

Assignment -2

Assignment Date	25 September 2022
Student Name	Dharmaraj R
Student Roll Number	811519104023
Maximum Marks	2 Marks

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

1 Load the dataset

```
[3]: data=pd.read_csv("Churn_Modelling.csv")
data.head()
```

```
[3]: RowNumber CustomerId Surname CreditScore Geography Gender Age
```

```
\
```

0	1	15634602	Hargrave	619	France	Female	42
1	2	15647311	Hill	608	Spain	Female	41
2	3	15619304	Onio	502	France	Female	42
3	4	15701354	Boni	699	France	Female	39
4	5	15737888	Mitchell	850	Spain	Female	43

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	
0	2	0.00	1	1	1	
1	1	83807.86	1	0	1	
2	8	159660.80	3	1	0	
3	1	0.00	2	0	0	
4	2	125510.82	1	1	1	

	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1
3	93826.63	0
4	79084.10	0

```
[4]: data.info()
```

```
<class
'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to
9999 Data columns (total 14
columns):
```

	#	Column	Non-Null Count	Dtype
0	10000	RowNumber	non-null	int64
1	10000	CustomerId	non-null	int64
2	10000	Surname	non-null	object
3	10000	CreditScore	non-null	int64
4	10000	Geography	non-null	object
5	10000	Gender	non-null	object
6	10000	Age	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

```

2 Data Cleaning/Preprocessing

Handle Missing values

```
[5]: data.isnull().sum()
```

```

[5]: 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

```

```

[6]: data["Gender"].value_counts()
    data["Gender"].replace({"Male":1,"Female":0},inplace=True)
    data["Surname"]=data["Surname"].astype(str)

```

```
[7]: data["Geography"].value_counts()
```

```

[7]: France      5014
    Germany    2509
    Spain      2477
    Name: Geography, dtype: int64

```

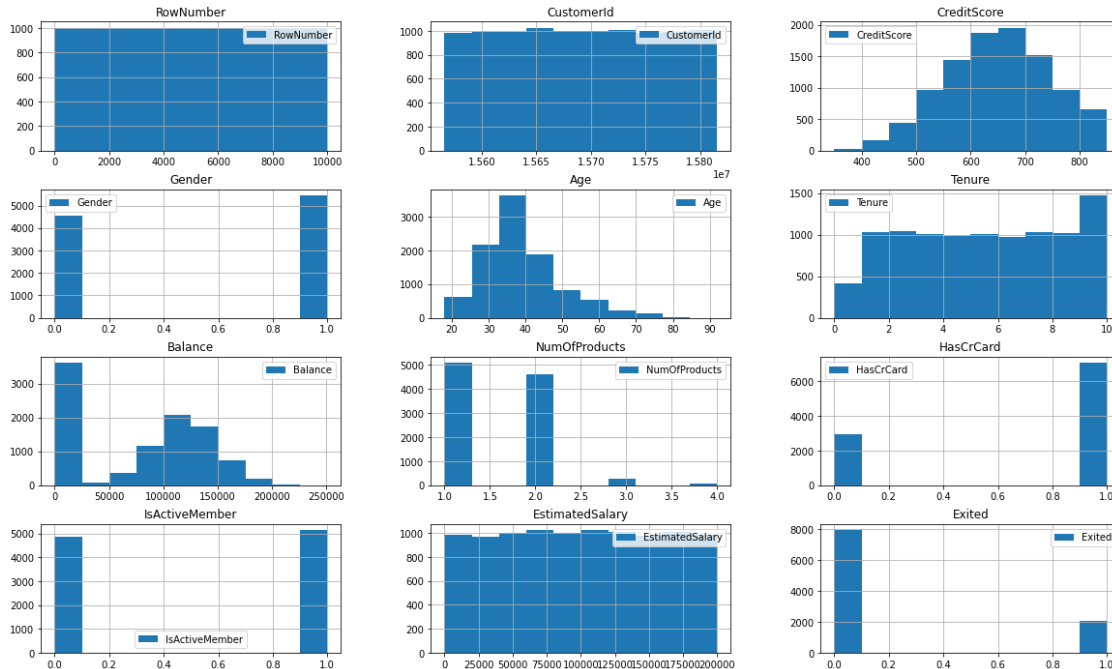
```
[8]: data["Geography"].str.strip()  
data["Surname"].str.strip()
```

```
[8]: 0      Hargrave  
1      Hill  
2      Onio  
3      Boni  
4      Mitchell  
...  
9995    Obijiaku  
9996    Johnstone  
9997    Liu  
9998    Sabbatini  
9999    Walker
```

```
Name: Surname, Length: 10000, dtype: object
```

```
[9]: data.hist(figsize=(20,12),legend=True)
```

```
[9]: array([[<AxesSubplot:title={'center':'RowNumber'}>,  
          <AxesSubplot:title={'center':'CustomerId'}>,  
          <AxesSubplot:title={'center':'CreditScore'}>],  
          [<AxesSubplot:title={'center':'Gender'}>,  
          <AxesSubplot:title={'center':'Age'}>,  
          <AxesSubplot:title={'center':'Tenure'}>],  
          [<AxesSubplot:title={'center':'Balance'}>,  
          <AxesSubplot:title={'center':'NumOfProducts'}>,  
          <AxesSubplot:title={'center':'HasCrCard'}>],  
          [<AxesSubplot:title={'center':'IsActiveMember'}>,  
          <AxesSubplot:title={'center':'EstimatedSalary'}>,  
          <AxesSubplot:title={'center':'Exited'}>]], dtype=object)
```



```
[10]: data.drop(columns=["RowNumber", "CustomerId"], inplace=True)
data.head()
```

```
[10]: Surname CreditScore Geography Gender Age Tenure Balance \
0 Hargrave 619 France 0 42 2 0.00
1 Hill 608 Spain 0 41 1 83807.86
2 Onio 502 France 0 42 8
159660.80
3 Boni 699 France 0 39 1 0.00
4 Mitchell 850 Spain 0 43 2
125510.82

NumOfProducts HasCrCard IsActiveMember EstimatedSalary Exited
0 1 1 1 101348.88 1
1 1 0 1 112542.58 0
2 3 1 0 113931.57 1
3 2 0 0 93826.63 0
4 1 1 1 79084.10 0
```

```
[11]: data["Surname"].value_counts()
```

```
[11]: Smith 32
Scott 29
Martin 29
Walker 28
Brown 26
```

```

Izmailov      1
Bold          1
Bonham        1
Poninski      1
Burbidge      1

Name: Surname, Length: 2932, dtype: int64

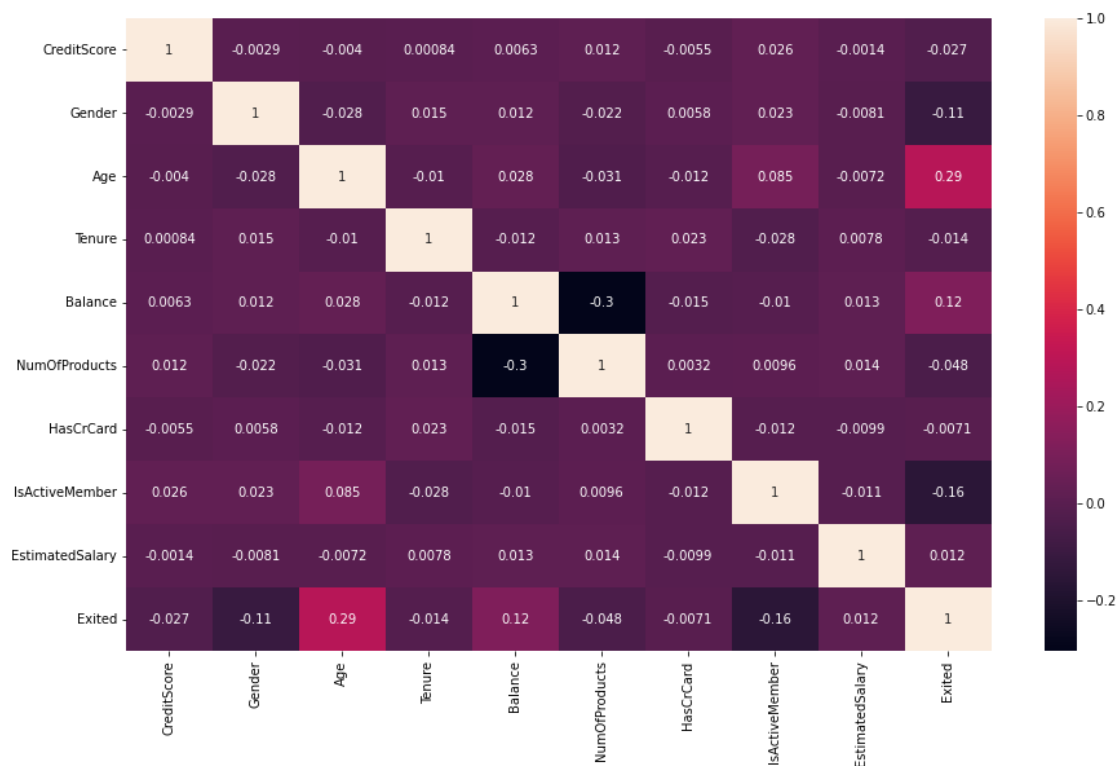
```

```
[12]: data.duplicated().value_counts()
```

```
[12]: False    10000
      dtype: int64
```

```
[13]: plt.figure(figsize=(15,9))
      sns.heatmap(data.corr(),annot=True)
```

```
[13]: <AxesSubplot:>
```



```
[14]: data.describe()
```

```
[14]:      CreditScore      Gender      Age      Tenure      Balance \
count    10000.000000    10000.000000    10000.000000    10000.000000
10000.000000  mean     650.528800     0.545700     38.921800     5.012800
76485.889288 std     96.653299     0.497932    10.487806     2.892174
62397.405202 min     350.000000     0.000000     18.000000     0.000000
0.000000  25%     584.000000     0.000000     32.000000     3.000000
0.000000
```

50%	652.000000	1.000000	37.000000	5.000000	97198.540000
75%	718.000000	1.000000	44.000000	7.000000	127644.240000
max	850.000000	1.000000	92.000000	10.000000	250898.090000

	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
count	10000.000000	10000.000000	10000.000000	10000.000000
mean	1.530200	0.70550	0.515100	100090.239881
std	0.581654	0.45584	0.499797	57510.492818
min	1.000000	0.00000	0.000000	11.580000
25%	1.000000	0.00000	0.000000	51002.110000
50%	1.000000	1.00000	1.000000	100193.915000
75%	2.000000	1.00000	1.000000	149388.247500
max	4.000000	1.00000	1.000000	199992.480000

	Exited
count	10000.000000
mean	0.203700
std	0.402769
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

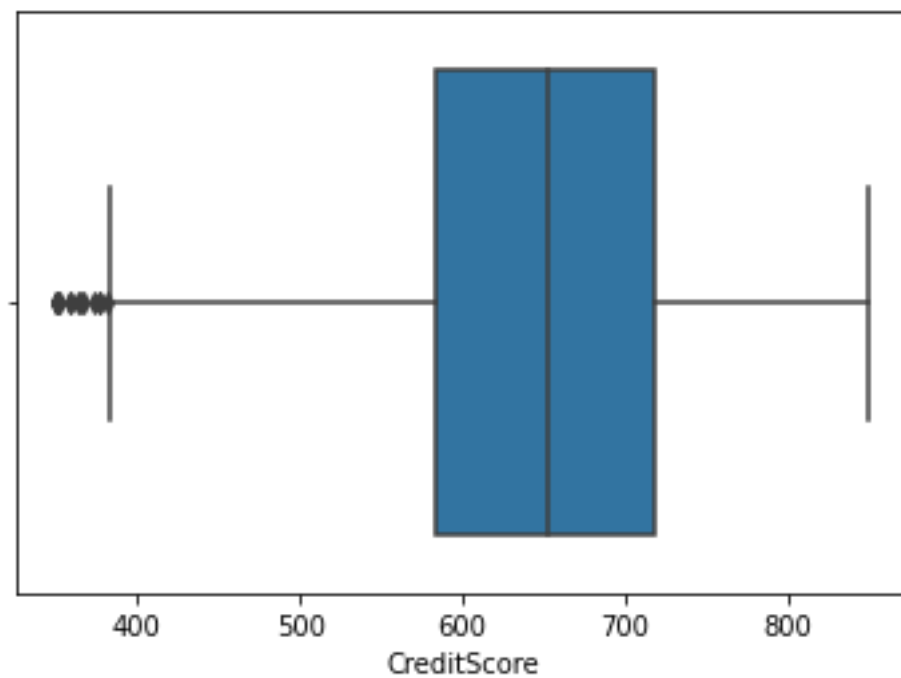
3 EDA analysis

Univariate Data Visualization

```
[15]: sns.boxplot(data["CreditScore"])
```

```
c:\users\arvin\appdata\local\programs\python\python39\lib\sitepackages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. warnings.warn(
```

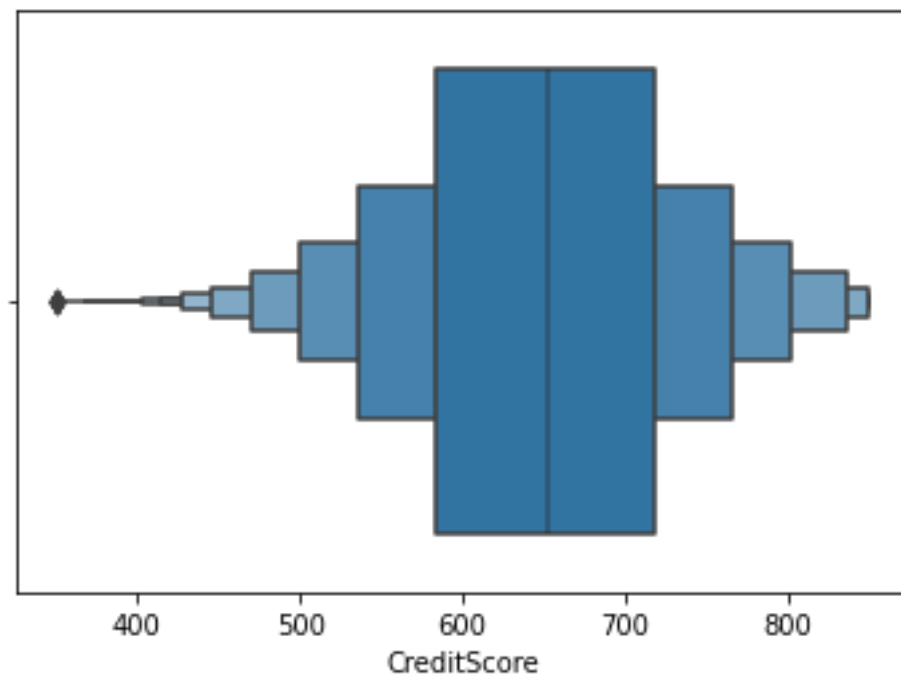
```
[15]: <AxesSubplot:xlabel='CreditScore'>
```



```
[16]: sns.boxenplot(data.CreditScore)
```

c:\users\arvin\appdata\local\programs\python\python39\lib\sitepackages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. warnings.warn(

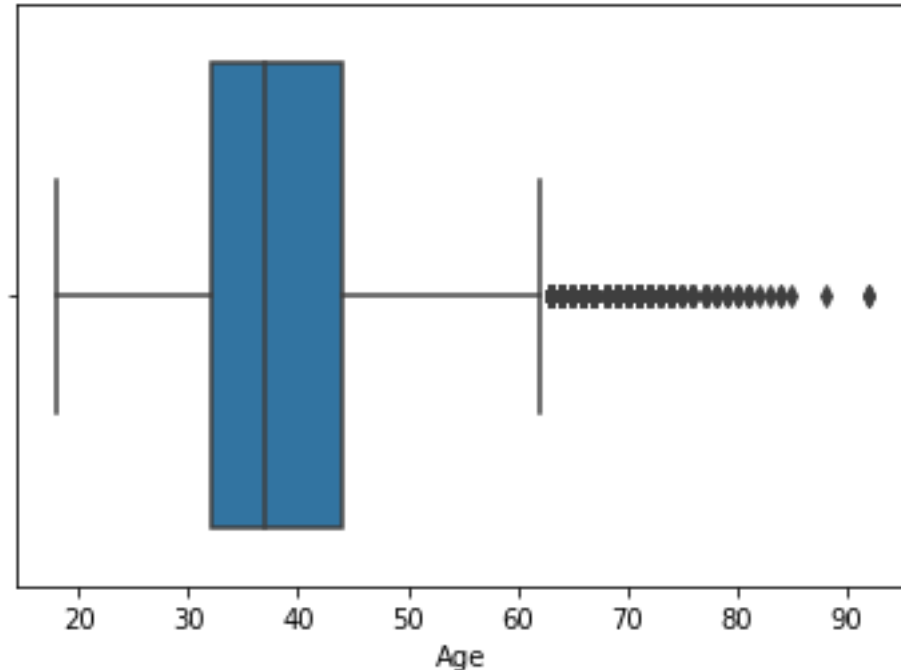
```
[16]: <AxesSubplot:xlabel='CreditScore'>
```




```
[17]: sns.boxplot(data["Age"])
```

```
c:\users\arvin\appdata\local\programs\python\python39\lib\sitepackages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. warnings.warn(
```

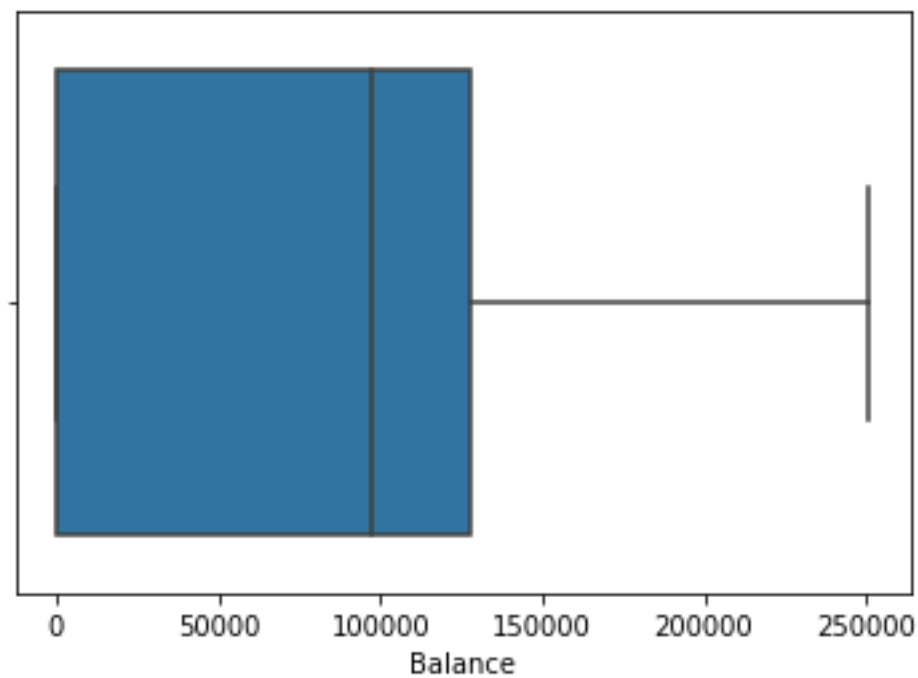
```
[17]: <AxesSubplot:xlabel='Age'>
```



```
[18]: sns.boxplot(data["Balance"])
```

```
c:\users\arvin\appdata\local\programs\python\python39\lib\sitepackages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. warnings.warn(
```

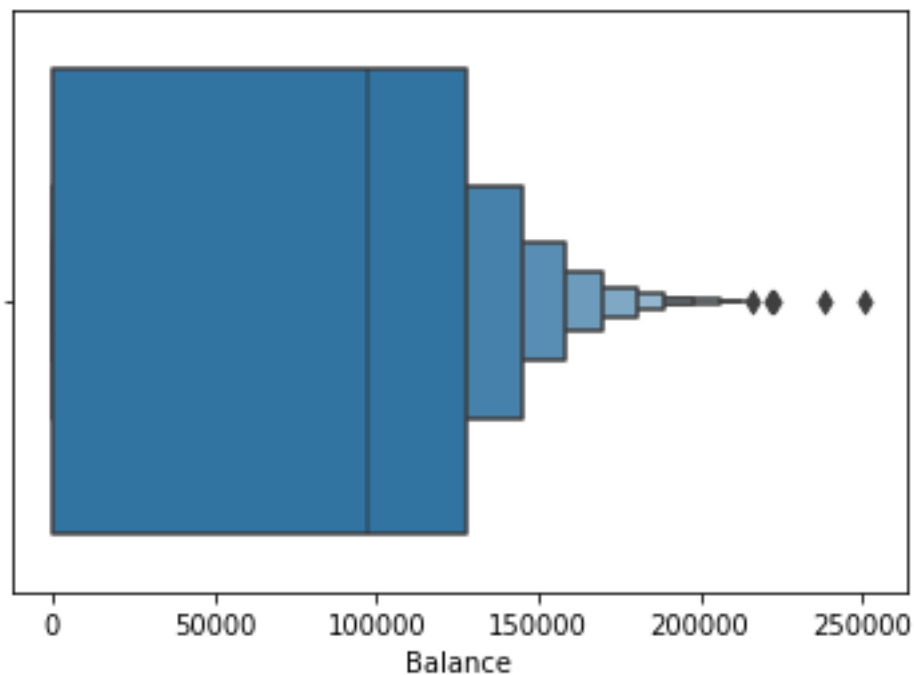
```
[18]: <AxesSubplot:xlabel='Balance'>
```



```
[19]: sns.boxenplot(data.Balance)
```

c:\users\arvin\appdata\local\programs\python\python39\lib\sitepackages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. warnings.warn(

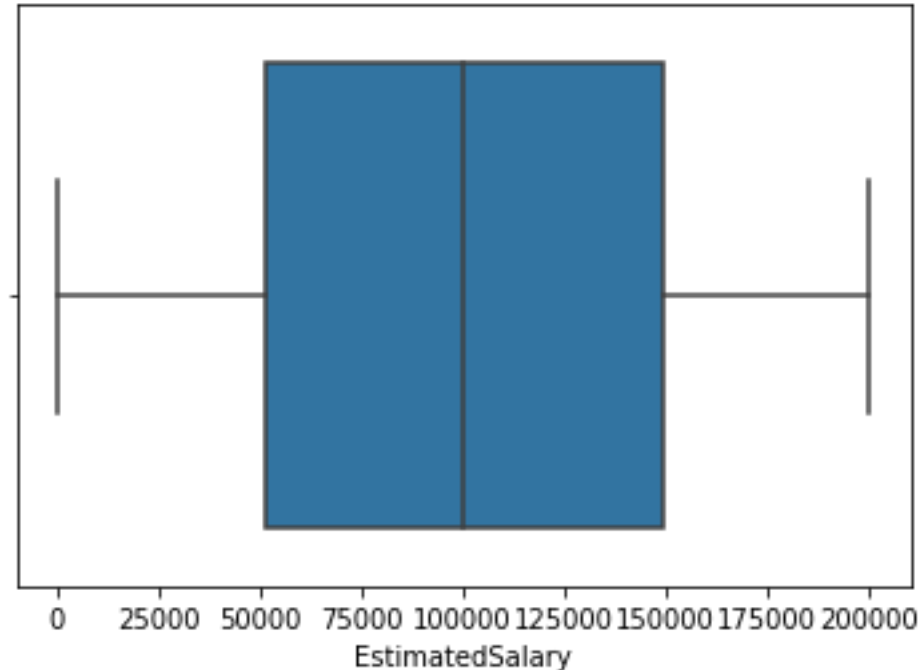
```
[19]: <AxesSubplot:xlabel='Balance'>
```



```
[21]: sns.boxplot(data.EstimatedSalary)
```

```
c:\users\arvin\appdata\local\programs\python\python39\lib\sitepackages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. warnings.warn(
```

```
[21]: <AxesSubplot:xlabel='EstimatedSalary'>
```

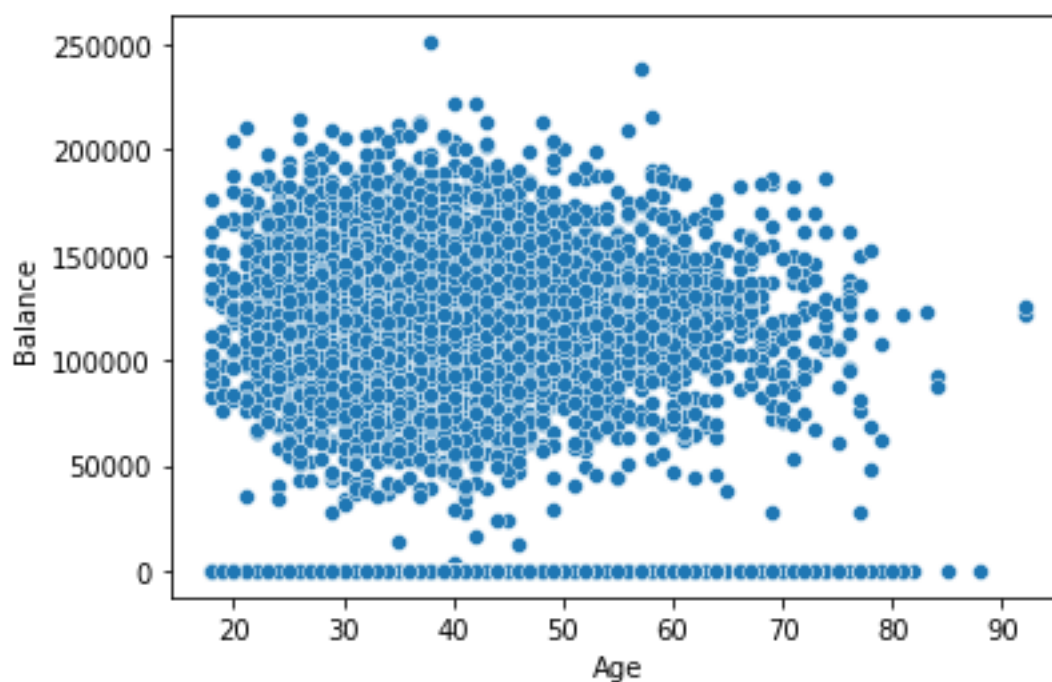


4 Bi-variate Data Visualization

```
[22]: sns.scatterplot(data.Age, data.Balance)
```

```
c:\users\arvin\appdata\local\programs\python\python39\lib\sitepackages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. warnings.warn(
```

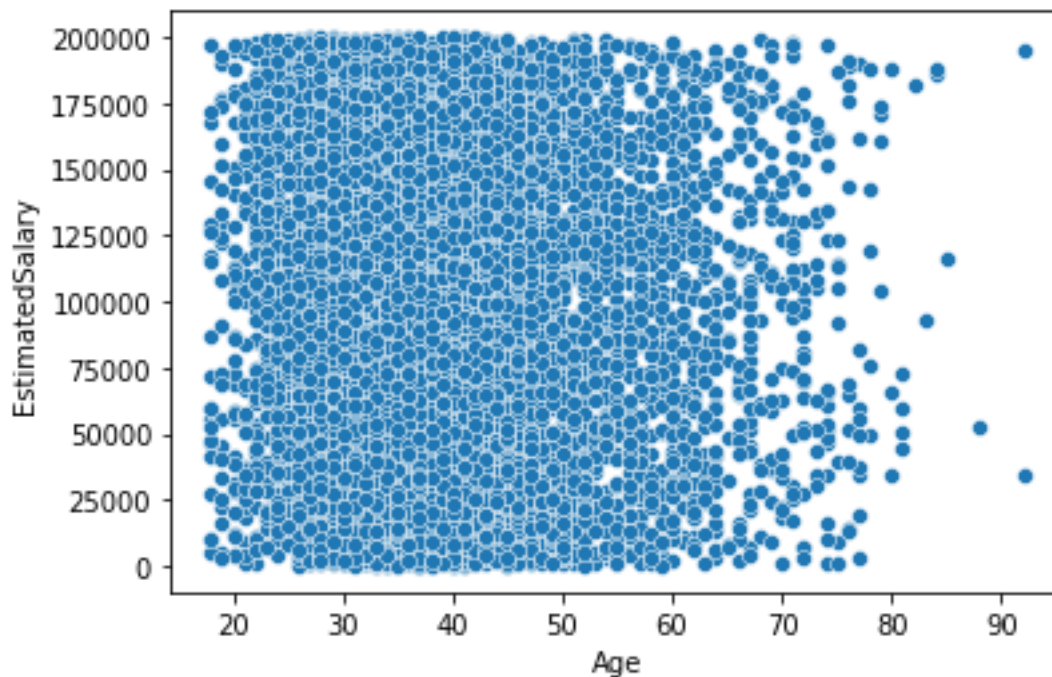
```
[22]: <AxesSubplot:xlabel='Age', ylabel='Balance'>
```



```
[23]: sns.scatterplot(data.Age,data.EstimatedSalary)
```

c:\users\arvin\appdata\local\programs\python\python39\lib\sitepackages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. warnings.warn(

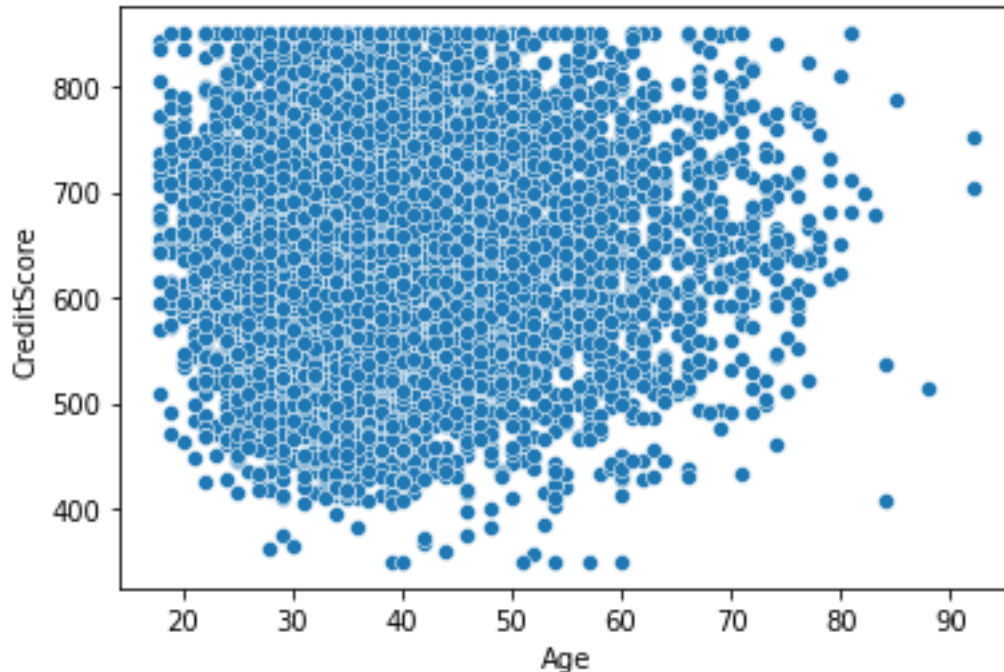
```
[23]: <AxesSubplot:xlabel='Age', ylabel='EstimatedSalary'>
```



```
[24]: sns.scatterplot(data.Age, data.CreditScore)
```

```
c:\users\arvin\appdata\local\programs\python\python39\lib\sitepackages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. warnings.warn(
```

```
[24]: <AxesSubplot:xlabel='Age', ylabel='CreditScore'>
```



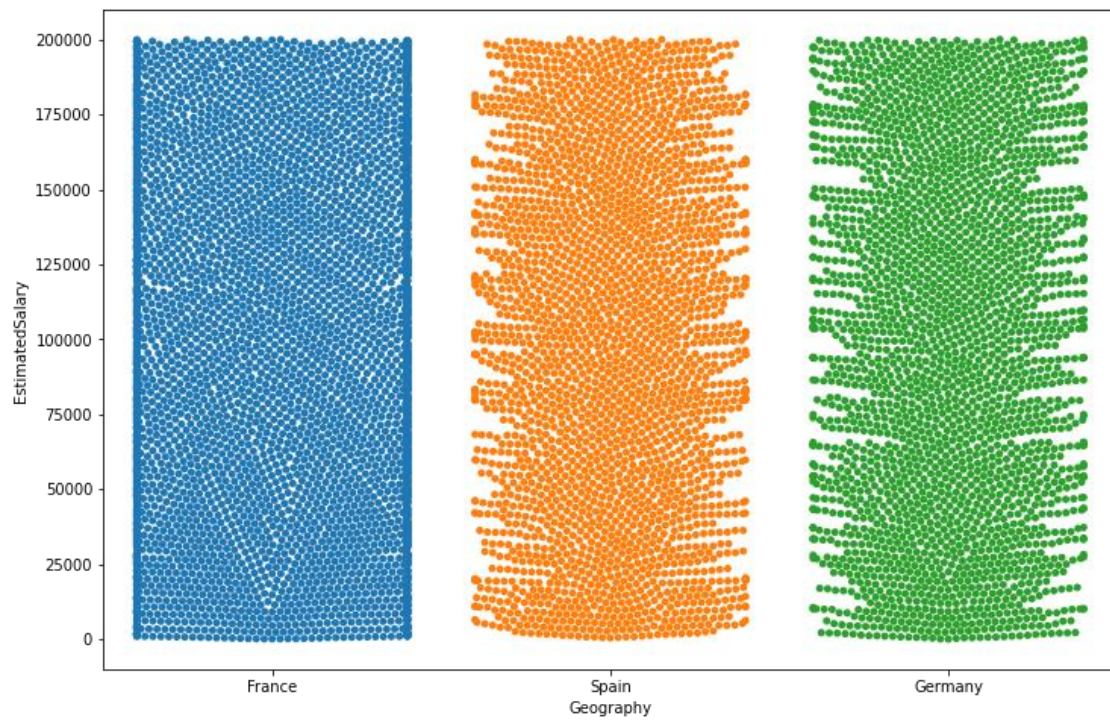
```
[25]: plt.figure(figsize=(12,8))
sns.swarmplot(data.Geography, data.EstimatedSalary, data=data)
```

```
c:\users\arvin\appdata\local\programs\python\python39\lib\sitepackages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
warnings.warn(
```

```
c:\users\arvin\appdata\local\programs\python\python39\lib\sitepackages\seaborn\categorical.py:1296: UserWarning: 43.3% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
warnings.warn(msg, UserWarning)
```

```
c:\users\arvin\appdata\local\programs\python\python39\lib\sitepackages\seaborn\categorical.py:1296: UserWarning: 5.5% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot. warnings.warn(msg, UserWarning)
```

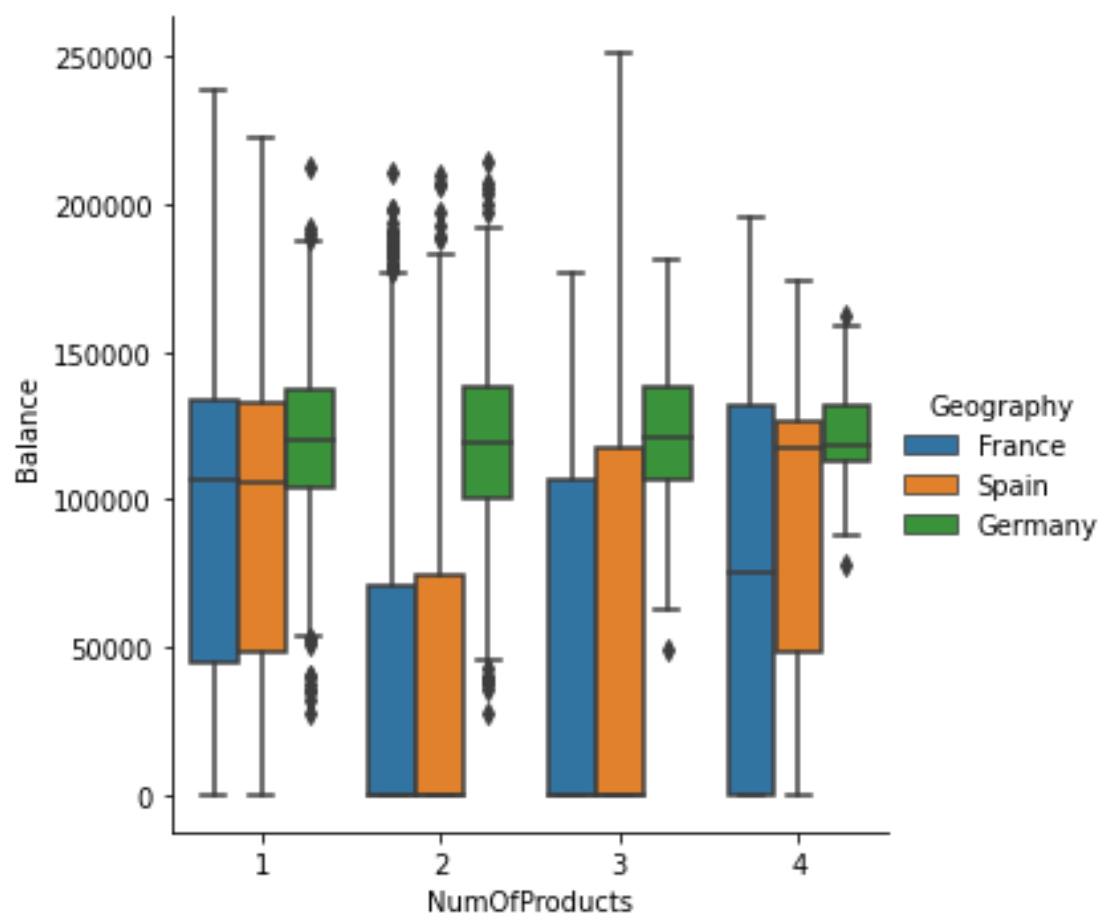
```
[25]: <AxesSubplot:xlabel='Geography', ylabel='EstimatedSalary'>
```



4.1 Multivariate Datavisualization

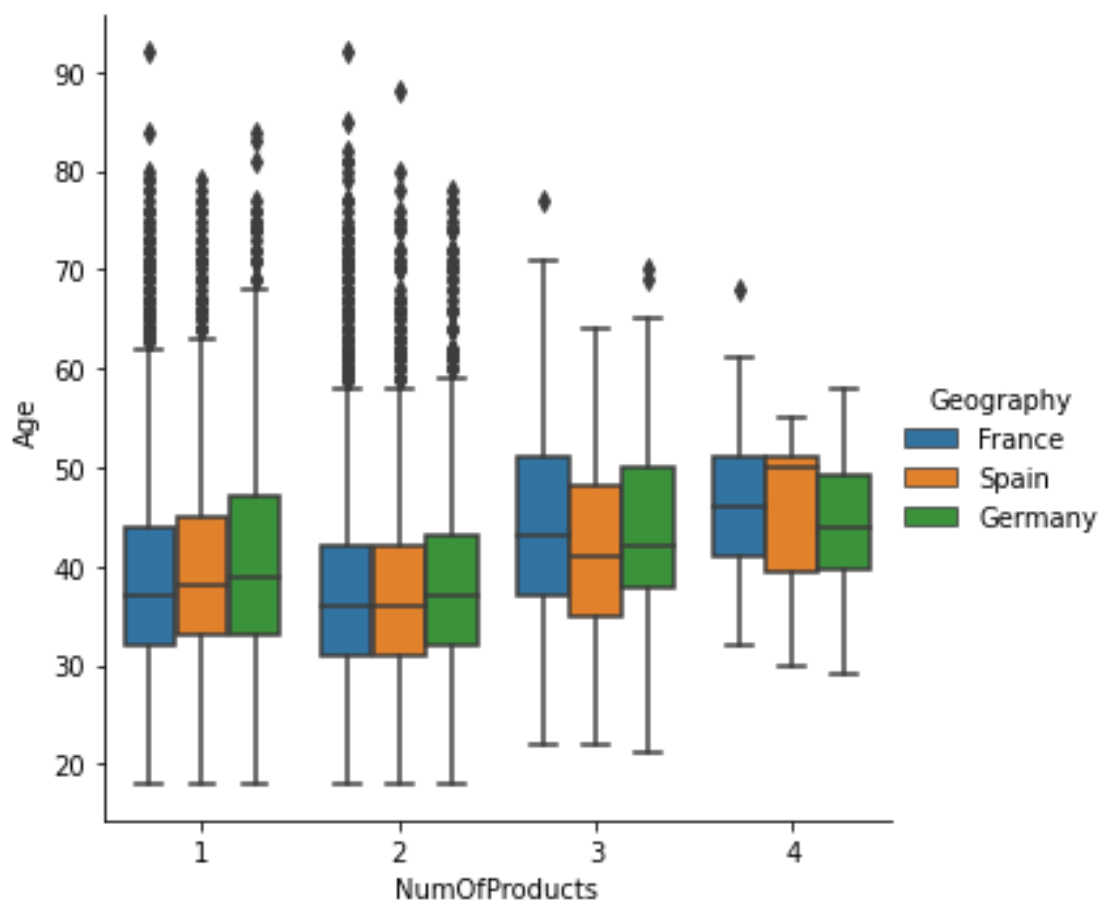
```
[26]: sns.catplot(x="NumOfProducts", y="Balance", data=data, hue="Geography", kind='box')
```

```
[26]: <seaborn.axisgrid.FacetGrid at 0x188ce353790>
```



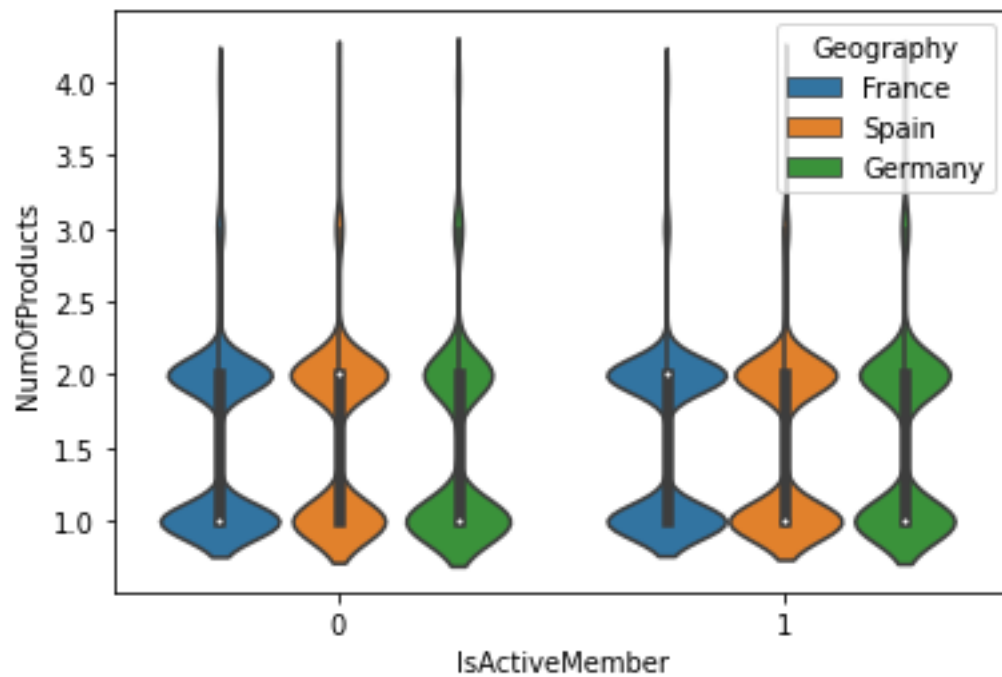
```
[27]: sns.catplot(x="NumOfProducts",y="Age",data=data,hue="Geography",kind='box')
```

```
[27]: <seaborn.axisgrid.FacetGrid at 0x188cf3fed90>
```

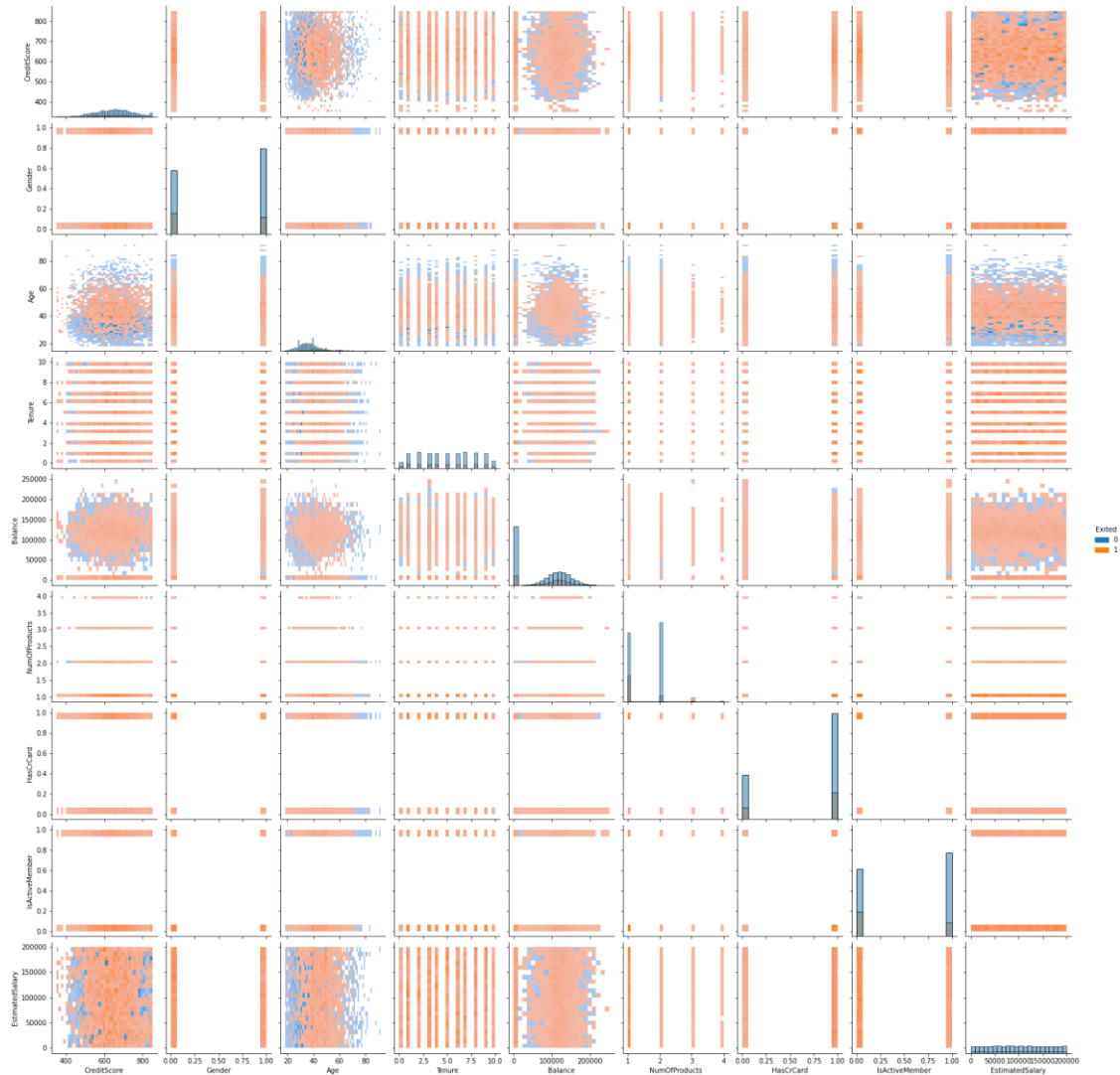
```
[28]: sns.violinplot(x="IsActiveMember", y="NumOfProducts", data=data, hue="Geography")
```

```
[28]: <AxesSubplot:xlabel='IsActiveMember', ylabel='NumOfProducts'>
```



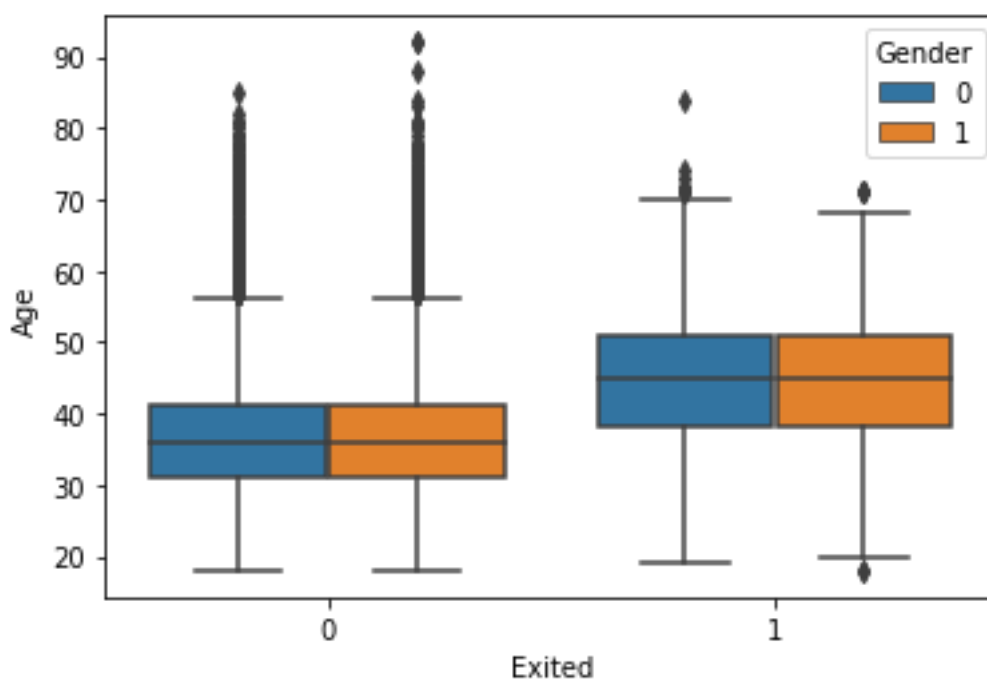

```
[29]: sns.pairplot(data=data, hue="Exited", kind='hist')
```

```
[29]: <seaborn.axisgrid.PairGrid at 0x188ce3a7ca0>
```



```
[30]: sns.boxplot(x="Exited", y="Age", data=data, hue="Gender")
```

```
[30]: <AxesSubplot:xlabel='Exited', ylabel='Age'>
```



5 Identify remove outliers

```
[31]: q1=data["Age"].quantile(0.25)
      q3=data["Age"].quantile(0.75)
      iq=q3-q1
      data=data[~((data["Age"]<(q1-1.5*iq))|(data["Age"]>(q3+1.5*iq)))]
```

```
[32]: q1=data["CreditScore"].quantile(0.25)
      q3=data["CreditScore"].quantile(0.75) iq=q3-q1
      data=data[~((data["CreditScore"]<(q1-1.5*iq))|(data["CreditScore"]>(q3+1.5*iq)))]
```

```
[33]: data[(data["Age"]<(data["Age"].mean()-3*data["Age"].std()))|(data["Age"]>(data["Age"].mean()+3*data["Age"].std()))]
```

```
[33]: Empty DataFrame
      Columns: [Surname, CreditScore, Geography, Gender, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary, Exited]
      Index: []
```

6 encoding Geography column by dummy variable technique

```
[34]: geo=pd.get_dummies(data["Geography"],drop_first=True)
      data1=pd.concat([data,geo],axis=1)
      data1
```

```
[34]:   Surname  CreditScore  Geography  Gender  Age  Tenure  Balance \
0  Hargrave         619    France      0    42      2      0.00
```

1	Hill	608	Spain	0	41	1	83807.86
2	Onio	502	France	0	42	8	159660.80
3	Boni	699	France	0	39	1	0.00
4	Mitchell	850	Spain	0	43	2	125510.82
...
9995	Obijiaku	771	France	1	39	5	0.00
9996	Johnstone	516	France	1	35	10	57369.61
9997	Liu	709	France	0	36	7	0.00
9998	Sabbatini	772	Germany	1	42	3	75075.31
9999	Walker	792	France	0	28	4	130142.79
NumOfProducts HasCrCard IsActiveMember EstimatedSalary Exited \							
0		1	1	1	101348.88	1	
1		1	0	1	112542.58	0	
2		3	1	0	113931.57	1	
3		2	0	0	93826.63	0	
4		1	1	1	79084.10	0	
...
9995		2	1	0	96270.64	0	
9996		1	1	1	101699.77	0	
9997		1	0	1	42085.58	1	
9998		2	1	0	92888.52	1	
9999		1	1	0	38190.78	0	

	Germany	Spain
0	0	0
1	0	1
2	0	0
3	0	0
4	0	1
...
9995	0	0
9996	0	0
9997	0	0
9998	1	0
9999	0	0

[9627 rows x 14 columns]

7 Split data into dependent and independent futures

```
[43]: x=data1.drop(columns=["Surname", "Geography", "Exited"])
      y=data["Exited"]
      x.head()
```

```
[43]: CreditScore Gender Age Tenure Balance NumOfProducts HasCrCard \
0      619      0 42 2      0.00 1      1
```

1	608	0	41	1	83807.86	1	0
2	502	0	42	8	159660.80	3	1
3	699	0	39	1	0.00	2	0
4	850	0	43	2	125510.82	1	1

	IsActiveMember	EstimatedSalary	Germany	Spain
0	1	101348.88	0	0
1	1	112542.58	0	1
2	0	113931.57	0	0
3	0	93826.63	0	0
4	1	79084.10	0	1

8 Scaling independent futures

```
[44]: from sklearn.preprocessing import
StandardScaler from sklearn.model_selection
import train_test_split sc=StandardScaler()
scaled_data=sc.fit_transform(x)
scaled_data=pd.
↳DataFrame(scaled_data,columns=["CreditScore","Gender","Age","Tenure","Balance",
"NumOfProduct scaled_data.head()
```

```
[44]: CreditScore  Gender      Age  Tenure  Balance  NumOfProducts \
0    -0.329901 -1.097262  0.479327 -1.044311 -1.226614  -0.914075
1    -0.444342 -1.097262  0.365664 -1.390532  0.116511  -0.914075
2    -1.547136 -1.097262  0.479327  1.033018  1.332148    2.529401
3     0.502395 -1.097262  0.138339 -1.390532 -1.226614    0.807663
4     2.073356 -1.097262  0.592990 -1.044311  0.784853  -0.914075

HasCrCard  IsActiveMember  EstimatedSalary  Germany  Spain
0  0.646875  0.992858  0.021336 -0.579629 -0.573072  1  -
1  1.545894  0.992858  0.215937 -0.579629  1.744981
2  0.646875 -1.007193  0.240084 -0.579629 -0.573072  3  -
1  1.545894 -1.007193 -0.109438 -0.579629 -0.573072
4  0.646875      0.992858      -0.365734 -0.579629  1.744981
```

9 Splitting data into train and test datasets

```
[45]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
```

```
[46]: x_train.shape,x_test.shape
```

```
[46]: ((7701, 11), (1926, 11))
```

```
[ ]:
```