```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
import nltk
import re
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
df = pd.read_csv('/content/drive/MyDrive/Subway/Churn_Modelling.csv')
```

```
df.columns
```

```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

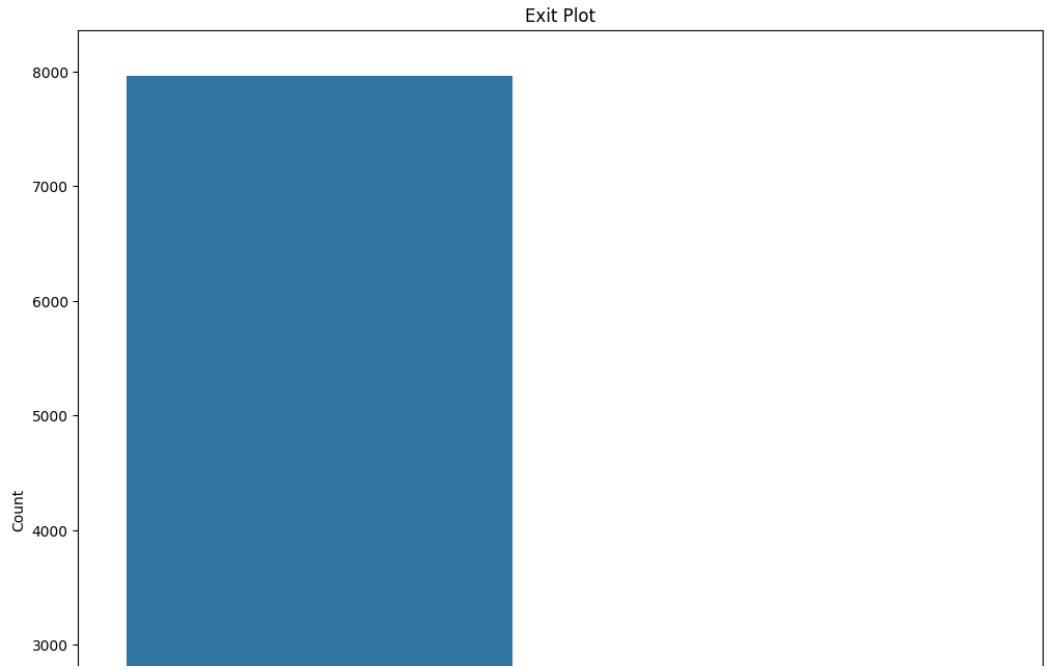
```
df.shape
```

```
(10000, 14)
```

```
df.head(5)
```

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

```
plt.figure(figsize=(12,12))
sns.countplot(x='Exited', data=df)
plt.xlabel('Exits')
plt.ylabel('Count')
plt.title('Exit Plot')
plt.show()
```




```
exits = list(df['Exited'].unique())
exits.sort()
exits
```

```
[0, 1]
```

```
df.isna().any()
```

```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

```
df.drop('RowNumber', axis=1, inplace=True)
df.drop('CustomerId', axis=1, inplace=True)
df.drop('Surname', axis=1, inplace=True)
```

```
import pandas as pd
from sklearn.preprocessing import OrdinalEncoder
from sklearn.naive_bayes import CategoricalNB
from sklearn import metrics

data = df
le = OrdinalEncoder()
data[['CreditScore', 'Geography','Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard','IsActiveMember', 'EstimatedSalary',
    data[[ 'CreditScore','Geography','Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard','IsActiveMember', 'EstimatedSala

Features = ['CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary']
X = data[Features]
Y = data['Exited']

clf_nb = CategoricalNB()
clf_nb.fit(X,Y)
```

```
▾ CategoricalNB
CategoricalNB()
```

```
from sklearn.ensemble import RandomForestClassifier
clf_rf = RandomForestClassifier(n_estimators=100, random_state=0)  # You can adjust n_estimators as needed
clf_rf.fit(X, Y)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=0)
```

```
from sklearn.ensemble import GradientBoostingClassifier


clf_gb = GradientBoostingClassifier(n_estimators=100, random_state=0)  # You can adjust n_estimators as needed
clf_gb.fit(X, Y)
```

```
GradientBoostingClassifier
GradientBoostingClassifier(random_state=0)
```

```
y_pred = clf_nb.predict(X)
print(Y)
print(y_pred)

print("Accuracy : " , metrics.accuracy_score(y_pred,Y) * 100)
```

```
0       1.0
1       0.0
2       1.0
3       0.0
4       0.0
       ...
9995    0.0
9996    0.0
9997    1.0
9998    1.0
9999    0.0
Name: Exited, Length: 10000, dtype: float64
[0. 0. 1. ... 0. 1. 0.]
Accuracy :  94.22
```

```
y_pred = clf_rf.predict(X)
print(Y)
print(y_pred)

print("Accuracy : " , metrics.accuracy_score(y_pred,Y) * 100)
```

```
0       1.0
1       0.0
2       1.0
3       0.0
4       0.0
       ...
9995    0.0
9996    0.0
9997    1.0
9998    1.0
9999    0.0
Name: Exited, Length: 10000, dtype: float64
[1. 0. 1. ... 1. 1. 0.]
Accuracy :  100.0
```

```
y_pred = clf_gb.predict(X)
print(Y)
print(y_pred)

print("Accuracy : " , metrics.accuracy_score(y_pred,Y) * 100)
```

```
0       1.0
1       0.0
2       1.0
3       0.0
4       0.0
       ...
9995    0.0
9996    0.0
9997    1.0
9998    1.0
9999    0.0
Name: Exited, Length: 10000, dtype: float64
[0. 0. 1. ... 0. 0. 0.]
Accuracy :  87.15
```

✓ 0s completed at 18:00