```
# Installing Necessary library for the working.
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings(
    'ignore'
)
from sklearn import set_config
set_config(display = 'diagrams')

df = pd.read_csv("/kaggle/input/uci-credit-card-csv/UCI_Credit_Card.csv")

df.head()
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	\
0	1	20000.0	2	2	1	24	2	2	-1	-1	
1	2	120000.0	2	2	2	26	-1	2	0	0	
2	3	90000.0	2	2	2	34	0	0	0	0	
3	4	50000.0	2	2	1	37	0	0	0	0	
4	5	50000.0	1	2	1	57	-1	0	-1	0	

	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	\
0	...	0.0	0.0	0.0	0.0	689.0	0.0	
1	...	3272.0	3455.0	3261.0	0.0	1000.0	1000.0	
2	...	14331.0	14948.0	15549.0	1518.0	1500.0	1000.0	
3	...	28314.0	28959.0	29547.0	2000.0	2019.0	1200.0	
4	...	20940.0	19146.0	19131.0	2000.0	36681.0	10000.0	

	PAY_AMT4	PAY_AMT5	PAY_AMT6	default.payment.next.month
0	0.0	0.0	0.0	1
1	1000.0	0.0	2000.0	1
2	1000.0	1000.0	5000.0	0
3	1100.0	1069.0	1000.0	0
4	9000.0	689.0	679.0	0

[5 rows x 25 columns]

```
df.describe() # Describition of the Dataset like mean, std and spread of Data.
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	\
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	
std	8660.398374	129747.661567	0.489129	0.790349	0.521970	
min	1.000000	10000.000000	1.000000	0.000000	0.000000	
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	

	AGE	PAY_0	PAY_2	PAY_3	PAY_4	\
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	
mean	35.485500	-0.016700	-0.133767	-0.166200	-0.220667	
std	9.217904	1.123802	1.197186	1.196868	1.169139	
min	21.000000	-2.000000	-2.000000	-2.000000	-2.000000	
25%	28.000000	-1.000000	-1.000000	-1.000000	-1.000000	
50%	34.000000	0.000000	0.000000	0.000000	0.000000	
75%	41.000000	0.000000	0.000000	0.000000	0.000000	
max	79.000000	8.000000	8.000000	8.000000	8.000000	

	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	\
count	...	30000.000000	30000.000000	30000.000000	30000.000000	
mean	...	43262.948967	40311.400967	38871.760400	5663.580500	

```
std    ...   64332.856134   60797.155770   59554.107537   16563.280354
min    ... -170000.000000  -81334.000000 -339603.000000       0.000000
25%    ...    2326.750000    1763.000000    1256.000000    1000.000000
50%    ...   19052.000000   18104.500000   17071.000000    2100.000000
75%    ...   54506.000000   50190.500000   49198.250000    5006.000000
max    ...  891586.000000  927171.000000  961664.000000  873552.000000

           PAY_AMT2      PAY_AMT3       PAY_AMT4       PAY_AMT5  \
count  3.000000e+04   30000.00000   30000.000000   30000.000000
mean   5.921163e+03    5225.68150    4826.076867    4799.387633
std    2.304087e+04   17606.96147   15666.159744   15278.305679
min    0.000000e+00       0.00000       0.000000       0.000000
25%    8.330000e+02     390.00000     296.000000     252.500000
50%    2.009000e+03    1800.00000    1500.000000    1500.000000
75%    5.000000e+03    4505.00000    4013.250000    4031.500000
max    1.684259e+06  896040.00000  621000.000000  426529.000000

            PAY_AMT6  default.payment.next.month
count   30000.000000                30000.000000
mean     5215.502567                    0.221200
std     17777.465775                    0.415062
min         0.000000                    0.000000
25%       117.750000                    0.000000
50%      1500.000000                    0.000000
75%      4000.000000                    0.000000
max    528666.000000                    1.000000

[8 rows x 25 columns]
```

```
df.isnull().sum() # No missing values
```

```
ID                            0
LIMIT_BAL                     0
SEX                           0
EDUCATION                     0
MARRIAGE                      0
AGE                           0
PAY_0                         0
PAY_2                         0
PAY_3                         0
PAY_4                         0
PAY_5                         0
PAY_6                         0
BILL_AMT1                     0
BILL_AMT2                     0
BILL_AMT3                     0
BILL_AMT4                     0
BILL_AMT5                     0
BILL_AMT6                     0
PAY_AMT1                      0
PAY_AMT2                      0
PAY_AMT3                      0
PAY_AMT4                      0
PAY_AMT5                      0
PAY_AMT6                      0
default.payment.next.month    0
dtype: int64
```

```
df.drop_duplicates() # In the Dataset there is no duplicate values
```

```
          ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  PAY_2  PAY_3  \
0          1    20000.0    2          2         1   24      2      2     -1
1          2   120000.0    2          2         2   26     -1      2      0
```

```
2          3    90000.0    2          2         2   34      0      0      0
3          4    50000.0    2          2         1   37      0      0      0
4          5    50000.0    1          2         1   57     -1      0     -1
...      ...        ...  ...        ...       ...  ...    ...    ...    ...
29995  29996   220000.0    1          3         1   39      0      0      0
29996  29997   150000.0    1          3         2   43     -1     -1     -1
29997  29998    30000.0    1          2         2   37      4      3      2
29998  29999    80000.0    1          3         1   41      1     -1      0
29999  30000    50000.0    1          2         1   46      0      0      0

       PAY_4  ...  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  \
0         -1  ...        0.0        0.0        0.0       0.0     689.0
1          0  ...     3272.0     3455.0     3261.0       0.0    1000.0
2          0  ...    14331.0    14948.0    15549.0    1518.0    1500.0
3          0  ...    28314.0    28959.0    29547.0    2000.0    2019.0
4          0  ...    20940.0    19146.0    19131.0    2000.0   36681.0
...      ...  ...        ...        ...        ...       ...       ...
29995      0  ...    88004.0    31237.0    15980.0    8500.0   20000.0
29996     -1  ...     8979.0     5190.0        0.0    1837.0    3526.0
29997     -1  ...    20878.0    20582.0    19357.0       0.0       0.0
29998      0  ...    52774.0    11855.0    48944.0   85900.0    3409.0
29999      0  ...    36535.0    32428.0    15313.0    2078.0    1800.0

       PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6  default.payment.next.month
0           0.0       0.0       0.0       0.0                          1
1        1000.0    1000.0       0.0    2000.0                          1
2        1000.0    1000.0    1000.0    5000.0                          0
3        1200.0    1100.0    1069.0    1000.0                          0
4       10000.0    9000.0     689.0     679.0                          0
...         ...       ...       ...       ...                        ...
29995    5003.0    3047.0    5000.0    1000.0                          0
29996    8998.0     129.0       0.0       0.0                          0
29997   22000.0    4200.0    2000.0    3100.0                          1
29998    1178.0    1926.0   52964.0    1804.0                          1
29999    1430.0    1000.0    1000.0    1000.0                          1

[30000 rows x 25 columns]
```

df.corr()

```
                                ID  LIMIT_BAL       SEX  EDUCATION  \
ID                        1.000000   0.026179  0.018497   0.039177
LIMIT_BAL                 0.026179   1.000000  0.024755  -0.219161
SEX                       0.018497   0.024755  1.000000   0.014232
EDUCATION                 0.039177  -0.219161  0.014232   1.000000
MARRIAGE                 -0.029079  -0.108139 -0.031389  -0.143464
AGE                       0.018678   0.144713 -0.090874   0.175061
PAY_0                    -0.030575  -0.271214 -0.057643   0.105364
PAY_2                    -0.011215  -0.296382 -0.070771   0.121566
PAY_3                    -0.018494  -0.286123 -0.066096   0.114025
PAY_4                    -0.002735  -0.267460 -0.060173   0.108793
PAY_5                    -0.022199  -0.249411 -0.055064   0.097520
PAY_6                    -0.020270  -0.235195 -0.044008   0.082316
BILL_AMT1                 0.019389   0.285430 -0.033642   0.023581
BILL_AMT2                 0.017982   0.278314 -0.031183   0.018749
BILL_AMT3                 0.024354   0.283236 -0.024563   0.013002
BILL_AMT4                 0.040351   0.293988 -0.021880  -0.000451
BILL_AMT5                 0.016705   0.295562 -0.017005  -0.007567
BILL_AMT6                 0.016730   0.290389 -0.016733  -0.009099
PAY_AMT1                  0.009742   0.195236 -0.000242  -0.037456
PAY_AMT2                  0.008406   0.178408 -0.001391  -0.030038
PAY_AMT3                  0.039151   0.210167 -0.008597  -0.039943
PAY_AMT4                  0.007793   0.203242 -0.002229  -0.038218
```

```
PAY_AMT5                     0.000652   0.217202 -0.001667  -0.040358
PAY_AMT6                     0.003000   0.219595 -0.002766  -0.037200
default.payment.next.month  -0.013952  -0.153520 -0.039961   0.028006

                             MARRIAGE       AGE     PAY_0     PAY_2     PAY_3  \
ID                          -0.029079  0.018678 -0.030575 -0.011215 -0.018494
LIMIT_BAL                   -0.108139  0.144713 -0.271214 -0.296382 -0.286123
SEX                         -0.031389 -0.090874 -0.057643 -0.070771 -0.066096
EDUCATION                   -0.143464  0.175061  0.105364  0.121566  0.114025
MARRIAGE                     1.000000 -0.414170  0.019917  0.024199  0.032688
AGE                         -0.414170  1.000000 -0.039447 -0.050148 -0.053048
PAY_0                        0.019917 -0.039447  1.000000  0.672164  0.574245
PAY_2                        0.024199 -0.050148  0.672164  1.000000  0.766552
PAY_3                        0.032688 -0.053048  0.574245  0.766552  1.000000
PAY_4                        0.033122 -0.049722  0.538841  0.662067  0.777359
PAY_5                        0.035629 -0.053826  0.509426  0.622780  0.686775
PAY_6                        0.034345 -0.048773  0.474553  0.575501  0.632684
BILL_AMT1                   -0.023472  0.056239  0.187068  0.234887  0.208473
BILL_AMT2                   -0.021602  0.054283  0.189859  0.235257  0.237295
BILL_AMT3                   -0.024909  0.053710  0.179785  0.224146  0.227494
BILL_AMT4                   -0.023344  0.051353  0.179125  0.222237  0.227202
BILL_AMT5                   -0.025393  0.049345  0.180635  0.221348  0.225145
BILL_AMT6                   -0.021207  0.047613  0.176980  0.219403  0.222327
PAY_AMT1                    -0.005979  0.026147 -0.079269 -0.080701  0.001295
PAY_AMT2                    -0.008093  0.021785 -0.070101 -0.058990 -0.066793
PAY_AMT3                    -0.003541  0.029247 -0.070561 -0.055901 -0.053311
PAY_AMT4                    -0.012659  0.021379 -0.064005 -0.046858 -0.046067
PAY_AMT5                    -0.001205  0.022850 -0.058190 -0.037093 -0.035863
PAY_AMT6                    -0.006641  0.019478 -0.058673 -0.036500 -0.035861
default.payment.next.month  -0.024339  0.013890  0.324794  0.263551  0.235253

                                PAY_4  ...  BILL_AMT4  BILL_AMT5  BILL_AMT6  \
ID                          -0.002735  ...   0.040351   0.016705   0.016730
LIMIT_BAL                   -0.267460  ...   0.293988   0.295562   0.290389
SEX                         -0.060173  ...  -0.021880  -0.017005  -0.016733
EDUCATION                    0.108793  ...  -0.000451  -0.007567  -0.009099
MARRIAGE                     0.033122  ...  -0.023344  -0.025393  -0.021207
AGE                         -0.049722  ...   0.051353   0.049345   0.047613
PAY_0                        0.538841  ...   0.179125   0.180635   0.176980
PAY_2                        0.662067  ...   0.222237   0.221348   0.219403
PAY_3                        0.777359  ...   0.227202   0.225145   0.222327
PAY_4                        1.000000  ...   0.245917   0.242902   0.239154
PAY_5                        0.819835  ...   0.271915   0.269783   0.262509
PAY_6                        0.716449  ...   0.266356   0.290894   0.285091
BILL_AMT1                    0.202812  ...   0.860272   0.829779   0.802650
BILL_AMT2                    0.225816  ...   0.892482   0.859778   0.831594
BILL_AMT3                    0.244983  ...   0.923969   0.883910   0.853320
BILL_AMT4                    0.245917  ...   1.000000   0.940134   0.900941
BILL_AMT5                    0.242902  ...   0.940134   1.000000   0.946197
BILL_AMT6                    0.239154  ...   0.900941   0.946197   1.000000
PAY_AMT1                    -0.009362  ...   0.233012   0.217031   0.199965
PAY_AMT2                    -0.001944  ...   0.207564   0.181246   0.172663
PAY_AMT3                    -0.069235  ...   0.300023   0.252305   0.233770
PAY_AMT4                    -0.043461  ...   0.130191   0.293118   0.250237
PAY_AMT5                    -0.033590  ...   0.160433   0.141574   0.307729
PAY_AMT6                    -0.026565  ...   0.177637   0.164184   0.115494
default.payment.next.month   0.216614  ...  -0.010156  -0.006760  -0.005372

                             PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  \
ID                           0.009742  0.008406  0.039151  0.007793  0.000652
LIMIT_BAL                    0.195236  0.178408  0.210167  0.203242  0.217202
SEX                         -0.000242 -0.001391 -0.008597 -0.002229 -0.001667
```

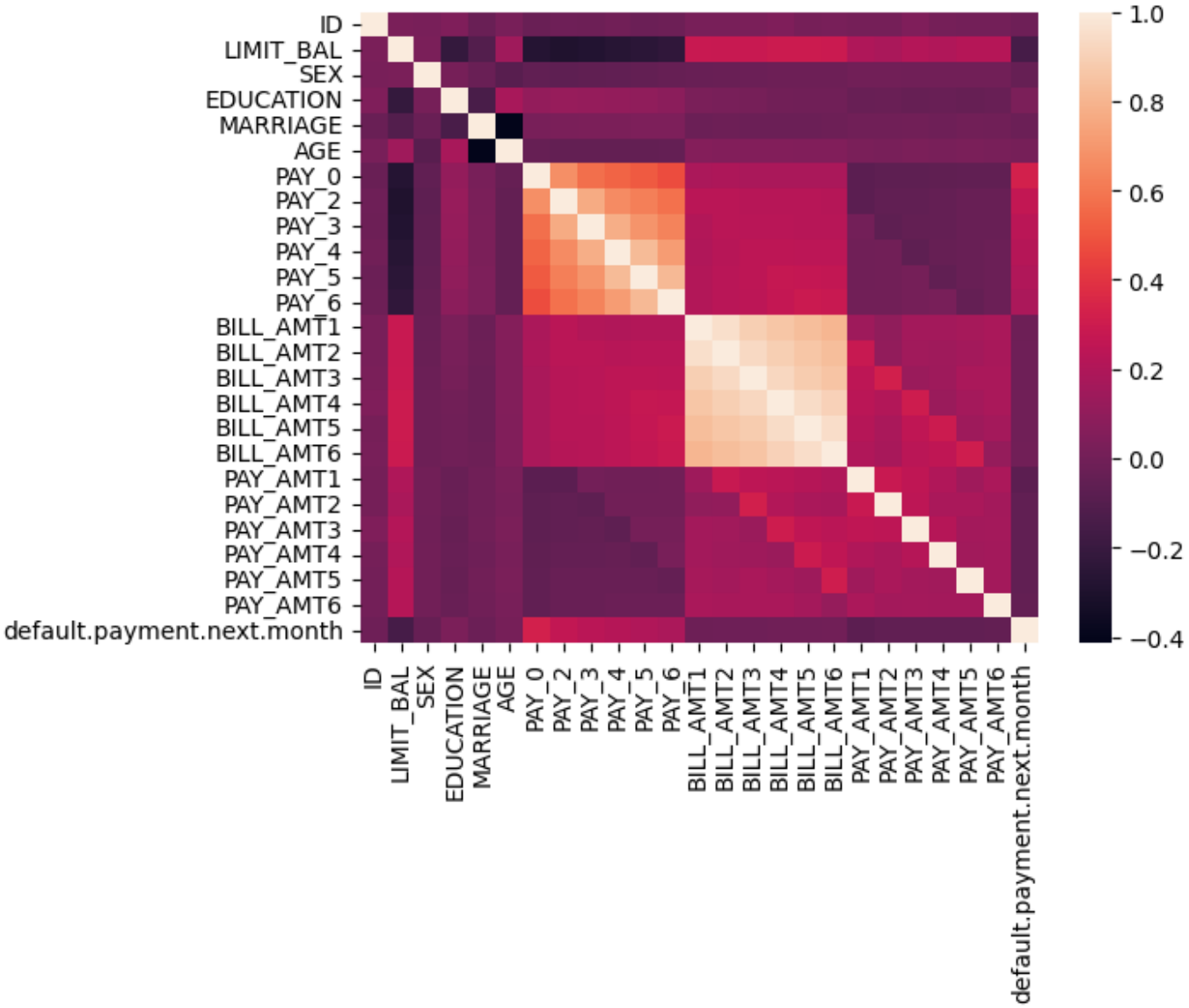
```
EDUCATION                  -0.037456 -0.030038 -0.039943 -0.038218 -0.040358
MARRIAGE                   -0.005979 -0.008093 -0.003541 -0.012659 -0.001205
AGE                         0.026147  0.021785  0.029247  0.021379  0.022850
PAY_0                      -0.079269 -0.070101 -0.070561 -0.064005 -0.058190
PAY_2                      -0.080701 -0.058990 -0.055901 -0.046858 -0.037093
PAY_3                       0.001295 -0.066793 -0.053311 -0.046067 -0.035863
PAY_4                      -0.009362 -0.001944 -0.069235 -0.043461 -0.033590
PAY_5                      -0.006089 -0.003191  0.009062 -0.058299 -0.033337
PAY_6                      -0.001496 -0.005223  0.005834  0.019018 -0.046434
BILL_AMT1                   0.140277  0.099355  0.156887  0.158303  0.167026
BILL_AMT2                   0.280365  0.100851  0.150718  0.147398  0.157957
BILL_AMT3                   0.244335  0.316936  0.130011  0.143405  0.179712
BILL_AMT4                   0.233012  0.207564  0.300023  0.130191  0.160433
BILL_AMT5                   0.217031  0.181246  0.252305  0.293118  0.141574
BILL_AMT6                   0.199965  0.172663  0.233770  0.250237  0.307729
PAY_AMT1                    1.000000  0.285576  0.252191  0.199558  0.148459
PAY_AMT2                    0.285576  1.000000  0.244770  0.180107  0.180908
PAY_AMT3                    0.252191  0.244770  1.000000  0.216325  0.159214
PAY_AMT4                    0.199558  0.180107  0.216325  1.000000  0.151830
PAY_AMT5                    0.148459  0.180908  0.159214  0.151830  1.000000
PAY_AMT6                    0.185735  0.157634  0.162740  0.157834  0.154896
default.payment.next.month -0.072929 -0.058579 -0.056250 -0.056827 -0.055124

                            PAY_AMT6  default.payment.next.month
ID                          0.003000                   -0.013952
LIMIT_BAL                   0.219595                   -0.153520
SEX                        -0.002766                   -0.039961
EDUCATION                  -0.037200                    0.028006
MARRIAGE                   -0.006641                   -0.024339
AGE                         0.019478                    0.013890
PAY_0                      -0.058673                    0.324794
PAY_2                      -0.036500                    0.263551
PAY_3                      -0.035861                    0.235253
PAY_4                      -0.026565                    0.216614
PAY_5                      -0.023027                    0.204149
PAY_6                      -0.025299                    0.186866
BILL_AMT1                   0.179341                   -0.019644
BILL_AMT2                   0.174256                   -0.014193
BILL_AMT3                   0.182326                   -0.014076
BILL_AMT4                   0.177637                   -0.010156
BILL_AMT5                   0.164184                   -0.006760
BILL_AMT6                   0.115494                   -0.005372
PAY_AMT1                    0.185735                   -0.072929
PAY_AMT2                    0.157634                   -0.058579
PAY_AMT3                    0.162740                   -0.056250
PAY_AMT4                    0.157834                   -0.056827
PAY_AMT5                    0.154896                   -0.055124
PAY_AMT6                    1.000000                   -0.053183
default.payment.next.month -0.053183                    1.000000

[25 rows x 25 columns]
```

```
df.shape
```

```
(30000, 25)
```

```
sns.heatmap(df.corr())
```

```
<Axes: >
```




```
df.columns
```

```
Index(['ID', 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0',
       'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6',
       'default.payment.next.month'],
      dtype='object')
```

The Dataset provide is hightly imbalanced Dataset

```
df['default.payment.next.month'].value_counts()
```

```
default.payment.next.month
0    23364
1     6636
Name: count, dtype: int64
```

```
X = df.drop(['ID','default.payment.next.month'],axis = 1)
y = df["default.payment.next.month"]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.33, random_state=42)

X_train.shape , X_test.shape
```

```
((20100, 23), (9900, 23))
```

```
y_train.shape, y_test.shape
```

```
((20100,), (9900,))
```

```
# Installing Machine models which could perform well over the same classifier data.
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegressionCV
!pip install Xgboost
import xgboost as xb
from xgboost import XGBClassifier
```

```
Requirement already satisfied: Xgboost in /opt/conda/lib/python3.10/site-packages
(2.0.3)
Requirement already satisfied: numpy in /opt/conda/lib/python3.10/site-packages (from
Xgboost) (1.26.4)
Requirement already satisfied: scipy in /opt/conda/lib/python3.10/site-packages (from
Xgboost) (1.11.4)
```

```
## Model Training Automation
models={
    'Random Forest':RandomForestClassifier(),
    'Logistic Regression':LogisticRegressionCV(),
    'Decision Tree':DecisionTreeClassifier(),
    'Adaboost': AdaBoostClassifier(),
    'xgboost': XGBClassifier()

}

from sklearn.metrics import accuracy_score

# Evaluare_model function to find out the best model which having highest Accuracy.
def Evaluate_model(X_train,y_train,X_test,y_test,models):
  report = {}
  for i in range(len(models)):
    model = list(models.values())[i]
    #train model
    model.fit(X_train,y_train)


    #predict Testing data
    y_test_pred = model.predict(X_test)

    #get accuracy for test data prediction
    test_model_score = accuracy_score(y_test,y_test_pred)

    report[list(models.keys())[i]] = test_model_score

  return report

Evaluate_model(X_train,y_train,X_test,y_test,models)
```

```
{'Random Forest': 0.8174747474747475,
 'Logistic Regression': 0.7820202020202021,
 'Decision Tree': 0.7251515151515151,
 'Adaboost': 0.8177777777777778,
 'xgboost': 0.8132323232323232}
```

Here we got that Random forest and Adaboost having higtest Accuracy, so we will now on focus on these two along with hyperparametering

before that As mentioned above the dataset is imbalance so, Now we will first use most popular way to balace(smote) unbalance data.

```
# So Now we have four choice to balance the Dataset(1- upsampling, 2- downsampling,3-smote, 4- class_weight of algo)
from imblearn.over_sampling import SMOTE

# Applying SMOTE
smote = SMOTE(random_state=42)
X_resampled_smote, y_resampled_smote = smote.fit_resample(X_train, y_train)

df.shape
```

```
(30000, 25)
```

```
df['default.payment.next.month'].value_counts()
```

```
default.payment.next.month
0    23364
1     6636
Name: count, dtype: int64
```

```
y_train.value_counts() ## Large amount of difference in both classes
```

```
default.payment.next.month
0    15622
1     4478
Name: count, dtype: int64
```

```
X_resampled_smote.shape
```

```
(31244, 23)
```

```
y_resampled_smote.shape
```

```
(31244,)
```

```
y_resampled_smote.value_counts() ## Now we having balanced information for both categories
```

```
default.payment.next.month
1    15622
0    15622
Name: count, dtype: int64
```

```
## Using these three models whcih have highest Accuracy over balance data.
models1={
    'Random Forest':RandomForestClassifier(),
    'Adaboost': AdaBoostClassifier(),
    'xgboost': XGBClassifier()

}
```

Trainiing and Evaluating the performance of model with balance Dataset

```
def Evaluate_model1(X_resampled_smote,yresampled_smote,X_test,y_test,models1):
  report = {}
  for i in range(len(models1)):
    model = list(models1.values())[i]
    #train model
    model.fit(X_train,y_train)


    #predict Testing data
    y_pred3 = model.predict(X_test)

    #get accuracy for test data prediction
    print([list(models1.keys())[i]])
```

```
    print(confusion_matrix(y_test,y_pred3))
    print(accuracy_score(y_test,y_pred3))
    print(classification_report(y_test,y_pred3))

  return report

Evaluate_model1(X_resampled_smote,y_resampled_smote,X_test,y_test,models1)
```

```
['Random Forest']
[[7286  456]
 [1371  787]]
0.8154545454545454
              precision    recall  f1-score   support

           0       0.84      0.94      0.89      7742
           1       0.63      0.36      0.46      2158

    accuracy                           0.82      9900
   macro avg       0.74      0.65      0.68      9900
weighted avg       0.80      0.82      0.80      9900

['Adaboost']
[[7411  331]
 [1473  685]]
0.8177777777777778
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      7742
           1       0.67      0.32      0.43      2158

    accuracy                           0.82      9900
   macro avg       0.75      0.64      0.66      9900
weighted avg       0.80      0.82      0.79      9900

['xgboost']
[[7274  468]
 [1381  777]]
0.8132323232323232
              precision    recall  f1-score   support

           0       0.84      0.94      0.89      7742
           1       0.62      0.36      0.46      2158

    accuracy                           0.81      9900
   macro avg       0.73      0.65      0.67      9900
weighted avg       0.79      0.81      0.79      9900


{}
```

```
from sklearn.metrics import confusion_matrix,classification_report
```

Using Hyperparametering to got more accuracy.

```
from sklearn.model_selection import RandomizedSearchCV

X_train = X_resampled_smote
y_train = y_resampled_smote

from sklearn.model_selection import train_test_split, RandomizedSearchCV,
cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

def find_best_classifier():
```

```
    # Define the classifiers
    classifiers = {
        'AdaBoost': AdaBoostClassifier(),
        'RandomForest': RandomForestClassifier(),

    }

    # Define the hyperparameter grids for each classifier
    param_grids = {
        'AdaBoost': {
            'n_estimators': [50, 100, 150, 200],
            'learning_rate': [0.01, 0.1,0.3,0.5,0.7, 1]
        },
        'RandomForest': {
            'n_estimators': [10,20,40, 50,70,90,100],
            'max_features': [5,8,10,15,20,35,40,50],
            'max_depth': [4, 6, 8, 10,15, None],
            'criterion':["gini", "entropy", 'log_loss']
        }
    }

    # Perform RandomizedSearchCV for each classifier
    best_classifiers = {}
    for clf_name, clf in classifiers.items():
        print(f"Training {clf_name}...")
        random_search = RandomizedSearchCV(clf,
param_distributions=param_grids[clf_name], n_iter=5, cv=5, random_state=42, n_jobs=-
1)
        random_search.fit(X_train, y_train)
        best_classifiers[clf_name] = random_search.best_estimator_
        print(random_search.best_estimator_)


    # Evaluate the classifiers on the test set
    best_scores = {}
    for clf_name, clf in best_classifiers.items():
        y_pred = clf.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred)
        recall = recall_score(y_test, y_pred)
        best_scores[clf_name] = {
            'accuracy': accuracy,
            'precision': precision,
            'recall': recall,

        }
        print(f"{clf_name} Test Scores:")
        print(f" - Accuracy: {accuracy:.4f}")
        print(f" - Precision: {precision:.4f}")
        print(f" - Recall: {recall:.4f}")


    # Identify the best classifier
    best_classifier_name = max(best_scores, key=lambda x: best_scores[x]['accuracy'])
    print(f"\nBest Classifier: {best_classifier_name} with accuracy:
{best_scores[best_classifier_name]['accuracy']:.4f}")
    return best_classifiers[best_classifier_name]

# Call the function to find the best classifier
best_classifier = find_best_classifier()
```

```
Training AdaBoost...
AdaBoostClassifier(learning_rate=0.7, n_estimators=150)
```

```
Training RandomForest...
RandomForestClassifier(max_depth=15, max_features=40, n_estimators=70)
AdaBoost Test Scores:
 - Accuracy: 0.7446
 - Precision: 0.4349
 - Recall: 0.5723
RandomForest Test Scores:
 - Accuracy: 0.7700
 - Precision: 0.4739
 - Recall: 0.5000

Best Classifier: RandomForest with accuracy: 0.7700
```

```
classifier = xb.XGBClassifier()
classifier.fit(X_train,y_train)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

```
from sklearn.metrics import confusion_matrix,classification_report
y_pred3=classifier.predict(X_test)
print(confusion_matrix(y_test,y_pred3))
print(accuracy_score(y_test,y_pred3))
print(classification_report(y_test,y_pred3))
```

```
[[7274  468]
 [1381  777]]
0.8132323232323232
              precision    recall  f1-score   support

           0       0.84      0.94      0.89      7742
           1       0.62      0.36      0.46      2158

    accuracy                           0.81      9900
   macro avg       0.73      0.65      0.67      9900
weighted avg       0.79      0.81      0.79      9900
```

```
# the now using best_params_ of each model
models2={
    'Random Forest':RandomForestClassifier(max_depth=6, max_features=50,
n_estimators=20),
    'Adaboost': AdaBoostClassifier(learning_rate=0.01, n_estimators=150)

}

def Evaluate_model2(X_resampled_smote,yresampled_smote,X_test,y_test,models2):
  report = {}
  for i in range(len(models2)):
    model = list(models2.values())[i]
    #train model
    model.fit(X_train,y_train)


    #predict Testing data
```

```
    y_pred3 = model.predict(X_test)

    #get accuracy for test data prediction
    print([list(models2.keys())[i]])
    print(confusion_matrix(y_test,y_pred3))
    print(accuracy_score(y_test,y_pred3))
    print(classification_report(y_test,y_pred3))

  return report

Evaluate_model2(X_resampled_smote,y_resampled_smote,X_test,y_test,models2)
```

```
['Random Forest']
[[7371  371]
 [1379  779]]
0.8232323232323232
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      7742
           1       0.68      0.36      0.47      2158

    accuracy                           0.82      9900
   macro avg       0.76      0.66      0.68      9900
weighted avg       0.81      0.82      0.80      9900

['Adaboost']
[[7432  310]
 [1457  701]]
0.8215151515151515
              precision    recall  f1-score   support

           0       0.84      0.96      0.89      7742
           1       0.69      0.32      0.44      2158

    accuracy                           0.82      9900
   macro avg       0.76      0.64      0.67      9900
weighted avg       0.80      0.82      0.80      9900


{}
```

As per above ReSearch we have found that, that Dataset is imbalance but after using smote technique to balace the row Data we getting worse performance.

As we are finding fault payment, so In this case Recall is more important and precision, With help of hyperparametering we getting hight recall in Adaboost along with hightest Accuracy which is 82%