

Assignment – 2

Assignment Date	25 September 2022
Student Name	V.Gokulakrishnan
Student Roll Number	820319205011
Maximum Marks	2 Marks

1. Download the dataset: Dataset

2 . Load the dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df =pd.read_csv("gdrive/My Drive/Churn_Modelling.csv")
df.head()
```

OUTPUT:

	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

```
from google.colab import drive drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
#dropping row number columns as we already have index column by default  
dataset.drop(['RowNumber'], axis=1,inplace=True)
```

3. Visualizations

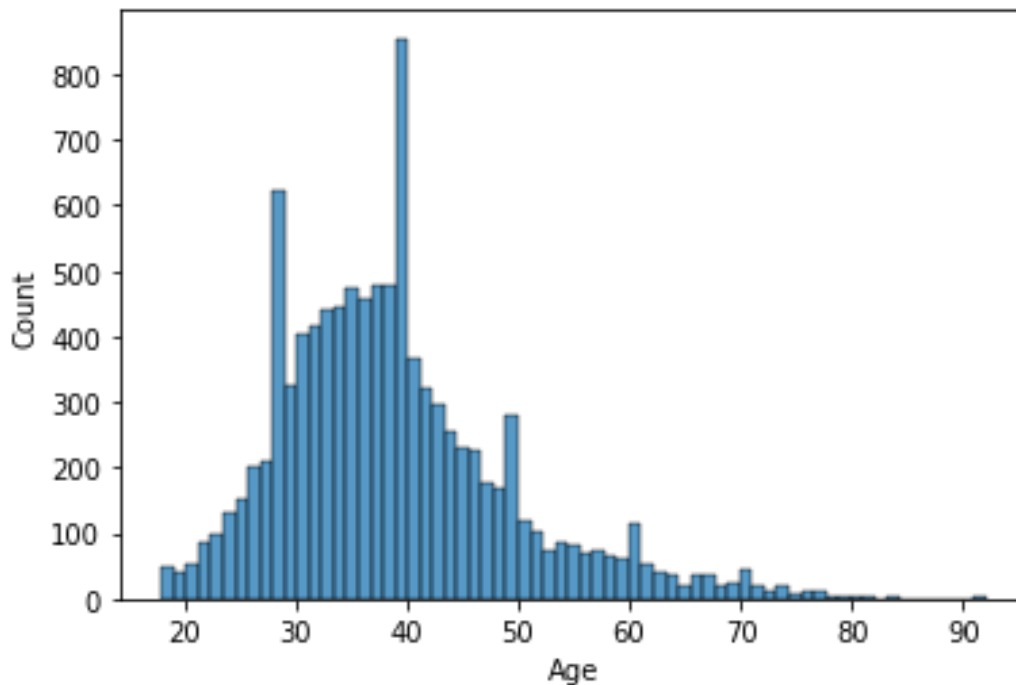
```
import matplotlib.pyplot as plt  
import seaborn as sns
```

##Univariate Analysis

```
# plt.scatter(churn.index, churn["Age"])  
# plt.show()
```

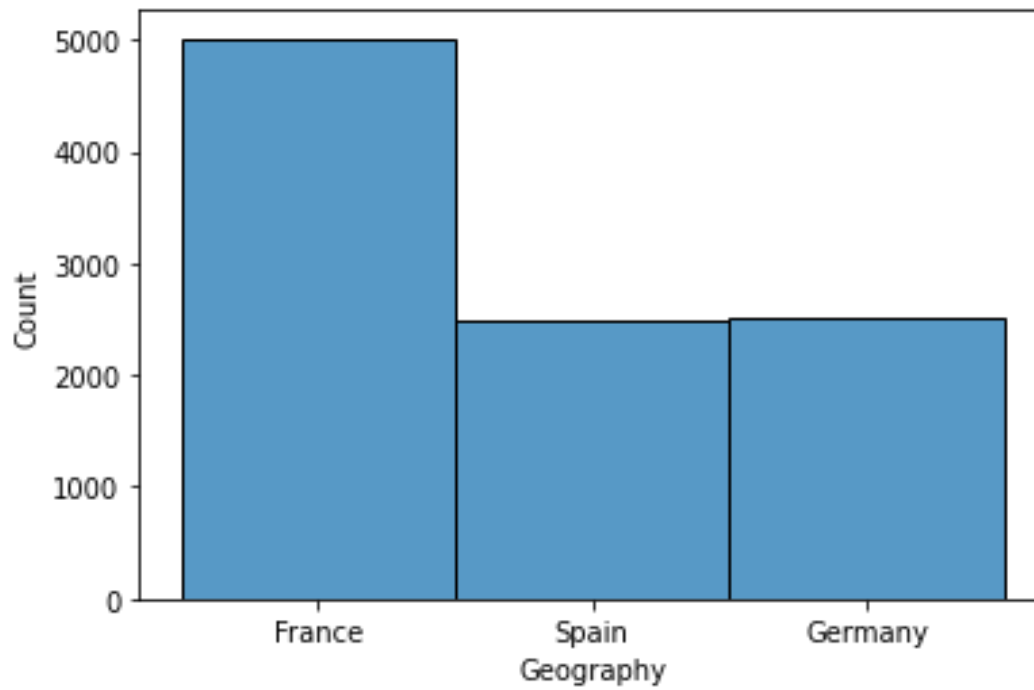
```
# Age Histogram sns.histplot(x='Age',  
data=dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f76872b9410>
```



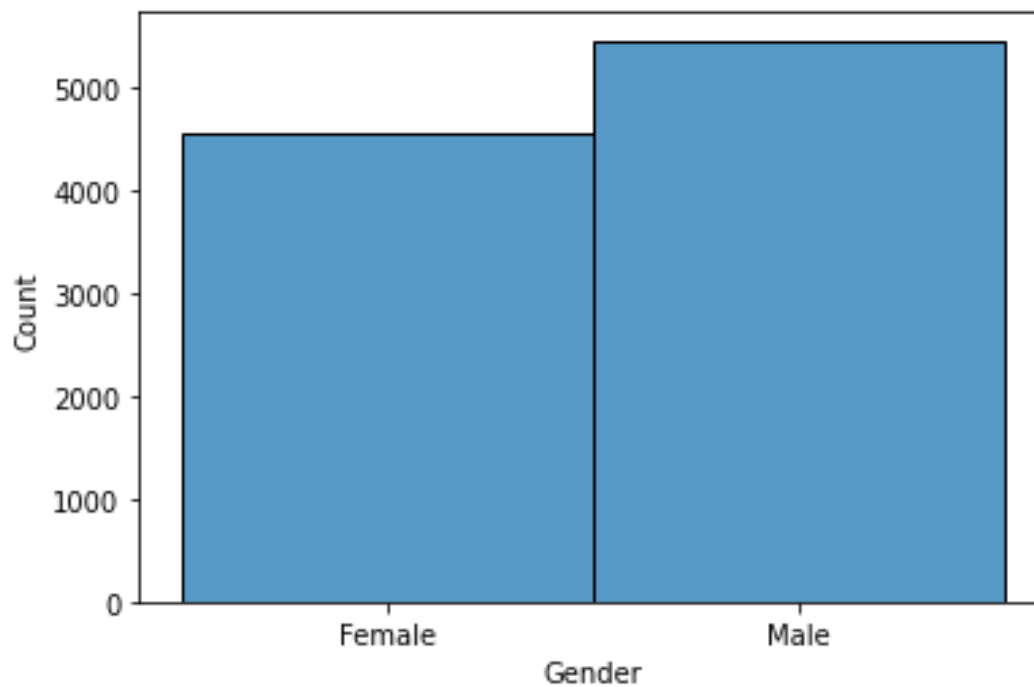
```
# Geography Histogram sns.histplot(x='Geography',  
data=dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f76864b6390>
```



```
# Geography Histogram sns.histplot(x='Gender',  
data=dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7685fdee90>
```



```
cols = 3 rows  
= 3
```

```

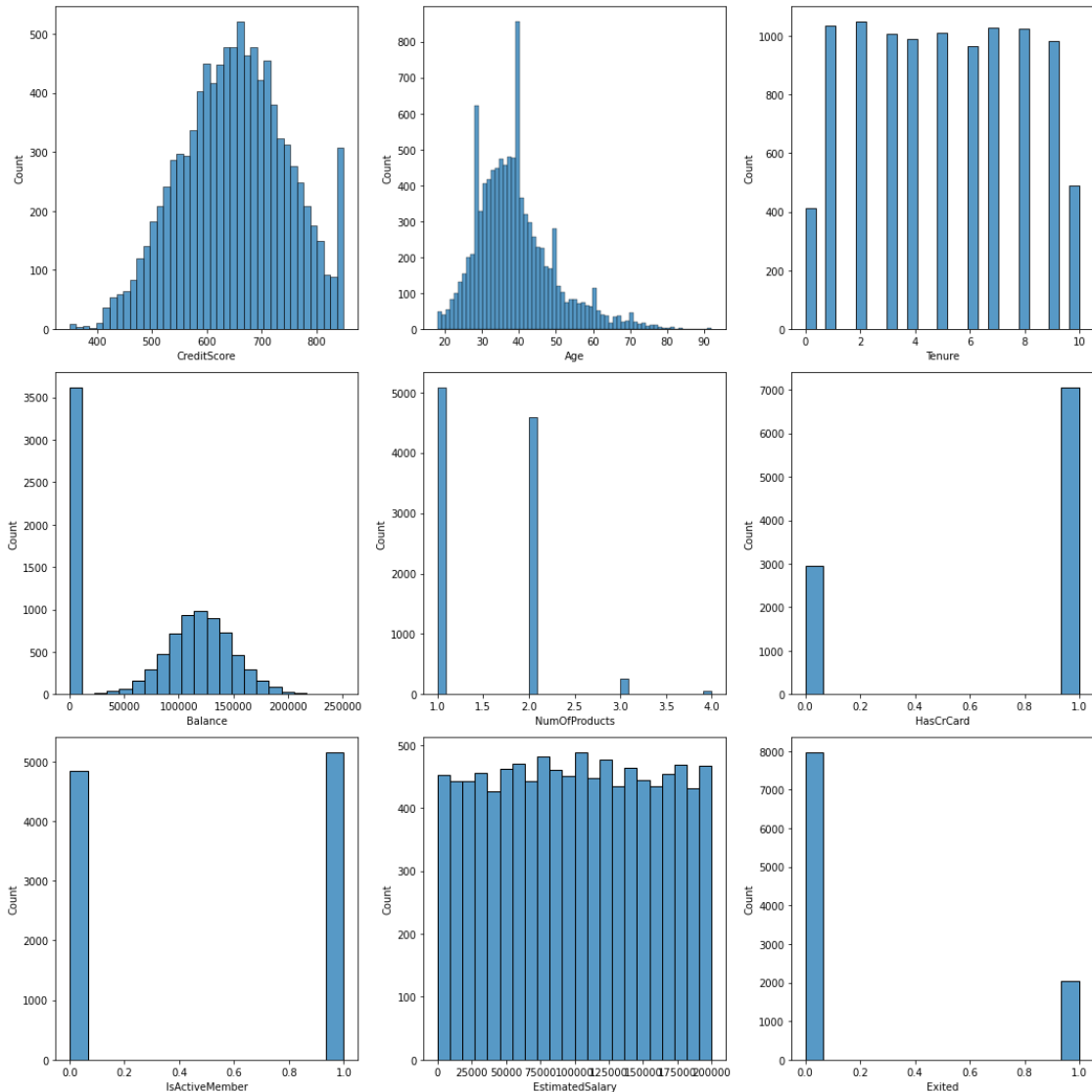
num_cols = dataset.select_dtypes(exclude='object').columns #exclude string
based columns namely Surname, Geography, Gender print(num_cols)
fig = plt.figure(figsize=(cols*5, rows*5))
for i, col in enumerate(num_cols[1:]): #exclude Customer ID

    ax=fig.add_subplot(rows,cols,i+1)
        sns.histplot(x = dataset[col], ax
= ax)

fig.tight_layout() plt.show()

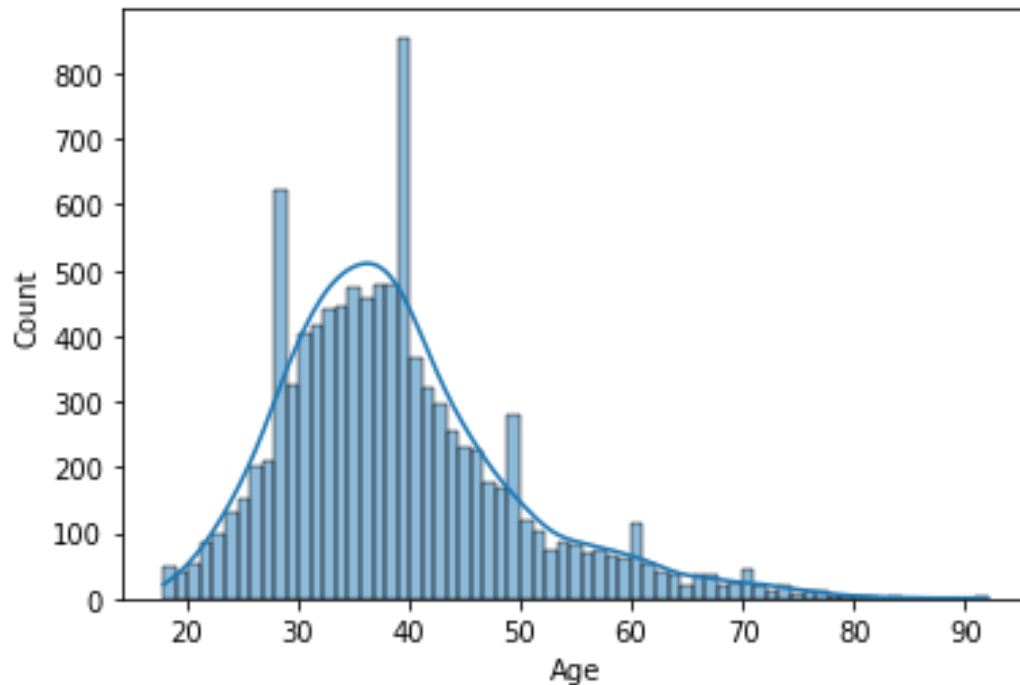
Index(['CustomerId', 'CreditScore', 'Age', 'Tenure', 'Balance',
      'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary',
      'Exited'],
      dtype='object')

```



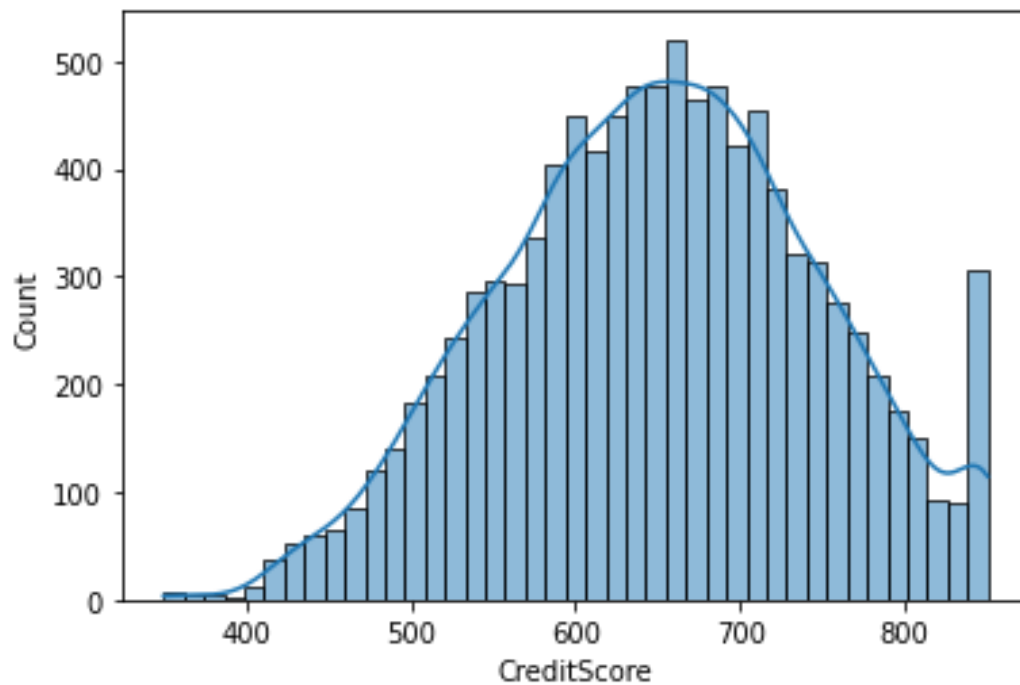
```
# sns.kdeplot(x='Age', data=churn, hue='Exited') sns.histplot(x='Age',  
data=dataset, kde=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7685ba8290>
```



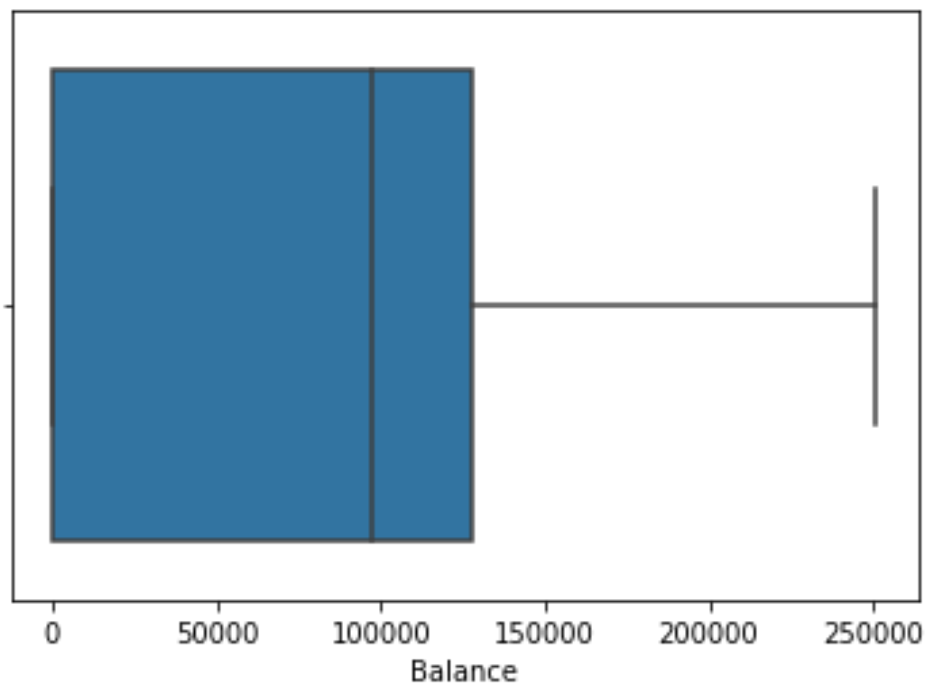
```
# sns.kdeplot(x='Age', data=churn, hue='IsActiveMember')  
sns.histplot(x='CreditScore', data=dataset, kde=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f768597f2d0>
```



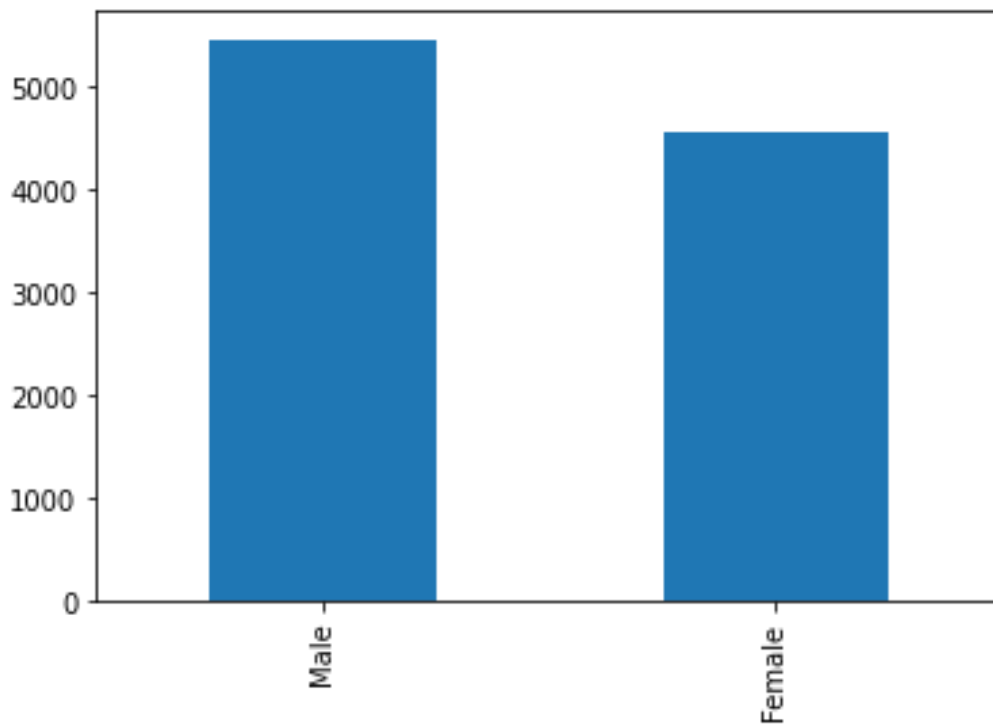
```
sns.boxplot(x=dataset['Balance'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7686032110>
```



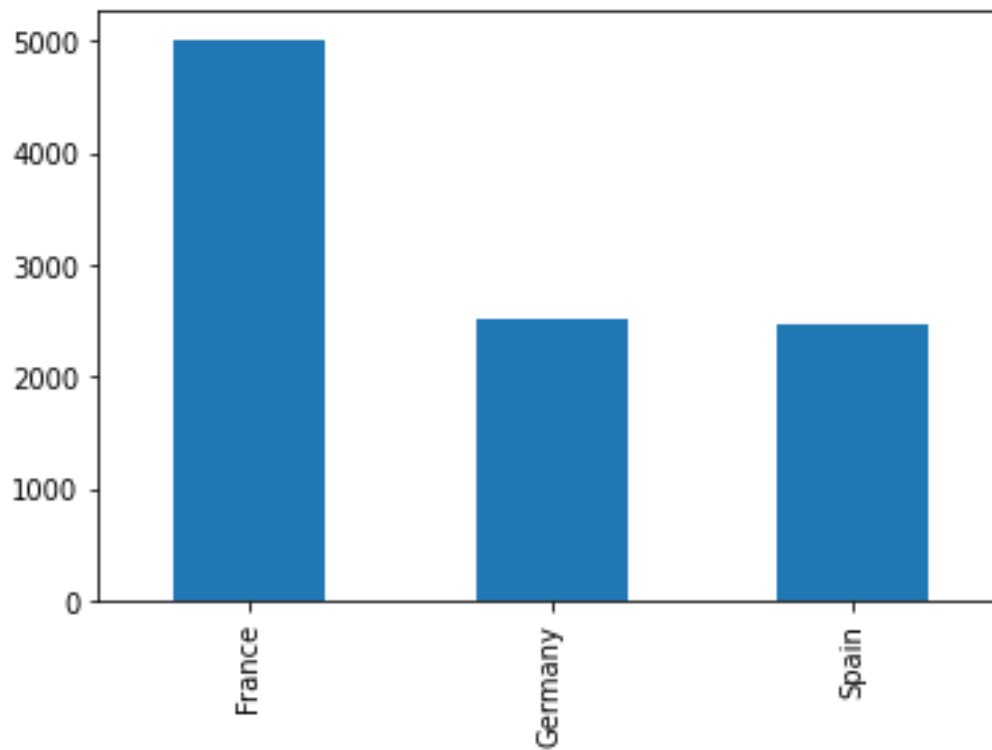
```
dataset['Gender'].value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7682e1ea50>
```



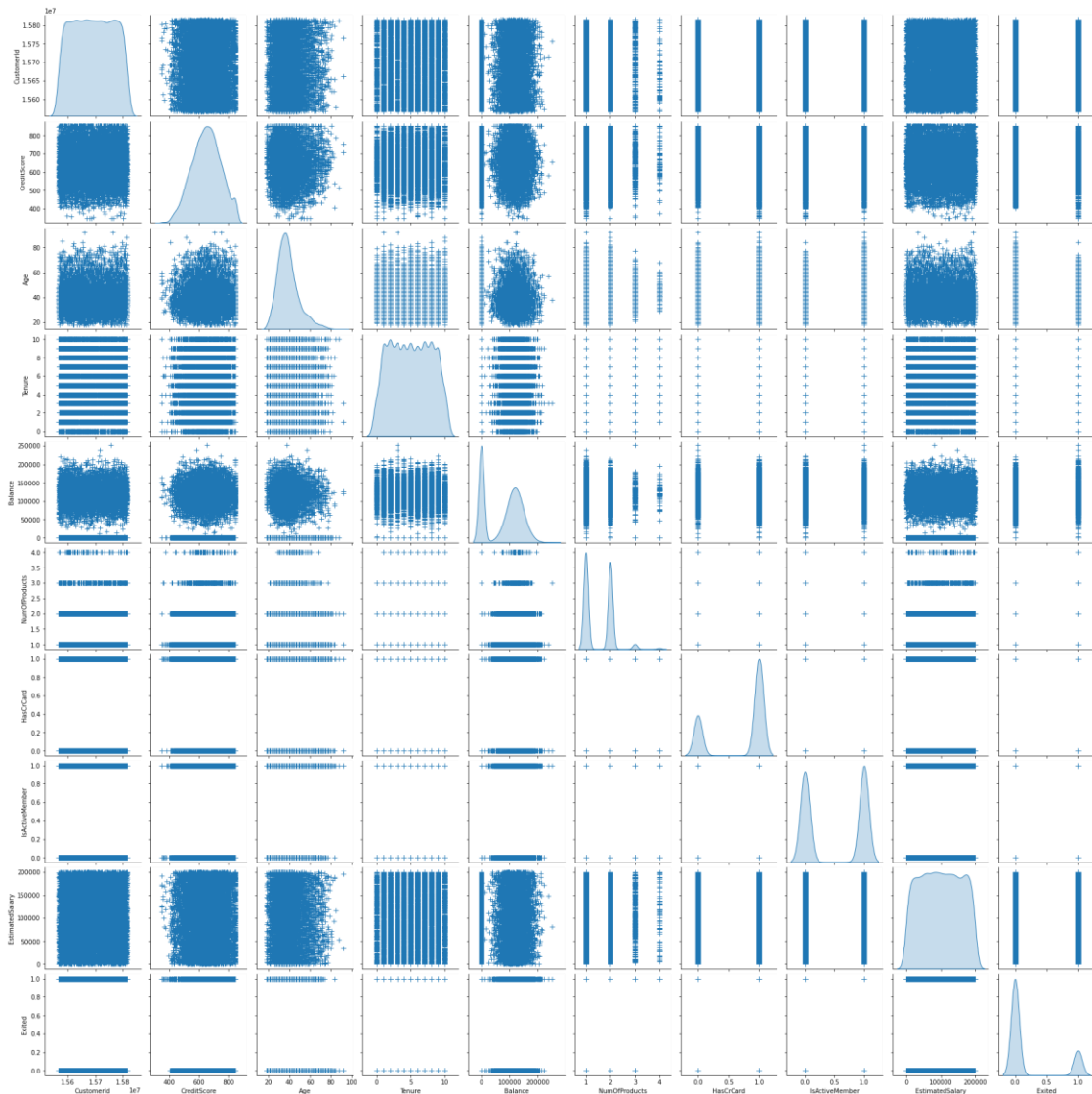
```
dataset['Geography'].value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7683120d90>
```



##Bi - Variate Analysis

```
g = sns.pairplot(dataset, diag_kind="kde", markers="+",  
plot_kws=dict(s=50, edgecolor="b", linewidth=1),  
diag_kws=dict(shade=True))
```



```
import matplotlib.pyplot as plt
```

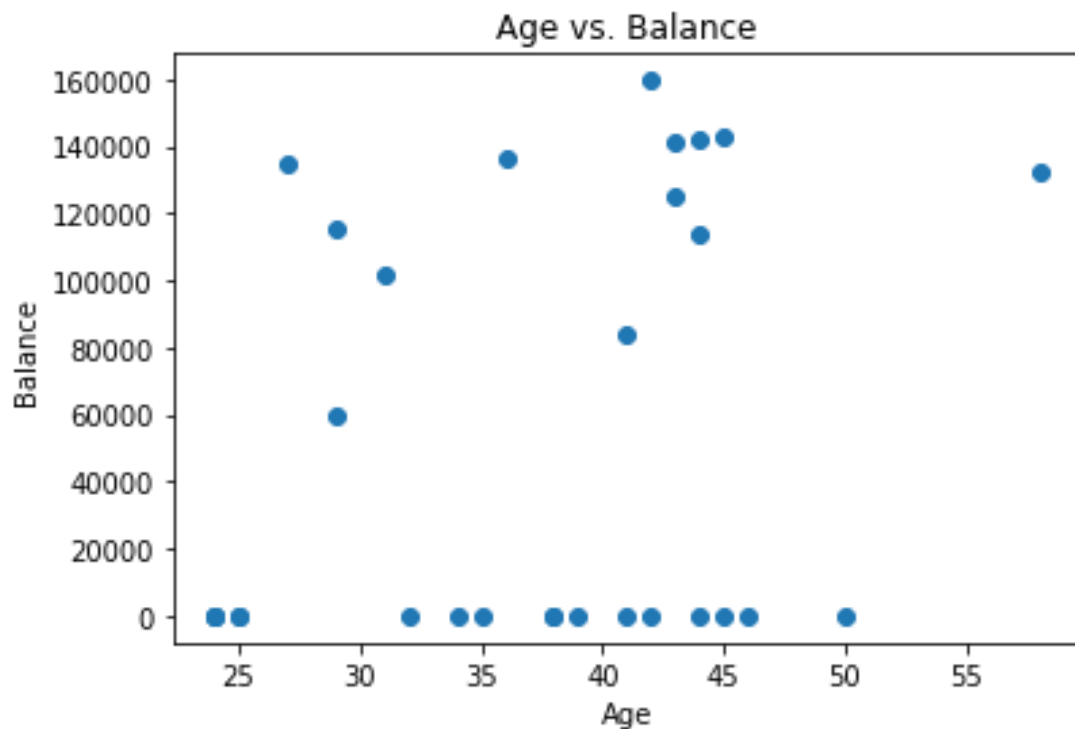
```
#create scatterplot of hours vs. score
```

```
plt.scatter(dataset.Age[:30], dataset.Balance[:30])
```

```
plt.title('Age vs. Balance') plt.xlabel('Age')
```

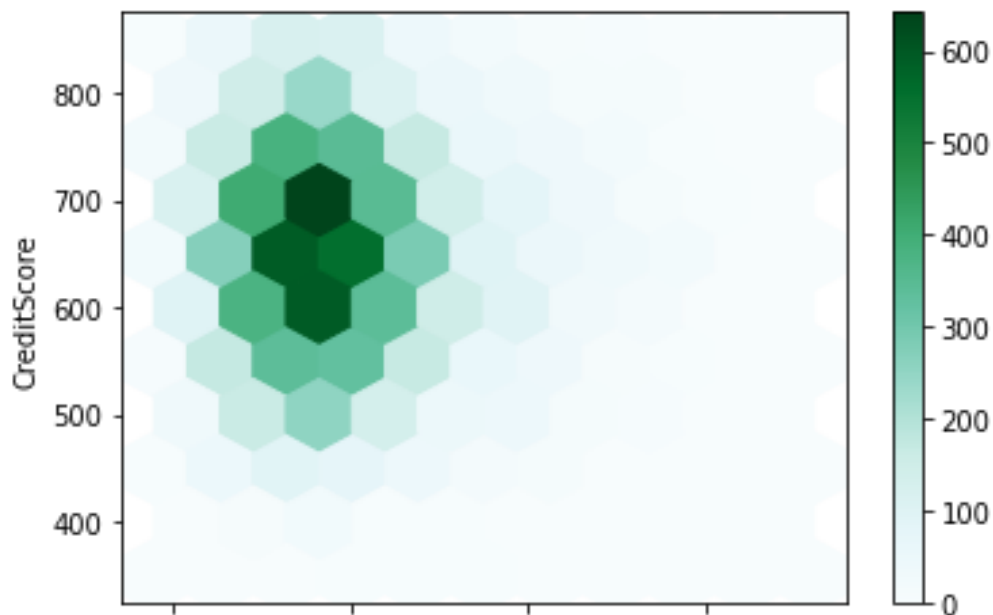
```
plt.ylabel('Balance')
```

```
Text(0, 0.5, 'Balance')
```

```
dataset.plot.hexbin(x='Age', y='CreditScore', gridsize=10)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7682d84690>
```



```
##Multi-variate Analysis dataset.corr()
```

	CustomerId	CreditScore	Age	Tenure	Balance	\
CustomerId	1.000000	0.005308	0.009497	-0.014883	-0.012419	
CreditScore	0.005308	1.000000	-0.003965	0.000842	0.006268	

Age	0.009497	-0.003965	1.000000	-0.009997	0.028308
Tenure	-0.014883	0.000842	-0.009997	1.000000	-0.012254
Balance	-0.012419	0.006268	0.028308	-0.012254	1.000000
NumOfProducts	0.016972	0.012238	-0.030680	0.013444	-0.304180
HasCrCard	-0.014025	-0.005458	-0.011721	0.022583	-0.014858
IsActiveMember	0.001665	0.025651	0.085472	-0.028362	-0.010084
EstimatedSalary	0.015271	-0.001384	-0.007201	0.007784	0.012797
Exited	-0.006248	-0.027094	0.285323	-0.014001	0.118533

	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	\
CustomerId	0.016972	-0.014025	0.001665	0.015271	
CreditScore	0.012238	-0.005458	0.025651	-0.001384	
Age	-0.030680	-0.011721	0.085472	-0.007201	
Tenure	0.013444	0.022583	-0.028362	0.007784	
Balance	-0.304180	-0.014858	-0.010084	0.012797	
NumOfProducts	1.000000	0.003183	0.009612	0.014204	
HasCrCard	0.003183	1.000000	-0.011866	-0.009933	
IsActiveMember	0.009612	-0.011866	1.000000	-0.011421	
EstimatedSalary	0.014204	-0.009933	-0.011421	1.000000	
Exited	-0.047820	-0.007138	-0.156128	0.012097	

	Exited
CustomerId	-0.006248
CreditScore	-0.027094
Age	0.285323
Tenure	-0.014001
Balance	0.118533
NumOfProducts	-0.047820
HasCrCard	-0.007138
IsActiveMember	-0.156128
EstimatedSalary	0.012097
Exited	1.000000

```
sns.set(font_scale=0.50) plt.figure(figsize=(8,4))
sns.heatmap(dataset.corr(),cmap='RdBu_r', annot=True, vmin=-1, vmax=1)
```

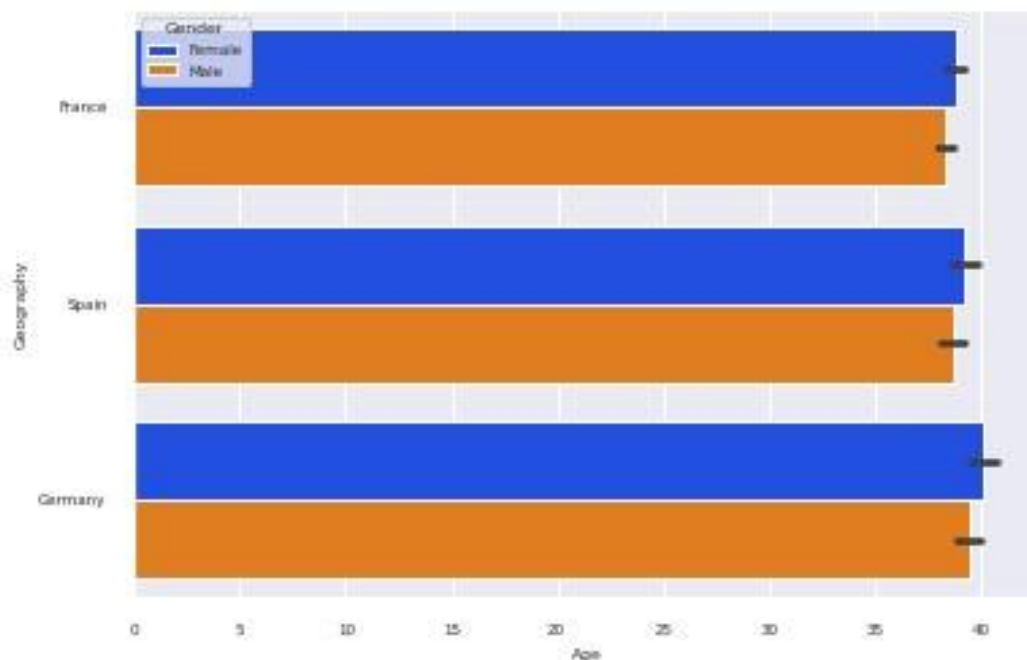
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7680979950>
```



#Three variables - Multivariate

```
sns.barplot(x='Age', y='Geography', data=dataset,
palette='bright', hue='Gender')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f767ec905d0>



4 . Descriptive statistics

```
import
statistics as st
dataset[['Age', 'Balance', 'EstimatedSalary']].mean()
```

```
Age                38.921800
Balance            76485.889288
EstimatedSalary    100090.239881
dtype: float64
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 13 columns):
```

#	Column	Non-Null Count	Dtype
---	--------	----------------	-------

0	CustomerId	10000 non-null	int64
1	Surname	10000 non-null	object
2	CreditScore	10000 non-null	int64
3	Geography	10000 non-null	object
4	Gender	10000 non-null	object
5	Age	10000 non-null	int64
6	Tenure	10000 non-null	int64
7	Balance	10000 non-null	float64
8	NumOfProducts	10000 non-null	int64
9	HasCrCard	10000 non-null	int64
10	IsActiveMember	10000 non-null	int64

11	EstimatedSalary	10000 non-null	float64	12	Exited	10000 non-null
----	-----------------	----------------	---------	----	--------	----------------

dtypes: float64(2), int64(8), object(3) memory usage: 1015.8+ KB

```
dataset.describe()
```

	CustomerId	CreditScore	Age	Tenure	Balance
count	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000
mean	1.569094e+07	650.528800	38.921800	5.012800	76485.889288
std	7.193619e+04	96.653299	10.487806	2.892174	62397.405202
min	1.556570e+07	350.000000	18.000000	0.000000	0.000000
25%	1.562853e+07	584.000000	32.000000	3.000000	0.000000
50%	1.569074e+07	652.000000	37.000000	5.000000	97198.540000
75%	1.575323e+07	718.000000	44.000000	7.000000	127644.240000
max	1.581569e+07	850.000000	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
dataset['Age'].median()
```

```
37.0
```

```
standard_deviation = dataset['CreditScore'].std()
```

```
print(standard_deviation) 96.65329873613035
```

```
st.mode(dataset['Geography'])
```

```
{"type": "string"} st.median(dataset['Age'])
```

```
37.0 st.variance(dataset['CreditScore'])
```

```
9341.860156575658
```

5 . Handle Missing Values `dataset.isnull().sum()`

#no missing values

```
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
```

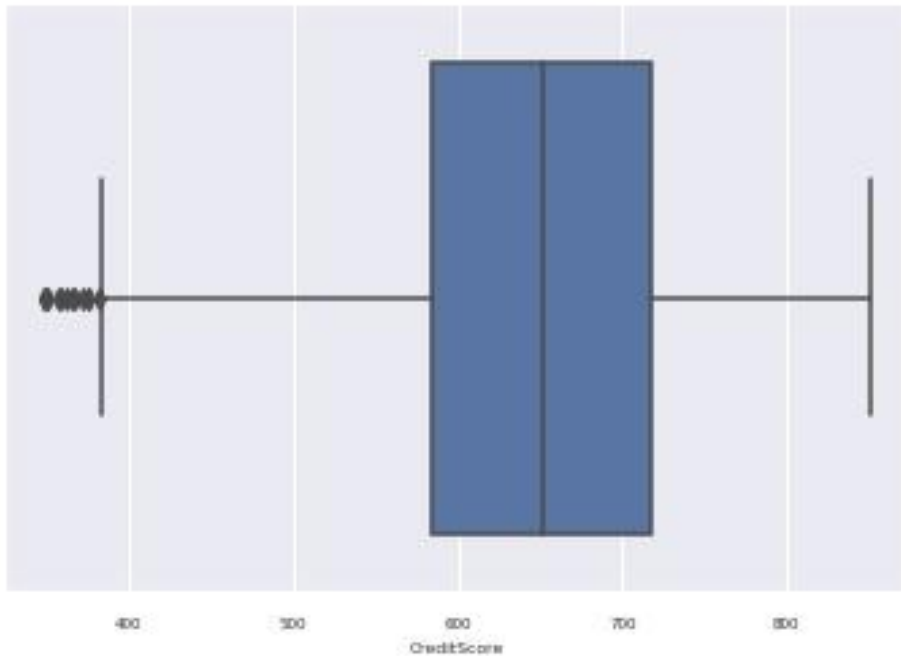
6 . Find and replace outliers

```
Visualize Outliers sns.boxplot(dataset['CreditScore'], data=dataset)
```

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

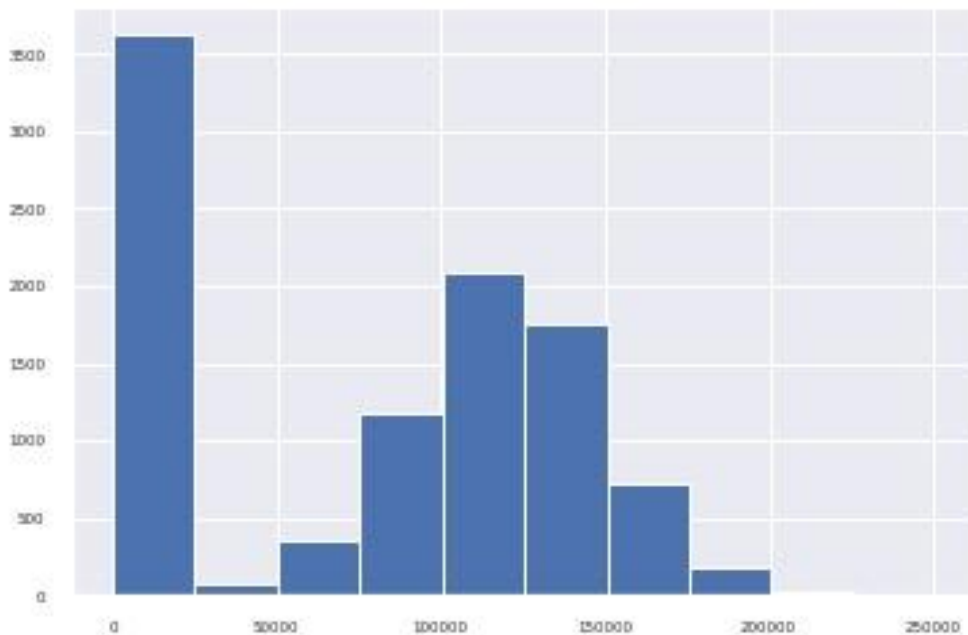
FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. FutureWarning

<matplotlib.axes._subplots.AxesSubplot at 0x7f767ebd9d90>



dataset['Balance'].hist()

<matplotlib.axes._subplots.AxesSubplot at 0x7f767ebbefd0>



```
for col in num_cols[1:]:
    print('skewness value of ',col,dataset[col].skew())
```

#Skewness should be in the range of -1 to 1, any columns with skewness outside of that range would have outliers

```
skewness value of  CreditScore -0.07160660820092675
skewness value of  Age 1.0113202630234552 skewness value
of  Tenure 0.01099145797717904 skewness value of
Balance -0.14110871094154384 skewness value of
NumOfProducts 0.7455678882823168 skewness value of
HasCrCard -0.9018115952400578 skewness value of
IsActiveMember -0.06043662833499078 skewness value of
EstimatedSalary 0.0020853576615585162 skewness value of
Exited 1.4716106649378211
```

```
Q1=dataset['Age'].quantile(0.25)
Q3=dataset['Age'].quantile(0.75)
IQR=Q3-Q1
```

IQR

12.0

Removing Outliers

#Values above than the upper bound and below than the lower bound are considered outliers

```
upper = dataset['Age'] >= (Q3+1.5*IQR)
```

```
# print("Upper bound:",upper) print(np.where(upper))
```

```
lower = dataset['Age'] <= (Q1-1.5*IQR)
# print("Lower bound:", lower)
print(np.where(lower))
```

```
(array([ 58,  85, 104, 158, 181, 230, 234, 243, 252, 276, 310,
        364, 371, 385, 387, 399, 538, 559, 567, 602, 612, 617,
        658, 678, 696, 736, 766, 769, 807, 811, 823, 859, 884,
        888, 948, 952, 957, 963, 969, 997, 1009, 1039, 1040, 1055,
        1114, 1205, 1234, 1235, 1246, 1252, 1278, 1285, 1328, 1342, 1387,
        1407, 1410, 1433, 1439, 1457, 1519, 1543, 1607, 1614, 1642, 1790,
        1810, 1866, 1901, 1904, 1907, 1933, 1981, 1996, 2002, 2012, 2039,
        2053, 2078, 2094, 2108, 2154, 2159, 2164, 2244, 2274, 2433, 2458,
        2459, 2519, 2553, 2599, 2615, 2659, 2670, 2713, 2717, 2760, 2772,
        2778, 2791, 2855, 2877, 2901, 2908, 2925, 2926, 3008, 3033, 3054,
        3110, 3142, 3166, 3192, 3203, 3229, 3305, 3308, 3311, 3314, 3317,
        3346, 3366, 3368, 3378, 3382, 3384, 3387, 3396, 3403, 3434, 3462,
```

```

3497, 3499, 3527, 3531, 3541, 3559, 3573, 3575, 3593, 3602, 3641,
3646, 3647, 3651, 3690, 3691, 3702, 3719, 3728, 3733, 3761, 3813,
3826, 3880, 3881, 3888, 3909, 3910, 3927, 3940, 3980, 3994, 4010,
4025, 4048, 4051, 4095, 4142, 4147, 4157, 4162, 4170, 4241, 4244,
4256, 4273, 4280, 4297, 4313, 4318, 4335, 4360, 4366, 4378, 4387,
4396, 4435, 4438, 4463, 4490, 4501, 4506, 4559, 4563, 4590, 4595,
4644, 4698, 4747, 4751, 4801, 4815, 4832, 4849, 4931, 4947, 4966,
4992, 5000, 5020, 5038, 5068, 5132, 5136, 5148, 5159, 5197, 5223,
5225, 5235, 5255, 5299, 5313, 5368, 5377, 5405, 5457, 5490, 5508,
5514, 5576, 5577, 5581, 5655, 5660, 5664, 5671, 5698, 5777, 5783,
5817, 5825, 5840, 5867, 5907, 5957, 5996, 6046, 6116, 6152, 6166,
6167, 6173, 6212, 6230, 6278, 6289, 6315, 6357, 6366, 6373, 6375,
6410, 6443, 6515, 6530, 6532, 6581, 6612, 6626, 6706, 6709, 6715,
6721, 6759, 6763, 6812, 6899, 6970, 6997, 7008, 7057, 7058, 7063,
7071, 7078, 7094, 7138, 7139, 7142, 7156, 7194, 7202, 7238, 7243,
7272, 7302, 7362, 7375, 7392, 7499, 7514, 7523, 7526, 7548, 7552,
7624, 7629, 7692, 7694, 7709, 7715, 7719, 7720, 7727, 7773, 7776,
7784, 7788, 7802, 7813, 7851, 7894, 7898, 7933, 7956, 7995, 8019,
8037, 8094, 8098, 8156, 8193, 8207, 8217, 8304, 8321, 8385, 8394,
8444, 8458, 8467, 8469, 8478, 8488, 8562, 8568, 8577, 8602, 8674,
8686, 8689, 8711, 8759, 8761, 8768, 8787, 8793, 8822, 8865, 8900,
8917, 8930, 9018, 9062, 9080, 9112, 9116, 9162, 9223, 9279, 9292,
9309, 9318, 9324, 9332, 9333, 9351, 9380, 9402, 9425, 9428, 9438,
9472, 9490, 9506, 9555, 9557, 9582, 9587, 9589, 9593, 9646, 9671,
9673, 9681, 9686, 9688, 9718, 9733, 9734, 9736, 9747, 9753, 9765,
9832, 9879, 9894, 9936]),) (array([],
dtype=int64),)

#Removing outliers based off Age column

# IQR
Q1 = np.percentile(dataset['Age'], 25,
interpolation = 'midpoint')

Q3 = np.percentile(dataset['Age'], 75,
interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", dataset.shape)

# Upper bound upper = np.where(dataset['Age'] >=
(Q3+1.5*IQR))
# Lower bound lower = np.where(dataset['Age'] <=
(Q1-1.5*IQR))

```



```
''' Removing the Outliers ''' dataset.drop(upper[0],
inplace = True) dataset.drop(lower[0], inplace =
True)
```

```
print("New Shape: ", dataset.shape)
```

```
Old Shape: (10000, 13) New
```

```
Shape: (9589, 13) dataset
```

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	\
0	15634602	Hargrave	619	France	Female	42	2	
1	15647311	Hill	608	Spain	Female	41	1	
2	15619304	Onio	502	France	Female	42	8	
3	15701354	Boni	699	France	Female	39	1	
4	15737888	Mitchell	850	Spain	Female	43	2	
...	
9995	15606229	Obijiaku	771	France	Male	39	5	
9996	15569892	Johnstone	516	France	Male	35	10	
9997	15584532	Liu	709	France	Female	36	7	
9998	15682355	Sabbatini	772	Germany	Male	42	3	
9999	15628319	Walker	792	France	Female	28		
4								

	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	\
0	0.00	1	1	1	101348.88	
1	83807.86		1	0	1	
	112542.58					
2	159660.80		3	1	0	
	113931.57					
3	0.00	2	0	0	93826.63	
4	125510.82		1	1	1	
	79084.10	
				
9995	0.00	2	1	0	96270.64	
9996	57369.61		1	1	1	
	101699.77					
9997	0.00	1	0	1	42085.58	
9998	75075.31		2	1	0	
	92888.52					
9999	130142.79		1	1	0	
	38190.78		Exited			
0	1					
1	0					
2	1					
3	0					
4	0			
9995	0					

```

9996      0
9997      1
9998      1
9999      0

```

```
[9589 rows x 13 columns]
```

```

for col in num_cols[1:]:
    print('skewness value of ',col,dataset[col].skew())

```

*# Now we have reduced the Age column's skewness values within -1 to 1 range #
We left the Exited column's skewness value as it is the dependent variable*

```

skewness value of  CreditScore -0.07274225895185718
skewness value of  Age 0.44721544739487257 skewness value
of  Tenure 0.008085830714996462 skewness value of
Balance -0.1409005824644143 skewness value of
NumOfProducts 0.7470530176747141 skewness value of
HasCrCard -0.9034483996482451 skewness value of
IsActiveMember -0.008552881368996219 skewness value of
EstimatedSalary -0.0025661797132480266 skewness value of
Exited 1.4798502461410206

```

7 . Check for Categorical columns and perform encoding

```
##Label encoding and One Hot encoding dataset.reset_index(inplace=True)
```

```

from sklearn.preprocessing import LabelEncoder from
sklearn.preprocessing import OneHotEncoder from
sklearn.compose import ColumnTransformer

```

```

categorical_feature_mask = dataset.dtypes==object categorical_cols =
dataset.columns[categorical_feature_mask].tolist()

```

```
categorical_cols=categorical_cols[1:] categorical_cols
```

```
['Geography', 'Gender']
```

```

le = LabelEncoder() dataset[categorical_cols] =
dataset[categorical_cols].apply(lambda col:
le.fit_transform(col)) dataset[categorical_cols].head(10)

```

	Geography	Gender
0	0	0
1	2	0
2	0	0
3	0	0
4	2	0
5	2	1

```

6          0          1
7          1          0
8          0          1 9          0          1

```

categorical_feature_mask

```

index          False CustomerId
False
Surname         True
CreditScore     False
Geography       True
Gender          True
Age            False
Tenure         False
Balance        False
NumOfProducts  False
HasCrCard      False
IsActiveMember False
EstimatedSalary False
Exited         False dtype:
bool

```

```
enc=OneHotEncoder()
```

```
enc_data=pd.DataFrame(enc.fit_transform(dataset[['Geography', 'Gender']]).toarray()) enc_data
```

```

      0    1    2    3    4
0    1.0  0.0  0.0  1.0  0.0
1    0.0  0.0  1.0  1.0  0.0
2    1.0  0.0  0.0  1.0  0.0
3    1.0  0.0  0.0  1.0  0.0
4    0.0  0.0  1.0  1.0  0.0 ...   ...   ...   ...   ...
9584  1.0  0.0  0.0  0.0  1.0
9585  1.0  0.0  0.0  0.0  1.0
9586  1.0  0.0  0.0  1.0  0.0
9587  0.0  1.0  0.0  0.0  1.0
9588  1.0  0.0  0.0  1.0  0.0

```

```
[9589 rows x 5 columns]
```

#First three columns of enc_data is for Geography and the next two columns is for Gender, we can replace the already existing categorical columns with these encoded values

#Dropping already existing Geography and Gender columns

```
dataset.drop(['Geography'], axis=1,inplace=True)
dataset.drop(['Gender'], axis=1,inplace=True)
```

```

dataset.insert(2, "Geography_France", enc_data.iloc[:,0], True)
dataset.insert(3, "Geography_Germany", enc_data.iloc[:,1], True)
dataset.insert(4, "Geography_Spain", enc_data.iloc[:,2], True)
dataset.insert(5, "Gender_Female", enc_data.iloc[:,3], True)
dataset.insert(6, "Gender_Male", enc_data.iloc[:,4], True) dataset

```

	index	CustomerId	Geography_France	Geography_Germany	Geography_Spain
\					
0	0	15634602	1.0	0.0	0.0
1	1	15647311	0.0	0.0	1.0
2	2	15619304	1.0	0.0	0.0
3	3	15701354	1.0	0.0	0.0
4	4	15737888	0.0	0.0	1.0

	...				
9584	9995	15606229	1.0	0.0	0.0
9585	9996	15569892	1.0	0.0	0.0
9586	9997	15584532	1.0	0.0	0.0
9587	9998	15682355	0.0	1.0	0.0
	9588	9999	15628319	1.0	0.0
	0.0				

	Gender_Female	Gender_Male	Surname	CreditScore	Age	Tenure	\
0	1.0	0.0	Hargrave	619	42	2	
1	1.0	0.0	Hill	608	41	1	
2	1.0	0.0	Onio	502	42	8	
3	1.0	0.0	Boni	699	39	1	
4	1.0	0.0	Mitchell	850	43	2	
...	
9584	0.0	1.0	Obijiaku	771	39	5	
9585	0.0	1.0	Johnstone	516	35	10	
9586	1.0	0.0	Liu	709	36	7	
9587	0.0	1.0	Sabbatini	772	42	3	9588
	1.0	0.0	Walker	792	28	4	

	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	\
0	0.00	1	1	1	101348.88	
1	83807.86		1	0	1	
	112542.58					
2	159660.80		3	1	0	
	113931.57					
3	0.00	2	0	0	93826.63	
4	125510.82		1	1	1	
	79084.10	
				
9584	0.00	2	1	0	96270.64	

9585	57369.61		1		1		1	
	101699.77							
9586	0.00		1		0		1	42085.58
9587	75075.31			2		1		0
	92888.52	9588	130142.79				1	1
	0		38190.78					

	Exited		
0	1		
1	0		
2	1		
3	0		
4	0
9584	0		
9585	0		
9586	1		
9587	1		
9588	0		

[9589 rows x 17 columns]

We drop some irrelevant columns that does not contribute to prediction

dataset.drop(columns="CustomerId",axis=1,inplace=True)

dataset.drop(columns="Surname",axis=1,inplace=True)

dataset.drop(columns="index",axis=1,inplace=True) dataset

	Geography_France	Geography_Germany	Geography_Spain	Gender_Female	\
0	1.0	0.0	0.0	1.0	
1	0.0	0.0	1.0	1.0	
2	1.0	0.0	0.0	1.0	
3	1.0	0.0	0.0	1.0	
4	0.0	0.0	1.0	1.0	

	...				
9584	1.0	0.0	0.0	0.0	
9585	1.0	0.0	0.0	0.0	
9586	1.0	0.0	0.0	1.0	
9587	0.0	1.0	0.0	0.0	
	9588	1.0	0.0		0.0
	1.0				

	Gender_Male	CreditScore	Age	Tenure	Balance	NumOfProducts	\
0	0.0	619	42	2	0.00	1	
1	0.0	608	41	1	83807.86	1	
2	0.0	502	42	8	159660.80	3	
3	0.0	699	39	1	0.00	2	

4	0.0	850	43	2	125510.82	1	...
	
9584	1.0	771	39	5	0.00	2	
9585	1.0	516	35	10	57369.61	1	
9586	0.0	709	36	7	0.00	1	
9587	1.0	772	42	3	75075.31	2	9588
	0.0	792	28	4	130142.79	1	

	HasCrCard	IsActiveMember	EstimatedSalary	Exited		
0	1	1	101348.88	1		
1	0	1	112542.58	0		
2	1	0	113931.57	1		
3	0	0	93826.63	0		
4	1	1	79084.10	0
			
9584	1	0	96270.64	0		
9585	1	1	101699.77	0		
9586	0	1	42085.58	1		
9587	1	0	92888.52	1		
9588	1	0	38190.78	0		

[9589 rows x 14 columns]

8 . Split the data into dependent and independent variables

```
X= dataset.iloc[:, :-1].values #Independent variables
y= dataset.iloc[:, -1].values #Dependent variables
```

X

```
array([[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 1.0134888e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 1.1254258e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 1.1393157e+05],
       ...,
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 4.2085580e+04],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 9.2888520e+04],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 3.8190780e+04]]) y
```

```
array([1, 0, 1, ..., 1, 1, 0])
```

9 . Scale the independent variable

```
from sklearn.preprocessing import StandardScaler
scale= StandardScaler() X
= scale.fit_transform(X)
X
array([[ 0.99718823, -0.57955796, -0.57297497, ...,  0.64561166,
         0.99573337,  0.01997639],
       [-1.0028197 , -0.57955796,  1.74527693, ..., -1.54891873,
         0.99573337,  0.21465635],
       [ 0.99718823, -0.57955796, -0.57297497, ...,  0.64561166,
        -1.00428491,  0.23881355],
       ...,
       [ 0.99718823, -0.57955796, -0.57297497, ..., -1.54891873,
         0.99573337, -1.01072631],
       [-1.0028197 ,  1.72545295, -0.57297497, ...,  0.64561166,
        -1.00428491, -0.12716553],
       [ 0.99718823, -0.57955796, -0.57297497, ...,  0.64561166,
        -1.00428491, -1.07846436]])
```

10 . Split the data into training and testing

```
from sklearn.model_selection import train_test_split

# We use train_test_split function to split the data such that 25% is used
for testing while the remaining 75% is used for training
X_train, X_test, y_train, y_test = train_test_split(X,y ,
random_state=104,test_size=0.25, shuffle=True)
```

X_train

```
array([[ 0.99718823, -0.57955796, -0.57297497, ...,  0.64561166,
         0.99573337, -1.74019169],
       [-1.0028197 , -0.57955796,  1.74527693, ..., -1.54891873,
        -1.00428491, -1.39787901],
       [-1.0028197 ,  1.72545295, -0.57297497, ..., -1.54891873,
         0.99573337, -1.48817335],
       ...,
       [ 0.99718823, -0.57955796, -0.57297497, ...,  0.64561166,
        -1.00428491,  0.71481237],
       [ 0.99718823, -0.57955796, -0.57297497, ..., -1.54891873,
        -1.00428491,  0.60834563],
       [-1.0028197 ,  1.72545295, -0.57297497, ...,  0.64561166,
         0.99573337,  0.0525285 ]]) X_test

array([[ -1.0028197 , -0.57955796,  1.74527693, ..., -1.54891873,
        -1.00428491, -0.90389608],
```

```

[ 0.99718823, -0.57955796, -0.57297497, ..., 0.64561166,
 0.99573337, -0.54087223],
[-1.0028197 , -0.57955796,  1.74527693, ..., 0.64561166,
 0.99573337, -1.02004733],
...,
[ 0.99718823, -0.57955796, -0.57297497, ..., 0.64561166,
 0.99573337, -0.23978536],
[ 0.99718823, -0.57955796, -0.57297497, ..., 0.64561166,
 0.99573337, -0.17457887],
[ 0.99718823, -0.57955796, -0.57297497, ..., 0.64561166,
-1.00428491, -0.0121091 ]]) y_train
array([0, 0, 0, ..., 0, 0, 0]) y_test
array([0, 1, 0, ..., 0, 0, 1])

```