```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.metrics import ConfusionMatrixDisplay, classification_report
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
```

```
from google.colab import files

uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving UCI_Credit_Card.csv to UCI_Credit_Card.csv

```
data=pd.read_csv('UCI_Credit_Card.csv')
```

```
data.head()
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BIl
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	...	

5 rows × 25 columns

```
data.describe()
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.0(
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	35.4{

```
data.corr()
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	
ID	1.000000	0.026179	0.018497	0.039177	-0.029079	0.018
LIMIT_BAL	0.026179	1.000000	0.024755	-0.219161	-0.108139	0.144
SEX	0.018497	0.024755	1.000000	0.014232	-0.031389	-0.090
EDUCATION	0.039177	-0.219161	0.014232	1.000000	-0.143464	0.175
MARRIAGE	-0.029079	-0.108139	-0.031389	-0.143464	1.000000	-0.414
AGE	0.018678	0.144713	-0.090874	0.175061	-0.414170	1.000
PAY_0	-0.030575	-0.271214	-0.057643	0.105364	0.019917	-0.039
PAY_2	-0.011215	-0.296382	-0.070771	0.121566	0.024199	-0.050
PAY_3	-0.018494	-0.286123	-0.066096	0.114025	0.032688	-0.053
PAY_4	-0.002735	-0.267460	-0.060173	0.108793	0.033122	-0.049
PAY_5	-0.022199	-0.249411	-0.055064	0.097520	0.035629	-0.053
PAY_6	-0.020270	-0.235195	-0.044008	0.082316	0.034345	-0.048
BILL_AMT1	0.019389	0.285430	-0.033642	0.023581	-0.023472	0.056
BILL_AMT2	0.017982	0.278314	-0.031183	0.018749	-0.021602	0.054
BILL_AMT3	0.024354	0.283236	-0.024563	0.013002	-0.024909	0.053
BILL_AMT4	0.040351	0.293988	-0.021880	-0.000451	-0.023344	0.051
BILL_AMT5	0.016705	0.295562	-0.017005	-0.007567	-0.025393	0.049
BILL_AMT6	0.016730	0.290389	-0.016733	-0.009099	-0.021207	0.047
PAY_AMT1	0.009742	0.195236	-0.000242	-0.037456	-0.005979	0.026
PAY_AMT2	0.008406	0.178408	-0.001391	-0.030038	-0.008093	0.021
PAY_AMT3	0.039151	0.210167	-0.008597	-0.039943	-0.003541	0.029
PAY_AMT4	0.007793	0.203242	-0.002229	-0.038218	-0.012659	0.021
PAY_AMT5	0.000652	0.217202	-0.001667	-0.040358	-0.001205	0.022
PAY_AMT6	0.003000	0.219595	-0.002766	-0.037200	-0.006641	0.019
default.payment.next.month	-0.013952	-0.153520	-0.039961	0.028006	-0.024339	0.013

25 rows × 25 columns

```
data.isnull().sum()
```

```
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
```

```
x=data.drop(['BILL_AMT6','BILL_AMT5','default.payment.next.month'],axis=1)
y=data['default.payment.next.month']
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y)
```

Decision Tree

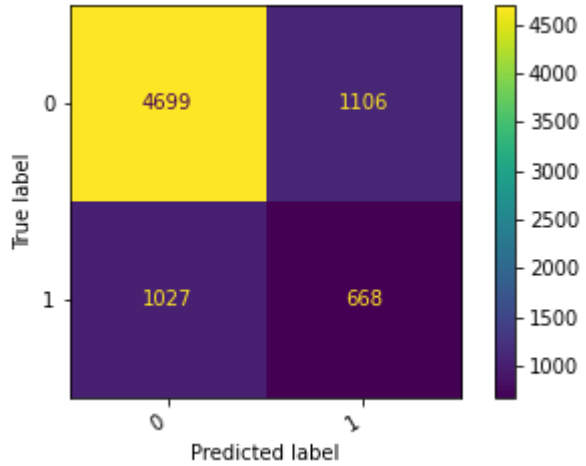
```
dt=DecisionTreeClassifier()
```

```
model=dt.fit(x_train,y_train)
```

```
predicts=model.predict(x_test)
```

```
confusion_matrix = metrics.confusion_matrix(y_test,predicts)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix)
cm_display.plot()
plt.gcf().autofmt_xdate()
plt.show()
```

```
print(classification_report(y_test, predicts))
```



	precision	recall	f1-score	support
0	0.82	0.81	0.82	5805
1	0.38	0.39	0.39	1695
accuracy			0.72	7500
macro avg	0.60	0.60	0.60	7500
weighted avg	0.72	0.72	0.72	7500

MLPclassifier

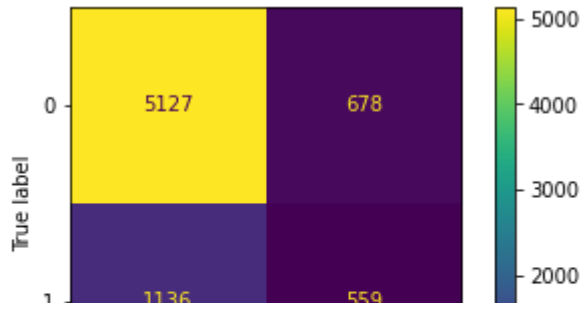
```
mlp=MLPClassifier()


model=mlp.fit(x_train,y_train)


predicts=model.predict(x_test)


confusion_matrix = metrics.confusion_matrix(y_test,predicts)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix)
cm_display.plot()
plt.gcf().autofmt_xdate()
plt.show()

print(classification_report(y_test, predicts))
```




```
model.score(x_test,y_test)
```

```
0.7581333333333333
```

Predicted label

Naive Bayes

```
              1       0.45      0.33      0.38      1695
```

```
nb=GaussianNB()
```

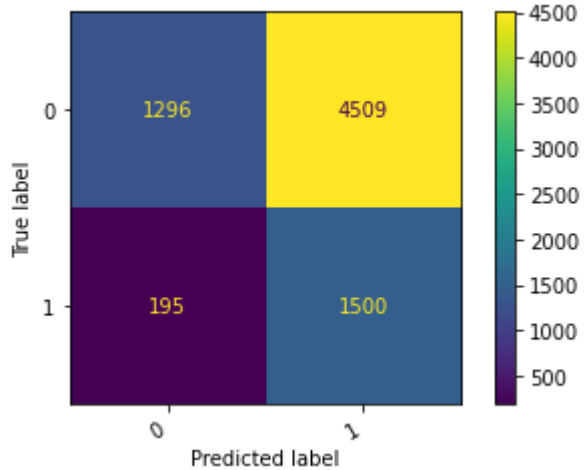
```
        macro avg       0.64      0.61      0.62      7500
```

```
model=nb.fit(x_train,y_train)
```

```
predicts=model.predict(x_test)
```

```
confusion_matrix = metrics.confusion_matrix(y_test,predicts)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix)
cm_display.plot()
plt.gcf().autofmt_xdate()
plt.show()
```

```
print(classification_report(y_test, predicts))
```




```
              precision    recall  f1-score   support

           0       0.87      0.22      0.36      5805
           1       0.25      0.88      0.39      1695

    accuracy                           0.37      7500
   macro avg       0.56      0.55      0.37      7500
weighted avg       0.73      0.37      0.36      7500
```

KNeighbour

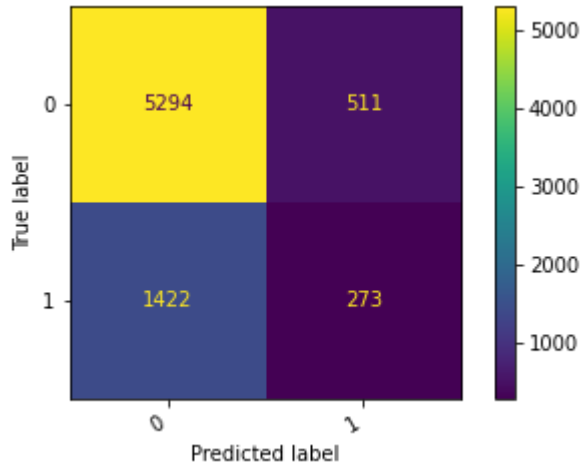
```
kn=KNeighborsClassifier()
```

```
model=kn.fit(x_train,y_train)
```

```
predicts=model.predict(x_test)
```

```
confusion_matrix = metrics.confusion_matrix(y_test,predicts)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix)
cm_display.plot()
plt.gcf().autofmt_xdate()
plt.show()
```

```
print(classification_report(y_test, predicts))
```




```
              precision    recall  f1-score   support

           0       0.79      0.91      0.85      5805
           1       0.35      0.16      0.22      1695

    accuracy                           0.74      7500
   macro avg       0.57      0.54      0.53      7500
weighted avg       0.69      0.74      0.70      7500
```

```
model.score(x_test,y_test)
```

```
0.7422666666666666
```