237652, 237654, 237655, 239722,
239724, 239887, 239889, 240453, 240454, 240456, 240457, 240459,
240460, 240462, 240464, 241678, 241680, 242415, 242417, 242459,
242461, 243482, 243484, 244188, 244189, 244190, 244574, 244575,
244576, 245416, 245925, 245927, 247037, 247039, 248779, 248781,
249280, 249282, 250553, 250555, 250746, 250748, 252532, 252534,
252539, 252540, 252541, 252938, 252940, 252950, 252951, 252953,
252954, 252956, 252957, 252965, 252967, 253040, 253042, 254804,
254805, 254806, 255212, 255213, 255214, 255405, 255406, 255407,
256675, 256677, 257104, 257106, 259335, 259337, 259362, 259364,
259580, 261153, 261154, 261155, 261908, 261910, 262982, 262984,
263489, 263491, 263543, 263544, 263545, 266289, 266290, 266291,
267422, 267423, 267424, 268291, 268292, 268293, 268294, 268295,
268297, 268298, 268299, 268300, 268301, 268302, 268303, 268304,
268305, 268306, 268307, 268308, 268309, 268310, 268311, 268312,
268313, 268314, 268315, 268316, 268317, 268318, 268319, 268320,
268321, 268322, 268323, 268324, 268325, 268697, 268698, 268699,
269519, 269521, 269924, 269926, 271331, 271333, 272266, 272268,
272634, 272635, 272636, 274429, 274431, 275762, 275764, 275962,
275963, 275964, 276129, 276131, 276508, 276510, 276779, 276780,
276781, 276991, 276992, 276994, 276995, 276997, 276998, 278174,
278176, 278671, 278673, 279188, 279190, 279884, 279885, 279886,
280183, 280185, 280926, 280927, 280929, 280930, 280932, 280933,
281132, 281134, 281815, 281817, 282040, 282042, 282206, 282207,
282208, 282209, 282210, 282211, 282212, 282213, 282985, 282987,
283483, 283485, 284191, 284193]
```

```
df = df.drop_duplicates(keep='first')
df.reset_index(drop = True, inplace = True)
```

```
print(f"Rows, Cols - After: {df.shape[0], df.shape[1]}")
```

```
Rows, Cols - After: (283726, 31)
```

Outliers treatment

As the whole dataset is transformed with PCA, so assuming that the outliers are already treated. Hence, we are not performing any outliers treatment on the dataframe, though we still see outliers available

Visualizeing and Analysing the Data

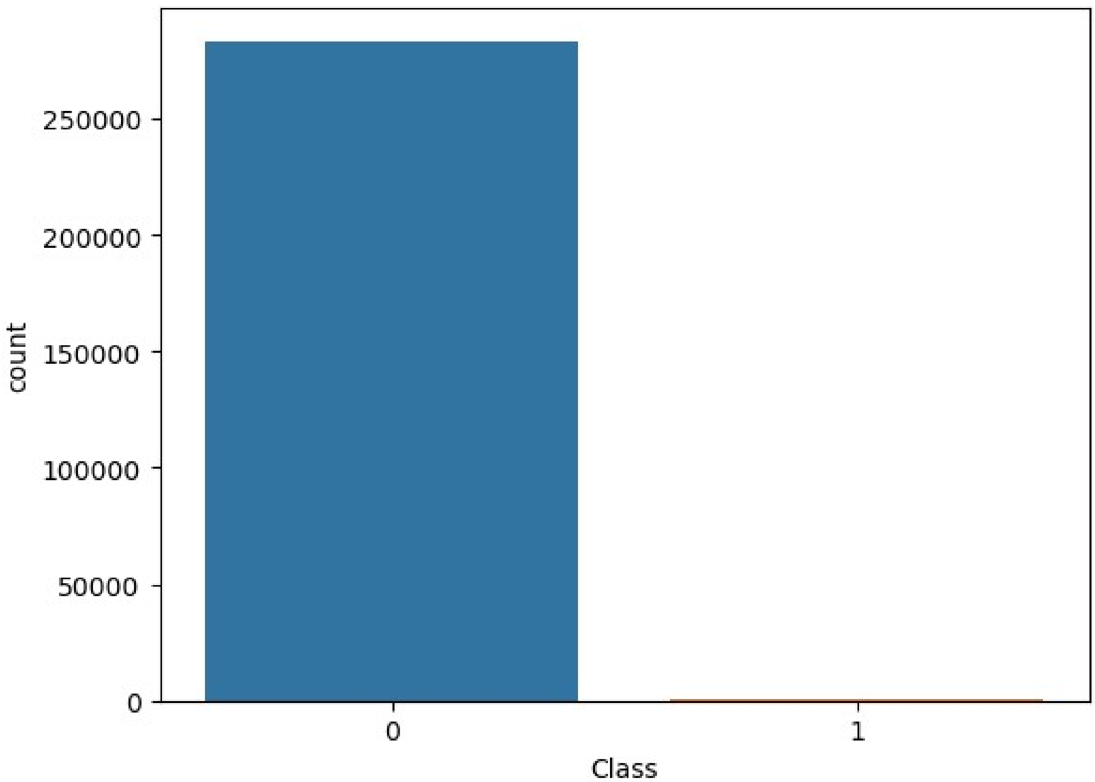
1.Fraudulent and Non Fraudulent activitys

```
classes=df['Class'].value_counts()
normal_share=round(classes[0]/df['Class'].count()*100,2)
fraud_share=round(classes[1]/df['Class'].count()*100, 2)
print("Non-Fraudulent : {} %".format(normal_share))
print("    Fraudulent : {} %".format(fraud_share))
```

```
Non-Fraudulent : 99.83 %
    Fraudulent : 0.17 %
```

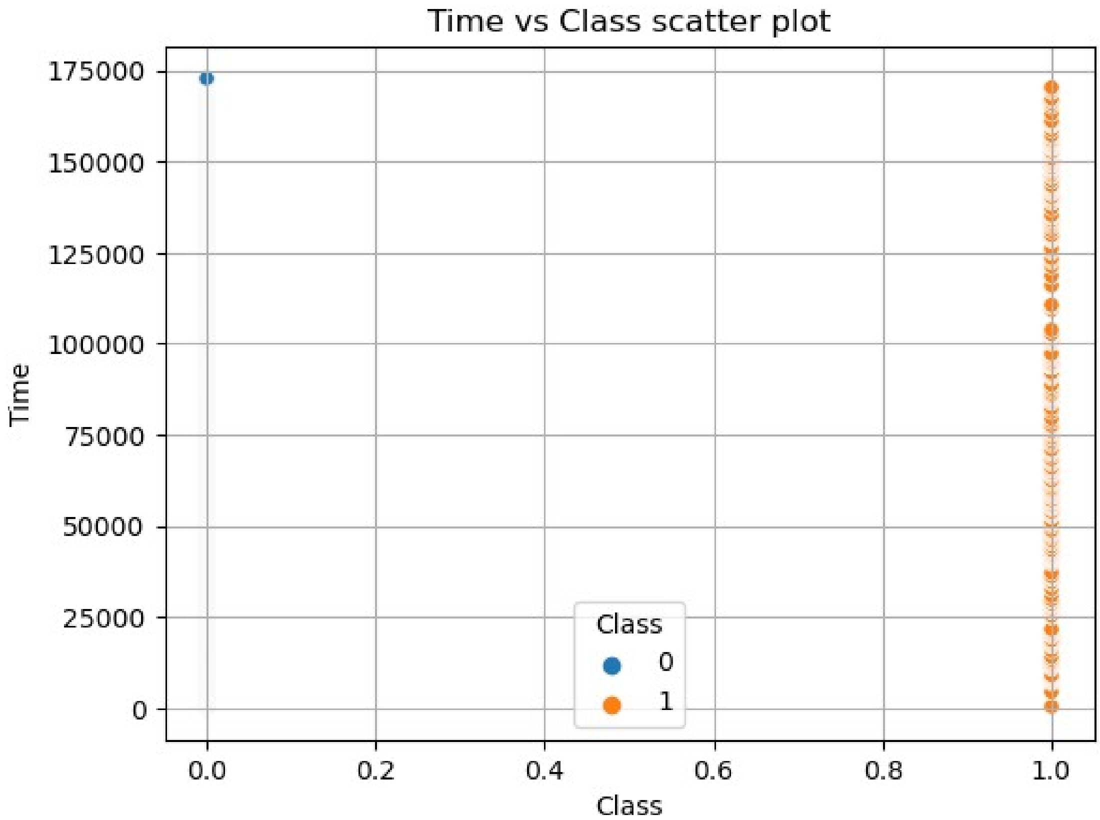
2.Visualizeing Count of Fraudulent and Non Fraudulent activitys preasent in transaction

```
sns.countplot(data=df,x="Class")
plt.show()
```



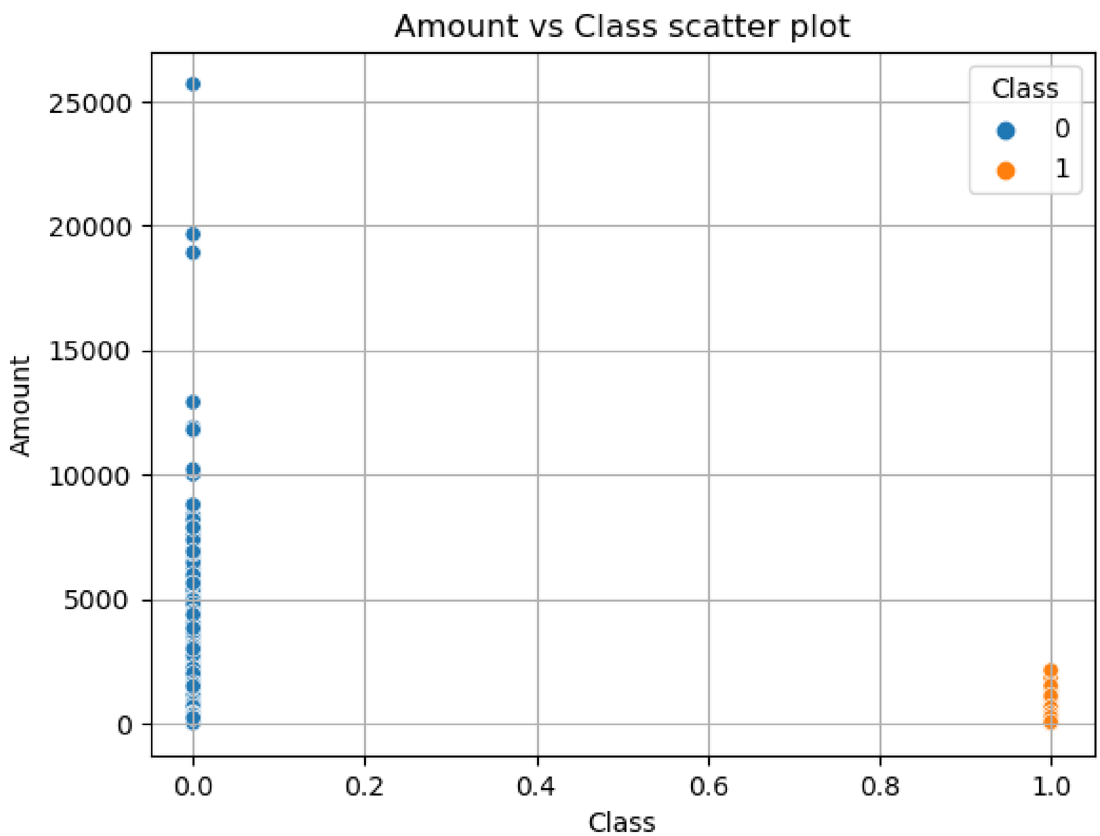
3.Created a scatter plot to observe the distribution of classes with time

```
sns.scatterplot( df["Class"],df["Time"],hue=df["Class"])
plt.title("Time vs Class scatter plot")
plt.grid()
plt.show()
```



4.Create a scatter plot to observe the distribution of classes with Amount

```
sns.scatterplot(df["Class"],df["Amount"],hue=df["Class"])
plt.title("Amount vs Class scatter plot")
plt.grid()
```



Observation

Clearly low amount transactions are more likely to be fraudulent than high amount transaction

```
corr=df.corr()
print(corr)
```

```
             Time        V1        V2        V3        V4        V5
V6  \
Time     1.000000  0.117927 -0.010556 -0.422054 -0.105845  0.173223 -
0.063279
V1       0.117927  1.000000  0.006875 -0.008112  0.002257 -0.007036
0.000413
V2      -0.010556  0.006875  1.000000  0.005278 -0.001495  0.005210 -
0.000594
V3      -0.422054 -0.008112  0.005278  1.000000  0.002829 -0.006879 -
0.001511
V4      -0.105845  0.002257 -0.001495  0.002829  1.000000  0.001744 -
0.000880
V5       0.173223 -0.007036  0.005210 -0.006879  0.001744  1.000000 -
0.000938
V6      -0.063279  0.000413 -0.000594 -0.001511 -0.000880 -0.000938
1.000000
V7       0.085335 -0.009173  0.007425 -0.011721  0.004657 -0.008709
```

```
0.000436
V8     -0.038203 -0.001168  0.002899 -0.001815  0.000890  0.001430  
0.003036
V9     -0.007861  0.001828 -0.000274 -0.003579  0.002154 -0.001213 -
0.000734
V10     0.031068  0.000815  0.000620 -0.009632  0.002753 -0.006050 -
0.002180
V11    -0.248536  0.001028 -0.000633  0.002339 -0.001223  0.000411 -
0.000211
V12     0.125500 -0.001524  0.002266 -0.005900  0.003366 -0.002342 -
0.001185
V13    -0.065958 -0.000568  0.000680  0.000113  0.000177  0.000019  
0.000397
V14    -0.100316 -0.002663  0.002711 -0.003027  0.002801 -0.001000  
0.000184
V15    -0.184392 -0.000602  0.001538 -0.001230  0.000572 -0.001171 -
0.000470
V16     0.011286 -0.003345  0.004013 -0.004430  0.003346 -0.002373  
0.000122
V17    -0.073819 -0.003491  0.003244 -0.008159  0.003655 -0.004466 -
0.001716
V18     0.090305 -0.003535  0.002477 -0.003495  0.002325 -0.002685  
0.000541
V19     0.029537  0.000919 -0.000358 -0.000016 -0.000560  0.000436  
0.000106
V20    -0.051022 -0.001393 -0.001287 -0.002269  0.000318 -0.001185 -
0.000181
V21     0.045913  0.002818 -0.004897  0.003500 -0.001034  0.001622 -
0.002134
V22     0.143727 -0.001436  0.001237 -0.000275  0.000115 -0.000559  
0.001104
V23     0.051474 -0.001330 -0.003855  0.000449  0.000732  0.001183 -
0.000755
V24    -0.015954 -0.000723  0.000701 -0.000072 -0.000120  0.000198  
0.001202
V25    -0.233262 -0.000222 -0.001569  0.000425  0.000162  0.000069  
0.000697
V26    -0.041818 -0.000684  0.000253 -0.000094  0.000777  0.000390 -
0.000028
V27    -0.005171 -0.015706  0.007555 -0.007051  0.001322 -0.005798  
0.000289
V28    -0.009305 -0.004861  0.001611 -0.000134  0.000231 -0.000820  
0.000925
Amount -0.010559 -0.230105 -0.533428 -0.212410  0.099514 -0.387685  
0.216389
Class  -0.012359 -0.094486  0.084624 -0.182322  0.129326 -0.087812 -
0.043915

              V7        V8        V9  ...       V21       V22       
V23  \
```

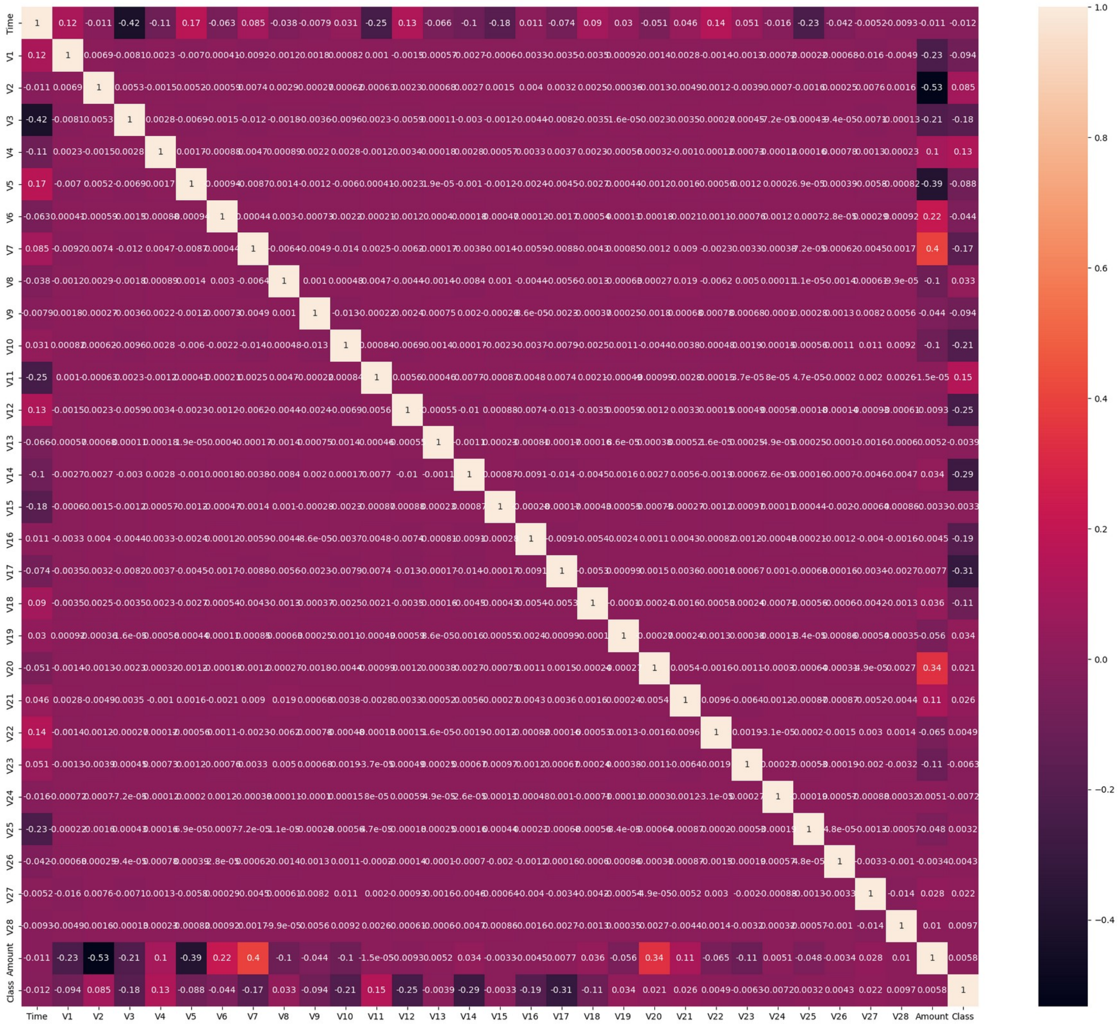
```
Time     0.085335 -0.038203 -0.007861  ...  0.045913  0.143727
0.051474
V1      -0.009173 -0.001168  0.001828  ...  0.002818 -0.001436 -
0.001330
V2       0.007425  0.002899 -0.000274  ... -0.004897  0.001237 -
0.003855
V3      -0.011721 -0.001815 -0.003579  ...  0.003500 -0.000275
0.000449
V4       0.004657  0.000890  0.002154  ... -0.001034  0.000115
0.000732
V5      -0.008709  0.001430 -0.001213  ...  0.001622 -0.000559
0.001183
V6       0.000436  0.003036 -0.000734  ... -0.002134  0.001104 -
0.000755
V7       1.000000 -0.006419 -0.004921  ...  0.009010 -0.002280
0.003303
V8      -0.006419  1.000000  0.001038  ...  0.018892 -0.006156
0.004994
V9      -0.004921  0.001038  1.000000  ...  0.000679  0.000785
0.000677
V10     -0.013617  0.000481 -0.012613  ...  0.003777 -0.000481
0.001917
V11      0.002454  0.004688 -0.000217  ... -0.002760 -0.000150 -
0.000037
V12     -0.006153 -0.004414 -0.002385  ...  0.003285  0.000151
0.000486
V13     -0.000170 -0.001381  0.000745  ...  0.000522  0.000016
0.000252
V14     -0.003816 -0.008387  0.001981  ...  0.005633 -0.001906
0.000666
V15     -0.001394  0.001044 -0.000283  ... -0.000271 -0.001197
0.000969
V16     -0.005944 -0.004376 -0.000086  ...  0.004326 -0.000820
0.001209
V17     -0.008794 -0.005576 -0.002318  ...  0.003560 -0.000162
0.000667
V18     -0.004279 -0.001323 -0.000373  ...  0.001629 -0.000533
0.000240
V19      0.000846 -0.000626  0.000247  ...  0.000244  0.001342
0.000381
V20     -0.001192  0.000271 -0.001838  ...  0.005372 -0.001617 -
0.001094
V21      0.009010  0.018892  0.000679  ...  1.000000  0.009645 -
0.006391
V22     -0.002280 -0.006156  0.000785  ...  0.009645  1.000000
0.001929
V23      0.003303  0.004994  0.000677  ... -0.006391  0.001929
1.000000
V24     -0.000384  0.000113 -0.000103  ...  0.001210 -0.000031
0.000273
```

```
V25    -0.000072  0.000011 -0.000275  ... -0.000872  0.000197 -
0.000532
V26     0.000624 -0.001407  0.001253  ... -0.000874 -0.001495 -
0.000185
V27    -0.004537  0.000613  0.008221  ... -0.005216  0.003037 -
0.002028
V28     0.001657 -0.000099  0.005591  ... -0.004436  0.001392 -
0.003224
Amount  0.400408 -0.104662 -0.044123  ...  0.108058 -0.064965 -
0.112833
Class  -0.172347  0.033068 -0.094021  ...  0.026357  0.004887 -
0.006333

             V24       V25       V26       V27       V28    Amount  
Class
Time   -0.015954 -0.233262 -0.041818 -0.005171 -0.009305 -0.010559 -
0.012359
V1     -0.000723 -0.000222 -0.000684 -0.015706 -0.004861 -0.230105 -
0.094486
V2      0.000701 -0.001569  0.000253  0.007555  0.001611 -0.533428  
0.084624
V3     -0.000072  0.000425 -0.000094 -0.007051 -0.000134 -0.212410 -
0.182322
V4     -0.000120  0.000162  0.000777  0.001322  0.000231  0.099514  
0.129326
V5      0.000198  0.000069  0.000390 -0.005798 -0.000820 -0.387685 -
0.087812
V6      0.001202  0.000697 -0.000028  0.000289  0.000925  0.216389 -
0.043915
V7     -0.000384 -0.000072  0.000624 -0.004537  0.001657  0.400408 -
0.172347
V8      0.000113  0.000011 -0.001407  0.000613 -0.000099 -0.104662  
0.033068
V9     -0.000103 -0.000275  0.001253  0.008221  0.005591 -0.044123 -
0.094021
V10     0.000154 -0.000565  0.001089  0.010769  0.009159 -0.102255 -
0.206971
V11     0.000080  0.000047 -0.000204  0.001987  0.002562 -0.000015  
0.149067
V12     0.000588 -0.000181 -0.000138 -0.000929 -0.000613 -0.009254 -
0.250711
V13    -0.000049  0.000248 -0.000101 -0.001577 -0.000604  0.005209 -
0.003897
V14    -0.000026  0.000155 -0.000702 -0.004556 -0.004664  0.034122 -
0.293375
V15     0.000113  0.000445 -0.002034 -0.000641  0.000858 -0.003265 -
0.003300
V16    -0.000482  0.000215 -0.001245 -0.003974 -0.001629 -0.004488 -
0.187186
V17     0.001006 -0.000685  0.000157 -0.003421 -0.002703  0.007730 -
```

```
0.313498
V18    -0.000710 -0.000559 -0.000596 -0.004231 -0.001256  0.035775 -
0.105340
V19    -0.000112 -0.000084  0.000856 -0.000544  0.000353 -0.055994
0.033631
V20    -0.000303 -0.000643 -0.000310 -0.000049  0.002671  0.340729
0.021486
V21     0.001210 -0.000872 -0.000874 -0.005216 -0.004436  0.108058
0.026357
V22    -0.000031  0.000197 -0.001495  0.003037  0.001392 -0.064965
0.004887
V23     0.000273 -0.000532 -0.000185 -0.002028 -0.003224 -0.112833 -
0.006333
V24     1.000000 -0.000188  0.000568 -0.000885  0.000322  0.005055 -
0.007210
V25    -0.000188  1.000000  0.000048 -0.001339 -0.000565 -0.047596
0.003202
V26     0.000568  0.000048  1.000000 -0.003294 -0.000999 -0.003425
0.004265
V27    -0.000885 -0.001339 -0.003294  1.000000 -0.013950  0.027922
0.021892
V28     0.000322 -0.000565 -0.000999 -0.013950  1.000000  0.010143
0.009682
Amount  0.005055 -0.047596 -0.003425  0.027922  0.010143  1.000000
0.005777
Class  -0.007210  0.003202  0.004265  0.021892  0.009682  0.005777
1.000000

[31 rows x 31 columns]
```

5.Plotting the correlation matrix

```
plt.figure(figsize=[25,21])
sns.heatmap(data=corr,annot=True)
plt.show()
```

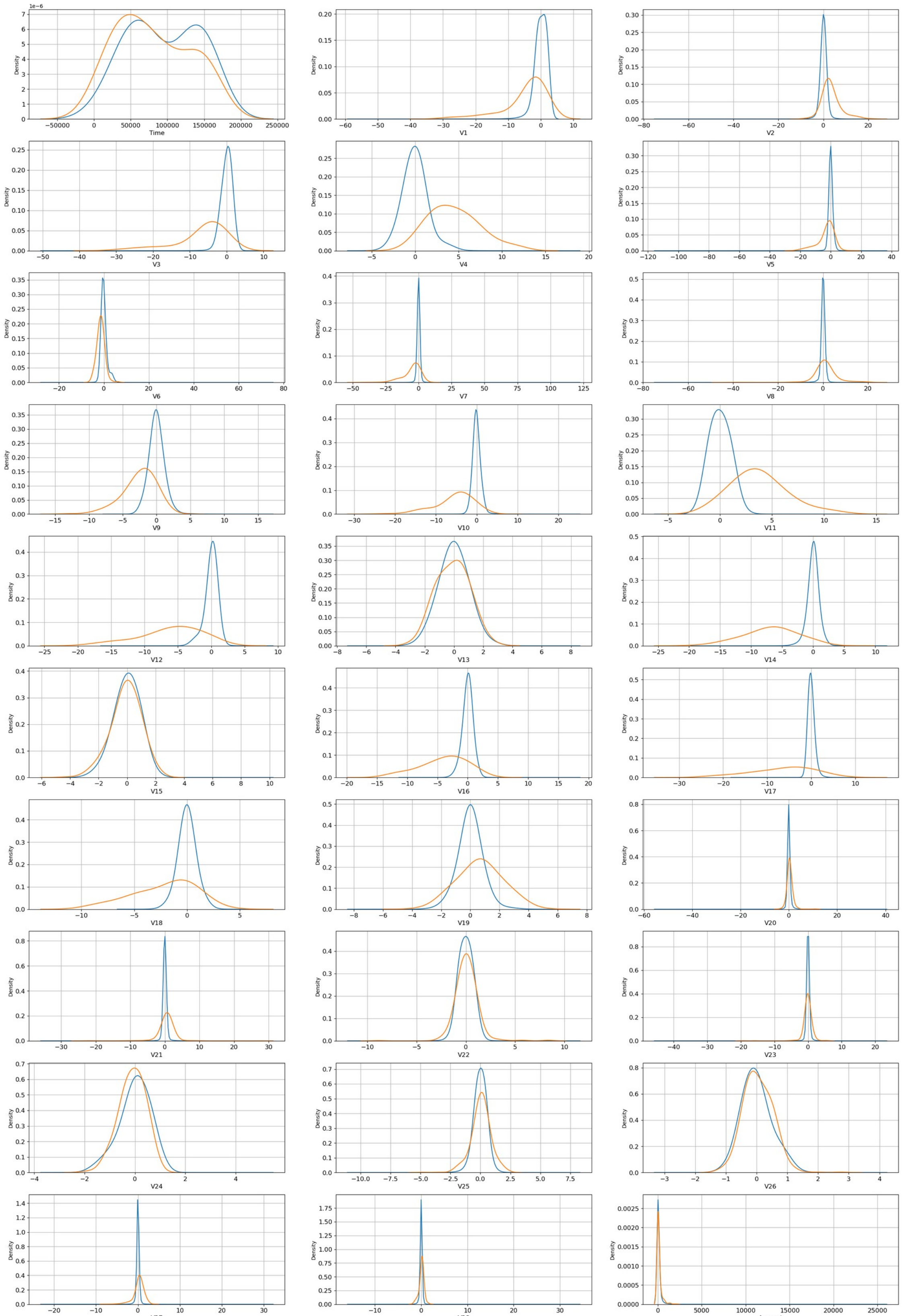


6.Plotting all the variable in displot to visualise the distribution

```
var = list(df.columns.values)
var.remove("Class") # dropping Class columns from the list

i = 0
t0 = df.loc[df['Class'] == 0]
t1 = df.loc[df['Class'] == 1]

plt.figure()
fig, ax = plt.subplots(10,3,figsize=(30,45));

for feature in var:
    i += 1
    plt.subplot(10,3,i)
    sns.kdeplot(t0[feature], bw=0.5,label="0")
    sns.kdeplot(t1[feature], bw=0.5,label="1")
    plt.xlabel(feature, fontsize=12)
    locs, labels = plt.xticks()
    plt.tick_params(axis='both', which='major', labelsize=12)
```

```
    plt.grid()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```


1e−6
Density
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

Observation

We can see most of the features distributions are overlapping for both the fraud and non-fraud transactions.

7. Droping Time column

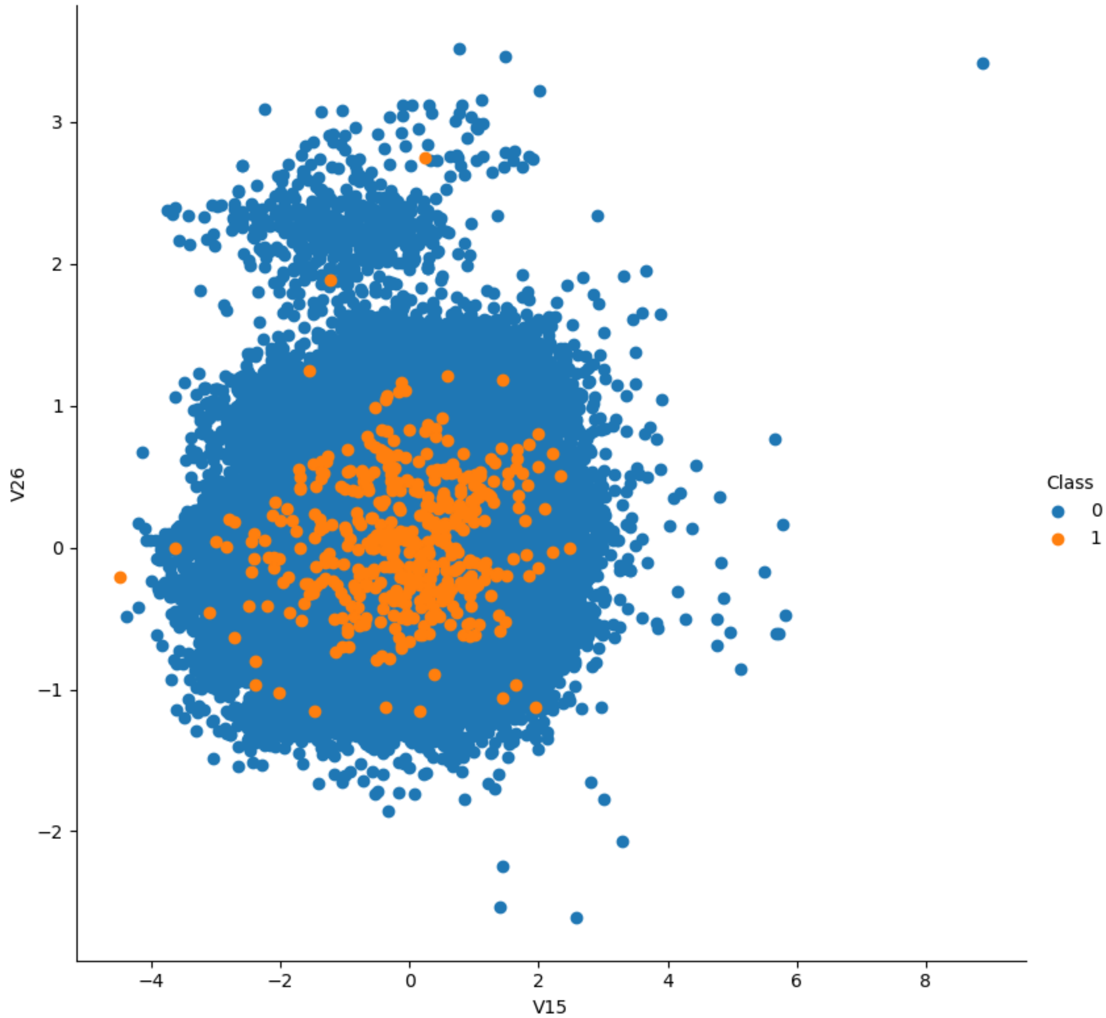
We are droping the “Time” column as it is irrelevant and this feature is not going to help in the model building becouse the time in not gona make fraud transaction only the activity makes it, get into the core of banking.

```
df = df.drop("Time", axis = 1)
print(df.head())
```

```
         V1        V2        V3        V4        V5        V6
V7  \
0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388
0.239599
1  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -
0.078803
2 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499
0.791461
3 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203
0.237609
4 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921
0.592941

         V8        V9       V10  ...       V21       V22       V23
V24  \
0  0.098698  0.363787  0.090794  ... -0.018307  0.277838 -0.110474
0.066928
1  0.085102 -0.255425 -0.166974  ... -0.225775 -0.638672  0.101288 -
0.339846
2  0.247676 -1.514654  0.207643  ...  0.247998  0.771679  0.909412 -
0.689281
3  0.377436 -1.387024 -0.054952  ... -0.108300  0.005274 -0.190321 -
1.175575
4 -0.270533  0.817739  0.753074  ... -0.009431  0.798278 -0.137458
0.141267

        V25       V26       V27       V28  Amount  Class
0  0.128539 -0.189115  0.133558 -0.021053  149.62      0
1  0.167170  0.125895 -0.008983  0.014724    2.69      0
2 -0.327642 -0.139097 -0.055353 -0.059752  378.66      0
3  0.647376 -0.221929  0.062723  0.061458  123.50      0
4 -0.206010  0.502292  0.219422  0.215153   69.99      0

[5 rows x 30 columns]
```

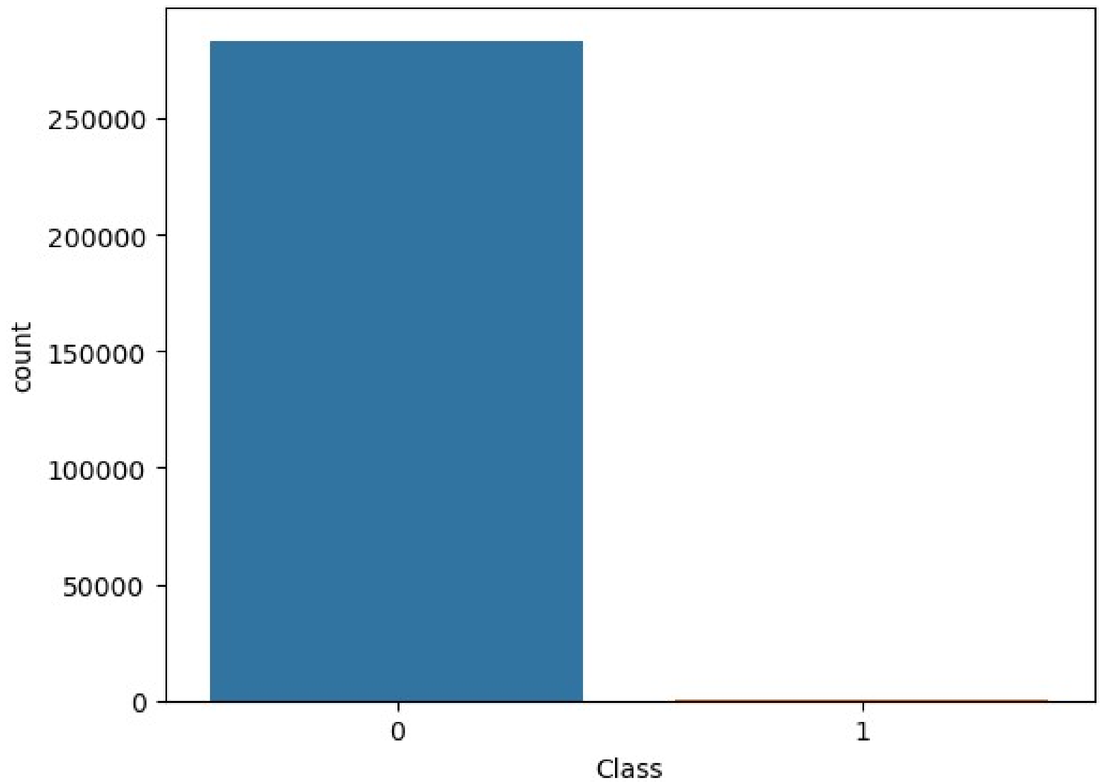
8.Visualizeing how the plots lay on V15 and V26 becouse we can see the relation on density of them in above graph

```
sns.FacetGrid(df,hue="Class",size=8).map(plt.scatter,"V15","V26").add_
legend();
plt.show()
```



9. Number of Fraudulent

```
sns.countplot(data=df,x="Class")
plt.show()
```




```
print(df["Class"].value_counts())
```

```
0    283253
1       473
Name: Class, dtype: int64
```

Note:-

Now we can see the data is highly Imbalanced

In Conclusion, everyone should know that the overall performance of ML models built on imbalanced datasets, will be constrained by its ability to predict rare and minority points. Identifying and resolving the imbalance of those points is crucial to the quality and performance of the generated models.

To handle that we are going to use SMOTE

Smote

SMOTE stands for Synthetic Minority Oversampling Technique. The method was proposed in a 2002 paper in the Journal of Artificial Intelligence Research. SMOTE is an improved method of dealing with imbalanced data in classification problems.

As an example, imagine a data set about sales of a new product for mountain sports. For simplicity, let's say that the website sells to two types of clients: skiers and climbers.

For each visitor, we also record whether the visitor buys the new mountain product. Imagine that we want to make a classification model that allows us to use customer data to make a prediction of whether the visitor will buy the new product.

Most e-commerce shoppers do not buy: often, many come for looking at products and only a small percentage of visitors actually buy something. Our data set will be imbalanced, because we have a huge number of non-buyers and a very small number of buyers.

10. Doing scalling to get better results

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_df = df
scaled_df = pd.DataFrame(scaler.fit_transform(scaled_df),
columns=scaled_df.columns)
```

11.Seperating Dependent and Indipendent Feature

```
x=scaled_df.drop("Class",axis=1)

y=scaled_df.loc[:,("Class")]

print(x)
```

```
                V1        V2        V3        V4        V5        V6
V7  \
0         0.935192  0.766490  0.881365  0.313023  0.763439  0.267669
0.266815
1         0.978542  0.770067  0.840298  0.271796  0.766120  0.262192
0.264875
2         0.935217  0.753118  0.868141  0.268766  0.762329  0.281122
0.270177
3         0.941878  0.765304  0.868484  0.213661  0.765647  0.275559
0.266803
4         0.938617  0.776520  0.864251  0.269796  0.762975  0.263984
0.268968
...            ...       ...       ...       ...       ...       ...
...
283721    0.756448  0.873531  0.666991  0.160317  0.729603  0.236810
0.235393
283722    0.945845  0.766677  0.872678  0.219189  0.771561  0.273661
0.265504
283723    0.990905  0.764080  0.781102  0.227202  0.783425  0.293496
0.263547
283724    0.954209  0.772856  0.849587  0.282508  0.763172  0.269291
0.261175
283725    0.949232  0.765256  0.849601  0.229488  0.765632  0.256488
0.274963

                V8        V9       V10  ...       V20       V21
V22  \
0         0.786444  0.475312  0.510600  ...  0.582942  0.561184
0.522992
```

```
1       0.786298  0.453981  0.505267  ...  0.579530  0.557840
0.480237
2       0.788042  0.410603  0.513018  ...  0.585855  0.565477
0.546030
3       0.789434  0.414999  0.507585  ...  0.578050  0.559734
0.510277
4       0.782484  0.490950  0.524303  ...  0.584615  0.561327
0.547271
...          ...       ...       ...  ...       ...       ...       ..
.
283721  0.863749  0.528729  0.598850  ...  0.595979  0.564920
0.515249
283722  0.788548  0.482925  0.488530  ...  0.580900  0.564933
0.553153
283723  0.792985  0.477677  0.498692  ...  0.580280  0.565220
0.537005
283724  0.792671  0.476287  0.500464  ...  0.581622  0.565755
0.547353
283725  0.780938  0.479528  0.489782  ...  0.584343  0.565688
0.540031

             V23       V24       V25       V26       V27       V28
Amount
0       0.663793  0.391253  0.585122  0.394557  0.418976  0.312697
0.005824
1       0.666938  0.336440  0.587290  0.446013  0.416345  0.313423
0.000105
2       0.678939  0.289354  0.559515  0.402727  0.415489  0.311911
0.014739
3       0.662607  0.223826  0.614245  0.389197  0.417669  0.314371
0.004807
4       0.663392  0.401270  0.566343  0.507497  0.420561  0.317490
0.002724
...          ...       ...       ...       ...       ...       ...
...
283721  0.680500  0.313600  0.658558  0.466291  0.433929  0.329840
0.000030
283722  0.665619  0.245298  0.543855  0.360884  0.417775  0.312038
0.000965
283723  0.664877  0.468492  0.592824  0.411177  0.416593  0.312585
0.002642
283724  0.663008  0.398836  0.545958  0.514746  0.418520  0.315245
0.000389
283725  0.671029  0.383420  0.551319  0.291786  0.416466  0.313401
0.008446

[283726 rows x 29 columns]
```

```
y=pd.DataFrame(y)
y.columns=["Class"]
print(y)
```

```
        Class
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0
...       ...
283721    0.0
283722    0.0
283723    0.0
283724    0.0
283725    0.0

[283726 rows x 1 columns]
```

```
print(y.value_counts())
```

```
Class
0.0      283253
1.0         473
dtype: int64
```

12.Implementing with SMOTE

```
#pip install imbalanced-learn
```

```
import imblearn
```

```
print(imblearn.__version__)
```

```
0.9.1
```

```
from imblearn import under_sampling, over_sampling
from imblearn.over_sampling import SMOTE
```

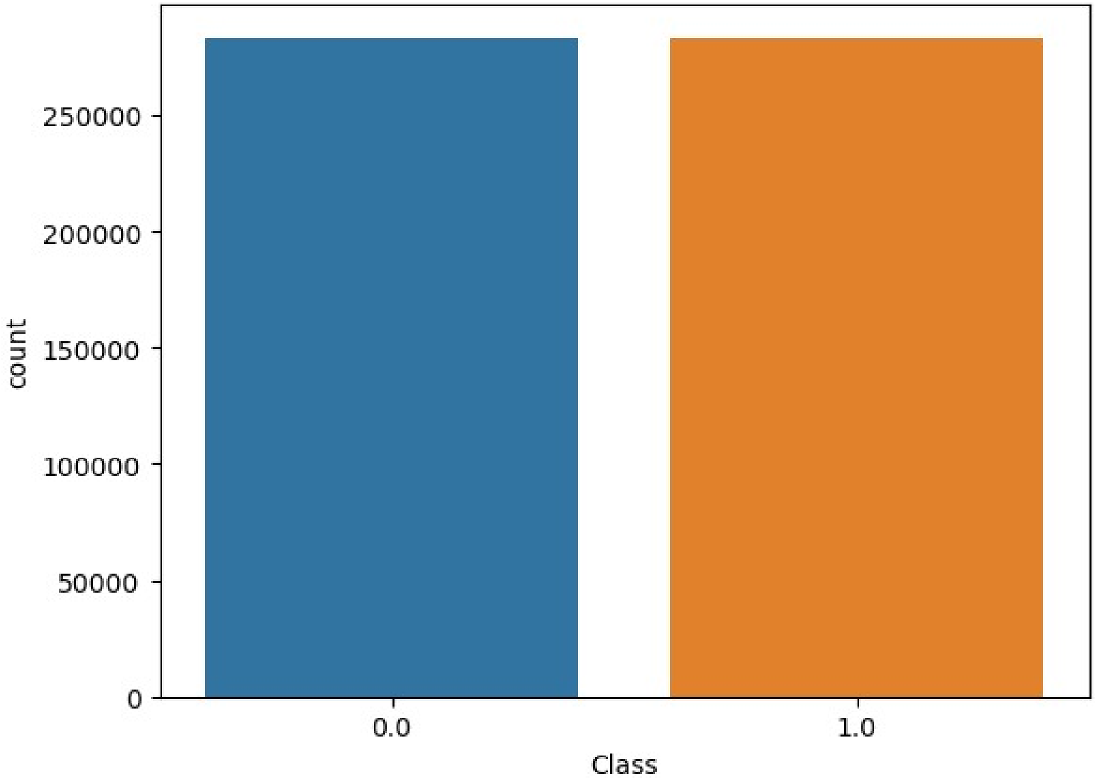
```
sm=SMOTE(random_state=14, sampling_strategy=1)
```

```
x_res,y_res=sm.fit_resample(x,y)
```

```
print("total columns and rows in x",x_res.shape)
print("total columns and rows in y",y_res.shape)
print("total value counts in our target
variable",y_res.value_counts())
```

```
total columns and rows in x (566506, 29)
total columns and rows in y (566506, 1)
total value counts in our target variable Class
0.0      283253
1.0      283253
dtype: int64
```

```
sns.countplot(data=y_res,x="Class")
plt.show()
```



Note

Now we can see our target variable it gets balanced

13.Spliting training and testing daa

```
x_train,x_test,y_train,y_test=train_test_split(x_res,y_res,test_size=0
.25,random_state=0)
```

```
y_test.value_counts()
```

```
Class
1.0      71091
0.0      70536
dtype: int64
```

Predection 1

1.Using Random forest

Random forest is used on the job by data scientists in many industries including banking, stock trading, medicine, and e-commerce. It's used to predict the things which help these industries run efficiently, such as customer activity, patient history, and safety.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import linear_model

cls = RandomForestClassifier(n_estimators=100, random_state=0)

cls.fit(x_train, y_train)

RandomForestClassifier(random_state=0)

pred = cls.predict(x_test)

print(pred)

[0. 1. 0. ... 0. 0. 1.]

pred=pd.DataFrame(pred)

print(pred)

          0
0       0.0
1       1.0
2       0.0
3       1.0
4       0.0
...     ...
141622  1.0
141623  0.0
141624  0.0
141625  0.0
141626  1.0

[141627 rows x 1 columns]

pred=pd.DataFrame(pred)
print(pred.head())
y_test=np.array(y_test)
y_test=pd.DataFrame(y_test)
print(y_test.head())

     0
0  0.0
1  1.0
2  0.0
3  1.0
4  0.0
     0
0  0.0
1  1.0
2  0.0
3  1.0
4  0.0
```

```
y_test.columns=["Actual"]
pred.columns=["Predection"]

ps=pd.DataFrame()
ps["pred"]=pred.Predection
ps["Actual"]=y_test.Actual
print(ps.head())
```

```
   pred  Actual
0   0.0     0.0
1   1.0     1.0
2   0.0     0.0
3   1.0     1.0
4   0.0     0.0
```

2.Analysing the predected Random forest values

```
print(ps["pred"].value_counts())
```

```
1.0    71116
0.0    70511
Name: pred, dtype: int64
```

```
print(ps["Actual"].value_counts())
```

```
1.0    71091
0.0    70536
Name: Actual, dtype: int64
```

3.Mean square error

```
Mean_Sq_Error1=((ps["pred"])-(ps["Actual"]))
Mean_Sq_Error2=((Mean_Sq_Error1)**2)
Mean_Sq_Error_cls=Mean_Sq_Error2.sum()
print("Mean_Sq_Error for random forest classifier
=",Mean_Sq_Error_cls)
```

```
Mean_Sq_Error for random forest classifier = 25.0
```

4.Model score

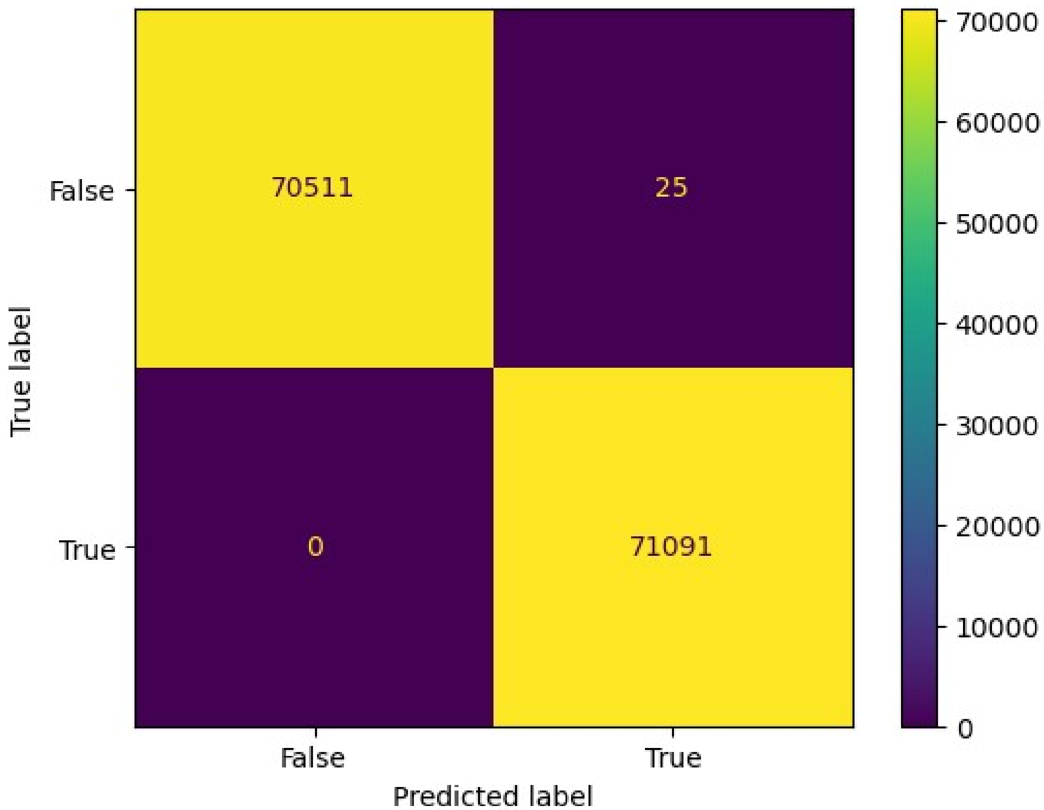
```
scoreOfModel1_cls = cls.score(x_test,y_test)
print(scoreOfModel1_cls)
```

```
0.9998234799861608
```

5.Confusion_Matrix

```
y_test_RFC=ps["Actual"]
y_pred_RFC=ps["pred"]

from sklearn import metrics
import matplotlib.pyplot as plt
def display_confusion_matrix(y_test_RFC,y_pred_RFC):
    matrix = metrics.confusion_matrix(y_test_RFC,y_pred_RFC)
    matrixDisplay = metrics.ConfusionMatrixDisplay(confusion_matrix =
```

```
matrix, display_labels = [False, True])
    matrixDisplay.plot()
    plt.show()
display_confusion_matrix(y_test_RFC,y_pred_RFC)
```




```
print(classification_report(y_test_RFC,y_pred_RFC))

              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     70536
         1.0       1.00      1.00      1.00     71091

    accuracy                           1.00    141627
   macro avg       1.00      1.00      1.00    141627
weighted avg       1.00      1.00      1.00    141627
```

6.Error Percentage

```
a=(ps.shape)
b=pd.DataFrame(a)
c=b.head(1)
c=np.array(c)
Error_Percentage_cls=(Mean_Sq_Error_cls/(np.array(c))*100)
print("Error occured in Randoem forest classifier
=",Error_Percentage_cls,"%")
```

```
Error occured in Randoem forest classifier = [[0.017652]] %
```

7.Result percentage

```
print("Result Percentage=",(100-Error_Percentage_cls),"%")
```

```
Result Percentage= [[99.982348]] %
```

8.Conclusion

Now we can see the results of RF model of 99.98234% accurate

Predection 2

1. Using XGBoost

The XGBoost (eXtreme Gradient Boosting) is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm that attempts to accurately predict a target variable by combining an ensemble of estimates from a set of simpler and weaker models.

```
from xgboost import XGBClassifier
xgb = XGBClassifier(n_estimators=100, random_state=0)
```

```
xgb.fit(x_train, y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1,
colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, feature_types=None, gamma=0, gpu_id=-
1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_bin=256, max_cat_threshold=64, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0,
min_child_weight=1,
              missing=nan, monotone_constraints='()',
n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto',
random_state=0, ...)
```

```
xgb_pred = xgb.predict(x_test)
```

```
print(xgb_pred)
```

```
[0 1 0 ... 0 0 1]
```

```
xgb_pred=pd.DataFrame(xgb_pred)
print(xgb_pred)
```

```
        0
0       0
```

```
1       1
2       0
3       1
4       0
...    ..
141622  1
141623  0
141624  0
141625  0
141626  1

[141627 rows x 1 columns]
```

```
xgb_pred=pd.DataFrame(xgb_pred)
print(xgb_pred.head())
y_test_xgb=np.array(y_test)
y_test_xgb=pd.DataFrame(y_test)
print(y_test_xgb.head())
```

```
   0
0  0
1  1
2  0
3  1
4  0
   Actual
0     0.0
1     1.0
2     0.0
3     1.0
4     0.0
```

```
y_test_xgb.columns=["Actual"]
xgb_pred.columns=["Predection"]

xgb_ps=pd.DataFrame()
xgb_ps["pred"]=xgb_pred.Predection

xgb_ps["Actual"]=y_test_xgb.Actual
print(xgb_ps.head())
```

```
   pred  Actual
0     0     0.0
1     1     1.0
2     0     0.0
3     1     1.0
4     0     0.0
```

2.Analysing the predected XG Boosting values

```
print(xgb_ps["pred"].value_counts())
```

```
1    71136
0    70491
Name: pred, dtype: int64
```

```
print(xgb_ps["Actual"].value_counts())
```

```
1.0    71091
0.0    70536
Name: Actual, dtype: int64
```

3.Mean square error

```
Mean_Sq_Error1=((xgb_ps["pred"])-(xgb_ps["Actual"]))
Mean_Sq_Error2=((Mean_Sq_Error1)**2)
Mean_Sq_Error_xgb=Mean_Sq_Error2.sum()
print("Mean_Sq_Error for random forest classifier
=",Mean_Sq_Error_xgb)
```

```
Mean_Sq_Error for random forest classifier = 45.0
```

4.Model score

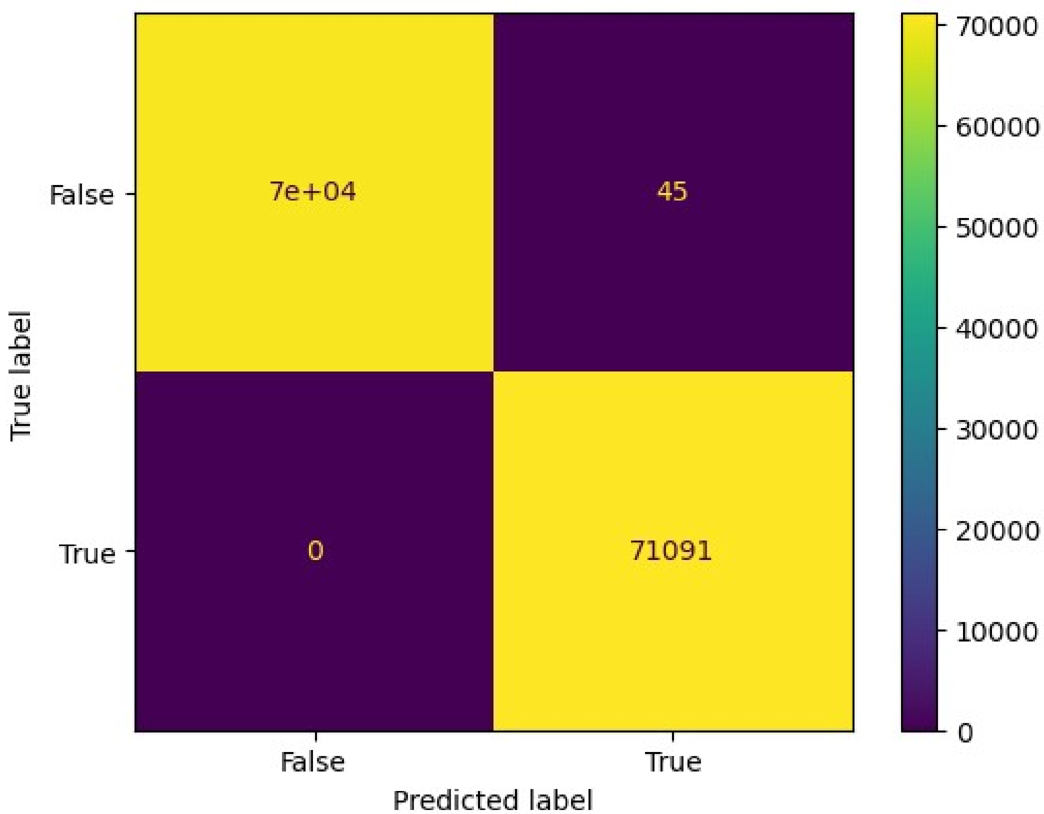
```
scoreOfModel1_xgb = xgb.score(x_test,y_test)
print(scoreOfModel1_xgb)
```

```
0.9996822639750895
```

5.Confusion_Matrix

```
y_test_xgb=xgb_ps["Actual"]
y_pred_xgb=xgb_ps["pred"]
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
def display_confusion_matrix(y_test_xgb,y_pred_xgb):
    matrix = metrics.confusion_matrix(y_test_xgb,y_pred_xgb)
    matrixDisplay = metrics.ConfusionMatrixDisplay(confusion_matrix =
matrix, display_labels = [False, True])
    matrixDisplay.plot()
    plt.show()
display_confusion_matrix(y_test_xgb,y_pred_xgb)
```




```
print(classification_report(y_test_xgb,y_pred_xgb))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     70536
         1.0       1.00      1.00      1.00     71091

    accuracy                           1.00    141627
   macro avg       1.00      1.00      1.00    141627
weighted avg       1.00      1.00      1.00    141627
```

6.Error Percentage

```
a=(xgb_ps.shape)
b=pd.DataFrame(a)
c=b.head(1)
c=np.array(c)
Error_Percentage_xgb=(Mean_Sq_Error_xgb/(np.array(c))*100)
print("Error occured in XG boosting =",Error_Percentage_xgb,"%")
```

```
Error occured in XG boosting = [[0.0317736]] %
```

7.Result Percentage

```
print("Result Percentage=",(100-Error_Percentage_xgb),"%")
```

```
Result Percentage= [[99.9682264]] %
```

8.Conclusion

Now we can see the results of XGB model of 99.96822% accurate, It is little lower than RF model

Predection 3

1.Using Ada Boost Classifier

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.

```
from sklearn.ensemble import AdaBoostClassifier

abc =AdaBoostClassifier(n_estimators=50,learning_rate=1,
random_state=0)

adb = abc.fit(x_train, y_train)

adb_pred = adb.predict(x_test)

print(adb_pred)

[0. 1. 0. ... 0. 0. 1.]

adb_pred=pd.DataFrame(adb_pred)
print(adb_pred)

          0
0       0.0
1       1.0
2       0.0
3       0.0
4       0.0
...     ...
141622  1.0
141623  0.0
141624  0.0
141625  0.0
141626  1.0

[141627 rows x 1 columns]

adb_pred=pd.DataFrame(adb_pred)
print(adb_pred.head())
y_test_adb=np.array(y_test)
y_test_adb=pd.DataFrame(y_test)
print(y_test_adb.head())
```

```
     0
0  0.0
1  1.0
2  0.0
3  0.0
4  0.0
   Actual
0     0.0
1     1.0
2     0.0
3     1.0
4     0.0
```

```
y_test_adb.columns=["Actual"]
adb_pred.columns=["Predection"]

adb_ps=pd.DataFrame()
adb_ps["pred"]=adb_pred.Predection

adb_ps["Actual"]=y_test_adb.Actual
print(adb_ps.head())
```

```
   pred  Actual
0   0.0     0.0
1   1.0     1.0
2   0.0     0.0
3   0.0     1.0
4   0.0     0.0
```

2.Analysing the predected ADB Gradient boosting classifier values

```
print(adb_ps["pred"].value_counts())
```

```
0.0    72144
1.0    69483
Name: pred, dtype: int64
```

```
print(adb_ps["Actual"].value_counts())
```

```
1.0    71091
0.0    70536
Name: Actual, dtype: int64
```

3.Mean square error

```
Mean_Sq_Error1=((adb_ps["pred"])-(adb_ps["Actual"]))
Mean_Sq_Error2=((Mean_Sq_Error1)**2)
Mean_Sq_Error_adb=Mean_Sq_Error2.sum()
print("Mean_Sq_Error for Adb Gradient boosting classifier
=",Mean_Sq_Error_adb)
```

```
Mean_Sq_Error for Adb Gradient boosting classifier = 5050.0
```

4.Model score

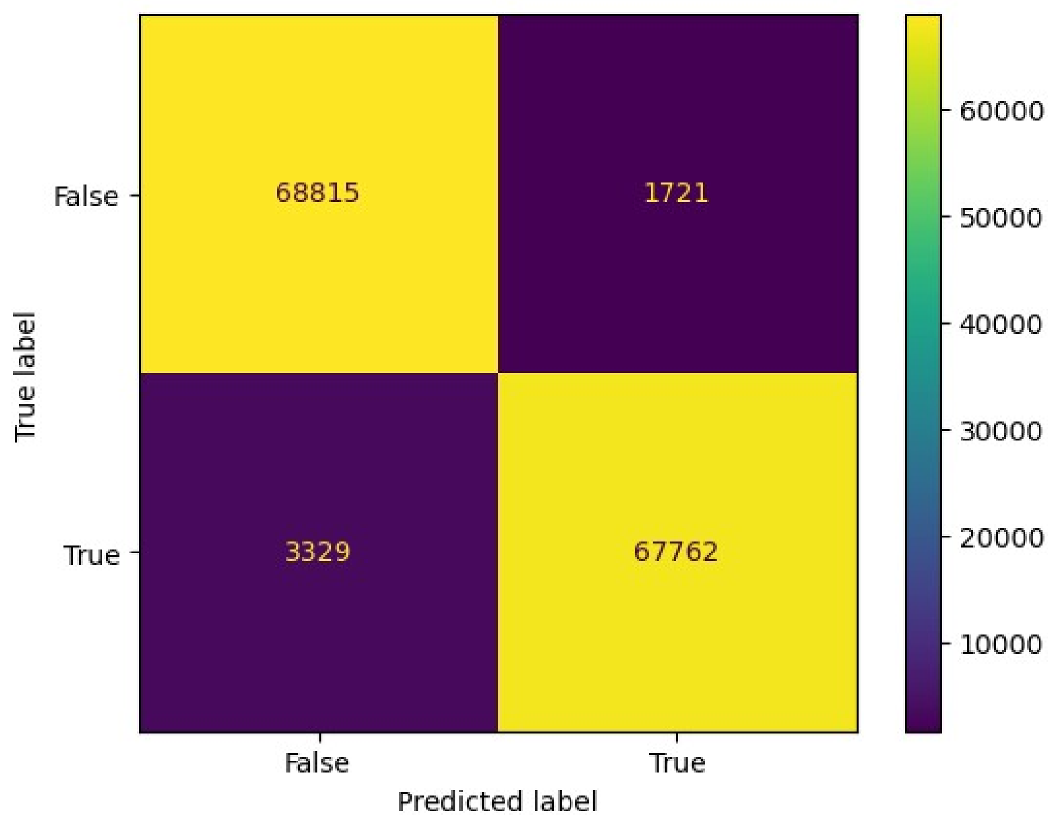
```
scoreOfModel1_adb = abc.score(x_test,y_test)
print(scoreOfModel1_adb)
```

```
0.9643429572044878
```

5.Confusion_Matrix

```
y_test_adb=adb_ps["Actual"]
y_pred_adb=adb_ps["pred"]
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
def display_confusion_matrix(y_test_adb,y_pred_adb):
    matrix = metrics.confusion_matrix(y_test_adb,y_pred_adb)
    matrixDisplay = metrics.ConfusionMatrixDisplay(confusion_matrix = 
matrix, display_labels = [False, True])
    matrixDisplay.plot()
    plt.show()
display_confusion_matrix(y_test_adb,y_pred_adb)
```




```
print(classification_report(y_test_adb,y_pred_adb))
```

```
              precision    recall  f1-score   support

         0.0       0.95      0.98      0.96     70536
         1.0       0.98      0.95      0.96     71091
```

```
    accuracy                           0.96    141627
   macro avg       0.96      0.96      0.96    141627
weighted avg       0.96      0.96      0.96    141627
```

6.Error Percentage

```
a=(adb_ps.shape)
b=pd.DataFrame(a)
c=b.head(1)
c=np.array(c)
Error_Percentage_adb=(Mean_Sq_Error_adb/(np.array(c))*100)
print("Error occured in adb boosting =",Error_Percentage_adb,"%")
```

```
Error occured in adb boosting = [[3.56570428]] %
```

7.Result Percentage

```
print("Result Percentage=",(100-Error_Percentage_adb),"%")
```

```
Result Percentage= [[96.43429572]] %
```

8.Conclusion

Take a look of the results of ADB boosting model of 96.434295% accurate, It is little lower than Above two predection models

Predection 4

1.Using K Nereast Neighbors Classifier

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows.

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn=KNeighborsClassifier()
```

```
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier()
```

```
knn_pred = knn.predict(x_test)
```

```
print(knn_pred)
```

```
[0. 1. 0. ... 0. 0. 1.]
```

```
knn_pred=pd.DataFrame(knn_pred)
print(knn_pred)
```

```
          0
0       0.0
1       1.0
2       0.0
3       1.0
4       0.0
...     ...
141622  1.0
141623  0.0
141624  0.0
141625  0.0
141626  1.0

[141627 rows x 1 columns]
```

```
knn_pred=pd.DataFrame(knn_pred)
print(knn_pred.head())
y_test_knn=np.array(y_test)
y_test_knn=pd.DataFrame(y_test)
print(y_test_knn.head())
```

```
     0
0  0.0
1  1.0
2  0.0
3  1.0
4  0.0
   Actual
0     0.0
1     1.0
2     0.0
3     1.0
4     0.0
```

```
y_test_knn.columns=["Actual"]
knn_pred.columns=["Predection"]
```

```
knn_ps=pd.DataFrame()
knn_ps["pred"]=knn_pred.Predection
```

```
knn_ps["Actual"]=y_test_knn.Actual
print(knn_ps.head())
```

```
   pred  Actual
0   0.0     0.0
1   1.0     1.0
2   0.0     0.0
3   1.0     1.0
4   0.0     0.0
```

2.Analysing the predected KNN

```
print(knn_ps["pred"].value_counts())
```

```
1.0    71226
0.0    70401
Name: pred, dtype: int64
```

```
print(knn_ps["Actual"].value_counts())
```

```
1.0    71091
0.0    70536
Name: Actual, dtype: int64
```

3.Mean square error

```
Mean_Sq_Error1=((knn_ps["pred"])-(knn_ps["Actual"]))
Mean_Sq_Error2=((Mean_Sq_Error1)**2)
Mean_Sq_Error_knn=Mean_Sq_Error2.sum()
print("Mean_Sq_Error for KNN classifier =",Mean_Sq_Error_knn)
```

```
Mean_Sq_Error for KNN classifier = 135.0
```

4.Model score

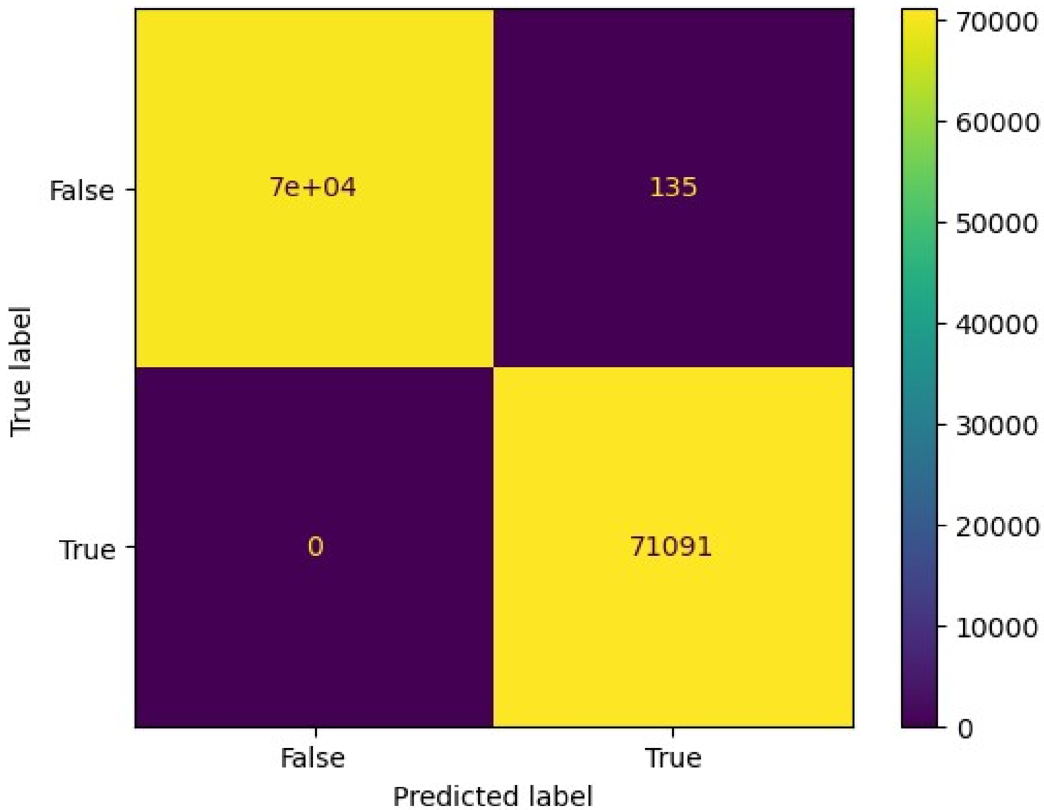
```
scoreOfModel1_knn = knn.score(x_test,y_test)
print(scoreOfModel1_knn)
```

```
0.9990467919252685
```

5.Confusion_Matrix

```
y_test_knn=knn_ps["Actual"]
y_pred_knn=knn_ps["pred"]
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
def display_confusion_matrix(y_test_knn,y_pred_knn):
    matrix = metrics.confusion_matrix(y_test_knn,y_pred_knn)
    matrixDisplay = metrics.ConfusionMatrixDisplay(confusion_matrix =
matrix, display_labels = [False, True])
    matrixDisplay.plot()
    plt.show()
display_confusion_matrix(y_test_knn,y_pred_knn)
```




```
print(classification_report(y_test_knn,y_pred_knn))

              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     70536
         1.0       1.00      1.00      1.00     71091

    accuracy                           1.00    141627
   macro avg       1.00      1.00      1.00    141627
weighted avg       1.00      1.00      1.00    141627
```

6.Error Percentage

```
a=(knn_ps.shape)
b=pd.DataFrame(a)
c=b.head(1)
c=np.array(c)
Error_Percentage_knn=(Mean_Sq_Error_knn/(np.array(c))*100)
print("Error occured in KNN Clasifier =",Error_Percentage_knn,"%")

Error occured in KNN Clasifier = [[0.09532081]] %
```

7.Result Percentage

```
print("Result Percentage=",(100-Error_Percentage_knn),"%")
```

```
Result Percentage= [[99.90467919]] %
```

8.Conclusion

Well the results of KNN clasifier model is quite good of 99.904679% accurate, but not as good as Random forest classifier model

Predection 5

```
#pip install catboost

from catboost import CatBoostClassifier

cbc = CatBoostClassifier(
    learning_rate=0.1,
)

cbc.fit(x_train,y_train)
```

```
0:	learn: 0.5033341	total: 581ms	remaining: 9m 40s
1:	learn: 0.3956935	total: 891ms	remaining: 7m 24s
2:	learn: 0.3132168	total: 1.2s	remaining: 6m 39s
3:	learn: 0.2578766	total: 1.47s	remaining: 6m 5s
4:	learn: 0.2118644	total: 1.78s	remaining: 5m 55s
5:	learn: 0.1840176	total: 2.07s	remaining: 5m 42s
6:	learn: 0.1673491	total: 2.33s	remaining: 5m 31s
7:	learn: 0.1533325	total: 2.62s	remaining: 5m 24s
8:	learn: 0.1443765	total: 2.94s	remaining: 5m 23s
9:	learn: 0.1343810	total: 3.24s	remaining: 5m 21s
10:	learn: 0.1268920	total: 3.52s	remaining: 5m 16s
11:	learn: 0.1188018	total: 3.87s	remaining: 5m 18s
12:	learn: 0.1140100	total: 4.17s	remaining: 5m 16s
13:	learn: 0.1092083	total: 4.46s	remaining: 5m 14s
14:	learn: 0.1054809	total: 4.72s	remaining: 5m 9s
15:	learn: 0.0991064	total: 5.02s	remaining: 5m 9s
16:	learn: 0.0967850	total: 5.29s	remaining: 5m 5s
17:	learn: 0.0927375	total: 5.58s	remaining: 5m 4s
18:	learn: 0.0908760	total: 5.9s	remaining: 5m 4s
19:	learn: 0.0871318	total: 6.19s	remaining: 5m 3s
20:	learn: 0.0840641	total: 6.6s	remaining: 5m 7s
21:	learn: 0.0814714	total: 7.04s	remaining: 5m 12s
22:	learn: 0.0797379	total: 7.45s	remaining: 5m 16s
23:	learn: 0.0782583	total: 7.87s	remaining: 5m 20s
24:	learn: 0.0762988	total: 8.29s	remaining: 5m 23s
25:	learn: 0.0746177	total: 8.73s	remaining: 5m 27s
26:	learn: 0.0724271	total: 9.24s	remaining: 5m 32s
27:	learn: 0.0704851	total: 9.61s	remaining: 5m 33s
28:	learn: 0.0683574	total: 10s	remaining: 5m 35s
29:	learn: 0.0670610	total: 10.4s	remaining: 5m 34s
30:	learn: 0.0658768	total: 10.6s	remaining: 5m 32s
31:	learn: 0.0646693	total: 10.9s	remaining: 5m 29s
32:	learn: 0.0627621	total: 11.2s	remaining: 5m 27s
```

```
33:	learn: 0.0611757	total: 11.4s	remaining: 5m 25s
34:	learn: 0.0598674	total: 11.7s	remaining: 5m 23s
35:	learn: 0.0579955	total: 12s	remaining: 5m 21s
36:	learn: 0.0567060	total: 12.3s	remaining: 5m 19s
37:	learn: 0.0553512	total: 12.6s	remaining: 5m 18s
38:	learn: 0.0539827	total: 12.9s	remaining: 5m 17s
39:	learn: 0.0530438	total: 13.2s	remaining: 5m 16s
40:	learn: 0.0521525	total: 13.5s	remaining: 5m 15s
41:	learn: 0.0511736	total: 13.9s	remaining: 5m 16s
42:	learn: 0.0501554	total: 14.3s	remaining: 5m 17s
43:	learn: 0.0494417	total: 14.6s	remaining: 5m 16s
44:	learn: 0.0483379	total: 14.8s	remaining: 5m 14s
45:	learn: 0.0474845	total: 15.1s	remaining: 5m 13s
46:	learn: 0.0464166	total: 15.4s	remaining: 5m 11s
47:	learn: 0.0456917	total: 15.6s	remaining: 5m 9s
48:	learn: 0.0446962	total: 15.9s	remaining: 5m 9s
49:	learn: 0.0441089	total: 16.2s	remaining: 5m 8s
50:	learn: 0.0434000	total: 16.5s	remaining: 5m 6s
51:	learn: 0.0426393	total: 16.7s	remaining: 5m 5s
52:	learn: 0.0420492	total: 17s	remaining: 5m 3s
53:	learn: 0.0412143	total: 17.3s	remaining: 5m 3s
54:	learn: 0.0406158	total: 17.6s	remaining: 5m 1s
55:	learn: 0.0402537	total: 17.8s	remaining: 5m
56:	learn: 0.0395588	total: 18.1s	remaining: 5m
57:	learn: 0.0388410	total: 18.5s	remaining: 4m 59s
58:	learn: 0.0380850	total: 18.8s	remaining: 5m
59:	learn: 0.0372057	total: 19.2s	remaining: 5m
60:	learn: 0.0368224	total: 19.5s	remaining: 4m 59s
61:	learn: 0.0364648	total: 19.7s	remaining: 4m 57s
62:	learn: 0.0357028	total: 20s	remaining: 4m 56s
63:	learn: 0.0351576	total: 20.2s	remaining: 4m 55s
64:	learn: 0.0344877	total: 20.5s	remaining: 4m 54s
65:	learn: 0.0339578	total: 21s	remaining: 4m 57s
66:	learn: 0.0332404	total: 21.4s	remaining: 4m 57s
67:	learn: 0.0327968	total: 21.7s	remaining: 4m 57s
68:	learn: 0.0322024	total: 22s	remaining: 4m 57s
69:	learn: 0.0316143	total: 22.4s	remaining: 4m 57s
70:	learn: 0.0311961	total: 22.9s	remaining: 5m
71:	learn: 0.0305764	total: 23.6s	remaining: 5m 3s
72:	learn: 0.0300902	total: 24.1s	remaining: 5m 5s
73:	learn: 0.0295750	total: 24.6s	remaining: 5m 7s
74:	learn: 0.0292898	total: 25.1s	remaining: 5m 9s
75:	learn: 0.0288288	total: 25.5s	remaining: 5m 10s
76:	learn: 0.0284394	total: 26s	remaining: 5m 12s
77:	learn: 0.0280395	total: 26.3s	remaining: 5m 11s
78:	learn: 0.0276192	total: 26.6s	remaining: 5m 10s
79:	learn: 0.0271762	total: 26.9s	remaining: 5m 9s
80:	learn: 0.0265876	total: 27.3s	remaining: 5m 9s
81:	learn: 0.0260212	total: 27.6s	remaining: 5m 8s
82:	learn: 0.0255925	total: 27.9s	remaining: 5m 8s
```

```
83:	learn: 0.0253104	total: 28.3s	remaining: 5m 8s
84:	learn: 0.0250244	total: 28.6s	remaining: 5m 7s
85:	learn: 0.0248340	total: 29s	remaining: 5m 8s
86:	learn: 0.0244883	total: 29.4s	remaining: 5m 8s
87:	learn: 0.0242236	total: 29.9s	remaining: 5m 10s
88:	learn: 0.0239612	total: 30.3s	remaining: 5m 10s
89:	learn: 0.0236728	total: 30.6s	remaining: 5m 9s
90:	learn: 0.0232436	total: 30.9s	remaining: 5m 8s
91:	learn: 0.0229311	total: 31.1s	remaining: 5m 7s
92:	learn: 0.0224874	total: 31.4s	remaining: 5m 5s
93:	learn: 0.0222148	total: 31.6s	remaining: 5m 4s
94:	learn: 0.0218815	total: 31.9s	remaining: 5m 4s
95:	learn: 0.0216817	total: 32.2s	remaining: 5m 3s
96:	learn: 0.0213489	total: 32.6s	remaining: 5m 3s
97:	learn: 0.0209790	total: 32.9s	remaining: 5m 2s
98:	learn: 0.0207379	total: 33.1s	remaining: 5m 1s
99:	learn: 0.0204507	total: 33.4s	remaining: 5m
100:	learn: 0.0202118	total: 33.6s	remaining: 4m 59s
101:	learn: 0.0199163	total: 33.9s	remaining: 4m 58s
102:	learn: 0.0197736	total: 34.1s	remaining: 4m 57s
103:	learn: 0.0194401	total: 34.4s	remaining: 4m 56s
104:	learn: 0.0192717	total: 34.7s	remaining: 4m 55s
105:	learn: 0.0189531	total: 34.9s	remaining: 4m 54s
106:	learn: 0.0187008	total: 35.2s	remaining: 4m 53s
107:	learn: 0.0185012	total: 35.5s	remaining: 4m 53s
108:	learn: 0.0183448	total: 35.7s	remaining: 4m 51s
109:	learn: 0.0180819	total: 36s	remaining: 4m 51s
110:	learn: 0.0179633	total: 36.2s	remaining: 4m 50s
111:	learn: 0.0178224	total: 36.5s	remaining: 4m 49s
112:	learn: 0.0175597	total: 36.7s	remaining: 4m 48s
113:	learn: 0.0174143	total: 37s	remaining: 4m 47s
114:	learn: 0.0172170	total: 37.3s	remaining: 4m 46s
115:	learn: 0.0170559	total: 37.5s	remaining: 4m 45s
116:	learn: 0.0168822	total: 37.8s	remaining: 4m 44s
117:	learn: 0.0166743	total: 38s	remaining: 4m 44s
118:	learn: 0.0165502	total: 38.3s	remaining: 4m 43s
119:	learn: 0.0163274	total: 38.7s	remaining: 4m 43s
120:	learn: 0.0162206	total: 39s	remaining: 4m 43s
121:	learn: 0.0161240	total: 39.4s	remaining: 4m 43s
122:	learn: 0.0159529	total: 39.9s	remaining: 4m 44s
123:	learn: 0.0157451	total: 40.3s	remaining: 4m 44s
124:	learn: 0.0153326	total: 40.7s	remaining: 4m 44s
125:	learn: 0.0152385	total: 41.1s	remaining: 4m 45s
126:	learn: 0.0150916	total: 41.5s	remaining: 4m 45s
127:	learn: 0.0148729	total: 41.8s	remaining: 4m 44s
128:	learn: 0.0146916	total: 42.1s	remaining: 4m 43s
129:	learn: 0.0145949	total: 42.3s	remaining: 4m 43s
130:	learn: 0.0144335	total: 42.6s	remaining: 4m 42s
131:	learn: 0.0143689	total: 42.9s	remaining: 4m 41s
132:	learn: 0.0142473	total: 43.1s	remaining: 4m 40s
```

```
133:	learn: 0.0140683	total: 43.3s	remaining: 4m 40s
134:	learn: 0.0139678	total: 43.6s	remaining: 4m 39s
135:	learn: 0.0137464	total: 43.8s	remaining: 4m 38s
136:	learn: 0.0136809	total: 44.1s	remaining: 4m 37s
137:	learn: 0.0136203	total: 44.3s	remaining: 4m 36s
138:	learn: 0.0134120	total: 44.6s	remaining: 4m 36s
139:	learn: 0.0132915	total: 44.8s	remaining: 4m 35s
140:	learn: 0.0132398	total: 45s	remaining: 4m 34s
141:	learn: 0.0130954	total: 45.3s	remaining: 4m 33s
142:	learn: 0.0128828	total: 45.6s	remaining: 4m 33s
143:	learn: 0.0127669	total: 45.8s	remaining: 4m 32s
144:	learn: 0.0127077	total: 46.1s	remaining: 4m 31s
145:	learn: 0.0125844	total: 46.3s	remaining: 4m 30s
146:	learn: 0.0123985	total: 46.6s	remaining: 4m 30s
147:	learn: 0.0123081	total: 46.8s	remaining: 4m 29s
148:	learn: 0.0121972	total: 47.1s	remaining: 4m 28s
149:	learn: 0.0120573	total: 47.3s	remaining: 4m 28s
150:	learn: 0.0119942	total: 47.6s	remaining: 4m 27s
151:	learn: 0.0118960	total: 47.8s	remaining: 4m 26s
152:	learn: 0.0118112	total: 48.1s	remaining: 4m 26s
153:	learn: 0.0116645	total: 48.4s	remaining: 4m 25s
154:	learn: 0.0115933	total: 48.6s	remaining: 4m 24s
155:	learn: 0.0115272	total: 48.8s	remaining: 4m 24s
156:	learn: 0.0113953	total: 49.1s	remaining: 4m 23s
157:	learn: 0.0112639	total: 49.4s	remaining: 4m 23s
158:	learn: 0.0111754	total: 49.6s	remaining: 4m 22s

159:	learn: 0.0110876	total: 49.8s	remaining: 4m 21s
160:	learn: 0.0109743	total: 50.1s	remaining: 4m 21s
161:	learn: 0.0109060	total: 50.4s	remaining: 4m 20s
162:	learn: 0.0107263	total: 50.6s	remaining: 4m 19s
163:	learn: 0.0106466	total: 50.9s	remaining: 4m 19s
164:	learn: 0.0105779	total: 51.1s	remaining: 4m 18s
165:	learn: 0.0104220	total: 51.4s	remaining: 4m 18s
166:	learn: 0.0103349	total: 51.6s	remaining: 4m 17s
167:	learn: 0.0102088	total: 51.9s	remaining: 4m 17s
168:	learn: 0.0101163	total: 52.2s	remaining: 4m 16s
169:	learn: 0.0100311	total: 52.5s	remaining: 4m 16s
170:	learn: 0.0099793	total: 52.7s	remaining: 4m 15s
171:	learn: 0.0098931	total: 52.9s	remaining: 4m 14s
172:	learn: 0.0097862	total: 53.2s	remaining: 4m 14s
173:	learn: 0.0096774	total: 53.5s	remaining: 4m 13s
174:	learn: 0.0096072	total: 53.7s	remaining: 4m 13s
175:	learn: 0.0095650	total: 54s	remaining: 4m 12s
176:	learn: 0.0094913	total: 54.2s	remaining: 4m 12s
177:	learn: 0.0094109	total: 54.5s	remaining: 4m 11s
178:	learn: 0.0093518	total: 54.9s	remaining: 4m 11s
179:	learn: 0.0092389	total: 55.4s	remaining: 4m 12s
180:	learn: 0.0091799	total: 55.8s	remaining: 4m 12s
181:	learn: 0.0091186	total: 56.2s	remaining: 4m 12s
```

```
182:	learn: 0.0090515	total: 56.6s	remaining: 4m 12s
183:	learn: 0.0089754	total: 57s	remaining: 4m 12s
184:	learn: 0.0088325	total: 57.5s	remaining: 4m 13s
185:	learn: 0.0087036	total: 57.7s	remaining: 4m 12s
186:	learn: 0.0085682	total: 58s	remaining: 4m 12s
187:	learn: 0.0085011	total: 58.3s	remaining: 4m 11s
188:	learn: 0.0084450	total: 58.5s	remaining: 4m 11s
189:	learn: 0.0083843	total: 58.8s	remaining: 4m 10s
190:	learn: 0.0083531	total: 59s	remaining: 4m 10s
191:	learn: 0.0082843	total: 59.3s	remaining: 4m 9s
192:	learn: 0.0082488	total: 59.5s	remaining: 4m 8s
193:	learn: 0.0081979	total: 59.7s	remaining: 4m 8s
194:	learn: 0.0081580	total: 60s	remaining: 4m 7s
195:	learn: 0.0081096	total: 1m	remaining: 4m 6s
196:	learn: 0.0080568	total: 1m	remaining: 4m 6s
197:	learn: 0.0079649	total: 1m	remaining: 4m 5s
198:	learn: 0.0078746	total: 1m	remaining: 4m 5s
199:	learn: 0.0078169	total: 1m 1s	remaining: 4m 5s
200:	learn: 0.0077724	total: 1m 1s	remaining: 4m 4s
201:	learn: 0.0076893	total: 1m 1s	remaining: 4m 4s
202:	learn: 0.0076293	total: 1m 2s	remaining: 4m 3s
203:	learn: 0.0075561	total: 1m 2s	remaining: 4m 3s
204:	learn: 0.0074826	total: 1m 2s	remaining: 4m 2s
205:	learn: 0.0074164	total: 1m 2s	remaining: 4m 2s
206:	learn: 0.0073098	total: 1m 3s	remaining: 4m 1s
207:	learn: 0.0072449	total: 1m 3s	remaining: 4m 1s
208:	learn: 0.0071529	total: 1m 3s	remaining: 4m
209:	learn: 0.0070774	total: 1m 3s	remaining: 4m
210:	learn: 0.0070315	total: 1m 4s	remaining: 3m 59s
211:	learn: 0.0070168	total: 1m 4s	remaining: 3m 59s
212:	learn: 0.0069810	total: 1m 4s	remaining: 3m 58s
213:	learn: 0.0069359	total: 1m 4s	remaining: 3m 58s
214:	learn: 0.0068984	total: 1m 5s	remaining: 3m 57s
215:	learn: 0.0067989	total: 1m 5s	remaining: 3m 57s
216:	learn: 0.0067462	total: 1m 5s	remaining: 3m 56s
217:	learn: 0.0066974	total: 1m 5s	remaining: 3m 56s
218:	learn: 0.0065973	total: 1m 6s	remaining: 3m 56s
219:	learn: 0.0065069	total: 1m 6s	remaining: 3m 55s
220:	learn: 0.0064414	total: 1m 6s	remaining: 3m 55s
221:	learn: 0.0063706	total: 1m 6s	remaining: 3m 54s
222:	learn: 0.0063343	total: 1m 7s	remaining: 3m 54s
223:	learn: 0.0062993	total: 1m 7s	remaining: 3m 53s
224:	learn: 0.0062540	total: 1m 7s	remaining: 3m 53s
225:	learn: 0.0062107	total: 1m 7s	remaining: 3m 52s
226:	learn: 0.0061987	total: 1m 8s	remaining: 3m 52s
227:	learn: 0.0061759	total: 1m 8s	remaining: 3m 51s
228:	learn: 0.0061047	total: 1m 8s	remaining: 3m 51s
229:	learn: 0.0060465	total: 1m 9s	remaining: 3m 51s
230:	learn: 0.0059819	total: 1m 9s	remaining: 3m 50s
231:	learn: 0.0059673	total: 1m 9s	remaining: 3m 50s
```

```
232:	learn: 0.0059230	total: 1m 9s	remaining: 3m 49s
233:	learn: 0.0058508	total: 1m 10s	remaining: 3m 49s
234:	learn: 0.0058404	total: 1m 10s	remaining: 3m 48s
235:	learn: 0.0057960	total: 1m 10s	remaining: 3m 48s
236:	learn: 0.0057452	total: 1m 11s	remaining: 3m 48s
237:	learn: 0.0057040	total: 1m 11s	remaining: 3m 48s
238:	learn: 0.0056498	total: 1m 11s	remaining: 3m 48s
239:	learn: 0.0056004	total: 1m 12s	remaining: 3m 49s
240:	learn: 0.0055249	total: 1m 12s	remaining: 3m 49s
241:	learn: 0.0054963	total: 1m 13s	remaining: 3m 49s
242:	learn: 0.0054447	total: 1m 13s	remaining: 3m 49s
243:	learn: 0.0054286	total: 1m 13s	remaining: 3m 48s
244:	learn: 0.0053858	total: 1m 14s	remaining: 3m 48s
245:	learn: 0.0053365	total: 1m 14s	remaining: 3m 47s
246:	learn: 0.0053096	total: 1m 14s	remaining: 3m 47s
247:	learn: 0.0053096	total: 1m 14s	remaining: 3m 46s
248:	learn: 0.0053096	total: 1m 14s	remaining: 3m 46s
249:	learn: 0.0053096	total: 1m 15s	remaining: 3m 45s
250:	learn: 0.0053096	total: 1m 15s	remaining: 3m 44s
251:	learn: 0.0053096	total: 1m 15s	remaining: 3m 44s
252:	learn: 0.0053096	total: 1m 15s	remaining: 3m 43s
253:	learn: 0.0053096	total: 1m 15s	remaining: 3m 42s
254:	learn: 0.0053096	total: 1m 16s	remaining: 3m 42s
255:	learn: 0.0053096	total: 1m 16s	remaining: 3m 41s
256:	learn: 0.0053095	total: 1m 16s	remaining: 3m 40s
257:	learn: 0.0053095	total: 1m 16s	remaining: 3m 40s
258:	learn: 0.0053095	total: 1m 16s	remaining: 3m 39s
259:	learn: 0.0053095	total: 1m 16s	remaining: 3m 38s
260:	learn: 0.0053095	total: 1m 17s	remaining: 3m 38s
261:	learn: 0.0053095	total: 1m 17s	remaining: 3m 37s
262:	learn: 0.0053095	total: 1m 17s	remaining: 3m 37s
263:	learn: 0.0053095	total: 1m 17s	remaining: 3m 36s
264:	learn: 0.0053095	total: 1m 17s	remaining: 3m 35s
265:	learn: 0.0053095	total: 1m 18s	remaining: 3m 35s
266:	learn: 0.0052676	total: 1m 18s	remaining: 3m 34s
267:	learn: 0.0052527	total: 1m 18s	remaining: 3m 34s
268:	learn: 0.0052296	total: 1m 18s	remaining: 3m 34s
269:	learn: 0.0052026	total: 1m 19s	remaining: 3m 33s
270:	learn: 0.0052026	total: 1m 19s	remaining: 3m 33s
271:	learn: 0.0052026	total: 1m 19s	remaining: 3m 32s
272:	learn: 0.0052026	total: 1m 19s	remaining: 3m 31s
273:	learn: 0.0052026	total: 1m 19s	remaining: 3m 31s
274:	learn: 0.0052025	total: 1m 19s	remaining: 3m 30s
275:	learn: 0.0052025	total: 1m 20s	remaining: 3m 30s
276:	learn: 0.0051873	total: 1m 20s	remaining: 3m 29s
277:	learn: 0.0051873	total: 1m 20s	remaining: 3m 29s
278:	learn: 0.0051873	total: 1m 20s	remaining: 3m 28s
279:	learn: 0.0051873	total: 1m 20s	remaining: 3m 27s
280:	learn: 0.0051873	total: 1m 21s	remaining: 3m 27s
281:	learn: 0.0051873	total: 1m 21s	remaining: 3m 26s
```

```
282:	learn: 0.0051873	total: 1m 21s	remaining: 3m 26s
283:	learn: 0.0051873	total: 1m 21s	remaining: 3m 25s
284:	learn: 0.0051873	total: 1m 21s	remaining: 3m 25s
285:	learn: 0.0051873	total: 1m 21s	remaining: 3m 24s
286:	learn: 0.0051873	total: 1m 22s	remaining: 3m 23s
287:	learn: 0.0051873	total: 1m 22s	remaining: 3m 23s
288:	learn: 0.0051873	total: 1m 22s	remaining: 3m 22s
289:	learn: 0.0051873	total: 1m 22s	remaining: 3m 22s
290:	learn: 0.0051873	total: 1m 22s	remaining: 3m 21s
291:	learn: 0.0051873	total: 1m 22s	remaining: 3m 21s
292:	learn: 0.0051873	total: 1m 23s	remaining: 3m 20s
293:	learn: 0.0051872	total: 1m 23s	remaining: 3m 20s
294:	learn: 0.0051872	total: 1m 23s	remaining: 3m 19s
295:	learn: 0.0051872	total: 1m 23s	remaining: 3m 18s
296:	learn: 0.0051872	total: 1m 23s	remaining: 3m 18s
297:	learn: 0.0051872	total: 1m 24s	remaining: 3m 17s
298:	learn: 0.0051872	total: 1m 24s	remaining: 3m 17s
299:	learn: 0.0051872	total: 1m 24s	remaining: 3m 16s
300:	learn: 0.0051872	total: 1m 24s	remaining: 3m 16s
301:	learn: 0.0051872	total: 1m 24s	remaining: 3m 15s
302:	learn: 0.0051872	total: 1m 24s	remaining: 3m 15s
303:	learn: 0.0051871	total: 1m 25s	remaining: 3m 14s
304:	learn: 0.0051871	total: 1m 25s	remaining: 3m 14s
305:	learn: 0.0051871	total: 1m 25s	remaining: 3m 13s
306:	learn: 0.0051871	total: 1m 25s	remaining: 3m 13s
307:	learn: 0.0051871	total: 1m 25s	remaining: 3m 12s
308:	learn: 0.0051871	total: 1m 25s	remaining: 3m 12s
309:	learn: 0.0051871	total: 1m 26s	remaining: 3m 11s
310:	learn: 0.0051871	total: 1m 26s	remaining: 3m 11s
311:	learn: 0.0051871	total: 1m 26s	remaining: 3m 10s
312:	learn: 0.0051871	total: 1m 26s	remaining: 3m 10s

313:	learn: 0.0051871	total: 1m 27s	remaining: 3m 10s
314:	learn: 0.0051871	total: 1m 27s	remaining: 3m 10s
315:	learn: 0.0051870	total: 1m 27s	remaining: 3m 9s
316:	learn: 0.0051870	total: 1m 28s	remaining: 3m 9s
317:	learn: 0.0051870	total: 1m 28s	remaining: 3m 9s
318:	learn: 0.0051870	total: 1m 28s	remaining: 3m 9s
319:	learn: 0.0051870	total: 1m 28s	remaining: 3m 9s
320:	learn: 0.0051870	total: 1m 29s	remaining: 3m 8s
321:	learn: 0.0051870	total: 1m 29s	remaining: 3m 8s
322:	learn: 0.0051870	total: 1m 29s	remaining: 3m 8s
323:	learn: 0.0051870	total: 1m 29s	remaining: 3m 7s
324:	learn: 0.0051869	total: 1m 30s	remaining: 3m 7s
325:	learn: 0.0051869	total: 1m 30s	remaining: 3m 6s
326:	learn: 0.0051869	total: 1m 30s	remaining: 3m 6s
327:	learn: 0.0051869	total: 1m 30s	remaining: 3m 5s
328:	learn: 0.0051869	total: 1m 30s	remaining: 3m 5s
329:	learn: 0.0051869	total: 1m 31s	remaining: 3m 4s
330:	learn: 0.0051869	total: 1m 31s	remaining: 3m 4s
```

```
331:	learn: 0.0051869	total: 1m 31s	remaining: 3m 3s
332:	learn: 0.0051869	total: 1m 31s	remaining: 3m 3s
333:	learn: 0.0051869	total: 1m 31s	remaining: 3m 2s
334:	learn: 0.0051869	total: 1m 31s	remaining: 3m 2s
335:	learn: 0.0051868	total: 1m 32s	remaining: 3m 1s
336:	learn: 0.0051868	total: 1m 32s	remaining: 3m 1s
337:	learn: 0.0051868	total: 1m 32s	remaining: 3m 1s
338:	learn: 0.0051696	total: 1m 32s	remaining: 3m
339:	learn: 0.0051165	total: 1m 32s	remaining: 3m
340:	learn: 0.0050466	total: 1m 33s	remaining: 3m
341:	learn: 0.0050107	total: 1m 33s	remaining: 2m 59s
342:	learn: 0.0049862	total: 1m 33s	remaining: 2m 59s
343:	learn: 0.0049814	total: 1m 33s	remaining: 2m 59s
344:	learn: 0.0049519	total: 1m 34s	remaining: 2m 58s
345:	learn: 0.0049274	total: 1m 34s	remaining: 2m 58s
346:	learn: 0.0048850	total: 1m 34s	remaining: 2m 58s
347:	learn: 0.0048777	total: 1m 34s	remaining: 2m 57s
348:	learn: 0.0048663	total: 1m 35s	remaining: 2m 57s
349:	learn: 0.0048314	total: 1m 35s	remaining: 2m 57s
350:	learn: 0.0048275	total: 1m 35s	remaining: 2m 56s
351:	learn: 0.0047983	total: 1m 35s	remaining: 2m 56s
352:	learn: 0.0047577	total: 1m 36s	remaining: 2m 56s
353:	learn: 0.0047351	total: 1m 36s	remaining: 2m 56s
354:	learn: 0.0047015	total: 1m 36s	remaining: 2m 55s
355:	learn: 0.0046679	total: 1m 36s	remaining: 2m 55s
356:	learn: 0.0046408	total: 1m 37s	remaining: 2m 55s
357:	learn: 0.0046090	total: 1m 37s	remaining: 2m 54s
358:	learn: 0.0045762	total: 1m 37s	remaining: 2m 54s
359:	learn: 0.0045358	total: 1m 37s	remaining: 2m 54s
360:	learn: 0.0045084	total: 1m 38s	remaining: 2m 53s
361:	learn: 0.0045084	total: 1m 38s	remaining: 2m 53s
362:	learn: 0.0044949	total: 1m 38s	remaining: 2m 53s
363:	learn: 0.0044949	total: 1m 38s	remaining: 2m 52s
364:	learn: 0.0044949	total: 1m 39s	remaining: 2m 52s
365:	learn: 0.0044949	total: 1m 39s	remaining: 2m 51s
366:	learn: 0.0044949	total: 1m 39s	remaining: 2m 51s
367:	learn: 0.0044949	total: 1m 39s	remaining: 2m 51s
368:	learn: 0.0044949	total: 1m 39s	remaining: 2m 50s
369:	learn: 0.0044829	total: 1m 40s	remaining: 2m 50s
370:	learn: 0.0044828	total: 1m 40s	remaining: 2m 50s
371:	learn: 0.0044828	total: 1m 40s	remaining: 2m 49s
372:	learn: 0.0044828	total: 1m 40s	remaining: 2m 49s
373:	learn: 0.0044828	total: 1m 40s	remaining: 2m 48s
374:	learn: 0.0044828	total: 1m 41s	remaining: 2m 48s
375:	learn: 0.0044828	total: 1m 41s	remaining: 2m 48s
376:	learn: 0.0044828	total: 1m 41s	remaining: 2m 47s
377:	learn: 0.0044828	total: 1m 41s	remaining: 2m 47s
378:	learn: 0.0044827	total: 1m 41s	remaining: 2m 46s
379:	learn: 0.0044827	total: 1m 42s	remaining: 2m 46s
380:	learn: 0.0044827	total: 1m 42s	remaining: 2m 46s
```

```
381:	learn: 0.0044827	total: 1m 42s	remaining: 2m 45s
382:	learn: 0.0044827	total: 1m 42s	remaining: 2m 45s
383:	learn: 0.0044827	total: 1m 43s	remaining: 2m 45s
384:	learn: 0.0044827	total: 1m 43s	remaining: 2m 45s
385:	learn: 0.0044827	total: 1m 43s	remaining: 2m 44s
386:	learn: 0.0044827	total: 1m 44s	remaining: 2m 44s
387:	learn: 0.0044827	total: 1m 44s	remaining: 2m 44s
388:	learn: 0.0044827	total: 1m 44s	remaining: 2m 44s
389:	learn: 0.0044827	total: 1m 45s	remaining: 2m 44s
390:	learn: 0.0044826	total: 1m 45s	remaining: 2m 44s
391:	learn: 0.0044826	total: 1m 45s	remaining: 2m 43s
392:	learn: 0.0044826	total: 1m 45s	remaining: 2m 43s
393:	learn: 0.0044826	total: 1m 46s	remaining: 2m 43s
394:	learn: 0.0044826	total: 1m 46s	remaining: 2m 42s
395:	learn: 0.0044826	total: 1m 46s	remaining: 2m 42s
396:	learn: 0.0044826	total: 1m 46s	remaining: 2m 41s
397:	learn: 0.0044826	total: 1m 46s	remaining: 2m 41s
398:	learn: 0.0044510	total: 1m 47s	remaining: 2m 41s
399:	learn: 0.0043955	total: 1m 47s	remaining: 2m 41s
400:	learn: 0.0043914	total: 1m 47s	remaining: 2m 40s
401:	learn: 0.0043914	total: 1m 47s	remaining: 2m 40s
402:	learn: 0.0043914	total: 1m 47s	remaining: 2m 39s
403:	learn: 0.0043914	total: 1m 48s	remaining: 2m 39s
404:	learn: 0.0043914	total: 1m 48s	remaining: 2m 39s
405:	learn: 0.0043914	total: 1m 48s	remaining: 2m 38s
406:	learn: 0.0043914	total: 1m 48s	remaining: 2m 38s
407:	learn: 0.0043914	total: 1m 48s	remaining: 2m 38s
408:	learn: 0.0043914	total: 1m 49s	remaining: 2m 37s
409:	learn: 0.0043914	total: 1m 49s	remaining: 2m 37s
410:	learn: 0.0043914	total: 1m 49s	remaining: 2m 36s
411:	learn: 0.0043914	total: 1m 49s	remaining: 2m 36s
412:	learn: 0.0043914	total: 1m 49s	remaining: 2m 36s
413:	learn: 0.0043914	total: 1m 50s	remaining: 2m 35s
414:	learn: 0.0043914	total: 1m 50s	remaining: 2m 35s
415:	learn: 0.0043914	total: 1m 50s	remaining: 2m 35s
416:	learn: 0.0043913	total: 1m 50s	remaining: 2m 34s
417:	learn: 0.0043913	total: 1m 50s	remaining: 2m 34s
418:	learn: 0.0043913	total: 1m 51s	remaining: 2m 34s
419:	learn: 0.0043913	total: 1m 51s	remaining: 2m 33s
420:	learn: 0.0043913	total: 1m 51s	remaining: 2m 33s
421:	learn: 0.0043913	total: 1m 51s	remaining: 2m 32s
422:	learn: 0.0043913	total: 1m 51s	remaining: 2m 32s
423:	learn: 0.0043913	total: 1m 52s	remaining: 2m 32s
424:	learn: 0.0043913	total: 1m 52s	remaining: 2m 31s
425:	learn: 0.0043913	total: 1m 52s	remaining: 2m 31s
426:	learn: 0.0043913	total: 1m 52s	remaining: 2m 31s
427:	learn: 0.0043913	total: 1m 52s	remaining: 2m 30s
428:	learn: 0.0043913	total: 1m 53s	remaining: 2m 30s
429:	learn: 0.0043913	total: 1m 53s	remaining: 2m 30s
430:	learn: 0.0043913	total: 1m 53s	remaining: 2m 29s
```

```
431:  learn: 0.0043913 total: 1m 53s    remaining: 2m 29s
432:  learn: 0.0043913 total: 1m 53s    remaining: 2m 28s
433:  learn: 0.0043912 total: 1m 53s    remaining: 2m 28s
434:  learn: 0.0043912 total: 1m 54s    remaining: 2m 28s
435:  learn: 0.0043912 total: 1m 54s    remaining: 2m 27s
436:  learn: 0.0043912 total: 1m 54s    remaining: 2m 27s
437:  learn: 0.0043912 total: 1m 54s    remaining: 2m 27s
438:  learn: 0.0043912 total: 1m 54s    remaining: 2m 26s
439:  learn: 0.0043912 total: 1m 55s    remaining: 2m 26s
440:  learn: 0.0043912 total: 1m 55s    remaining: 2m 26s
441:  learn: 0.0043912 total: 1m 55s    remaining: 2m 25s
442:  learn: 0.0043912 total: 1m 55s    remaining: 2m 25s
443:  learn: 0.0043912 total: 1m 55s    remaining: 2m 25s
444:  learn: 0.0043912 total: 1m 56s    remaining: 2m 24s
445:  learn: 0.0043912 total: 1m 56s    remaining: 2m 24s
446:  learn: 0.0043912 total: 1m 56s    remaining: 2m 24s
447:  learn: 0.0043912 total: 1m 56s    remaining: 2m 23s
448:  learn: 0.0043912 total: 1m 56s    remaining: 2m 23s
449:  learn: 0.0043912 total: 1m 57s    remaining: 2m 23s
450:  learn: 0.0043912 total: 1m 57s    remaining: 2m 22s
451:  learn: 0.0043912 total: 1m 57s    remaining: 2m 22s
452:  learn: 0.0043912 total: 1m 57s    remaining: 2m 22s
453:  learn: 0.0043912 total: 1m 57s    remaining: 2m 21s
454:  learn: 0.0043912 total: 1m 57s    remaining: 2m 21s
455:  learn: 0.0043912 total: 1m 58s    remaining: 2m 20s
456:  learn: 0.0043912 total: 1m 58s    remaining: 2m 20s
457:  learn: 0.0043912 total: 1m 58s    remaining: 2m 20s
458:  learn: 0.0043912 total: 1m 58s    remaining: 2m 20s
459:  learn: 0.0043912 total: 1m 59s    remaining: 2m 19s
460:  learn: 0.0043912 total: 1m 59s    remaining: 2m 19s
461:  learn: 0.0043912 total: 1m 59s    remaining: 2m 19s
462:  learn: 0.0043912 total: 2m  remaining: 2m 19s
463:  learn: 0.0043912 total: 2m  remaining: 2m 19s
464:  learn: 0.0043912 total: 2m  remaining: 2m 19s
465:  learn: 0.0043912 total: 2m 1s     remaining: 2m 18s

466:  learn: 0.0043912 total: 2m 1s     remaining: 2m 18s
467:  learn: 0.0043912 total: 2m 1s     remaining: 2m 18s
468:  learn: 0.0043912 total: 2m 1s     remaining: 2m 18s
469:  learn: 0.0043912 total: 2m 2s     remaining: 2m 17s
470:  learn: 0.0043912 total: 2m 2s     remaining: 2m 17s
471:  learn: 0.0043911 total: 2m 2s     remaining: 2m 17s
472:  learn: 0.0043911 total: 2m 2s     remaining: 2m 16s
473:  learn: 0.0043911 total: 2m 2s     remaining: 2m 16s
474:  learn: 0.0043911 total: 2m 3s     remaining: 2m 16s
475:  learn: 0.0043911 total: 2m 3s     remaining: 2m 15s
476:  learn: 0.0043911 total: 2m 3s     remaining: 2m 15s
477:  learn: 0.0043911 total: 2m 3s     remaining: 2m 15s
478:  learn: 0.0043911 total: 2m 3s     remaining: 2m 14s
479:  learn: 0.0043911 total: 2m 4s     remaining: 2m 14s
```

```
480:	learn: 0.0043911	total: 2m 4s	remaining: 2m 14s
481:	learn: 0.0043911	total: 2m 4s	remaining: 2m 13s
482:	learn: 0.0043911	total: 2m 4s	remaining: 2m 13s
483:	learn: 0.0043911	total: 2m 4s	remaining: 2m 13s
484:	learn: 0.0043911	total: 2m 5s	remaining: 2m 12s
485:	learn: 0.0043911	total: 2m 5s	remaining: 2m 12s
486:	learn: 0.0043911	total: 2m 5s	remaining: 2m 12s
487:	learn: 0.0043911	total: 2m 5s	remaining: 2m 11s
488:	learn: 0.0043911	total: 2m 5s	remaining: 2m 11s
489:	learn: 0.0043911	total: 2m 5s	remaining: 2m 11s
490:	learn: 0.0043911	total: 2m 6s	remaining: 2m 10s
491:	learn: 0.0043911	total: 2m 6s	remaining: 2m 10s
492:	learn: 0.0043911	total: 2m 6s	remaining: 2m 10s
493:	learn: 0.0043910	total: 2m 6s	remaining: 2m 9s
494:	learn: 0.0043910	total: 2m 6s	remaining: 2m 9s
495:	learn: 0.0043910	total: 2m 7s	remaining: 2m 9s
496:	learn: 0.0043910	total: 2m 7s	remaining: 2m 8s
497:	learn: 0.0043910	total: 2m 7s	remaining: 2m 8s
498:	learn: 0.0043910	total: 2m 7s	remaining: 2m 8s
499:	learn: 0.0043910	total: 2m 7s	remaining: 2m 7s
500:	learn: 0.0043910	total: 2m 8s	remaining: 2m 7s
501:	learn: 0.0043910	total: 2m 8s	remaining: 2m 7s
502:	learn: 0.0043910	total: 2m 8s	remaining: 2m 6s
503:	learn: 0.0043910	total: 2m 8s	remaining: 2m 6s
504:	learn: 0.0043910	total: 2m 8s	remaining: 2m 6s
505:	learn: 0.0043910	total: 2m 9s	remaining: 2m 6s
506:	learn: 0.0043910	total: 2m 9s	remaining: 2m 5s
507:	learn: 0.0043651	total: 2m 9s	remaining: 2m 5s
508:	learn: 0.0043540	total: 2m 9s	remaining: 2m 5s
509:	learn: 0.0043335	total: 2m 9s	remaining: 2m 4s
510:	learn: 0.0043334	total: 2m 10s	remaining: 2m 4s
511:	learn: 0.0043137	total: 2m 10s	remaining: 2m 4s
512:	learn: 0.0042961	total: 2m 10s	remaining: 2m 4s
513:	learn: 0.0042818	total: 2m 10s	remaining: 2m 3s
514:	learn: 0.0042713	total: 2m 11s	remaining: 2m 3s
515:	learn: 0.0042713	total: 2m 11s	remaining: 2m 3s
516:	learn: 0.0042713	total: 2m 11s	remaining: 2m 2s
517:	learn: 0.0042713	total: 2m 11s	remaining: 2m 2s
518:	learn: 0.0042713	total: 2m 11s	remaining: 2m 2s
519:	learn: 0.0042712	total: 2m 12s	remaining: 2m 1s
520:	learn: 0.0042712	total: 2m 12s	remaining: 2m 1s
521:	learn: 0.0042712	total: 2m 12s	remaining: 2m 1s
522:	learn: 0.0042454	total: 2m 12s	remaining: 2m 1s
523:	learn: 0.0042369	total: 2m 13s	remaining: 2m
524:	learn: 0.0042096	total: 2m 13s	remaining: 2m
525:	learn: 0.0041751	total: 2m 13s	remaining: 2m
526:	learn: 0.0041612	total: 2m 13s	remaining: 2m
527:	learn: 0.0041348	total: 2m 14s	remaining: 1m 59s
528:	learn: 0.0041321	total: 2m 14s	remaining: 1m 59s
529:	learn: 0.0041024	total: 2m 14s	remaining: 1m 59s
```

```
530:	learn: 0.0040975	total: 2m 14s	remaining: 1m 59s
531:	learn: 0.0040975	total: 2m 15s	remaining: 1m 58s
532:	learn: 0.0040975	total: 2m 15s	remaining: 1m 58s
533:	learn: 0.0040975	total: 2m 15s	remaining: 1m 58s
534:	learn: 0.0040975	total: 2m 16s	remaining: 1m 58s
535:	learn: 0.0040975	total: 2m 16s	remaining: 1m 58s
536:	learn: 0.0040975	total: 2m 16s	remaining: 1m 57s
537:	learn: 0.0040975	total: 2m 17s	remaining: 1m 57s
538:	learn: 0.0040796	total: 2m 17s	remaining: 1m 57s
539:	learn: 0.0040796	total: 2m 17s	remaining: 1m 57s
540:	learn: 0.0040796	total: 2m 17s	remaining: 1m 57s
541:	learn: 0.0040796	total: 2m 18s	remaining: 1m 56s
542:	learn: 0.0040716	total: 2m 18s	remaining: 1m 56s
543:	learn: 0.0040716	total: 2m 18s	remaining: 1m 56s
544:	learn: 0.0040716	total: 2m 18s	remaining: 1m 55s
545:	learn: 0.0040716	total: 2m 18s	remaining: 1m 55s
546:	learn: 0.0040716	total: 2m 19s	remaining: 1m 55s
547:	learn: 0.0040716	total: 2m 19s	remaining: 1m 54s
548:	learn: 0.0040716	total: 2m 19s	remaining: 1m 54s
549:	learn: 0.0040646	total: 2m 19s	remaining: 1m 54s
550:	learn: 0.0040645	total: 2m 19s	remaining: 1m 53s
551:	learn: 0.0040645	total: 2m 20s	remaining: 1m 53s
552:	learn: 0.0040645	total: 2m 20s	remaining: 1m 53s
553:	learn: 0.0040645	total: 2m 20s	remaining: 1m 53s
554:	learn: 0.0040645	total: 2m 20s	remaining: 1m 52s
555:	learn: 0.0040645	total: 2m 20s	remaining: 1m 52s
556:	learn: 0.0040645	total: 2m 20s	remaining: 1m 52s
557:	learn: 0.0040645	total: 2m 21s	remaining: 1m 51s
558:	learn: 0.0040645	total: 2m 21s	remaining: 1m 51s
559:	learn: 0.0040645	total: 2m 21s	remaining: 1m 51s
560:	learn: 0.0040645	total: 2m 21s	remaining: 1m 50s
561:	learn: 0.0040645	total: 2m 21s	remaining: 1m 50s
562:	learn: 0.0040645	total: 2m 22s	remaining: 1m 50s
563:	learn: 0.0040645	total: 2m 22s	remaining: 1m 50s
564:	learn: 0.0040645	total: 2m 22s	remaining: 1m 49s
565:	learn: 0.0040645	total: 2m 22s	remaining: 1m 49s
566:	learn: 0.0040645	total: 2m 22s	remaining: 1m 49s
567:	learn: 0.0040645	total: 2m 23s	remaining: 1m 48s
568:	learn: 0.0040645	total: 2m 23s	remaining: 1m 48s
569:	learn: 0.0040433	total: 2m 23s	remaining: 1m 48s
570:	learn: 0.0040196	total: 2m 23s	remaining: 1m 48s
571:	learn: 0.0039927	total: 2m 24s	remaining: 1m 47s
572:	learn: 0.0039618	total: 2m 24s	remaining: 1m 47s
573:	learn: 0.0039596	total: 2m 24s	remaining: 1m 47s
574:	learn: 0.0039596	total: 2m 24s	remaining: 1m 46s
575:	learn: 0.0039596	total: 2m 24s	remaining: 1m 46s
576:	learn: 0.0039596	total: 2m 25s	remaining: 1m 46s
577:	learn: 0.0039596	total: 2m 25s	remaining: 1m 46s
578:	learn: 0.0039596	total: 2m 25s	remaining: 1m 45s
579:	learn: 0.0039512	total: 2m 25s	remaining: 1m 45s
```

```
580:	learn: 0.0039346	total: 2m 26s	remaining: 1m 45s
581:	learn: 0.0039345	total: 2m 26s	remaining: 1m 45s
582:	learn: 0.0039345	total: 2m 26s	remaining: 1m 44s
583:	learn: 0.0039345	total: 2m 26s	remaining: 1m 44s
584:	learn: 0.0039345	total: 2m 26s	remaining: 1m 44s
585:	learn: 0.0039345	total: 2m 27s	remaining: 1m 43s
586:	learn: 0.0039345	total: 2m 27s	remaining: 1m 43s
587:	learn: 0.0039345	total: 2m 27s	remaining: 1m 43s
588:	learn: 0.0039345	total: 2m 27s	remaining: 1m 42s
589:	learn: 0.0039064	total: 2m 27s	remaining: 1m 42s
590:	learn: 0.0038822	total: 2m 28s	remaining: 1m 42s
591:	learn: 0.0038822	total: 2m 28s	remaining: 1m 42s
592:	learn: 0.0038822	total: 2m 28s	remaining: 1m 41s
593:	learn: 0.0038822	total: 2m 28s	remaining: 1m 41s
594:	learn: 0.0038822	total: 2m 28s	remaining: 1m 41s
595:	learn: 0.0038822	total: 2m 29s	remaining: 1m 41s
596:	learn: 0.0038822	total: 2m 29s	remaining: 1m 40s
597:	learn: 0.0038822	total: 2m 29s	remaining: 1m 40s
598:	learn: 0.0038822	total: 2m 29s	remaining: 1m 40s
599:	learn: 0.0038821	total: 2m 29s	remaining: 1m 39s
600:	learn: 0.0038821	total: 2m 30s	remaining: 1m 39s
601:	learn: 0.0038821	total: 2m 30s	remaining: 1m 39s
602:	learn: 0.0038821	total: 2m 30s	remaining: 1m 39s
603:	learn: 0.0038708	total: 2m 31s	remaining: 1m 39s
604:	learn: 0.0038708	total: 2m 31s	remaining: 1m 38s
605:	learn: 0.0038708	total: 2m 31s	remaining: 1m 38s
606:	learn: 0.0038708	total: 2m 31s	remaining: 1m 38s
607:	learn: 0.0038708	total: 2m 32s	remaining: 1m 38s
608:	learn: 0.0038708	total: 2m 32s	remaining: 1m 38s
609:	learn: 0.0038708	total: 2m 33s	remaining: 1m 37s
610:	learn: 0.0038708	total: 2m 33s	remaining: 1m 37s
611:	learn: 0.0038707	total: 2m 33s	remaining: 1m 37s
612:	learn: 0.0038707	total: 2m 33s	remaining: 1m 37s
613:	learn: 0.0038708	total: 2m 34s	remaining: 1m 36s
614:	learn: 0.0038707	total: 2m 34s	remaining: 1m 36s
615:	learn: 0.0038707	total: 2m 34s	remaining: 1m 36s
616:	learn: 0.0038707	total: 2m 34s	remaining: 1m 35s
617:	learn: 0.0038707	total: 2m 34s	remaining: 1m 35s
618:	learn: 0.0038707	total: 2m 35s	remaining: 1m 35s
619:	learn: 0.0038707	total: 2m 35s	remaining: 1m 35s

620:	learn: 0.0038707	total: 2m 35s	remaining: 1m 34s
621:	learn: 0.0038707	total: 2m 35s	remaining: 1m 34s
622:	learn: 0.0038707	total: 2m 35s	remaining: 1m 34s
623:	learn: 0.0038707	total: 2m 35s	remaining: 1m 34s
624:	learn: 0.0038610	total: 2m 36s	remaining: 1m 33s
625:	learn: 0.0038610	total: 2m 36s	remaining: 1m 33s
626:	learn: 0.0038609	total: 2m 36s	remaining: 1m 33s
627:	learn: 0.0038609	total: 2m 36s	remaining: 1m 32s
628:	learn: 0.0038609	total: 2m 37s	remaining: 1m 32s
```

```
629:	learn: 0.0038609	total: 2m 37s	remaining: 1m 32s
630:	learn: 0.0038609	total: 2m 37s	remaining: 1m 32s
631:	learn: 0.0038609	total: 2m 37s	remaining: 1m 31s
632:	learn: 0.0038609	total: 2m 37s	remaining: 1m 31s
633:	learn: 0.0038609	total: 2m 38s	remaining: 1m 31s
634:	learn: 0.0038609	total: 2m 38s	remaining: 1m 30s
635:	learn: 0.0038609	total: 2m 38s	remaining: 1m 30s
636:	learn: 0.0038609	total: 2m 38s	remaining: 1m 30s
637:	learn: 0.0038609	total: 2m 38s	remaining: 1m 30s
638:	learn: 0.0038609	total: 2m 38s	remaining: 1m 29s
639:	learn: 0.0038609	total: 2m 39s	remaining: 1m 29s
640:	learn: 0.0038609	total: 2m 39s	remaining: 1m 29s
641:	learn: 0.0038609	total: 2m 39s	remaining: 1m 28s
642:	learn: 0.0038609	total: 2m 39s	remaining: 1m 28s
643:	learn: 0.0038609	total: 2m 39s	remaining: 1m 28s
644:	learn: 0.0038609	total: 2m 40s	remaining: 1m 28s
645:	learn: 0.0038609	total: 2m 40s	remaining: 1m 27s
646:	learn: 0.0038609	total: 2m 40s	remaining: 1m 27s
647:	learn: 0.0038609	total: 2m 40s	remaining: 1m 27s
648:	learn: 0.0038609	total: 2m 40s	remaining: 1m 27s
649:	learn: 0.0038609	total: 2m 41s	remaining: 1m 26s
650:	learn: 0.0038608	total: 2m 41s	remaining: 1m 26s
651:	learn: 0.0038609	total: 2m 41s	remaining: 1m 26s
652:	learn: 0.0038608	total: 2m 41s	remaining: 1m 25s
653:	learn: 0.0038608	total: 2m 41s	remaining: 1m 25s
654:	learn: 0.0038608	total: 2m 42s	remaining: 1m 25s
655:	learn: 0.0038608	total: 2m 42s	remaining: 1m 25s
656:	learn: 0.0038608	total: 2m 42s	remaining: 1m 24s
657:	learn: 0.0038608	total: 2m 42s	remaining: 1m 24s
658:	learn: 0.0038608	total: 2m 42s	remaining: 1m 24s
659:	learn: 0.0038608	total: 2m 43s	remaining: 1m 23s
660:	learn: 0.0038608	total: 2m 43s	remaining: 1m 23s
661:	learn: 0.0038608	total: 2m 43s	remaining: 1m 23s
662:	learn: 0.0038608	total: 2m 43s	remaining: 1m 23s
663:	learn: 0.0038608	total: 2m 43s	remaining: 1m 22s
664:	learn: 0.0038608	total: 2m 44s	remaining: 1m 22s
665:	learn: 0.0038608	total: 2m 44s	remaining: 1m 22s
666:	learn: 0.0038608	total: 2m 44s	remaining: 1m 22s
667:	learn: 0.0038608	total: 2m 44s	remaining: 1m 21s
668:	learn: 0.0038608	total: 2m 44s	remaining: 1m 21s
669:	learn: 0.0038608	total: 2m 44s	remaining: 1m 21s
670:	learn: 0.0038608	total: 2m 45s	remaining: 1m 20s
671:	learn: 0.0038608	total: 2m 45s	remaining: 1m 20s
672:	learn: 0.0038607	total: 2m 45s	remaining: 1m 20s
673:	learn: 0.0038607	total: 2m 45s	remaining: 1m 20s
674:	learn: 0.0038607	total: 2m 45s	remaining: 1m 19s
675:	learn: 0.0038607	total: 2m 46s	remaining: 1m 19s
676:	learn: 0.0038607	total: 2m 46s	remaining: 1m 19s
677:	learn: 0.0038607	total: 2m 46s	remaining: 1m 19s
678:	learn: 0.0038607	total: 2m 46s	remaining: 1m 18s
```

```
679:	learn: 0.0038607	total: 2m 47s	remaining: 1m 18s
680:	learn: 0.0038607	total: 2m 47s	remaining: 1m 18s
681:	learn: 0.0038607	total: 2m 47s	remaining: 1m 18s
682:	learn: 0.0038607	total: 2m 48s	remaining: 1m 18s
683:	learn: 0.0038607	total: 2m 48s	remaining: 1m 17s
684:	learn: 0.0038607	total: 2m 48s	remaining: 1m 17s
685:	learn: 0.0038607	total: 2m 49s	remaining: 1m 17s
686:	learn: 0.0038607	total: 2m 49s	remaining: 1m 17s
687:	learn: 0.0038607	total: 2m 49s	remaining: 1m 16s
688:	learn: 0.0038607	total: 2m 49s	remaining: 1m 16s
689:	learn: 0.0038607	total: 2m 50s	remaining: 1m 16s
690:	learn: 0.0038607	total: 2m 50s	remaining: 1m 16s
691:	learn: 0.0038607	total: 2m 50s	remaining: 1m 15s
692:	learn: 0.0038606	total: 2m 50s	remaining: 1m 15s
693:	learn: 0.0038606	total: 2m 50s	remaining: 1m 15s
694:	learn: 0.0038607	total: 2m 51s	remaining: 1m 15s
695:	learn: 0.0038607	total: 2m 51s	remaining: 1m 14s
696:	learn: 0.0038606	total: 2m 51s	remaining: 1m 14s
697:	learn: 0.0038606	total: 2m 51s	remaining: 1m 14s
698:	learn: 0.0038606	total: 2m 51s	remaining: 1m 14s
699:	learn: 0.0038606	total: 2m 52s	remaining: 1m 13s
700:	learn: 0.0038606	total: 2m 52s	remaining: 1m 13s
701:	learn: 0.0038606	total: 2m 52s	remaining: 1m 13s
702:	learn: 0.0038606	total: 2m 52s	remaining: 1m 12s
703:	learn: 0.0038606	total: 2m 52s	remaining: 1m 12s
704:	learn: 0.0038606	total: 2m 53s	remaining: 1m 12s
705:	learn: 0.0038606	total: 2m 53s	remaining: 1m 12s
706:	learn: 0.0038606	total: 2m 53s	remaining: 1m 11s
707:	learn: 0.0038606	total: 2m 53s	remaining: 1m 11s
708:	learn: 0.0038606	total: 2m 53s	remaining: 1m 11s
709:	learn: 0.0038605	total: 2m 54s	remaining: 1m 11s
710:	learn: 0.0038606	total: 2m 54s	remaining: 1m 10s
711:	learn: 0.0038605	total: 2m 54s	remaining: 1m 10s
712:	learn: 0.0038605	total: 2m 54s	remaining: 1m 10s
713:	learn: 0.0038605	total: 2m 55s	remaining: 1m 10s
714:	learn: 0.0038605	total: 2m 55s	remaining: 1m 9s
715:	learn: 0.0038605	total: 2m 55s	remaining: 1m 9s
716:	learn: 0.0038605	total: 2m 55s	remaining: 1m 9s
717:	learn: 0.0038605	total: 2m 55s	remaining: 1m 9s
718:	learn: 0.0038605	total: 2m 55s	remaining: 1m 8s
719:	learn: 0.0038605	total: 2m 56s	remaining: 1m 8s
720:	learn: 0.0038605	total: 2m 56s	remaining: 1m 8s
721:	learn: 0.0038605	total: 2m 56s	remaining: 1m 7s
722:	learn: 0.0038605	total: 2m 56s	remaining: 1m 7s
723:	learn: 0.0038604	total: 2m 56s	remaining: 1m 7s
724:	learn: 0.0038604	total: 2m 57s	remaining: 1m 7s
725:	learn: 0.0038604	total: 2m 57s	remaining: 1m 6s
726:	learn: 0.0038604	total: 2m 57s	remaining: 1m 6s
727:	learn: 0.0038604	total: 2m 57s	remaining: 1m 6s
728:	learn: 0.0038604	total: 2m 57s	remaining: 1m 6s
```

```
729:	learn: 0.0038604	total: 2m 58s	remaining: 1m 5s
730:	learn: 0.0038604	total: 2m 58s	remaining: 1m 5s
731:	learn: 0.0038604	total: 2m 58s	remaining: 1m 5s
732:	learn: 0.0038604	total: 2m 58s	remaining: 1m 5s
733:	learn: 0.0038604	total: 2m 58s	remaining: 1m 4s
734:	learn: 0.0038604	total: 2m 59s	remaining: 1m 4s
735:	learn: 0.0038604	total: 2m 59s	remaining: 1m 4s
736:	learn: 0.0038604	total: 2m 59s	remaining: 1m 4s
737:	learn: 0.0038604	total: 2m 59s	remaining: 1m 3s
738:	learn: 0.0038434	total: 2m 59s	remaining: 1m 3s
739:	learn: 0.0038434	total: 3m	remaining: 1m 3s
740:	learn: 0.0038434	total: 3m	remaining: 1m 3s
741:	learn: 0.0038434	total: 3m	remaining: 1m 2s
742:	learn: 0.0038434	total: 3m	remaining: 1m 2s
743:	learn: 0.0038433	total: 3m	remaining: 1m 2s
744:	learn: 0.0038433	total: 3m 1s	remaining: 1m 1s
745:	learn: 0.0038433	total: 3m 1s	remaining: 1m 1s
746:	learn: 0.0038433	total: 3m 1s	remaining: 1m 1s
747:	learn: 0.0038433	total: 3m 1s	remaining: 1m 1s
748:	learn: 0.0038433	total: 3m 1s	remaining: 1m
749:	learn: 0.0038433	total: 3m 1s	remaining: 1m
750:	learn: 0.0038433	total: 3m 2s	remaining: 1m
751:	learn: 0.0038433	total: 3m 2s	remaining: 1m
752:	learn: 0.0038433	total: 3m 2s	remaining: 59.9s
753:	learn: 0.0038433	total: 3m 2s	remaining: 59.6s
754:	learn: 0.0038433	total: 3m 3s	remaining: 59.4s
755:	learn: 0.0038433	total: 3m 3s	remaining: 59.2s
756:	learn: 0.0038433	total: 3m 3s	remaining: 59s
757:	learn: 0.0038433	total: 3m 4s	remaining: 58.8s
758:	learn: 0.0038433	total: 3m 4s	remaining: 58.5s
759:	learn: 0.0038433	total: 3m 4s	remaining: 58.3s
760:	learn: 0.0038433	total: 3m 5s	remaining: 58.1s
761:	learn: 0.0038433	total: 3m 5s	remaining: 57.9s
762:	learn: 0.0038433	total: 3m 5s	remaining: 57.7s
763:	learn: 0.0038433	total: 3m 5s	remaining: 57.4s
764:	learn: 0.0038432	total: 3m 6s	remaining: 57.1s
765:	learn: 0.0038432	total: 3m 6s	remaining: 56.9s
766:	learn: 0.0038432	total: 3m 6s	remaining: 56.6s
767:	learn: 0.0038432	total: 3m 6s	remaining: 56.4s
768:	learn: 0.0038432	total: 3m 6s	remaining: 56.1s
769:	learn: 0.0038432	total: 3m 6s	remaining: 55.8s
770:	learn: 0.0038432	total: 3m 7s	remaining: 55.6s
771:	learn: 0.0038432	total: 3m 7s	remaining: 55.3s
772:	learn: 0.0038432	total: 3m 7s	remaining: 55.1s
773:	learn: 0.0038432	total: 3m 7s	remaining: 54.8s
774:	learn: 0.0038432	total: 3m 7s	remaining: 54.5s
775:	learn: 0.0038432	total: 3m 8s	remaining: 54.3s

776:	learn: 0.0038432	total: 3m 8s	remaining: 54s
777:	learn: 0.0038432	total: 3m 8s	remaining: 53.8s
```

```
778:	learn: 0.0038432	total: 3m 8s	remaining: 53.5s
779:	learn: 0.0038432	total: 3m 8s	remaining: 53.2s
780:	learn: 0.0038432	total: 3m 8s	remaining: 53s
781:	learn: 0.0038432	total: 3m 9s	remaining: 52.7s
782:	learn: 0.0038432	total: 3m 9s	remaining: 52.5s
783:	learn: 0.0038432	total: 3m 9s	remaining: 52.2s
784:	learn: 0.0038432	total: 3m 9s	remaining: 51.9s
785:	learn: 0.0038432	total: 3m 9s	remaining: 51.7s
786:	learn: 0.0038432	total: 3m 10s	remaining: 51.4s
787:	learn: 0.0038432	total: 3m 10s	remaining: 51.2s
788:	learn: 0.0038432	total: 3m 10s	remaining: 50.9s
789:	learn: 0.0038432	total: 3m 10s	remaining: 50.7s
790:	learn: 0.0038432	total: 3m 10s	remaining: 50.4s
791:	learn: 0.0038432	total: 3m 11s	remaining: 50.2s
792:	learn: 0.0038432	total: 3m 11s	remaining: 49.9s
793:	learn: 0.0038432	total: 3m 11s	remaining: 49.6s
794:	learn: 0.0038432	total: 3m 11s	remaining: 49.4s
795:	learn: 0.0038431	total: 3m 11s	remaining: 49.1s
796:	learn: 0.0038431	total: 3m 11s	remaining: 48.9s
797:	learn: 0.0038431	total: 3m 12s	remaining: 48.6s
798:	learn: 0.0038431	total: 3m 12s	remaining: 48.4s
799:	learn: 0.0038431	total: 3m 12s	remaining: 48.1s
800:	learn: 0.0038431	total: 3m 12s	remaining: 47.9s
801:	learn: 0.0038431	total: 3m 12s	remaining: 47.6s
802:	learn: 0.0038431	total: 3m 13s	remaining: 47.4s
803:	learn: 0.0038431	total: 3m 13s	remaining: 47.1s
804:	learn: 0.0038431	total: 3m 13s	remaining: 46.9s
805:	learn: 0.0038431	total: 3m 13s	remaining: 46.6s
806:	learn: 0.0038431	total: 3m 13s	remaining: 46.3s
807:	learn: 0.0038431	total: 3m 13s	remaining: 46.1s
808:	learn: 0.0038431	total: 3m 14s	remaining: 45.8s
809:	learn: 0.0038431	total: 3m 14s	remaining: 45.6s
810:	learn: 0.0038431	total: 3m 14s	remaining: 45.3s
811:	learn: 0.0038431	total: 3m 14s	remaining: 45.1s
812:	learn: 0.0038431	total: 3m 14s	remaining: 44.8s
813:	learn: 0.0038431	total: 3m 15s	remaining: 44.6s
814:	learn: 0.0038431	total: 3m 15s	remaining: 44.3s
815:	learn: 0.0038431	total: 3m 15s	remaining: 44.1s
816:	learn: 0.0038431	total: 3m 15s	remaining: 43.8s
817:	learn: 0.0038431	total: 3m 15s	remaining: 43.6s
818:	learn: 0.0038431	total: 3m 15s	remaining: 43.3s
819:	learn: 0.0038431	total: 3m 16s	remaining: 43.1s
820:	learn: 0.0038431	total: 3m 16s	remaining: 42.8s
821:	learn: 0.0038431	total: 3m 16s	remaining: 42.6s
822:	learn: 0.0038431	total: 3m 16s	remaining: 42.3s
823:	learn: 0.0038431	total: 3m 16s	remaining: 42.1s
824:	learn: 0.0038431	total: 3m 17s	remaining: 41.8s
825:	learn: 0.0038431	total: 3m 17s	remaining: 41.6s
826:	learn: 0.0038431	total: 3m 17s	remaining: 41.3s
827:	learn: 0.0038431	total: 3m 17s	remaining: 41.1s
```

```
828:	learn: 0.0038431	total: 3m 17s	remaining: 40.8s
829:	learn: 0.0038431	total: 3m 18s	remaining: 40.6s
830:	learn: 0.0038431	total: 3m 18s	remaining: 40.3s
831:	learn: 0.0038431	total: 3m 18s	remaining: 40.1s
832:	learn: 0.0038431	total: 3m 18s	remaining: 39.8s
833:	learn: 0.0038430	total: 3m 18s	remaining: 39.6s
834:	learn: 0.0038430	total: 3m 19s	remaining: 39.4s
835:	learn: 0.0038430	total: 3m 19s	remaining: 39.1s
836:	learn: 0.0038430	total: 3m 19s	remaining: 38.9s
837:	learn: 0.0038430	total: 3m 20s	remaining: 38.7s
838:	learn: 0.0038430	total: 3m 20s	remaining: 38.5s
839:	learn: 0.0038430	total: 3m 20s	remaining: 38.2s
840:	learn: 0.0038430	total: 3m 21s	remaining: 38s
841:	learn: 0.0038430	total: 3m 21s	remaining: 37.8s
842:	learn: 0.0038430	total: 3m 21s	remaining: 37.6s
843:	learn: 0.0038430	total: 3m 21s	remaining: 37.3s
844:	learn: 0.0038430	total: 3m 22s	remaining: 37.1s
845:	learn: 0.0038430	total: 3m 22s	remaining: 36.8s
846:	learn: 0.0038430	total: 3m 22s	remaining: 36.6s
847:	learn: 0.0038430	total: 3m 22s	remaining: 36.3s
848:	learn: 0.0038430	total: 3m 22s	remaining: 36.1s
849:	learn: 0.0038430	total: 3m 23s	remaining: 35.8s
850:	learn: 0.0038430	total: 3m 23s	remaining: 35.6s
851:	learn: 0.0038430	total: 3m 23s	remaining: 35.3s
852:	learn: 0.0038430	total: 3m 23s	remaining: 35.1s
853:	learn: 0.0038430	total: 3m 23s	remaining: 34.8s
854:	learn: 0.0038430	total: 3m 23s	remaining: 34.6s
855:	learn: 0.0038430	total: 3m 24s	remaining: 34.3s
856:	learn: 0.0038430	total: 3m 24s	remaining: 34.1s
857:	learn: 0.0038430	total: 3m 24s	remaining: 33.8s
858:	learn: 0.0038430	total: 3m 24s	remaining: 33.6s
859:	learn: 0.0038430	total: 3m 24s	remaining: 33.4s
860:	learn: 0.0038430	total: 3m 25s	remaining: 33.1s
861:	learn: 0.0038430	total: 3m 25s	remaining: 32.9s
862:	learn: 0.0038430	total: 3m 25s	remaining: 32.6s
863:	learn: 0.0038430	total: 3m 25s	remaining: 32.4s
864:	learn: 0.0038430	total: 3m 25s	remaining: 32.1s
865:	learn: 0.0038430	total: 3m 25s	remaining: 31.9s
866:	learn: 0.0038430	total: 3m 26s	remaining: 31.6s
867:	learn: 0.0038430	total: 3m 26s	remaining: 31.4s
868:	learn: 0.0038430	total: 3m 26s	remaining: 31.1s
869:	learn: 0.0038430	total: 3m 26s	remaining: 30.9s
870:	learn: 0.0038430	total: 3m 26s	remaining: 30.6s
871:	learn: 0.0038430	total: 3m 27s	remaining: 30.4s
872:	learn: 0.0038430	total: 3m 27s	remaining: 30.2s
873:	learn: 0.0038430	total: 3m 27s	remaining: 29.9s
874:	learn: 0.0038430	total: 3m 27s	remaining: 29.7s
875:	learn: 0.0038430	total: 3m 27s	remaining: 29.4s
876:	learn: 0.0038430	total: 3m 27s	remaining: 29.2s
877:	learn: 0.0038119	total: 3m 28s	remaining: 28.9s
```

```
878:	learn: 0.0038118	total: 3m 28s	remaining: 28.7s
879:	learn: 0.0037920	total: 3m 28s	remaining: 28.5s
880:	learn: 0.0037615	total: 3m 29s	remaining: 28.2s
881:	learn: 0.0037299	total: 3m 29s	remaining: 28s
882:	learn: 0.0037299	total: 3m 29s	remaining: 27.8s
883:	learn: 0.0037169	total: 3m 29s	remaining: 27.5s
884:	learn: 0.0037169	total: 3m 29s	remaining: 27.3s
885:	learn: 0.0037169	total: 3m 30s	remaining: 27s
886:	learn: 0.0037169	total: 3m 30s	remaining: 26.8s
887:	learn: 0.0037169	total: 3m 30s	remaining: 26.5s
888:	learn: 0.0037169	total: 3m 30s	remaining: 26.3s
889:	learn: 0.0037169	total: 3m 30s	remaining: 26.1s
890:	learn: 0.0037169	total: 3m 30s	remaining: 25.8s
891:	learn: 0.0037169	total: 3m 31s	remaining: 25.6s
892:	learn: 0.0037169	total: 3m 31s	remaining: 25.3s
893:	learn: 0.0037169	total: 3m 31s	remaining: 25.1s
894:	learn: 0.0037168	total: 3m 31s	remaining: 24.8s
895:	learn: 0.0037168	total: 3m 31s	remaining: 24.6s
896:	learn: 0.0037168	total: 3m 32s	remaining: 24.3s
897:	learn: 0.0037168	total: 3m 32s	remaining: 24.1s
898:	learn: 0.0037168	total: 3m 32s	remaining: 23.9s
899:	learn: 0.0037168	total: 3m 32s	remaining: 23.6s
900:	learn: 0.0037168	total: 3m 32s	remaining: 23.4s
901:	learn: 0.0037168	total: 3m 32s	remaining: 23.1s
902:	learn: 0.0037168	total: 3m 33s	remaining: 22.9s
903:	learn: 0.0037168	total: 3m 33s	remaining: 22.6s
904:	learn: 0.0037168	total: 3m 33s	remaining: 22.4s
905:	learn: 0.0037168	total: 3m 33s	remaining: 22.2s
906:	learn: 0.0037168	total: 3m 33s	remaining: 21.9s
907:	learn: 0.0037167	total: 3m 33s	remaining: 21.7s
908:	learn: 0.0037167	total: 3m 34s	remaining: 21.4s
909:	learn: 0.0037167	total: 3m 34s	remaining: 21.2s
910:	learn: 0.0037167	total: 3m 34s	remaining: 21s
911:	learn: 0.0037167	total: 3m 34s	remaining: 20.7s
912:	learn: 0.0037167	total: 3m 34s	remaining: 20.5s
913:	learn: 0.0037167	total: 3m 35s	remaining: 20.3s
914:	learn: 0.0037167	total: 3m 35s	remaining: 20s
915:	learn: 0.0037167	total: 3m 35s	remaining: 19.8s
916:	learn: 0.0037167	total: 3m 36s	remaining: 19.6s
917:	learn: 0.0037167	total: 3m 36s	remaining: 19.3s
918:	learn: 0.0037167	total: 3m 36s	remaining: 19.1s
919:	learn: 0.0037167	total: 3m 37s	remaining: 18.9s
920:	learn: 0.0037167	total: 3m 37s	remaining: 18.6s
921:	learn: 0.0037167	total: 3m 37s	remaining: 18.4s
922:	learn: 0.0037167	total: 3m 37s	remaining: 18.2s
923:	learn: 0.0037167	total: 3m 38s	remaining: 17.9s
924:	learn: 0.0037167	total: 3m 38s	remaining: 17.7s
925:	learn: 0.0037167	total: 3m 38s	remaining: 17.5s
926:	learn: 0.0037167	total: 3m 38s	remaining: 17.2s
927:	learn: 0.0037167	total: 3m 38s	remaining: 17s
```

```
928:	learn: 0.0037167	total: 3m 38s	remaining: 16.7s
929:	learn: 0.0037167	total: 3m 39s	remaining: 16.5s
930:	learn: 0.0037167	total: 3m 39s	remaining: 16.3s
931:	learn: 0.0037167	total: 3m 39s	remaining: 16s

932:	learn: 0.0037166	total: 3m 39s	remaining: 15.8s
933:	learn: 0.0037166	total: 3m 39s	remaining: 15.5s
934:	learn: 0.0037166	total: 3m 39s	remaining: 15.3s
935:	learn: 0.0037166	total: 3m 40s	remaining: 15.1s
936:	learn: 0.0037166	total: 3m 40s	remaining: 14.8s
937:	learn: 0.0037166	total: 3m 40s	remaining: 14.6s
938:	learn: 0.0037166	total: 3m 40s	remaining: 14.3s
939:	learn: 0.0037166	total: 3m 40s	remaining: 14.1s
940:	learn: 0.0037166	total: 3m 41s	remaining: 13.9s
941:	learn: 0.0037166	total: 3m 41s	remaining: 13.6s
942:	learn: 0.0037166	total: 3m 41s	remaining: 13.4s
943:	learn: 0.0037166	total: 3m 41s	remaining: 13.1s
944:	learn: 0.0037166	total: 3m 41s	remaining: 12.9s
945:	learn: 0.0037166	total: 3m 41s	remaining: 12.7s
946:	learn: 0.0037166	total: 3m 42s	remaining: 12.4s
947:	learn: 0.0037166	total: 3m 42s	remaining: 12.2s
948:	learn: 0.0037166	total: 3m 42s	remaining: 12s
949:	learn: 0.0037166	total: 3m 42s	remaining: 11.7s
950:	learn: 0.0037166	total: 3m 42s	remaining: 11.5s
951:	learn: 0.0037166	total: 3m 42s	remaining: 11.2s
952:	learn: 0.0037166	total: 3m 43s	remaining: 11s
953:	learn: 0.0037166	total: 3m 43s	remaining: 10.8s
954:	learn: 0.0037166	total: 3m 43s	remaining: 10.5s
955:	learn: 0.0037166	total: 3m 43s	remaining: 10.3s
956:	learn: 0.0037166	total: 3m 43s	remaining: 10.1s
957:	learn: 0.0037166	total: 3m 43s	remaining: 9.82s
958:	learn: 0.0037166	total: 3m 44s	remaining: 9.58s
959:	learn: 0.0037166	total: 3m 44s	remaining: 9.35s
960:	learn: 0.0037166	total: 3m 44s	remaining: 9.11s
961:	learn: 0.0037025	total: 3m 44s	remaining: 8.88s
962:	learn: 0.0036806	total: 3m 45s	remaining: 8.64s
963:	learn: 0.0036806	total: 3m 45s	remaining: 8.41s
964:	learn: 0.0036806	total: 3m 45s	remaining: 8.17s
965:	learn: 0.0036806	total: 3m 45s	remaining: 7.94s
966:	learn: 0.0036805	total: 3m 45s	remaining: 7.7s
967:	learn: 0.0036805	total: 3m 45s	remaining: 7.47s
968:	learn: 0.0036805	total: 3m 46s	remaining: 7.23s
969:	learn: 0.0036806	total: 3m 46s	remaining: 7s
970:	learn: 0.0036805	total: 3m 46s	remaining: 6.76s
971:	learn: 0.0036805	total: 3m 46s	remaining: 6.53s
972:	learn: 0.0036805	total: 3m 46s	remaining: 6.29s
973:	learn: 0.0036805	total: 3m 46s	remaining: 6.06s
974:	learn: 0.0036805	total: 3m 47s	remaining: 5.82s
975:	learn: 0.0036805	total: 3m 47s	remaining: 5.59s
976:	learn: 0.0036805	total: 3m 47s	remaining: 5.35s
```

```
977:	learn: 0.0036805	total: 3m 47s	remaining: 5.12s
978:	learn: 0.0036805	total: 3m 47s	remaining: 4.89s
979:	learn: 0.0036805	total: 3m 47s	remaining: 4.65s
980:	learn: 0.0036805	total: 3m 48s	remaining: 4.42s
981:	learn: 0.0036805	total: 3m 48s	remaining: 4.18s
982:	learn: 0.0036805	total: 3m 48s	remaining: 3.95s
983:	learn: 0.0036805	total: 3m 48s	remaining: 3.72s
984:	learn: 0.0036805	total: 3m 48s	remaining: 3.48s
985:	learn: 0.0036805	total: 3m 49s	remaining: 3.25s
986:	learn: 0.0036805	total: 3m 49s	remaining: 3.02s
987:	learn: 0.0036805	total: 3m 49s	remaining: 2.79s
988:	learn: 0.0036805	total: 3m 49s	remaining: 2.55s
989:	learn: 0.0036805	total: 3m 49s	remaining: 2.32s
990:	learn: 0.0036805	total: 3m 49s	remaining: 2.09s
991:	learn: 0.0036804	total: 3m 50s	remaining: 1.85s
992:	learn: 0.0036804	total: 3m 50s	remaining: 1.62s
993:	learn: 0.0036804	total: 3m 50s	remaining: 1.39s
994:	learn: 0.0036804	total: 3m 50s	remaining: 1.16s
995:	learn: 0.0036804	total: 3m 50s	remaining: 927ms
996:	learn: 0.0036804	total: 3m 51s	remaining: 696ms
997:	learn: 0.0036804	total: 3m 51s	remaining: 464ms
998:	learn: 0.0036804	total: 3m 51s	remaining: 232ms
999:	learn: 0.0036804	total: 3m 52s	remaining: 0us

<catboost.core.CatBoostClassifier at 0x67a159d1c0>
```

```
cbc_pred = cbc.predict(x_test)
```

```
print(cbc_pred)
```

```
[0. 1. 0. ... 0. 0. 1.]
```

```
cbc_pred=pd.DataFrame(cbc_pred)
print(cbc_pred)
```

```
          0
0       0.0
1       1.0
2       0.0
3       1.0
4       0.0
...     ...
141622  1.0
141623  0.0
141624  0.0
141625  0.0
141626  1.0

[141627 rows x 1 columns]
```

```
cbc_pred=pd.DataFrame(cbc_pred)
print(cbc_pred.head())
```

```
y_test_cbc=np.array(y_test)
y_test_cbc=pd.DataFrame(y_test)
print(y_test_cbc.head())
```

```
     0
0  0.0
1  1.0
2  0.0
3  1.0
4  0.0
   Actual
0     0.0
1     1.0
2     0.0
3     1.0
4     0.0
```

```
y_test_cbc.columns=["Actual"]
cbc_pred.columns=["Predection"]

cbc_ps=pd.DataFrame()
cbc_ps["pred"]=cbc_pred.Predection

cbc_ps["Actual"]=y_test_cbc.Actual
print(cbc_ps.head())
```

```
   pred  Actual
0   0.0     0.0
1   1.0     1.0
2   0.0     0.0
3   1.0     1.0
4   0.0     0.0
```

2.Analysing the predected Cat boosting model

```
print(cbc_ps["pred"].value_counts())
```

```
1.0    71165
0.0    70462
Name: pred, dtype: int64
```

```
print(cbc_ps["Actual"].value_counts())
```

```
1.0    71091
0.0    70536
Name: Actual, dtype: int64
```

3.Mean square error

```
Mean_Sq_Error1=((cbc_ps["pred"])-(cbc_ps["Actual"]))
Mean_Sq_Error2=((Mean_Sq_Error1)**2)
Mean_Sq_Error_cbc=Mean_Sq_Error2.sum()
print("Mean_Sq_Error for Cat boosting model classifier
=",Mean_Sq_Error_cbc)
```

```
Mean_Sq_Error for Cat boosting model classifier = 74.0
```

4.Model score

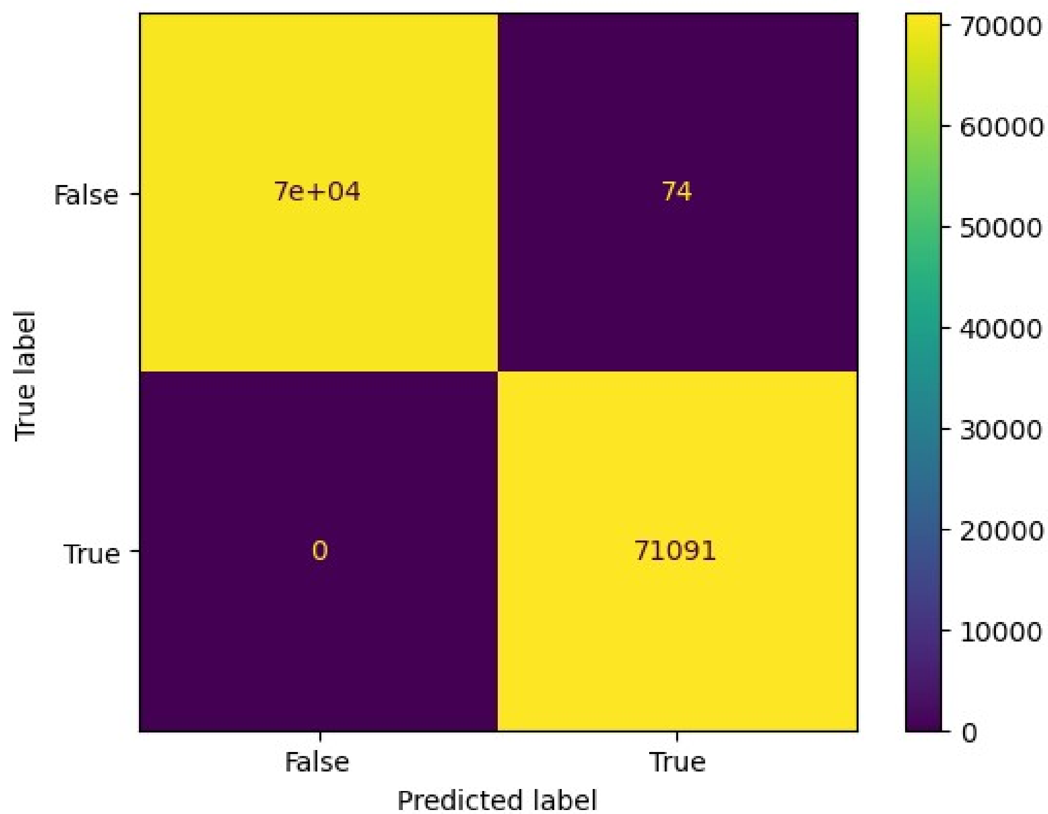
```
scoreOfModel1_cbc = cbc.score(x_test,y_test)
print(scoreOfModel1_cbc)
```

```
0.9994775007590361
```

5.Confusion_Matrix

```
y_test_cbc=cbc_ps["Actual"]
y_pred_cbc=cbc_ps["pred"]
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
def display_confusion_matrix(y_test_cbc,y_pred_cbc):
    matrix = metrics.confusion_matrix(y_test_cbc,y_pred_cbc)
    matrixDisplay = metrics.ConfusionMatrixDisplay(confusion_matrix = 
matrix, display_labels = [False, True])
    matrixDisplay.plot()
    plt.show()
display_confusion_matrix(y_test_cbc,y_pred_cbc)
```




```
print(classification_report(y_test_cbc,y_pred_cbc))
```

```
              precision    recall  f1-score   support
```

```
         0.0       1.00      1.00      1.00     70536
         1.0       1.00      1.00      1.00     71091

    accuracy                           1.00    141627
   macro avg       1.00      1.00      1.00    141627
weighted avg       1.00      1.00      1.00    141627
```

6.Error Percentage

```
a=(cbc_ps.shape)
b=pd.DataFrame(a)
c=b.head(1)
c=np.array(c)
Error_Percentage_cbc=(Mean_Sq_Error_cbc/(np.array(c))*100)
print("Error occured in Cat boosting Clasifier
=",Error_Percentage_cbc,"%")
```

```
Error occured in Cat boosting Clasifier = [[0.05224992]] %
```

7.Result Percentage

```
print("Result Percentage=",(100-Error_Percentage_cbc),"%")
```

```
Result Percentage= [[99.94775008]] %
```

8.Conclusion

Well the results of Cat boosting clasifier model is quite good of 99.94775008% accurate, but not as good as Random forest classifier model

Over all Conclusion

Random forest 99.98234% accurate

XGB model 99.96822% accurate

KNN clasifier model 99.904679% accurate

ADB boosting model 96.434295% accurate

Cat boosting clasifier model 99.94775008% accurate

Now we can see that Random forest preforms well compared with others