In [73]:
```
import pandas as pd
import seaborn as sns
```

In [74]:
```
df = pd.read_csv("D:\c drive settinf\Desktop\datasets\Churn_Modelling.csv")
df.head()
```

Out[74]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
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

In [75]:
```
df.shape
```

Out[75]:
```
(10000, 14)
```

In [76]:
```
df.describe()
```

Out[76]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	Num
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	1(
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	

In [77]:
```
df.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
```

In [78]:
```
df.isnull().sum()
```

Out[78]:
```
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
```

In [79]:
```
df.duplicated().sum()
```

Out[79]:
```
0
```

In [80]:
```
df["Geography"].value_counts()
```

Out[80]:
```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

In [81]:
```
df["Gender"].value_counts()
```

Out[81]:
```
Male      5457
Female    4543
Name: Gender, dtype: int64
```

In [82]:
```
df = df.drop(['RowNumber','CustomerId','Surname'], axis=1)
```

In [83]:
```
df = pd.DataFrame(df)
df
```

Out[83]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsAc
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 [84]:
```
# extract the data from the dataset
categorical_columns = df.select_dtypes(include=["object"]).columns.tolist()
```

categorical_columns

In [85]:
```
categorical_columns
```

Out[85]:
```
['Geography', 'Gender']
```

In [86]:
```
# import sklearn and onehotencoder
from sklearn.preprocessing import OneHotEncoder
```

In [87]:
```
# initialize the onehotencoding
ohe = OneHotEncoder(sparse=False)
```

In [88]:
```
one_hot_encoded = ohe.fit_transform(df[categorical_columns])
```

In [89]:
```
# creating dataframe for onehotencoded.
one_hot_df = pd.DataFrame(one_hot_encoded)
```

In [90]:
```
# to get the column name of the we use the get_feature_names_out()
columns_name = ohe.get_feature_names_out(categorical_columns)
```

In [91]:
```
# concatenate the ohe_dataframe with original one
df_encoded = pd.concat([df,one_hot_df],axis=1)
```

In [92]:
```
df_encoded_1 = df_encoded.drop(categorical_columns,axis=1)
```

In [96]:
```
df_encoded_1.columns
```

Out[96]:
```
Index([     'CreditScore',               'Age',            'Tenure',
                'Balance',     'NumOfProducts',         'HasCrCard',
         'IsActiveMember', 'EstimatedSalary',            'Exited',
                        0,                    1,                   2,
                        3,                    4],
      dtype='object')
```

In [98]:
```
# import libraries for performing the train_test_split
from sklearn.model_selection import train_test_split
```

In [101…
```
x_train,x_test,y_train,y_test = train_test_split(df_encoded_1.drop(columns=['Exitec
```

In [102…
```
x_train
```

Out[102]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimate
4322	508	31	8	72541.48	1	1	0	12
955	706	44	4	129605.99	1	0	0	6
4019	620	31	2	166833.86	2	1	1	13
8314	643	33	4	0.00	2	1	1	15
9272	739	42	2	141642.92	2	1	0	17
...	...	...	...	...	...	...	...	
9841	567	46	1	68238.51	2	1	1	10
3352	591	40	2	99886.42	2	1	1	8
1645	506	41	3	57745.76	1	1	0	
7531	692	40	6	163505.16	1	0	0	9
7554	706	30	6	87609.68	2	0	0	13

8000 rows × 13 columns

In [103…
```
x_test
```

Out[103]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimate
6759	705	92	1	126076.24	2	1	1	[illegible]
2177	667	24	4	0.00	2	0	1	[illegible]
818	497	27	9	75263.16	1	1	1	1(
5655	695	63	1	146202.93	1	1	1	1:
8531	723	30	1	0.00	3	1	0	1(
...	...	...	...	...	...	...	...	
886	739	38	0	128366.44	1	1	0	[illegible]
2900	626	26	8	148610.41	3	0	1	1(
2717	775	70	6	119684.88	2	1	1	[illegible]
7672	555	30	1	0.00	2	0	0	[illegible]
7838	569	32	8	145330.43	1	1	1	1:

2000 rows × 13 columns

In [104…

```
# feature scaling
from sklearn.tree import DecisionTreeClassifier
```

In [105…

```
clf = DecisionTreeClassifier(max_depth=5, random_state=42)
```

In [108…

```
clf.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:1688: Futur
eWarning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[108]:

```
DecisionTreeClassifier(max_depth=5, random_state=42)
```

In [109…

```
predict = clf.predict(x_test)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:1688: Futur
eWarning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [111…

```
from sklearn.metrics import accuracy_score, classification_report
print("Accuracy:", accuracy_score(y_test,predict)*100)
```

```
Accuracy: 84.39999999999999
```

In [ ]: