Importing Libraries & Loading Data

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv('CreditCardDefault.csv')
```

In [3]:
```
type(df)
```

Out[3]:
```
pandas.core.frame.DataFrame
```

In [4]:
```
df.shape
```

Out[4]:
```
(30000, 25)
```

In [5]:
```
df.head()
```

Out[5]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BI
0	1	20000	2	2	1	24	2	2	-1	-1	...	
1	2	120000	2	2	2	26	-1	2	0	0	...	
2	3	90000	2	2	2	34	0	0	0	0	...	
3	4	50000	2	2	1	37	0	0	0	0	...	
4	5	50000	1	2	1	57	-1	0	-1	0	...	

5 rows × 25 columns

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   ID                  30000 non-null  int64
 1   LIMIT_BAL           30000 non-null  int64
 2   SEX                 30000 non-null  int64
 3   EDUCATION           30000 non-null  int64
 4   MARRIAGE            30000 non-null  int64
 5   AGE                 30000 non-null  int64
 6   PAY_0               30000 non-null  int64
 7   PAY_2               30000 non-null  int64
 8   PAY_3               30000 non-null  int64
 9   PAY_4               30000 non-null  int64
 10  PAY_5               30000 non-null  int64
 11  PAY_6               30000 non-null  int64
 12  BILL_AMT1           30000 non-null  int64
 13  BILL_AMT2           30000 non-null  int64
 14  BILL_AMT3           30000 non-null  int64
 15  BILL_AMT4           30000 non-null  int64
 16  BILL_AMT5           30000 non-null  int64
 17  BILL_AMT6           30000 non-null  int64
 18  PAY_AMT1            30000 non-null  int64
 19  PAY_AMT2            30000 non-null  int64
 20  PAY_AMT3            30000 non-null  int64
 21  PAY_AMT4            30000 non-null  int64
 22  PAY_AMT5            30000 non-null  int64
 23  PAY_AMT6            30000 non-null  int64
 24  next_month_payment  30000 non-null  int64
dtypes: int64(25)
memory usage: 5.7 MB
```

In [7]:
```
df.describe()
```

Out[7]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	35.485500
std	8660.398374	129747.661567	0.489129	0.790349	0.521970	9.217904
min	1.000000	10000.000000	1.000000	0.000000	0.000000	21.000000
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	28.000000
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	34.000000
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	41.000000
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	79.000000

8 rows × 25 columns

Feature Engineering

In [8]:

```
X = df.iloc[:,:-1]
y = pd.DataFrame(df.iloc[:,-1])
```

In [9]:

```
X.shape, y.shape
```

Out[9]:

```
((30000, 24), (30000, 1))
```

In [10]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [11]:

```
scaler = StandardScaler()
```

In [12]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, str
```

In [13]:

```
X_train = scaler.fit_transform(X_train)
```

In [14]:

```
X_test = scaler.transform(X_test)
```

Base Model & Ensemble Model - SVM

In [15]:

```
from sklearn.ensemble import BaggingClassifier
from sklearn.svm import SVC
```

In [16]:

```
base_model = SVC(probability=True)
```

In [17]:

```
model = BaggingClassifier(base_estimator=base_model, n_estimators=10, max_sam
                  bootstrap_features=True, n_jobs=-1)
```

In [18]:
```
model.fit(X_train, y_train)
```

```
C:\Users\15516\anaconda3\lib\site-packages\sklearn\ensemble\_bagging.py:64
5: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

Out[18]:
```
BaggingClassifier(base_estimator=SVC(C=1.0, break_ties=False, cache_size=20
0,
                                     class_weight=None, coef0=0.0,
                                     decision_function_shape='ovr', degree=
3,
                                     gamma='scale', kernel='rbf', max_iter=
-1,
                                     probability=True, random_state=None,
                                     shrinking=True, tol=0.001, verbose=Fal
se),
                  bootstrap=True, bootstrap_features=True, max_features=15,
                  max_samples=0.4, n_estimators=10, n_jobs=-1, oob_score=Fa
lse,
                  random_state=None, verbose=0, warm_start=False)
```

In [19]:
```
y_pred = model.predict(X_test)
```

In [20]:
```
from sklearn.metrics import accuracy_score
```

In [21]:
```
accuracy_score(y_test, y_pred)
```

Out[21]:
```
0.8142222222222222
```

Base Model & Ensemble Model - Decision Tree

In [22]:
```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
```

In [23]:
```
base_model = DecisionTreeClassifier(criterion='entropy')
```

In [24]:
```
model = BaggingClassifier(base_estimator=base_model, n_estimators=10, max_sam
                  bootstrap_features=True, n_jobs=-1)
```

In [25]:

```
model.fit(X_train, y_train)
```

```
C:\Users\15516\anaconda3\lib\site-packages\sklearn\ensemble\_bagging.py:64
5: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

Out[25]:

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                        class_weight=None,
                                                        criterion='entrop
y',
                                                        max_depth=None,
                                                        max_features=None,
                                                        max_leaf_nodes=Non
e,
                                                        min_impurity_decrea
se=0.0,
                                                        min_impurity_split=
None,
                                                        min_samples_leaf=1,
                                                        min_samples_split=
2,
                                                        min_weight_fraction
_leaf=0.0,
                                                        presort='deprecate
d',
                                                        random_state=None,
                                                        splitter='best'),
                  bootstrap=True, bootstrap_features=True, max_features=15,
                  max_samples=0.4, n_estimators=10, n_jobs=-1, oob_score=Fa
lse,
                  random_state=None, verbose=0, warm_start=False)
```

In [26]:

```
y_pred = model.predict(X_test)
```

In [27]:

```
from sklearn.metrics import accuracy_score
```

In [28]:

```
accuracy_score(y_test, y_pred)
```

Out[28]:

```
0.8112222222222222
```

In [ ]: