```
# 1.  Load the basic libraries and packages

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, BernoulliNB, MultinomialNB
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
```

```
# 2.    Load the dataset

data = pd.read_excel('/content/default_of_credit_card_clients.xls' , skiprows=1)
data
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AI
0	1	20000	2	2	1	24	2	2	-1	-1	...	0	0	0	0	6
1	2	120000	2	2	2	26	-1	2	0	0	...	3272	3455	3261	0	1(
2	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	1!
3	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	2(
4	5	50000	1	2	1	57	-1	0	-1	0	...	20940	19146	19131	2000	36(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1	3	1	39	0	0	0	0	...	88004	31237	15980	8500	20(
29996	29997	150000	1	3	2	43	-1	-1	-1	-1	...	8979	5190	0	1837	3!
29997	29998	30000	1	2	2	37	4	3	2	-1	...	20878	20582	19357	0	
29998	29999	80000	1	3	1	41	1	-1	0	0	...	52774	11855	48944	85900	3
29999	30000	50000	1	2	1	46	0	0	0	0	...	36535	32428	15313	2078	1

30000 rows × 25 columns

```
# 3.    Analyse the dataset

data.describe()
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	35.485500	-0.016700	-0.133767	-0.166200	-0
std	8660.398374	129747.661567	0.489129	0.790349	0.521970	9.217904	1.123802	1.197186	1.196868	1
min	1.000000	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.000000	-2.000000	-2.000000	-2
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.000000	-1.000000	-1.000000	-1
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000	0.000000	0.000000	0
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	41.000000	0.000000	0.000000	0.000000	0
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.000000	8.000000	8.000000	8

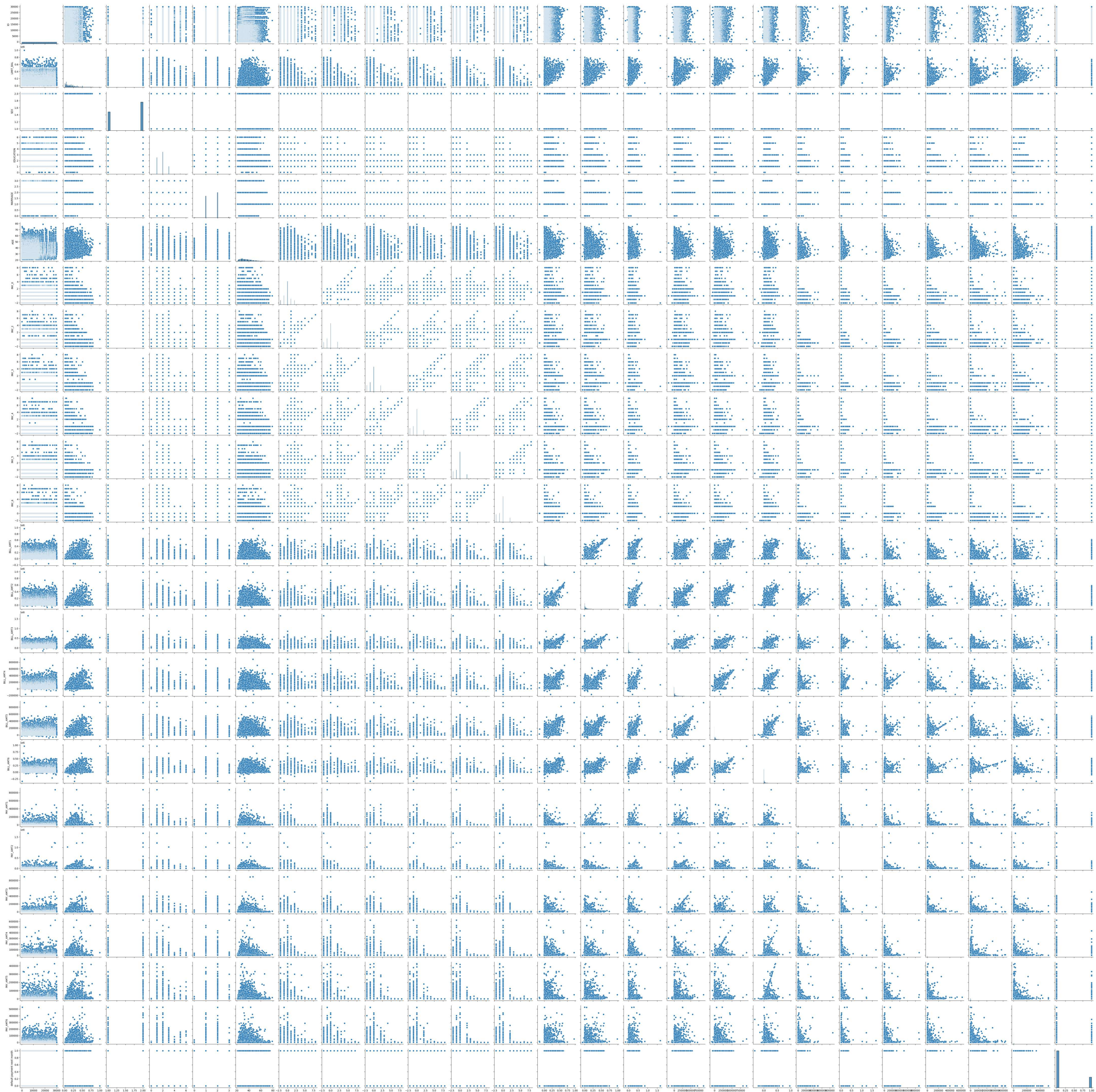
8 rows × 25 columns

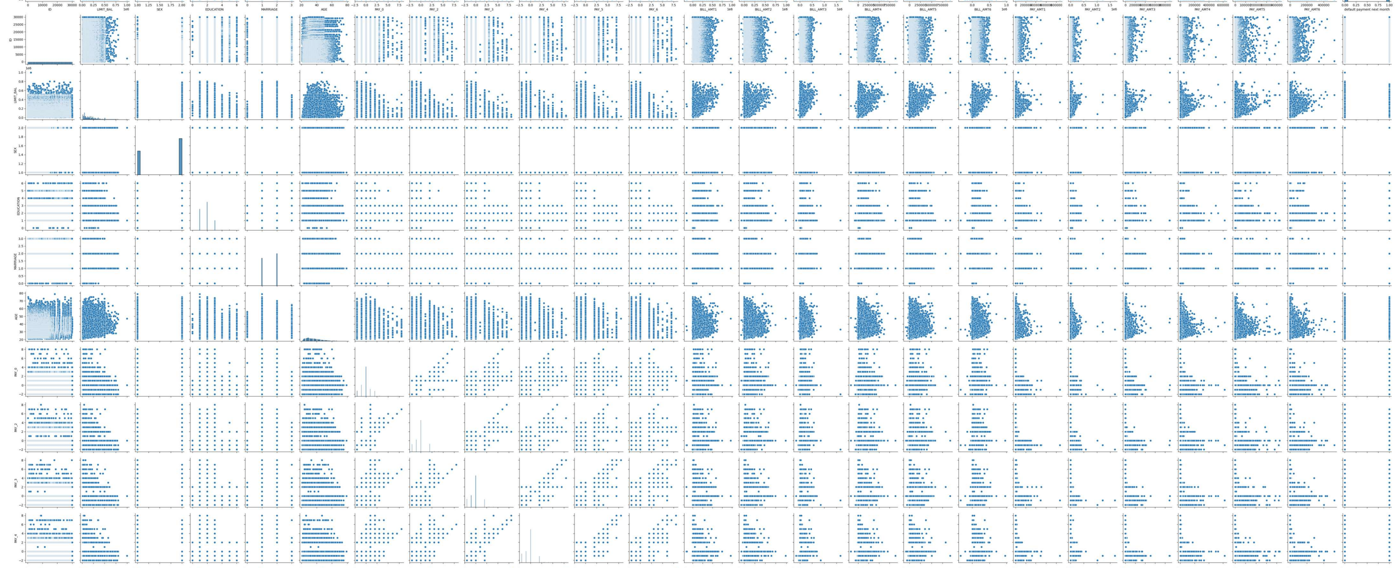
```
# 4.    Normalize the data

def Feature_Normalization(X):
  X = (X - np.mean(X)) / np.std(X)  # Calculate mean and std across the entire 1D array
  return X
```

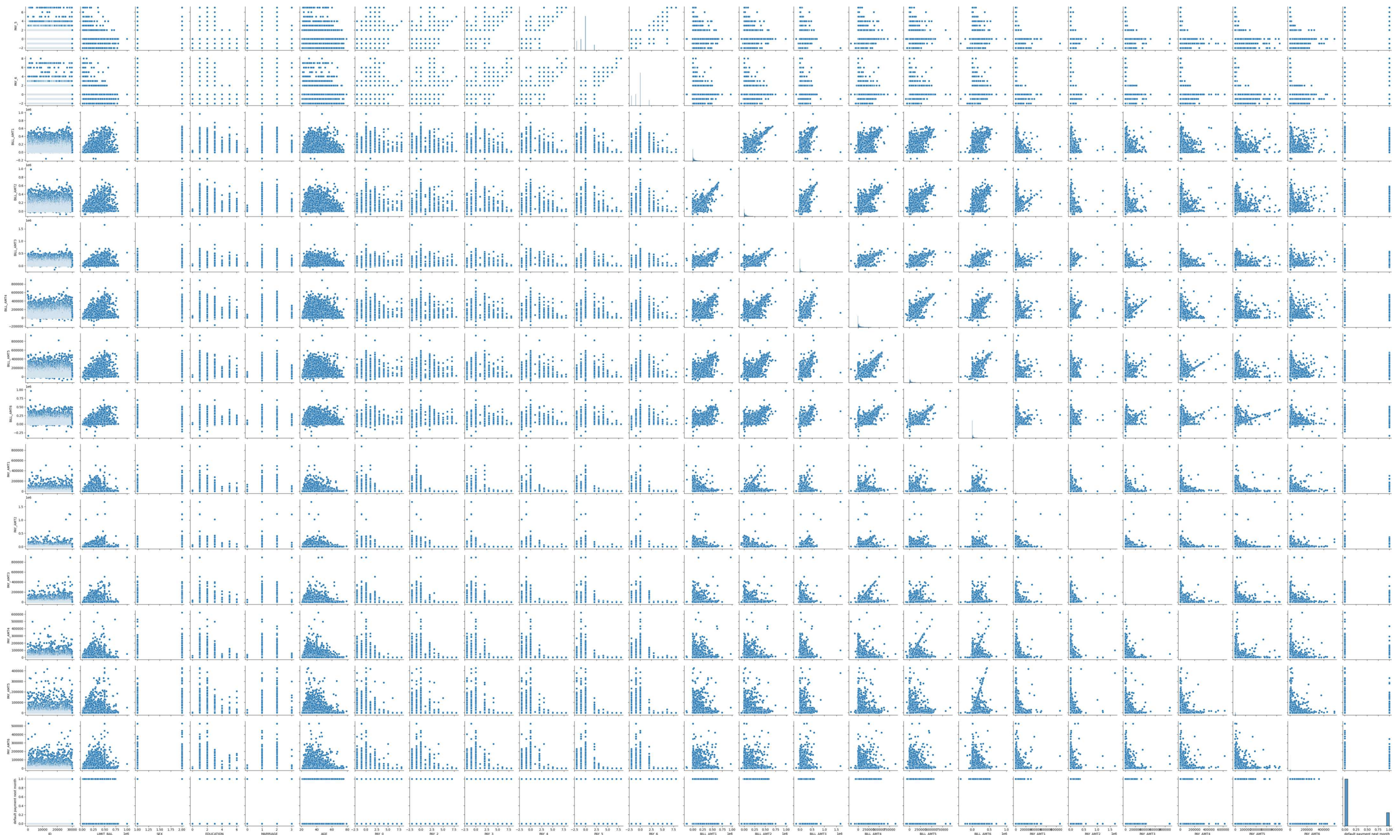
```
# 5.    Pre-process the data

x = data.iloc[: , :-1].values
y = data.iloc[: , -1].values

# Initialize x_norm as a list to store normalized features
x_norm = []

for i in range(x.shape[1]):  # Iterate through columns of x
    norm_feature = Feature_Normalization(x[:, i])
    x_norm.append(norm_feature)  # Append normalized feature to the list

# Convert the list of normalized features to a NumPy array
x_norm = np.array(x_norm).T  # Transpose to get the desired shape
x_norm
```

```
array([[-1.73199307, -1.13672015,  0.81016074, ..., -0.30806256,
        -0.31413612, -0.29338206],
       [-1.7318776 , -0.3659805 ,  0.81016074, ..., -0.24422965,
        -0.31413612, -0.18087821],
       [-1.73176213, -0.59720239,  0.81016074, ..., -0.24422965,
        -0.24868274, -0.01212243],
       ...,
       [ 1.73176213, -1.05964618, -1.23432296, ..., -0.03996431,
        -0.18322937, -0.11900109],
       [ 1.7318776 , -0.67427636, -1.23432296, ..., -0.18512036,
         3.15253642, -0.19190359],
       [ 1.73199307, -0.90549825, -1.23432296, ..., -0.24422965,
        -0.24868274, -0.23713013]])
```

```
# 6.    Visualize the Data

sns.pairplot(data)
plt.show()
```







```
# 7.    Separate the training and testing data

x_train , x_test , y_train , y_test = train_test_split(x_norm , y , test_size = 0.2 , random_state = 42)
```

```
# 8.    Apply the Bernoulli Naïve Bayes algorithm

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

```
BernoulliNB
BernoulliNB()
```

```
# 9.    Predict the testing dataset

y_pred_Bernoulli = model.predict(x_test)
```

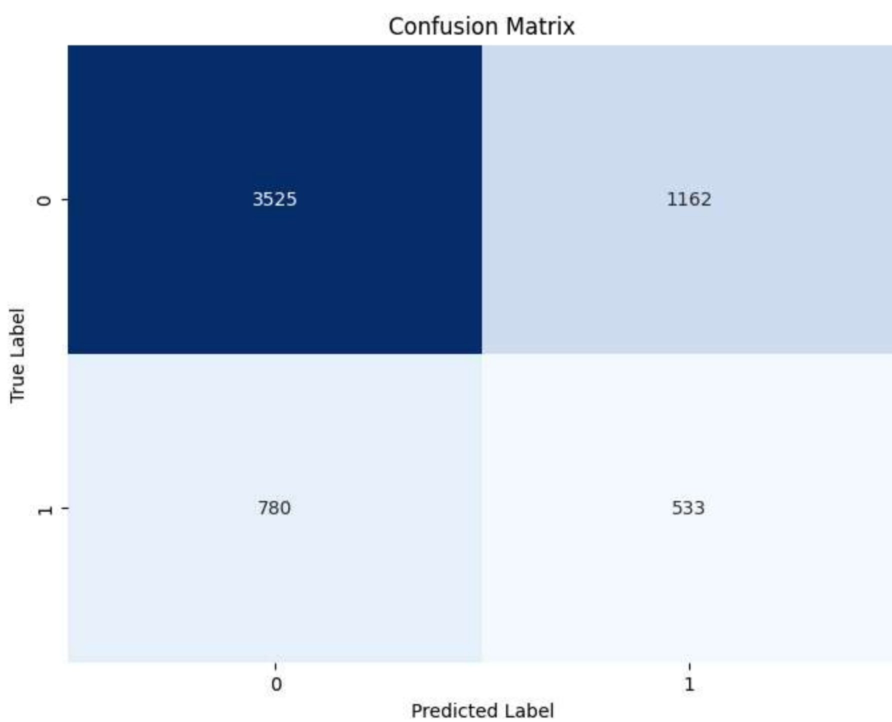
```
# 10.   Obtain the confusion matrix

cm = confusion_matrix(y_test , y_pred_Bernoulli)
cm
```

```
array([[3525, 1162],
       [ 780,  533]])
```

```
# 11.   Obtain the accuracy score

accuracy_score(y_test , y_pred_Bernoulli)
```

```
0.6763333333333333
```

```
# 12.   Visualize the classified dataset

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



```
# 13.   Apply the Gaussian Naïve Bayes algorithm

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

```
GaussianNB
GaussianNB()
```

```
# 14.   Predict the testing dataset

y_pred_GaussianNB = model.predict(x_test)
```

```
# 15.   Obtain the confusion matrix

cm = confusion_matrix(y_test , y_pred_GaussianNB)
cm
```

```
array([[3396, 1291],
       [ 458,  855]])
```

```
# 16.   Obtain the accuracy score

accuracy_score(y_test , y_pred_GaussianNB)
```

```
0.7085
```

```
# 17.   Visualize the classified dataset

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```