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

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import pydot
from sklearn.tree import export_graphviz
from sklearn import metrics
from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


# 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	P
0	1	20000	2	2	1	24	2	2	-1	-1	...	0	0	0	0	
1	2	120000	2	2	2	26	-1	2	0	0	...	3272	3455	3261	0	
2	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	
3	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	
4	5	50000	1	2	1	57	-1	0	-1	0	...	20940	19146	19131	2000	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1	3	1	39	0	0	0	0	...	88004	31237	15980	8500	
29996	29997	150000	1	3	2	43	-1	-1	-1	-1	...	8979	5190	0	1837	
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	
29999	30000	50000	1	2	1	46	0	0	0	0	...	36535	32428	15313	2078	

30000 rows × 25 columns

```
# 3.    Separate the feature and prediction value columns

X = data.drop("default payment next month",axis=1)
y = data["default payment next month"]


# 4.    Pre-process 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

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]])
```

```
# 5. Splitting 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)
```

```
# 6. Creating a Model

forest = RandomForestClassifier(n_estimators=200 , random_state = 42)
```

```
# 7. Training a Model

forest.fit(X_Train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(n_estimators=200, random_state=42)
```

```
# 8. Predicitng the Output Using model

Y_Predicted = forest.predict(X_Test)
```

```
# 9. Measuring the Accuracy

metrics.accuracy_score(Y_Test,Y_Predicted)
```

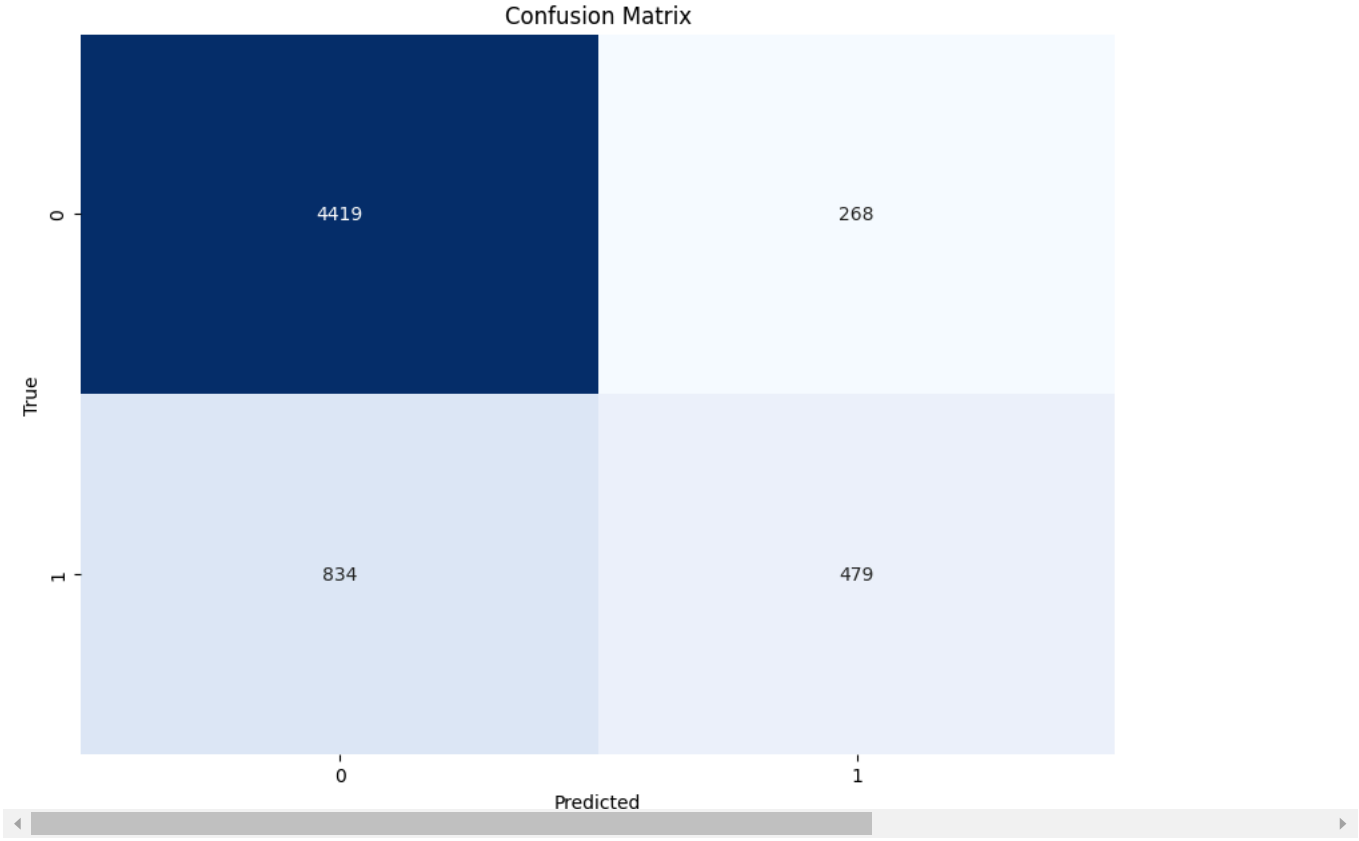
```
0.8163333333333334
```

```
# 10. Creating a Confusion Matrix

confusion_matrix = confusion_matrix(y_true = Y_Test , y_pred = Y_Predicted)
confusion_matrix
```

```
array([[4419,  268],
       [ 834,  479]])
```

```
# 11. Plotting a Confusion Matrix

plt.figure(figsize=(10, 7))
sns.heatmap(confusion_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
```

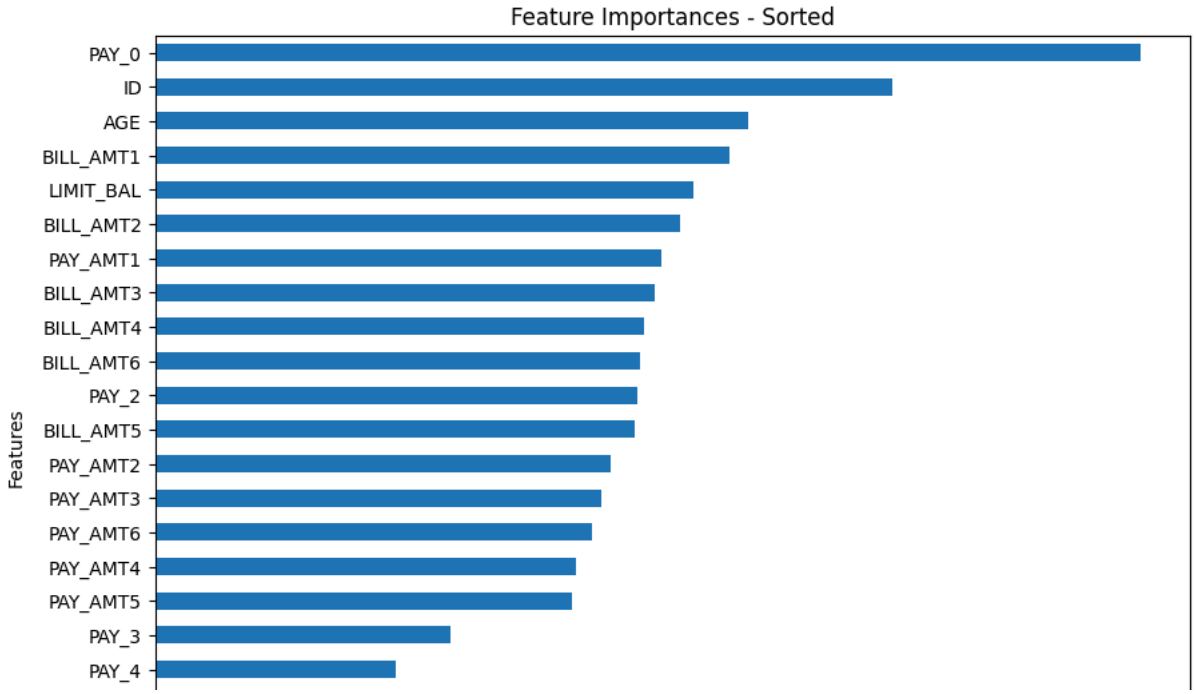
```
Text(0.5, 1.0, 'Confusion Matrix')
```




```
# 12. Accesing a Tree

tree  = forest.estimators_[5]


# 13. Exporitng a Tree

export_graphviz(tree, out_file='/content/tree.dot', feature_names=X.columns, rounded=True, precision=1)


# 14. Converting the Tree into png format

graph = pydot.graph_from_dot_file('/content/tree.dot')
graph[0].write_png('/content/tree.png')


# 15. Loading a Graph

(graph,) = pydot.graph_from_dot_file('/content/tree.dot')


# 16. Analysing the Feature Importance

forest.feature_importances_
```

```
array([0.07058186, 0.05159751, 0.01024649, 0.01839903, 0.012592  ,
       0.05684346, 0.09436883, 0.04614297, 0.02828079, 0.02307154,
       0.02144156, 0.01730312, 0.05498292, 0.05027022, 0.04780828,
       0.04682365, 0.04594865, 0.04644907, 0.04850121, 0.04363698,
       0.04270387, 0.04024435, 0.03989228, 0.04186938])
```

```
# 17. Visulizing the FFeature Importance

feature_importances = pd.Series(forest.feature_importances_, index=X.columns)

# Sort the feature importances in descending order
sorted_importances = feature_importances.sort_values(ascending=True)

# Plot the sorted feature importances
sorted_importances.plot(kind='barh', figsize=(10, 8))

# Add labels and title
plt.xlabel('Importance')
plt.ylabel('Features')
plt.title('Feature Importances - Sorted')
plt.show()
```


Feature Importances - Sorted
PAY_0
ID
AGE
BILL_AMT1
LIMIT_BAL
BILL_AMT2
PAY_AMT1
BILL_AMT3
BILL_AMT4
BILL_AMT6
PAY_2
BILL_AMT5
PAY_AMT2
PAY_AMT3
PAY_AMT6
PAY_AMT4
PAY_AMT5
PAY_3
PAY_4
Features