Assignment 2 Solution

Import necessary libraries

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Load the dataset

In [2]:

```
df = pd.read_csv("https://drive.google.com/uc?id=1_HcM0K8wt4b7FMLkc1V1dv0y6I_9ULzy")
df
```

Out[2]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

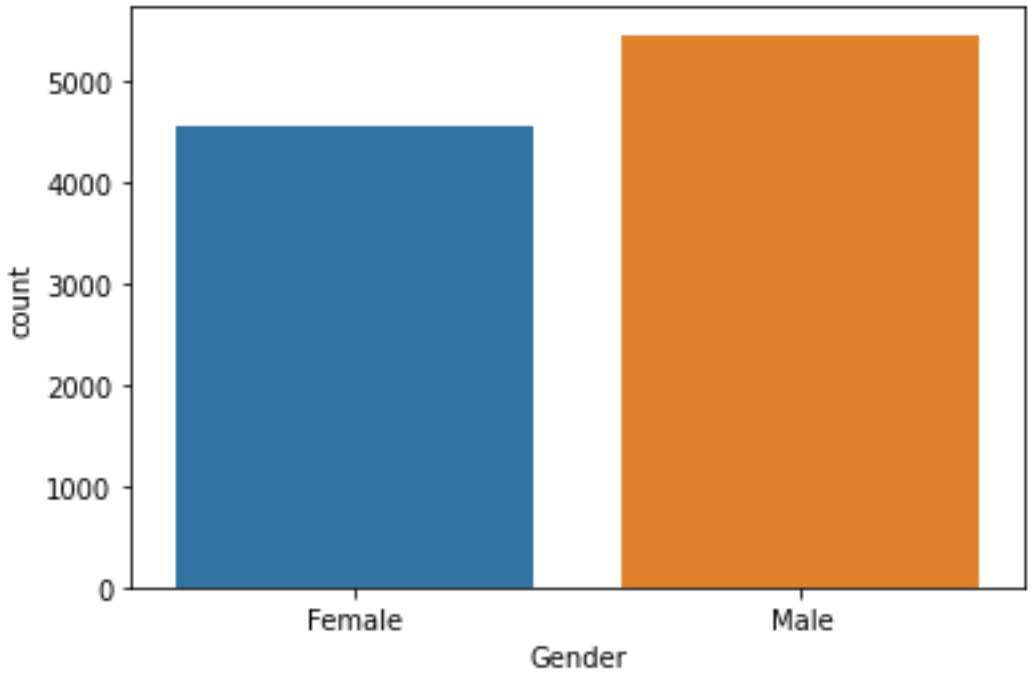
10000 rows × 14 columns

Visualizations

Uni-Variate Analysis

In [3]:

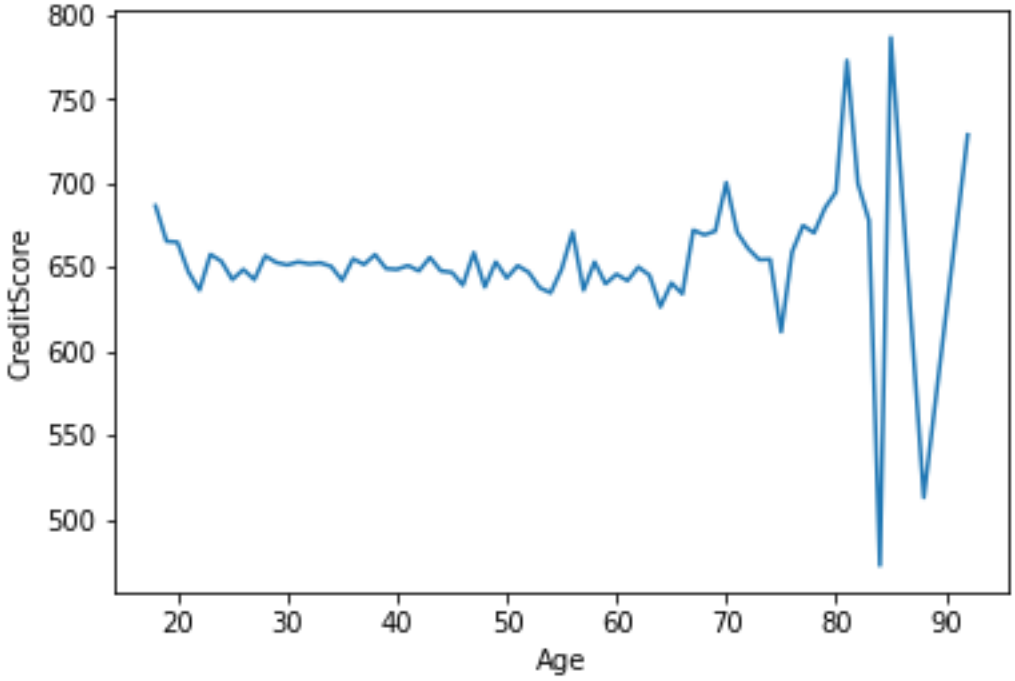
```
sns.countplot(x=df["Gender"])
plt.show()
```



Bi-Variate Analysis

In [4]:

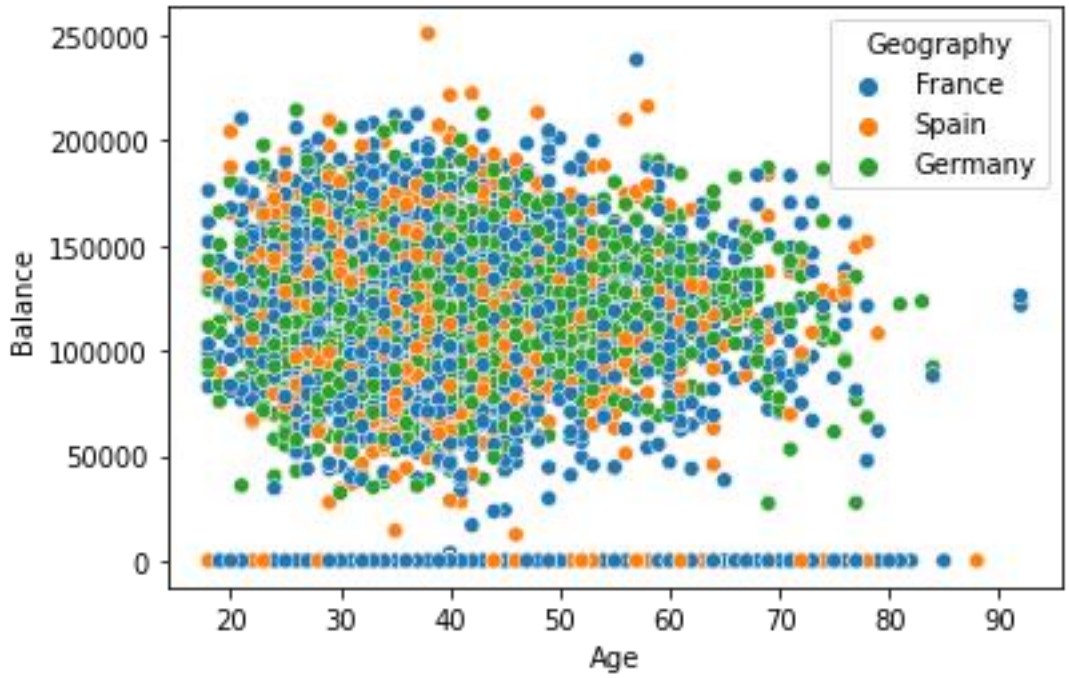
```
sns.lineplot(x="Age", y="CreditScore", data=df, ci=None)
plt.show()
```



Multi-Variate Analysis

In [5]:

```
sns.scatterplot(x="Age", y="Balance", hue="Geography", data=df)
plt.show()
```



Descriptive Statistics

In [6]:

```
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 [7]:

```
df.describe(include="all")
```

Out[7]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000	10000.000000	10000	10000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
unique	NaN	NaN	2932	NaN	3	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	NaN	Smith	NaN	France	Male	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	NaN	32	NaN	5014	5457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	5000.500000	1.569094e+07	NaN	650.528800	NaN	NaN	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	NaN	96.653299	NaN	NaN	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.000000	1.556570e+07	NaN	350.000000	NaN	NaN	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.750000	1.562853e+07	NaN	584.000000	NaN	NaN	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.500000	1.569074e+07	NaN	652.000000	NaN	NaN	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.250000	1.575323e+07	NaN	718.000000	NaN	NaN	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
max	10000.00000	1.581569e+07	NaN	850.000000	NaN	NaN	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

Handle missing values

In [8]:

```
df.isnull().sum()
```

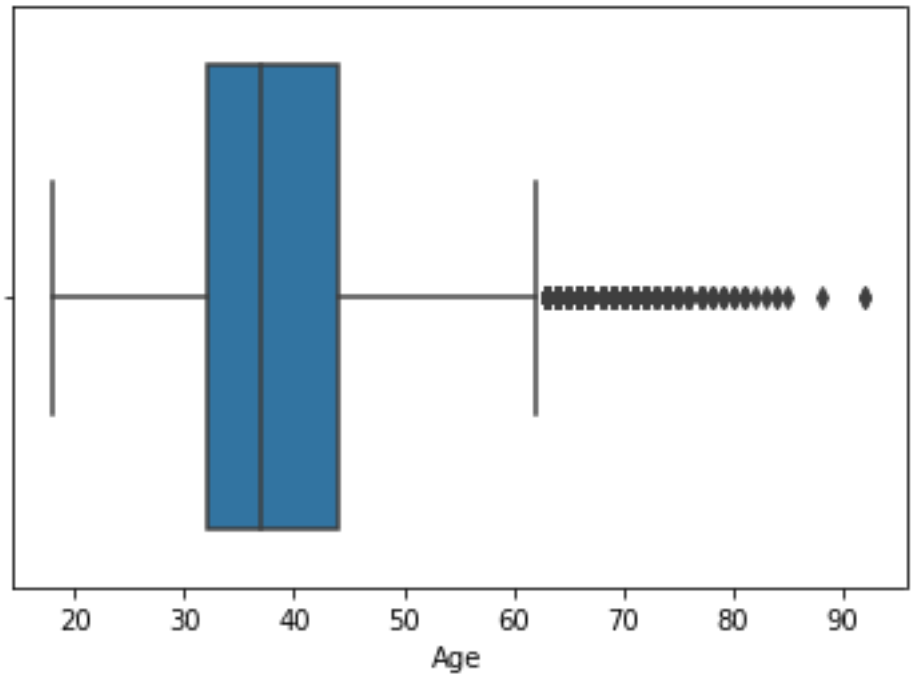
Out[8]:

```
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
```

Handle Outliers

In [9]:

```
sns.boxplot(x=df['Age'])
plt.show()
```



In [10]:

```
df = df[df["Age"] < 60].copy(deep=False)
```

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9474 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        9474 non-null   int64  
 1   CustomerId       9474 non-null   int64  
 2   Surname          9474 non-null   object 
 3   CreditScore      9474 non-null   int64  
 4   Geography        9474 non-null   object 
 5   Gender           9474 non-null   object 
 6   Age              9474 non-null   int64  
 7   Tenure           9474 non-null   int64  
 8   Balance          9474 non-null   float64
 9   NumOfProducts    9474 non-null   int64  
 10  HasCrCard        9474 non-null   int64  
 11  IsActiveMember   9474 non-null   int64  
 12  EstimatedSalary  9474 non-null   float64
 13  Exited           9474 non-null   int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Categorical label encoding

In [12]:

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
```

```
df["Geography"] = encoder.fit_transform(df["Geography"])
df["Gender"] = encoder.fit_transform(df["Gender"])

df
```

Out[12]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	2	0	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	0	1	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	0	1	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	0	0	36	7	0.00	1	0	1	42085.58	1

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9998	9999	15682355	Sabbatini	772	1	1	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	0	0	28	4	130142.79	1	1	0	38190.78	0

9474 rows × 14 columns

Split data into dependent and independent variables

In [13]:

```
X = df.iloc[:, 3:13]
X
```

Out[13]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	0	0	42	2	0.00	1	1	1	101348.88
1	608	2	0	41	1	83807.86	1	0	1	112542.58
2	502	0	0	42	8	159660.80	3	1	0	113931.57
3	699	0	0	39	1	0.00	2	0	0	93826.63
4	850	2	0	43	2	125510.82	1	1	1	79084.10
...	...	...	...	...	...	...	...	...	...	...
9995	771	0	1	39	5	0.00	2	1	0	96270.64

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
9996	516	0	1	35	10	57369.61	1	1	1	101699.77
9997	709	0	0	36	7	0.00	1	0	1	42085.58
9998	772	1	1	42	3	75075.31	2	1	0	92888.52
9999	792	0	0	28	4	130142.79	1	1	0	38190.78

9474 rows × 10 columns

In [14]:

```
y = df.iloc[:, 13]
y
```

Out[14]:

```
0       1
1       0
2       1
3       0
4       0
       ..
9995    0
9996    0
9997    1
9998    1
9999    0
Name: Exited, Length: 9474, dtype: int64
```

Scale independent variables

In [15]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X = scaler.fit_transform(X)

X
```

Out[15]:

```
array([[0.538     , 0.        , 0.        , ..., 1.        , 1.        ,
        0.50673489],
       [0.516     , 1.        , 0.        , ..., 0.        , 1.        ,
        0.56270874],
```

```
[0.304     , 0.        , 0.        , ..., 1.        , 0.        ,
 0.56965435],
...,
[0.718     , 0.        , 0.        , ..., 0.        , 1.        ,
 0.21039009],
[0.844     , 0.5       , 1.        , ..., 1.        , 0.        ,
 0.46442905],
[0.884     , 0.        , 0.        , ..., 1.        , 0.        ,
 0.19091423]])
```

Split the data into train and test data

In [16]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```