```
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.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix,accuracy_score
```

```
# Loading the dataset
data = pd.read_csv('/content/Churn_Modelling.csv')
data
```

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 |
| **1** | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 |
| **2** | 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 |
| **3** | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 |
| **4** | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **9995** | 9996 | 15606229 | Obijiaku | 771 | France | Male | 39 | 5 | 0.00 | 2 | 1 |
| **9996** | 9997 | 15569892 | Johnstone | 516 | France | Male | 35 | 10 | 57369.61 | 1 | 1 |
| **9997** | 9998 | 15584532 | Liu | 709 | France | Female | 36 | 7 | 0.00 | 1 | 0 |
| **9998** | 9999 | 15682355 | Sabbatini | 772 | Germany | Male | 42 | 3 | 75075.31 | 2 | 1 |
| **9999** | 10000 | 15628319 | Walker | 792 | France | Female | 28 | 4 | 130142.79 | 1 | 1 |

10000 rows × 14 columns

Next steps: Generate code with data View recommended plots

```
data.head()
```

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | Is |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 | |
| **1** | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 | |
| **2** | 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 | |
| **3** | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 | |
| **4** | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 | |

Next steps: Generate code with data View recommended plots

```
data.tail()
```

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Ba |
|---|---|---|---|---|---|---|---|---|---|
| **9995** | 9996 | 15606229 | Obijiaku | 771 | France | Male | 39 | 5 | |
| **9996** | 9997 | 15569892 | Johnstone | 516 | France | Male | 35 | 10 | 57 |
| **9997** | 9998 | 15584532 | Liu | 709 | France | Female | 36 | 7 | |
| **9998** | 9999 | 15682355 | Sabbatini | 772 | Germany | Male | 42 | 3 | 75 |
| **9999** | 10000 | 15628319 | Walker | 792 | France | Female | 28 | 4 | 130 |

```
data.shape
```

```
(10000, 14)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

```
data.dtypes
```

```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

```
data.isna().sum()
```

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
```

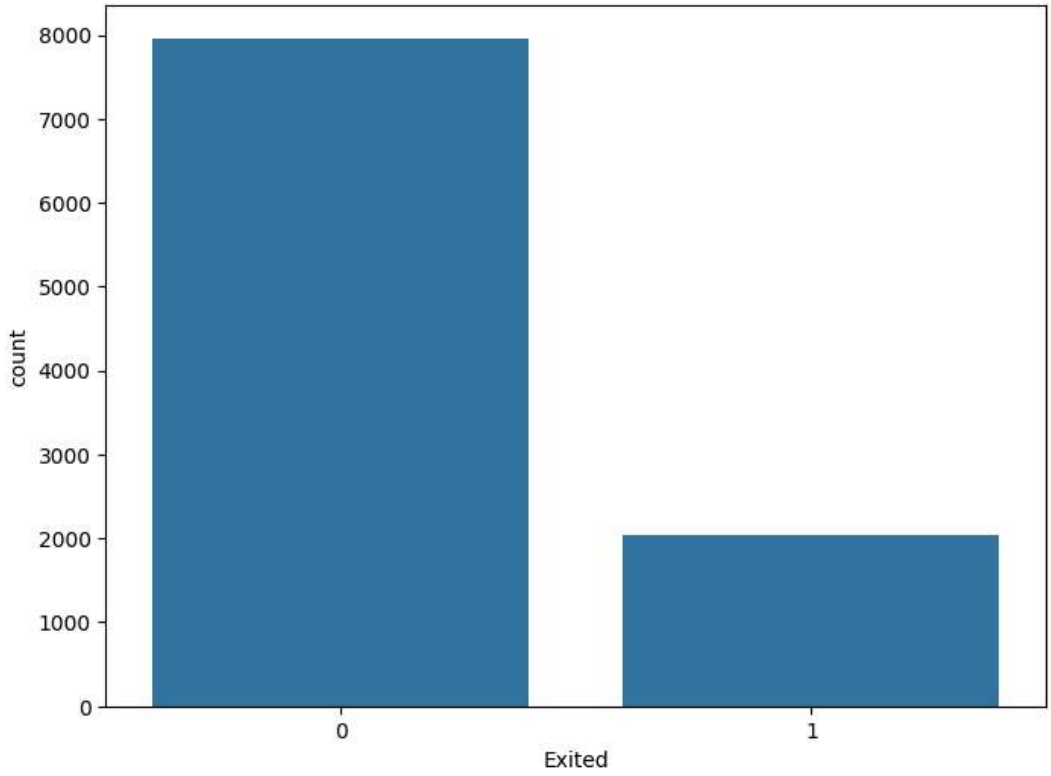
```
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

```
data['Exited'].value_counts()
```

```
Exited
0    7963
1    2037
Name: count, dtype: int64
```

```
plt.figure(figsize =(8,6))
sns.countplot(x='Exited',data = data)
```

```
<Axes: xlabel='Exited', ylabel='count'>
```




Start coding or generate with AI.

```
# Excluded non-numeric columns like 'Surname'
numeric_columns = ['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSa
X = data[numeric_columns]
```

```
# Splitting data in target label
y = data['Exited']
```

```
# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Building and train of model (Random Forest)
knn=KNeighborsClassifier(n_neighbors=7)
nb=GaussianNB()
model=SVC()
rf = RandomForestClassifier()
lst_model=[knn,nb,model,rf]
```

```
# Predicting &  Evaluating the model
for i in lst_model:
  print("Model name: ",i)
  i.fit(X_train,y_train)
  y_pred=i.predict(X_test)
  print("****************************************")
  print(confusion_matrix(y_test,y_pred))
  print("Accuracy Score.............")
  print(accuracy_score(y_test,y_pred))
```

```
Model name:  KNeighborsClassifier(n_neighbors=7)
****************************************
[[1536   71]
 [ 243  150]]
Accuracy Score.............
0.843
Model name:  GaussianNB()
****************************************
[[1570   37]
 [ 307   86]]
Accuracy Score.............
0.828
Model name:  SVC()
****************************************
[[1566   41]
 [ 256  137]]
Accuracy Score.............
0.8515
Model name:  RandomForestClassifier()
****************************************
[[1551   56]
 [ 230  163]]
Accuracy Score.............
0.857
```