Assignment -2

Data Visualization and Pre-processing

Assignment submission	27 September 2022
Student Name	AARTHI R
Student Roll Number	951919CS001
Maximum Marks	2 Marks

1. Download the dataset: Dataset

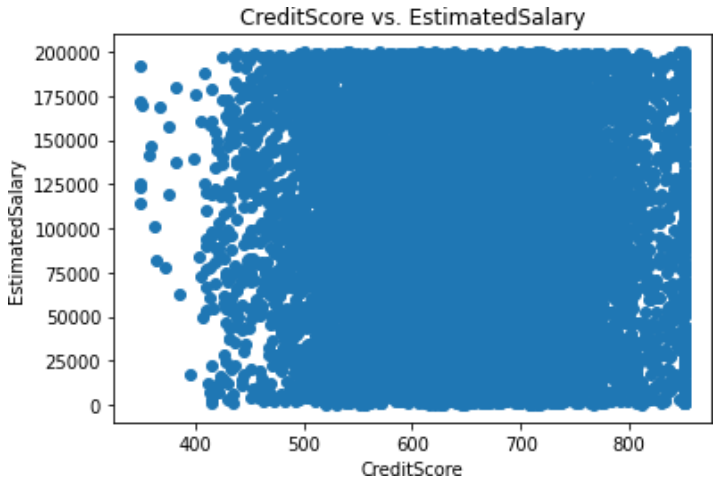
2. Load the dataset.

```
import pandas as pd
df=pd.read_csv('Churn_Modelling.csv')
```

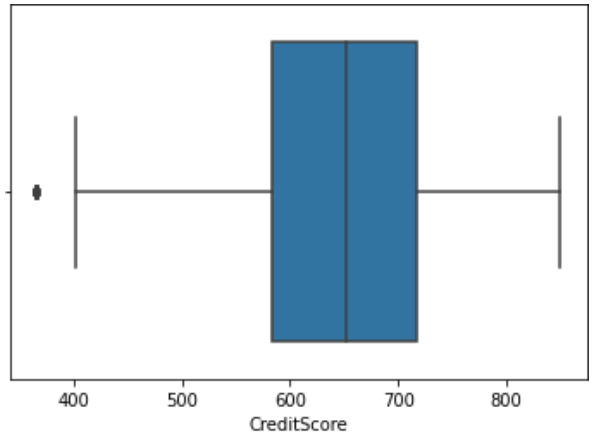
3. Perform Below Visualizations

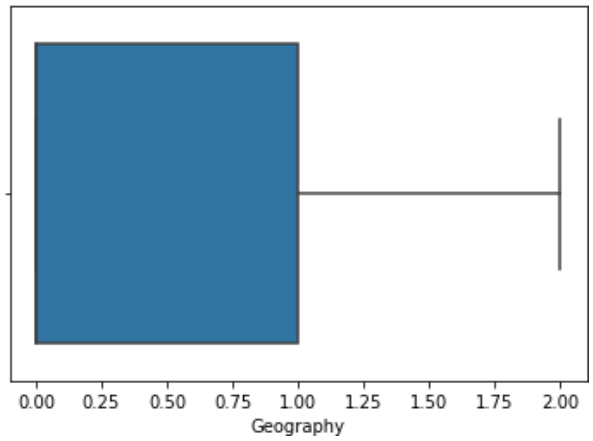
● Univariate Analysis

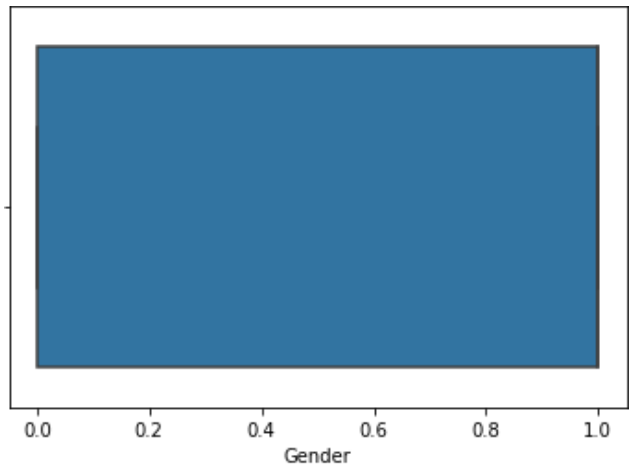
```
import matplotlib.pyplot as plt

%matplotlib inline
plt.scatter(df.CreditScore,df.EstimatedSalary)
plt.title('CreditScore vs. EstimatedSalary')
plt.xlabel('CreditScore')
plt.ylabel('EstimatedSalary')
plt.show()
```

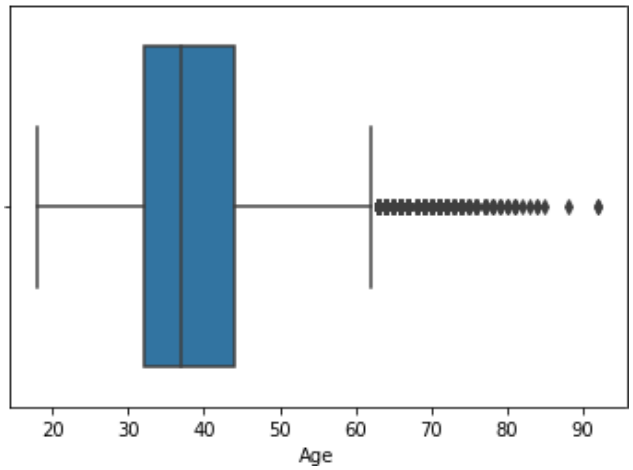


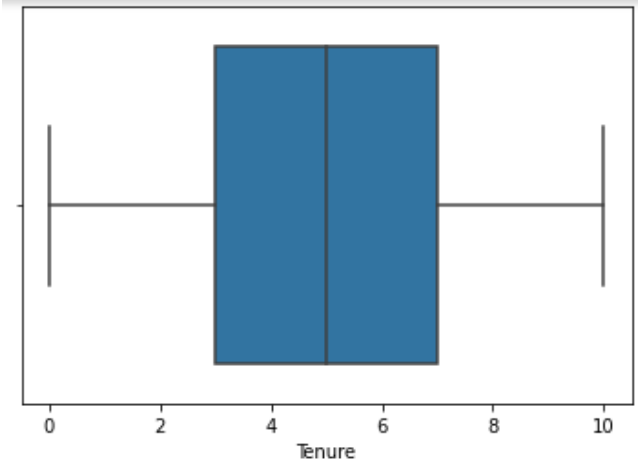

```
for col in df.columns:
  if(df.dtypes[col]=='int64' or df.dtypes[col]=='float64' ):
    sns.boxplot(x=df[col]).set( xlabel=col)
    plt.show()
```

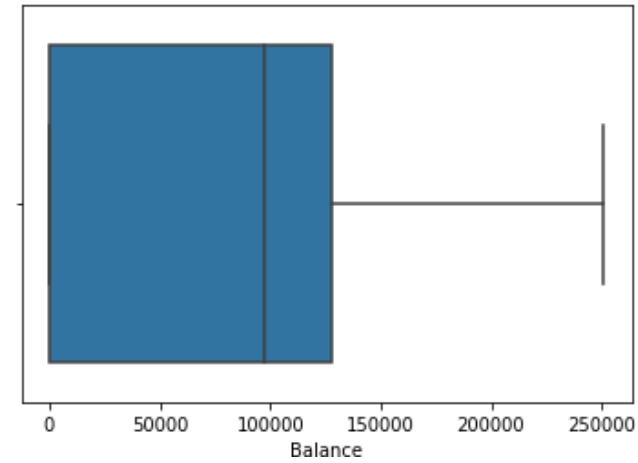

400
500
600
700
800
CreditScore

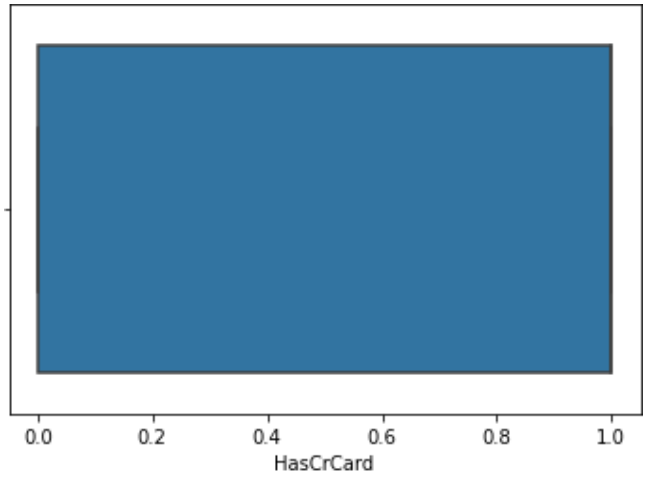

0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Geography

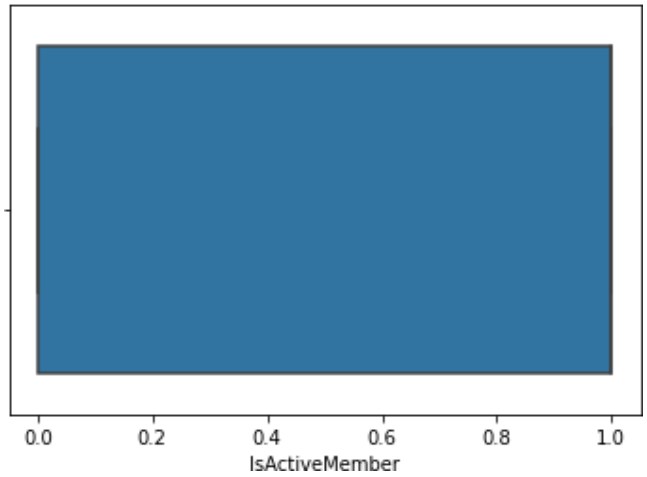

0.0
0.2
0.4
0.6
0.8
1.0
Gender

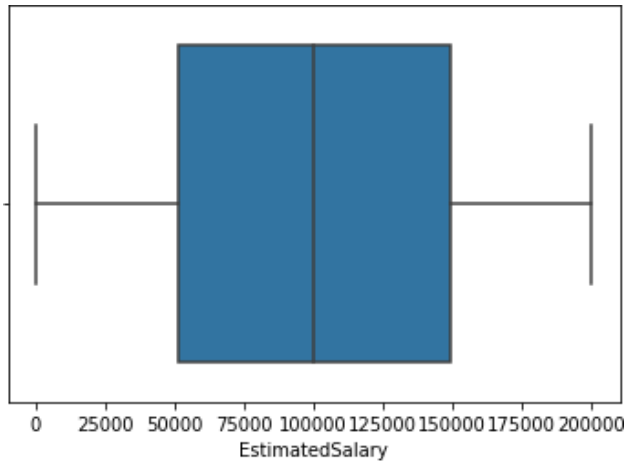

20
30
40
50
60
70
80
90
Age

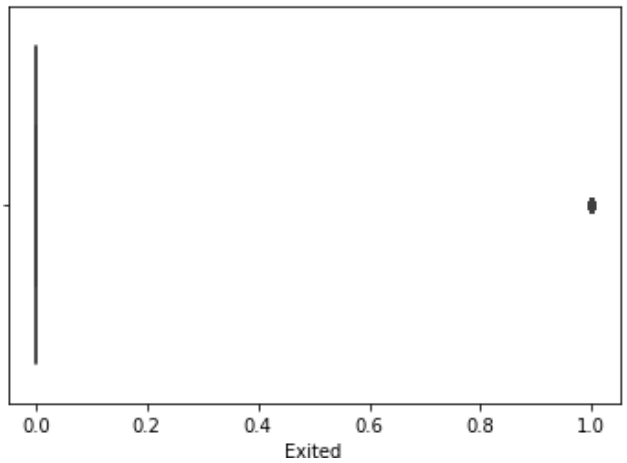

0
2
4
6
8
10
Tenure


0
50000
100000
150000
200000
250000
Balance


0.0
0.2
0.4
0.6
0.8
1.0
HasCrCard

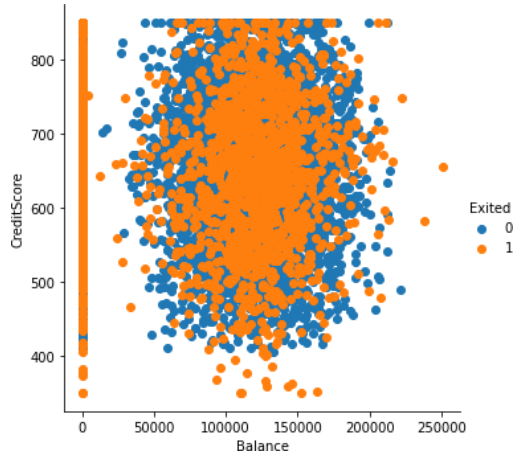

0.0
0.2
0.4
0.6
0.8
1.0
IsActiveMember





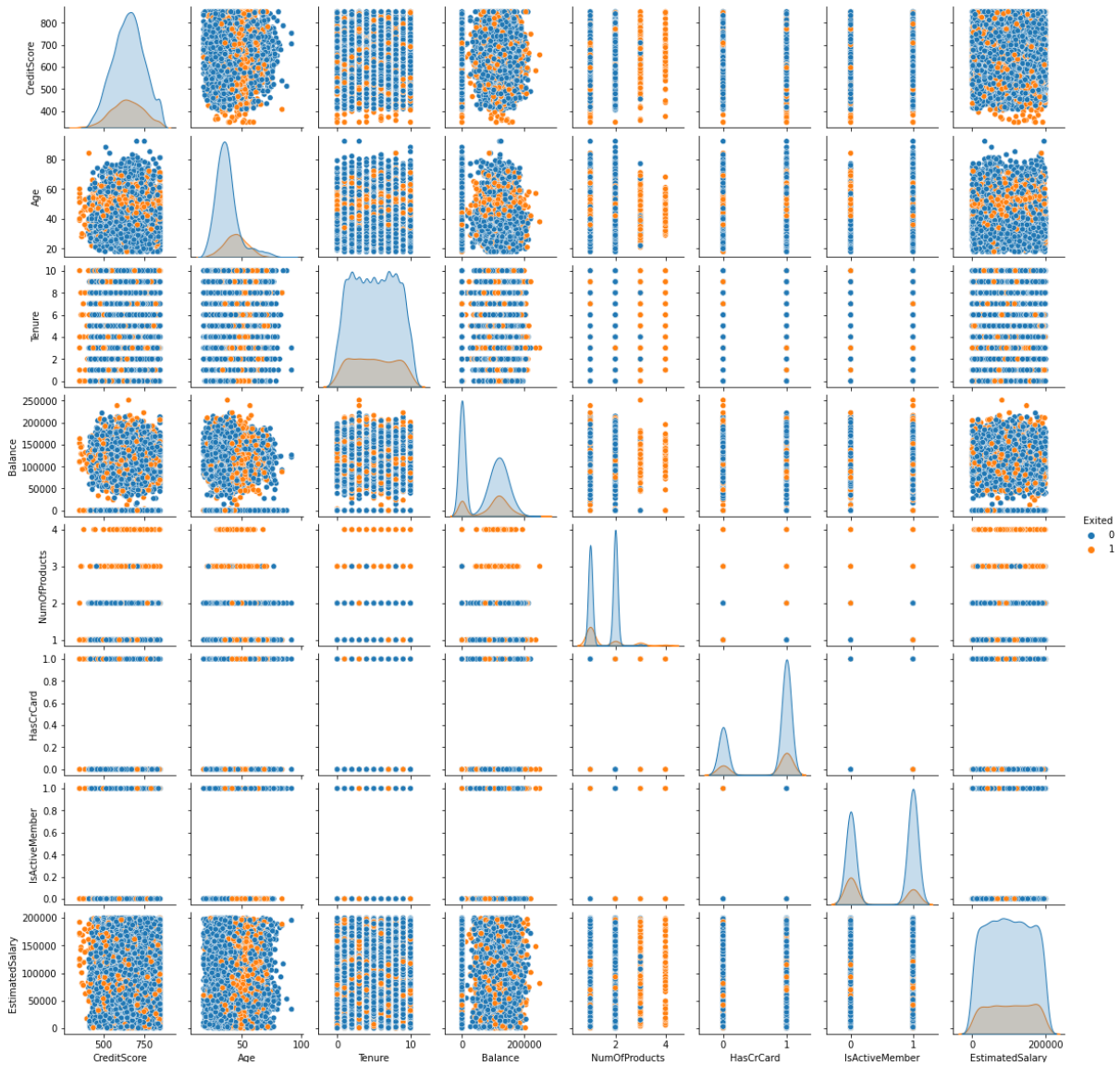
● Bi - Variate Analysis

```
import seaborn as sns
sns.FacetGrid(df,hue='Exited',height=5).map(plt.scatter,"Balance","CreditScore").add_legend()
plt.show()
```



● Multi - Variate Analysis

```
sns.pairplot(df, hue='Exited', height=2)
```



4. Perform descriptive statistics on the dataset.

df.describe()

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	9940.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	650.52400	0.746300	0.545700	38.921800	5.012800	76485.889288	1.515292	0.70550	0.515100	100090.239881	0.203700
std	96.66498	0.827529	0.497932	10.487806	2.892174	62397.405202	0.550743	0.45584	0.499797	57510.492818	0.402769
min	365.00000	0.000000	0.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	584.00000	0.000000	0.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	652.00000	0.000000	1.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	718.00000	1.000000	1.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	850.00000	2.000000	1.000000	92.000000	10.000000	250898.090000	3.000000	1.00000	1.000000	199992.480000	1.000000

5. Handle the Missing values.

df.isnull().sum()

```
CreditScore        0
Geography          0
```

```
Gender              0
Age                 0
Tenure              0
Balance             0
NumOfProducts       0
HasCrCard           0
IsActiveMember      0
EstimatedSalary     0
Exited              0
dtype: int64
```

#there is no missing values

6. Find the outliers and replace the outliers

import numpy as np
#Outliers are found using the univariate

CreditsMedian = df.loc[df['CreditScore']<400, 'CreditScore'].median()
ProdMedian = df.loc[df['NumOfProducts']>=3.5,'NumOfProducts'].median()

df.loc[df.CreditScore < 400, 'CreditScore'] = np.nan
df.fillna(CreditsMedian,inplace=True)
df.loc[df.NumOfProducts > 3, 'NumOfProducts'] = np.nan
df.fillna(ProdMedian,inplace=True)

df

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619.0	0	0	42	2	0.00	1.0	1	1	101348.88	1
1	608.0	2	0	41	1	83807.86	1.0	0	1	112542.58	0
2	502.0	0	0	42	8	159660.80	3.0	1	0	113931.57	1
3	699.0	0	0	39	1	0.00	2.0	0	0	93826.63	0
4	850.0	2	0	43	2	125510.82	1.0	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...
9995	771.0	0	1	39	5	0.00	2.0	1	0	96270.64	0
9996	516.0	0	1	35	10	57369.61	1.0	1	1	101699.77	0
9997	709.0	0	0	36	7	0.00	1.0	0	1	42085.58	1
9998	772.0	1	1	42	3	75075.31	2.0	1	0	92888.52	1
9999	792.0	0	0	28	4	130142.79	1.0	1	0	38190.78	0

10000 rows × 11 columns

7. Check for Categorical columns and perform encoding.

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

df.info()
#we have 2 categorial information
```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  --------         --------------- -------
 0   CreditScore      10000 non-null  int64
 1   Geography        10000 non-null  object
 2   Gender           10000 non-null  object
 3   Age              10000 non-null  int64
 4   Tenure           10000 non-null  int64
 5   Balance          10000 non-null  float64
 6   NumOfProducts    10000 non-null  int64
 7   HasCrCard        10000 non-null  int64
 8   IsActiveMember   10000 non-null  int64
 9   EstimatedSalary  10000 non-null  float64
 10  Exited           10000 non-null int64
dtypes: float64(2), int64(7), object(2)
memory usage: 859.5+ KB
```

```
from sklearn.preprocessing import LabelEncoder,MinMaxScaler
labelencoder = LabelEncoder()
df['Geography']= labelencoder.fit_transform(df['Geography'])
df['Gender'] = labelencoder.fit_transform(df['Gender'])
```

8. Split the data into dependent and independent variables.

```
x= df.iloc[:, :-1]
y= df.iloc[:,-1:]
```

9. Scale the independent variables

```
from sklearn.preprocessing import MinMaxScaler
nm =MinMaxScaler()
X = nm.fit_transform(x)
```

10. Split the data into training and testing

```
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,random_state=0)
```