In [1]:
```
"""
Given a bank customer, build a neural network-based classifier that can determine w
they will leave or not in the next 6 months.
Dataset Description: The case study is from an open-source dataset from Kaggle.
The dataset contains 10,000 sample points with 14 distinct features such as
CustomerId, CreditScore, Geography, Gender, Age, Tenure, Balance, etc.
Link to the Kaggle project:
https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling
Perform following steps:
1. Read the dataset.
2. Distinguish the feature and target set and divide the data set into training and
3. Normalize the train and test data.
4. Initialize and build the model. Identify the points of improvement and implement
5. Print the accuracy score and confusion matrix (5 points).
"""
```

Out[1]:
```
'\nGiven a bank customer, build a neural network-based classifier that can determi
ne whether\nthey will leave or not in the next 6 months.\nDataset Description: The
case study is from an open-source dataset from Kaggle.\nThe dataset contains 10,00
0 sample points with 14 distinct features such as\nCustomerId, CreditScore, Geogra
phy, Gender, Age, Tenure, Balance, etc.\nLink to the Kaggle project:\nhttps://www.
kaggle.com/barelydedicated/bank-customer-churn-modeling\nPerform following step
s:\n1. Read the dataset.\n2. Distinguish the feature and target set and divide the
data set into training and test sets.\n3. Normalize the train and test data.\n4. I
nitialize and build the model. Identify the points of improvement and implement th
e same.\n5. Print the accuracy score and confusion matrix (5 points).\n'
```

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [4]:
```
df = pd.read_csv('Bank Churn Modelling.csv')
df
```

Out[4]:

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	I P
0	15634602	Hargrave	619	France	Female	42	2	0.00	
1	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	15619304	Onio	502	France	Female	42	8	159660.80	
3	15701354	Boni	699	France	Female	39	1	0.00	
4	15737888	Mitchell	850	Spain	Female	43	2	125510.82	
...	...	...	...	...	...	...	...	...	
9995	15606229	Obijiaku	771	France	Male	39	5	0.00	
9996	15569892	Johnstone	516	France	Male	35	10	57369.61	
9997	15584532	Liu	709	France	Female	36	7	0.00	
9998	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	
9999	15628319	Walker	792	France	Female	28	4	130142.79	

10000 rows × 13 columns

In [5]:
```
df = df.drop(['CustomerId','Surname'],axis=1)
df
```

Out[5]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	Num Of Products	Has Credit Card	Is Active Membe
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	
2	502	France	Female	42	8	159660.80	3	1	
3	699	France	Female	39	1	0.00	2	0	
4	850	Spain	Female	43	2	125510.82	1	1	
...	...	...	...	...	...	...	...	...	
9995	771	France	Male	39	5	0.00	2	1	
9996	516	France	Male	35	10	57369.61	1	1	
9997	709	France	Female	36	7	0.00	1	0	
9998	772	Germany	Male	42	3	75075.31	2	1	
9999	792	France	Female	28	4	130142.79	1	1	

10000 rows × 11 columns

In [6]:
```
X = df.drop('Churn',axis=1)
y = df['Churn']
```

In [7]:
```
X
```

Out[7]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	Num Of Products	Has Credit Card	Is Active Member
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	
2	502	France	Female	42	8	159660.80	3	1	
3	699	France	Female	39	1	0.00	2	0	
4	850	Spain	Female	43	2	125510.82	1	1	
...	...	...	...	...	...	...	...	...	...
9995	771	France	Male	39	5	0.00	2	1	
9996	516	France	Male	35	10	57369.61	1	1	
9997	709	France	Female	36	7	0.00	1	0	
9998	772	Germany	Male	42	3	75075.31	2	1	
9999	792	France	Female	28	4	130142.79	1	1	

10000 rows × 10 columns

In [8]:
```
y
```

Out[8]:
```
0       1
1       0
2       1
3       0
4       0
       ..
9995    0
9996    0
9997    1
9998    1
9999    0
Name: Churn, Length: 10000, dtype: int64
```

In [9]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   CreditScore       10000 non-null  int64  
 1   Geography         10000 non-null  object 
 2   Gender            10000 non-null  object 
 3   Age               10000 non-null  int64  
 4   Tenure            10000 non-null  int64  
 5   Balance           10000 non-null  float64
 6   Num Of Products   10000 non-null  int64  
 7   Has Credit Card   10000 non-null  int64  
 8   Is Active Member  10000 non-null  int64  
 9   Estimated Salary  10000 non-null  float64
dtypes: float64(2), int64(6), object(2)
memory usage: 781.4+ KB
```

In [10]:
```
X = pd.get_dummies(X, columns=['Geography', 'Gender'], drop_first=True)
```

In [11]:
```
X
```

Out[11]:

	CreditScore	Age	Tenure	Balance	Num Of Products	Has Credit Card	Is Active Member	Estimated Salary	Geogra
0	619	42	2	0.00	1	1	1	101348.88	
1	608	41	1	83807.86	1	0	1	112542.58	
2	502	42	8	159660.80	3	1	0	113931.57	
3	699	39	1	0.00	2	0	0	93826.63	
4	850	43	2	125510.82	1	1	1	79084.10	
...	...	...	...	...	...	...	...	...	
9995	771	39	5	0.00	2	1	0	96270.64	
9996	516	35	10	57369.61	1	1	1	101699.77	
9997	709	36	7	0.00	1	0	1	42085.58	
9998	772	42	3	75075.31	2	1	0	92888.52	
9999	792	28	4	130142.79	1	1	0	38190.78	

10000 rows × 11 columns

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
```

In [13]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [29]:
```
svm_clf = SVC(kernel='rbf', random_state=50)
svm_clf.fit(X_train_scaled, y_train)
```

Out[29]:

SVC

▶ Parameters

In [30]:
```
y_pred_svm = svm_clf.predict(X_test_scaled)
```

In [31]:
```
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print("SVM Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svm))
```

```
SVM Accuracy: 0.8576666666666667
SVM Confusion Matrix:
 [[2327   62]
 [ 365  246]]
```

In [32]:
```
rf_clf = RandomForestClassifier(n_estimators=400, random_state=42)
rf_clf.fit(X_train_scaled, y_train)
```

Out[32]:

RandomForestClassifier

▶ Parameters

In [33]:
```
y_pred_rf = rf_clf.predict(X_test_scaled)
```

In [34]:
```
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Random Forest Confusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 0.8656666666666667
Random Forest Confusion Matrix:
 [[2315   74]
 [ 329  282]]
```

In [ ]: