

## Assignment -2

### Data Visualization and Pre-processing

Assignment Date	30 september 2022
Student Name	Prembabu.C
Student Roll Number	310819104061
Maximum Marks	2 Marks

## ASSIGNMENT 2

### Importing libraries

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

### Loading data set

```
ds=pd.read_csv('Churn_Modelling.csv')
```

```
ds.shape
```

```
(10000, 14)
```

```
ds.head()
```

	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

## Visualization

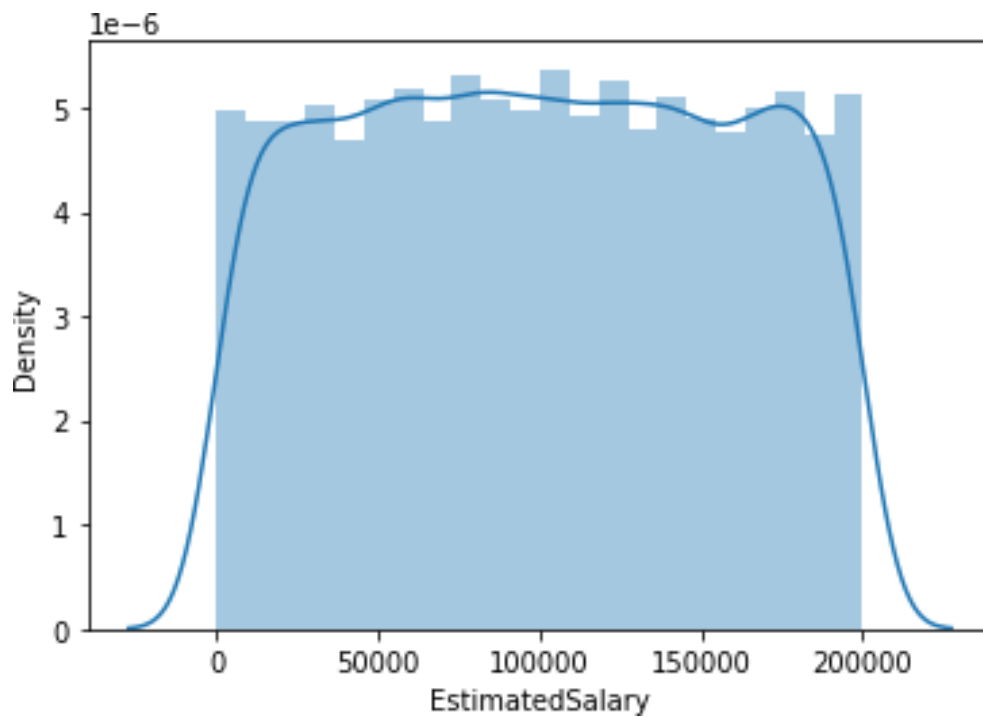
### 1. Univariate

```
sns.distplot(ds['EstimatedSalary'], hist=True)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
```

```
warnings.warn(msg, FutureWarning)
```

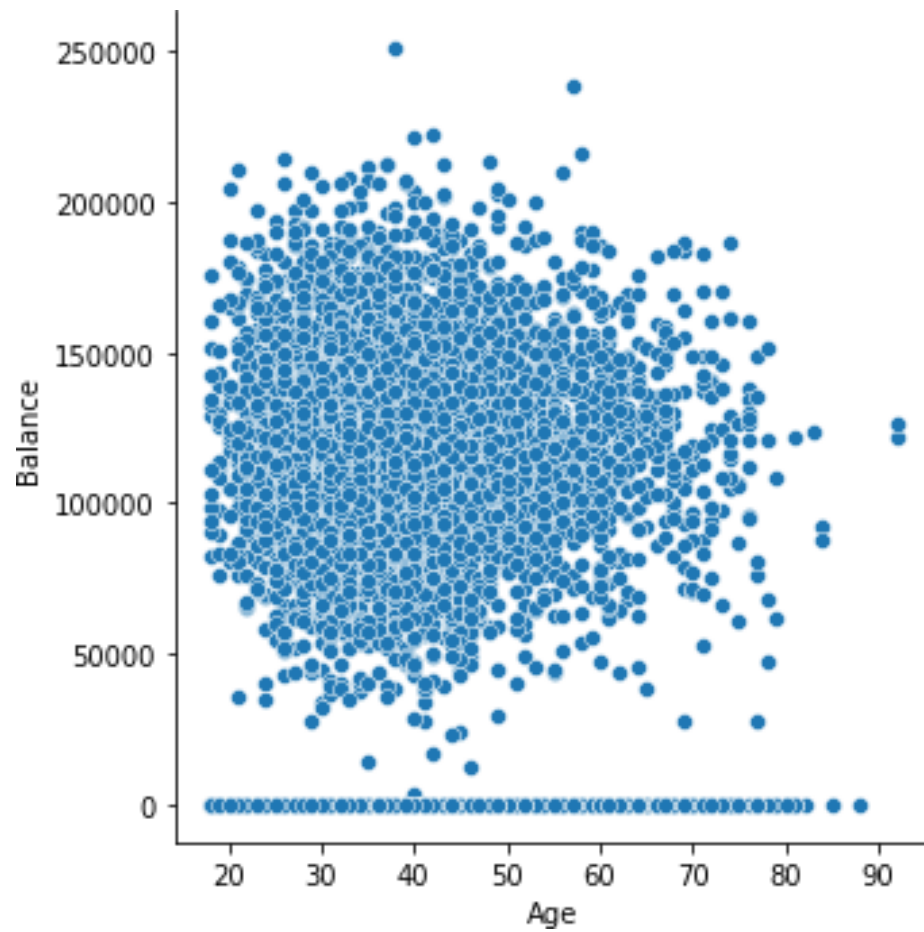
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8afae88250>
```



## 2. Bivariate

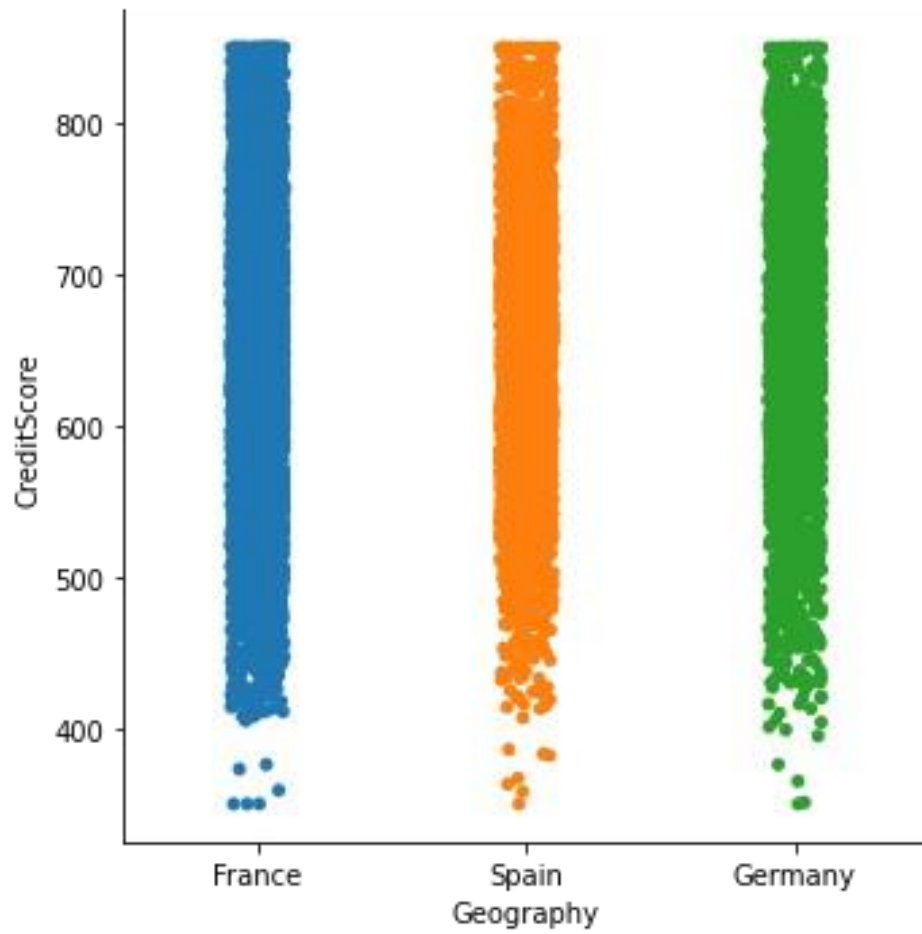
```
sns.relplot(x='Age', y='Balance', data=ds)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8afaaa3dd0>
```



```
sns.catplot(x='Geography',y='CreditScore',data=ds)
```

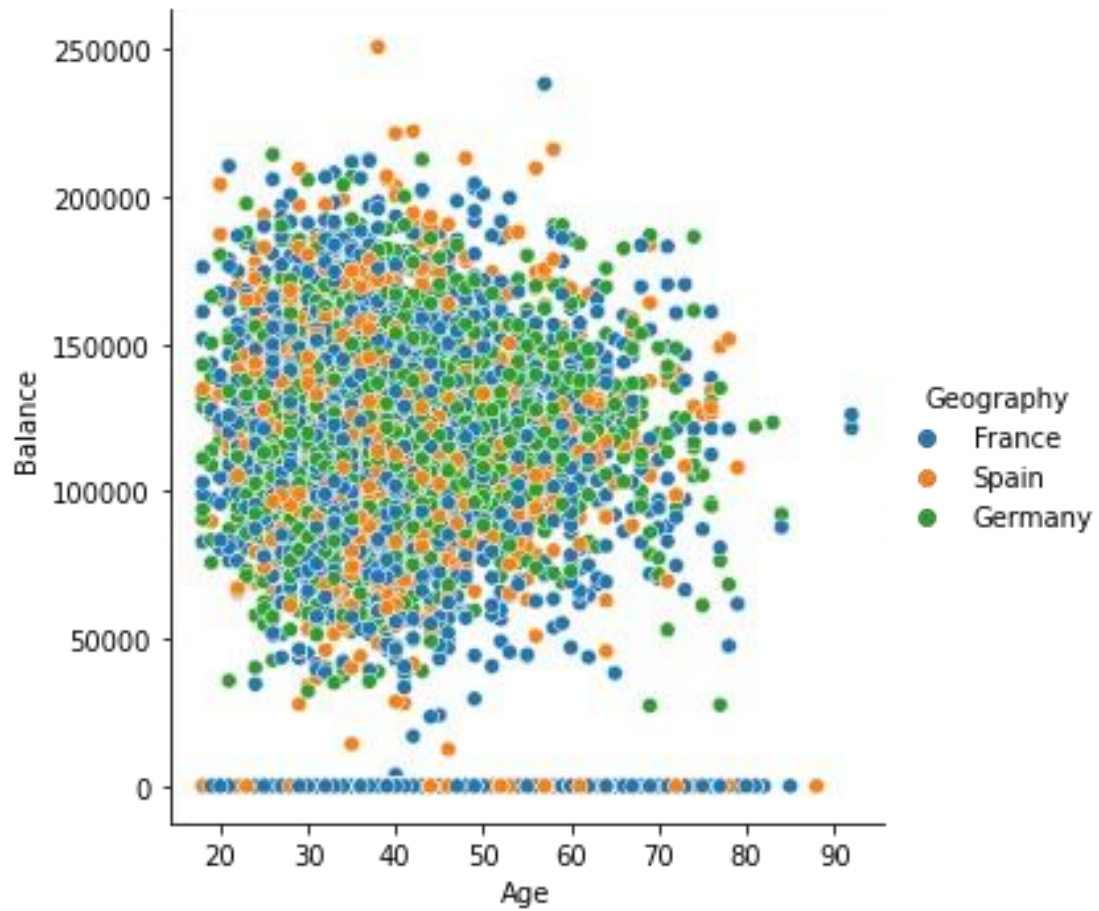
```
<seaborn.axisgrid.FacetGrid at 0x7f8afae83dd0>
```



### 3. Multivariate

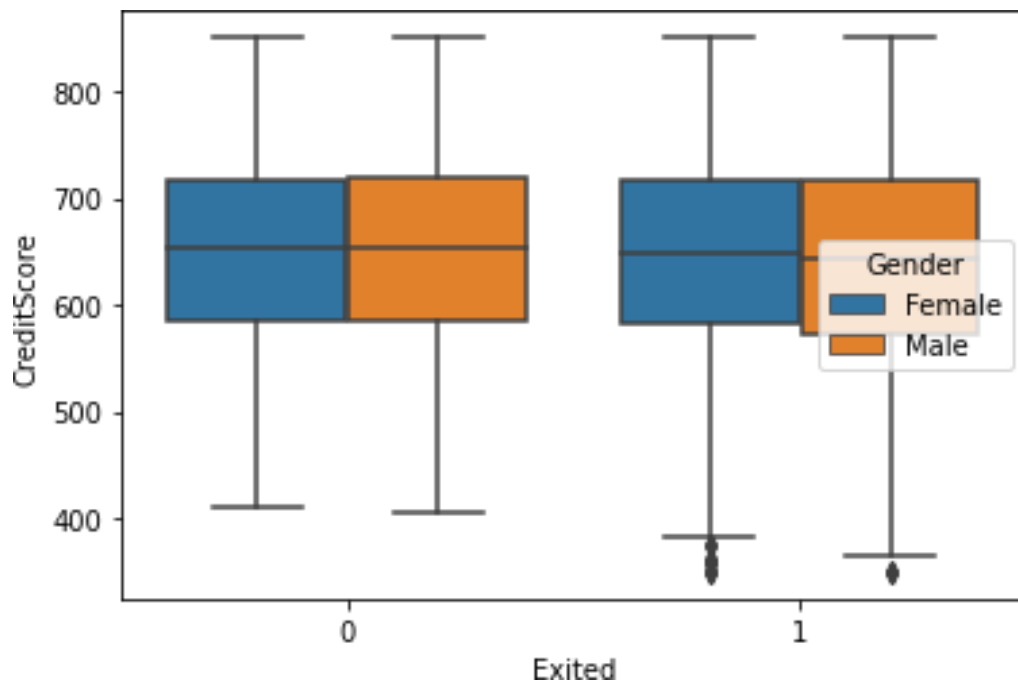
```
sns.relplot(x='Age',y='Balance',hue='Geography',data=ds)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8af64e6610>
```



```
sns.boxplot(x='Exited',y='CreditScore',hue='Gender',data=ds)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8af64ae810>
```



## Descriptive Statistics

ds.describe()

	RowNumber	CustomerId	CreditScore	Age
Tenure \				
count	10000.00000	1.000000e+04	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800
std	2886.89568	7.193619e+04	96.653299	10.487806
min	1.00000	1.556570e+07	350.000000	18.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000
max	10000.00000	1.581569e+07	850.000000	92.000000
	Balance	NumOfProducts	HasCrCard	IsActiveMember \
count	10000.000000	10000.000000	10000.00000	10000.000000
mean	76485.889288	1.530200	0.70550	0.515100
std	62397.405202	0.581654	0.45584	0.499797
min	0.000000	1.000000	0.00000	0.000000
25%	0.000000	1.000000	0.00000	0.000000

50%	97198.540000	1.000000	1.000000	1.000000
75%	127644.240000	2.000000	1.000000	1.000000
max	250898.090000	4.000000	1.000000	1.000000

	EstimatedSalary	Exited
count	10000.000000	10000.000000
mean	100090.239881	0.203700
std	57510.492818	0.402769
min	11.580000	0.000000
25%	51002.110000	0.000000
50%	100193.915000	0.000000
75%	149388.247500	0.000000
max	199992.480000	1.000000

## Handling the missing(null) values

```
ds.isnull().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

```

```
ds.isnull().sum()
```

```

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

```



## Split the data into dependent and independent variables

```
x=ds.iloc[:,3:13].values
print(x.shape)
y=ds.iloc[:,13:14].values
print(y.shape)
```

```
(10000, 10)
```

```
(10000, 1)
```

## Finding and Replacing Outliers

```
ds.skew()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this
will raise TypeError.  Select only valid columns before calling the
reduction.
```

```
    """Entry point for launching an IPython kernel.
```

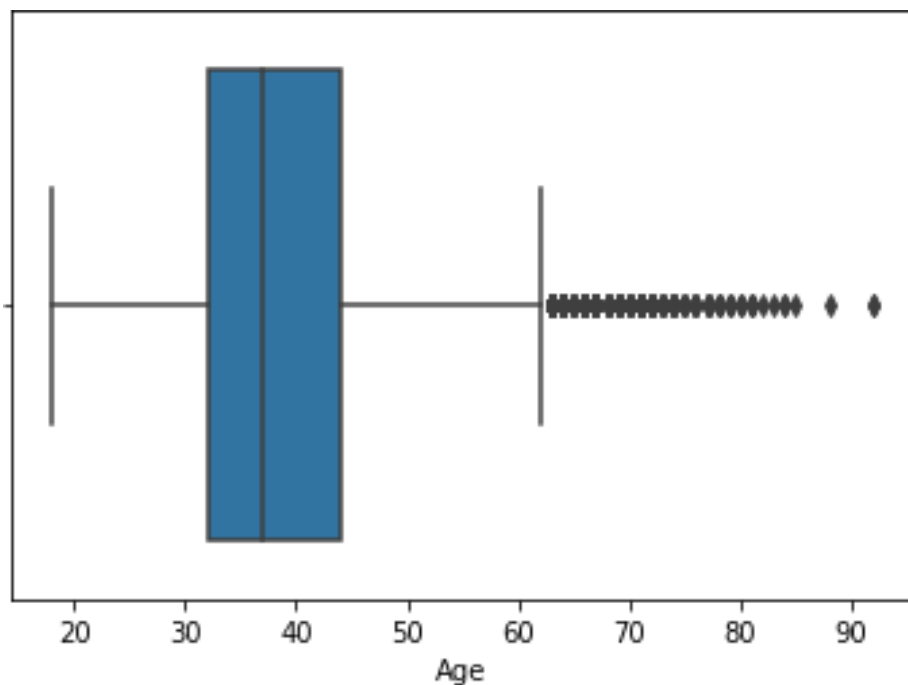
```
RowNumber          0.000000
CustomerId          0.001149
CreditScore        -0.071607
Age                 1.011320
Tenure              0.010991
Balance             -0.141109
NumOfProducts       0.745568
HasCrCard           -0.901812
IsActiveMember      -0.060437
EstimatedSalary     0.002085
Exited              1.471611
dtype: float64
```

```
sns.boxplot(ds["Age"])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
```

```
FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8af6283850>
```



```
q0 = ds["Age"].describe()["25%"]
q1 = ds["Age"].describe()["75%"]
iqr=q1-q0
```

```
lb = q0 -(1.5*iqr)
ub = q1 + (1.5*iqr)
```

```
ds[ds["Age"]<lb]
```

Empty DataFrame

Columns: [RowNumber, CustomerId, Surname, CreditScore, Geography, Gender, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary, Exited]  
Index: []

```
ds[ds["Age"]>ub]
```

	RowNumber	CustomerId	Surname	CreditScore	Geography
Gender \	Age \				
58	59	15623944	T'ien	511	Spain
Female	66				
85	86	15805254	Ndukaku	652	Spain
Female	75				
104	105	15804919	Dunbabin	670	Spain
Female	65				
158	159	15589975	Maclean	646	France
Female	73				
181	182	15789669	Hsia	510	France
Male	65				
...	...	...	...	...	...

```

. . . . .
9753      9754      15705174      Chiedozie      656      Germany
Male      68
9765      9766      15777067      Thomas      445      France
Male      64
9832      9833      15814690      Chukwujekwu      595      Germany
Female     64
9894      9895      15704795      Vagin      521      France
Female     77
9936      9937      15653037      Parks      609      France
Male      77

```

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
58	4	0.00	1	1	0	
85	10	0.00	2	1	1	
104	1	0.00	1	1	1	
158	6	97259.25	1	0	1	
181	2	0.00	2	1	1	
...	...	...	...	...	...	
9753	7	153545.11	1	1	1	
9765	2	136770.67	1	0	1	
9832	2	105736.32	1	1	1	
9894	6	0.00	2	1	1	
9936	1	0.00	1	0	1	

	EstimatedSalary	Exited
58	1643.11	1
85	114675.75	0
104	177655.68	1
158	104719.66	0
181	48071.61	0
...	...	...
9753	186574.68	0
9765	43678.06	0
9832	89935.73	1
9894	49054.10	0
9936	18708.76	0

```
[359 rows x 14 columns]
```

```
outlier_list = list(ds[ds["Age"] > ub]["Age"])
```

```
print(outlier_list)
```

```

[66, 75, 65, 73, 65, 72, 67, 67, 79, 80, 68, 75, 66, 66, 70, 63, 72,
64, 64, 70, 67, 82, 63, 69, 65, 69, 64, 65, 74, 67, 66, 67, 63, 70,
71, 72, 67, 74, 76, 66, 63, 66, 68, 67, 63, 71, 66, 69, 73, 65, 66,
64, 69, 64, 77, 74, 65, 70, 67, 69, 67, 74, 69, 74, 74, 64, 63, 63,
70, 74, 65, 72, 77, 66, 65, 74, 88, 63, 71, 63, 64, 67, 70, 68, 72,
71, 66, 75, 67, 73, 69, 76, 63, 85, 67, 74, 76, 66, 69, 66, 72, 63,

```

```
71, 63, 74, 67, 72, 72, 66, 84, 71, 66, 63, 74, 69, 84, 67, 64, 68,
66, 77, 70, 67, 79, 67, 76, 73, 66, 67, 64, 73, 76, 72, 64, 71, 63,
70, 65, 66, 65, 80, 66, 63, 63, 63, 63, 66, 74, 69, 63, 64, 76, 75,
68, 69, 77, 64, 66, 74, 71, 67, 68, 64, 68, 70, 64, 75, 66, 64, 78,
65, 74, 64, 64, 71, 77, 79, 70, 81, 64, 68, 68, 63, 79, 66, 64, 70,
69, 71, 72, 66, 68, 63, 71, 72, 72, 64, 78, 75, 65, 65, 67, 63, 68,
71, 73, 64, 66, 71, 69, 71, 66, 76, 69, 73, 64, 64, 75, 73, 71, 72,
63, 67, 68, 73, 67, 64, 63, 92, 65, 75, 67, 71, 64, 66, 64, 66, 67,
77, 92, 67, 63, 66, 66, 68, 65, 72, 71, 76, 63, 67, 67, 66, 67, 63,
65, 70, 72, 77, 74, 72, 73, 77, 67, 71, 64, 72, 81, 76, 69, 68, 74,
64, 64, 71, 68, 63, 67, 63, 64, 76, 63, 63, 68, 67, 72, 70, 81, 67,
73, 66, 68, 71, 66, 63, 75, 69, 64, 69, 70, 71, 71, 66, 70, 63, 64,
65, 63, 67, 71, 67, 65, 66, 63, 73, 66, 64, 72, 71, 69, 67, 64, 81,
73, 63, 67, 74, 83, 69, 71, 78, 63, 70, 69, 72, 70, 63, 74, 80, 69,
72, 67, 76, 71, 67, 71, 78, 63, 63, 68, 64, 70, 78, 69, 68, 64, 64,
77, 77]
```

```
outlier_dict = {}.fromkeys(outlier_list,ub)
```

```
print(outlier_dict)
```

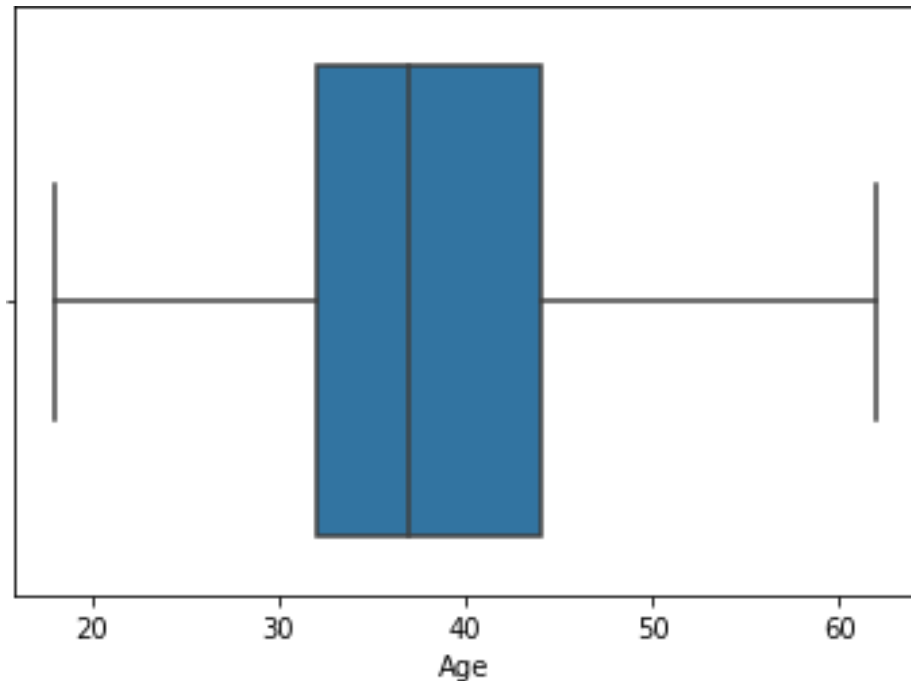
```
{66: 62.0, 75: 62.0, 65: 62.0, 73: 62.0, 72: 62.0, 67: 62.0, 79: 62.0,
80: 62.0, 68: 62.0, 70: 62.0, 63: 62.0, 64: 62.0, 82: 62.0, 69: 62.0,
74: 62.0, 71: 62.0, 76: 62.0, 77: 62.0, 88: 62.0, 85: 62.0, 84: 62.0,
78: 62.0, 81: 62.0, 92: 62.0, 83: 62.0}
```

```
ds["Age"] = ds["Age"].replace(outlier_dict)
sns.boxplot(ds["Age"])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
```

```
FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8afae88150>
```



```
ds[ds["Age"]>ub]
```

```
Empty DataFrame
```

```
Columns: [RowNumber, CustomerId, Surname, CreditScore, Geography, Gender, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary, Exited]
Index: []
```

### Check for Categorical columns and perform encoding

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct=ColumnTransformer([ ('oh',OneHotEncoder(),
[1,2])],remainder='passthrough')
x=ct.fit_transform(x)
print(x.shape)
```

```
(10000, 13)
```

```
import joblib
joblib.dump(ct,"churnct.pkl")
```

```
['churnct.pkl']
```

### 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)
print(x_train.shape)
print(x_test.shape)
```

```
(8000, 13)  
(2000, 13)
```

```
from sklearn.preprocessing import StandardScaler  
sc=StandardScaler()  
x_train=sc.fit_transform(x_train)  
x_test=sc.transform(x_test)  
joblib.dump(sc, "churnsc.pkl")  
['churnsc.pkl']
```