Assignment-2

AssignmentDate	21 September2022
StudentName	Swetha.S
StudentRegisterNumber	710119104019
MaximumMarks	2

In[1]:

```
importpandasaspdimpo
rt numpyas
npimportseabornassns
importmatplotlib.pyplotasplt
%matplotlibinline
importscipy.stats
#importstatsmodels.apiassms
importstatsmodels.formula.apiassmf
fromstatsmodels.stats.stattoolsimportjarque_bera
```

In[2]:

```
data=pd.read_csv('Churn_Modelling.csv')dat
a
```

Out[2]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfP
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	
...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	

10000rows ×14 columns

DescribeFunction

In[7]:

```
data[['Age','Surname','Tenure','Balance']].describe()
```

Out[7]:

	Age	Tenure	Balance
count	10000.000000	10000.000000	10000.000000
mean	38.921800	5.012800	76485.889288

std	10.487806	2.892174	62397.405202
min	18.000000	0.000000	0.000000
25%	32.000000	3.000000	0.000000
50%	37.000000	5.000000	97198.540000
75%	44.000000	7.000000	127644.240000
max	92.000000	10.000000	250898.090000

Data Type

Loading[MathJax]/extensions/Safe.js

In[15]:
```
data.dtypes
```

Out[15]:
```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype:object
```

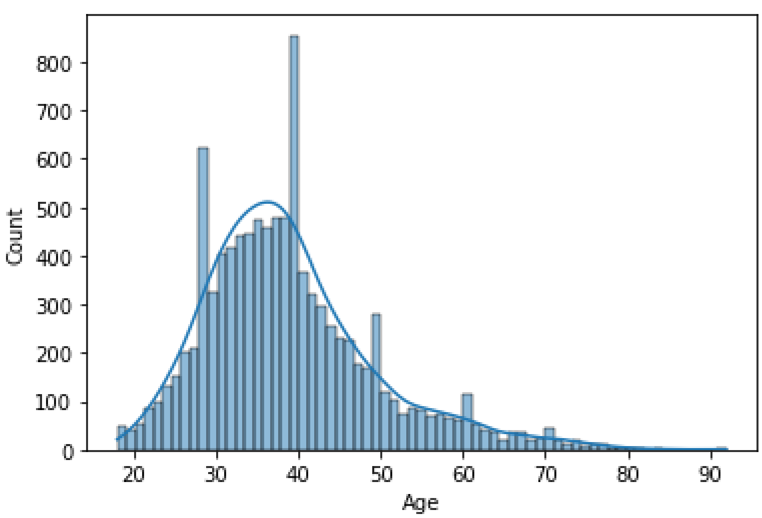
In[16]:
```
data.isnull().any()
```

Out[16]:
```
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
```

UNIVARIATEANALYSIS

In[18]:
```
sns.histplot(data.Age,kde=True)
```

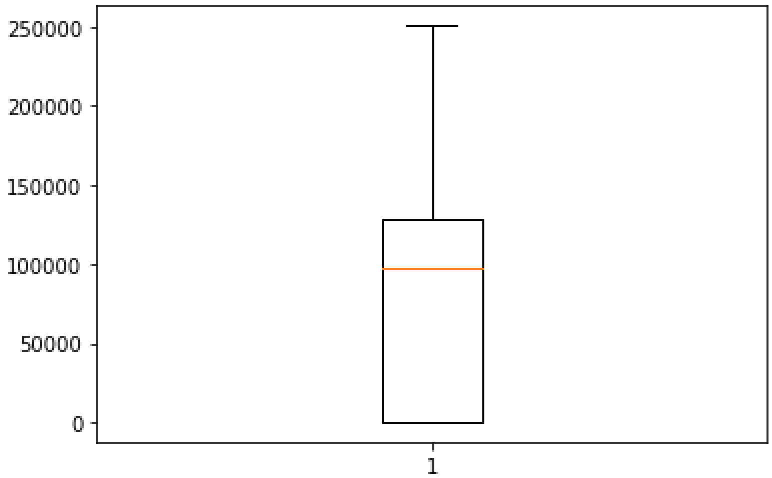
Out[18]:<AxesSubplot:xlabel='Age',ylabel='Count'>



BIVARIATEANALYSIS

In[29]:

```
plt.boxplot(data.Balance)
plt.show()
```

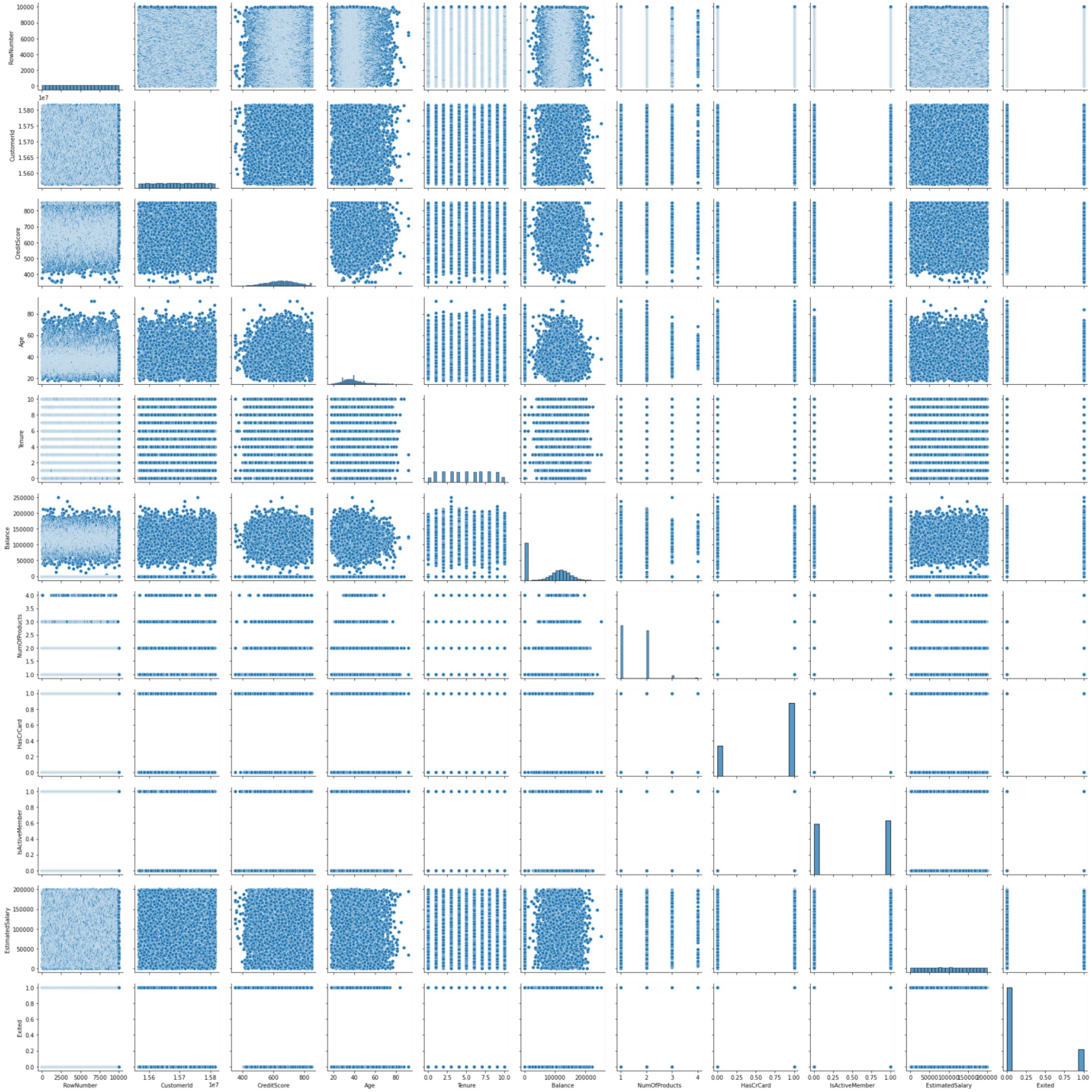


MULTIVARIATEANALYSIS

In[47]:

```
sns.pairplot(data)
```

Out[47]:<seaborn.axisgrid.PairGridat0x1cb8b759610>



Performdescriptivestatisticson thedataset

In[3]:

```
data.describe(include='all')
```

Out[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
count	10000.00000	1.000000e+04	10000	10000.000000	10000	10000	10000.000000	10000.000000
unique	NaN	NaN	2932	NaN	3	2	NaN	NaN
top	NaN	NaN	Smith	NaN	France	Male	NaN	NaN
freq	NaN	NaN	32	NaN	5014	5457	NaN	NaN
mean	5000.50000	1.569094e+07	NaN	650.528800	NaN	NaN	38.921800	5.012800
std	2886.89568	7.193619e+04	NaN	96.653299	NaN	NaN	10.487806	2.892174
min	1.00000	1.556570e+07	NaN	350.000000	NaN	NaN	18.000000	0.000000
25%	2500.75000	1.562853e+07	NaN	584.000000	NaN	NaN	32.000000	3.000000
50%	5000.50000	1.569074e+07	NaN	652.000000	NaN	NaN	37.000000	5.000000
75%	7500.25000	1.575323e+07	NaN	718.000000	NaN	NaN	44.000000	7.000000
max	10000.00000	1.581569e+07	NaN	850.000000	NaN	NaN	92.000000	10.000000

In[4]:

```
data.count()
```

Out[4]:

```
RowNumber          10000
CustomerId         10000
Surname            10000
CreditScore        10000
Geography          10000
Gender             10000
Age                10000
Tenure             10000
Balance            10000
NumOfProducts      10000
HasCrCard          10000
IsActiveMember     10000
EstimatedSalary    10000
Exited             10000
dtype:int64
```

HandletheMissingvalues.

FillwithZerosforNAN values

In[7]:

```
a=data.fillna(0)a
```

Out[7]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfP
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	
...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	

10000rows ×14 columns

Findthe outliersand replacetheoutliers

In[8]:

```
a
```

Out[8]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfP
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	
...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	

10000rows ×14 columns

In[9]:

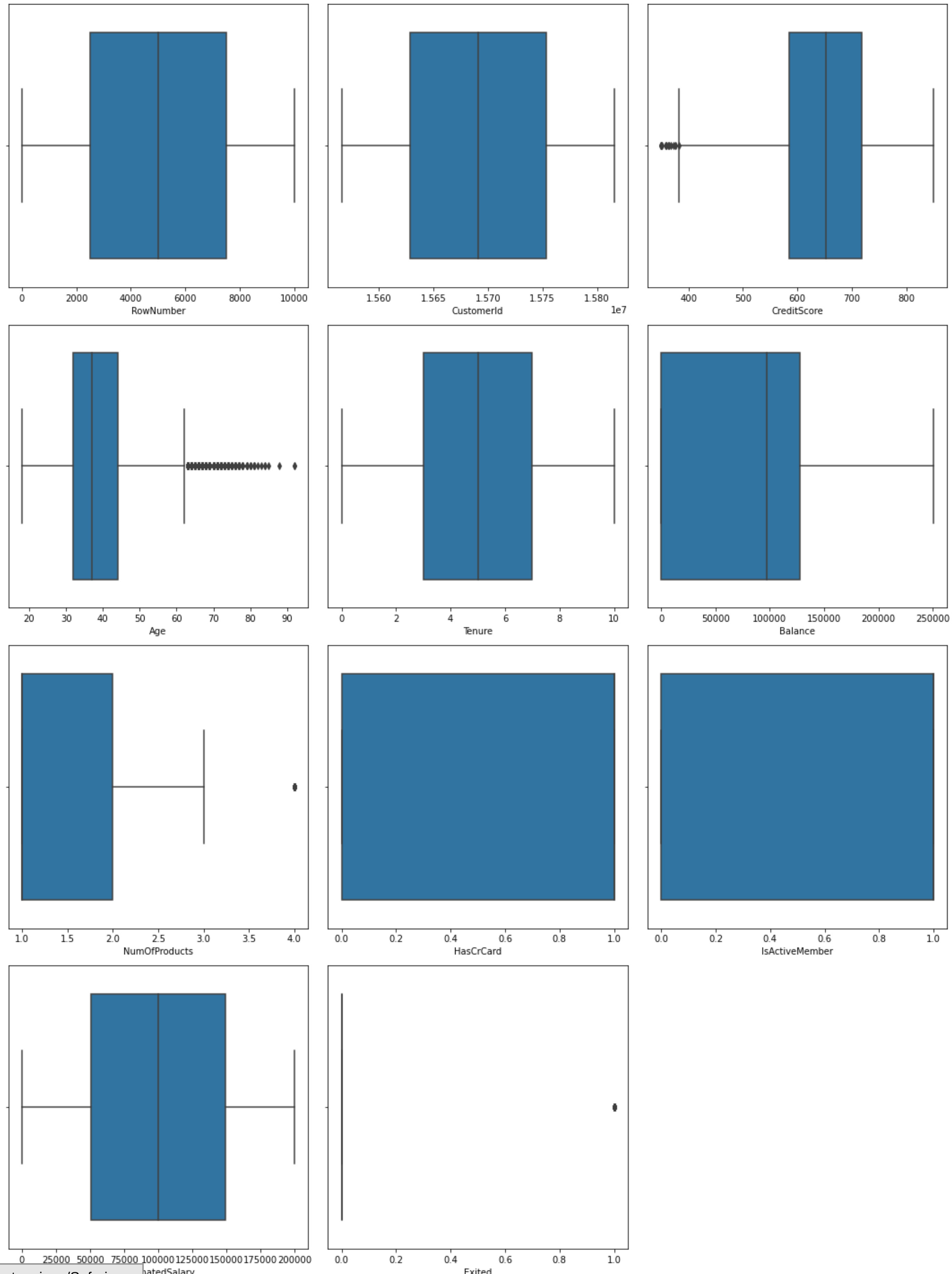
```
missing_values=data.isnull().sum()missing_values[
missing_values>0]/len(data)*100
```

Out[9]:Series([],dtype:float64)

In [13]:

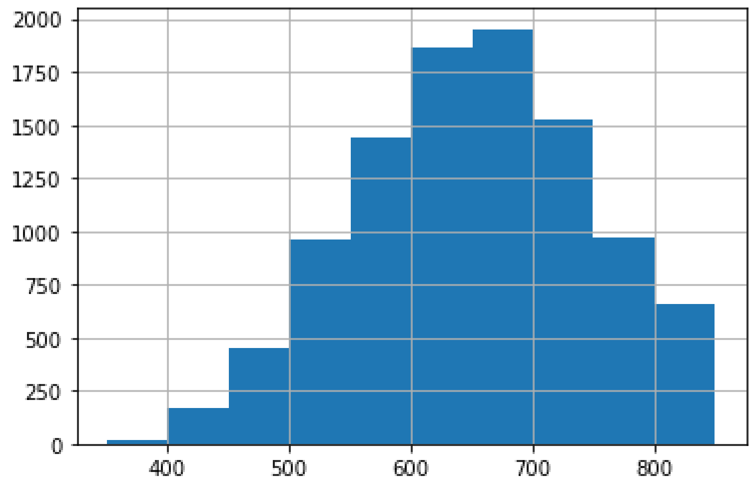
```
cols =3
rows =4
num_cols=data.select_dtypes(exclude='object').columns
fig = plt.figure( figsize=(cols*5, rows*5))
for i, col in enumerate(num_cols):
```

```
    ax=fig.add_subplot(rows,cols,i+1)

    sns.boxplot(x=data[col],ax=ax)

fig.tight_layout()
plt.show()
```



In[14]:
```
data['CreditScore'].hist()
```

Out[14]:
```
<AxesSubplot:>
```



In[15]:
```
print('SkewnessvalueofAge:',data['Age'].skew())Ag
e_mean=data['Age'].mean()
print('MeanofAgeis:',Age_mean)Ag
e_std=data['Age'].std()
print('StandardDeviationofAgeis:',Age_std)low=
Age_mean-(3*Age_std)
high=Age_mean+(3*Age_std)
Age_outliers=data[(data['Age']<low)|(data['Age']>high)]
#print('OutliersofAgeis:\n',Age_outliers)print
('Outliers of Age is:')Age_outliers.head()
```

```
SkewnessvalueofAge:1.0113202630234552Mean
of Ageis:38.9218
StandardDeviationofAgeis:10.4878064517045910utl
iers ofAge is:
```

Out[15]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfPro
85	86	15805254	Ndukaku	652	Spain	Female	75	10	0.00	
158	159	15589975	Maclean	646	France	Female	73	6	97259.25	
230	231	15808473	Ringrose	673	France	Male	72	1	0.00	
252	253	15793726	Matveyeva	681	France	Female	79	0	0.00	
310	311	15712287	Pokrovskii	652	France	Female	80	4	0.00	

CheckforCategoricalcolumnsandperformen coding.

In[]:
```
#data1=pd.read_csv('Churn_Modelling.csv')#data1.
head()
```

In [4]:
```
importnumpyasnp#fornumpyoperations
import pandas as pd#for creating DataFrame using
Pandas#tosplit thedataset usingsklearn
from sklearn.model_selectionimport train_test_split
#loadtitanicdataset
data1 = pd.read_csv('Churn_Modelling.csv',
```

```
 usecols=['Surname','Gender','Geography'])data1.
head()
```

Out[4]:

	Surname	Geography	Gender
0	Hargrave	France	Female
1	Hill	Spain	Female
2	Onio	France	Female
3	Boni	France	Female
4	Mitchell	Spain	Female

In[5]:

```
pd.get_dummies(data1)
```

Out[5]:

	Surname_Abazu	Surname_Abbie	Surname_Abbott	Surname_Abdullah	Surname_Abdulov	Surname_Abel
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
...	...	...	...	...	...	...
9995	0	0	0	0	0	0
9996	0	0	0	0	0	0
9997	0	0	0	0	0	0
9998	0	0	0	0	0	0
9999	0	0	0	0	0	0

10000rows×2937columns

In[17]:

```
#Returnsdictionaryhavingkeyascategoryandvaluesasnumber
deffind_category_mappings(data,variable):
 return{k:ifori,kinenumerate(data[variable].unique())}# Returns
the column after mapping with
dictionarydefinteger_encode(data,variable,ordinal_mapping):
 data[variable]=data[variable].map(ordinal_mapping)
forvariablein['Surname','Geography','Gender']:map
 pings=find_category_mappings(data1,variable)inte
 ger_encode(data1, variable,mappings)
data1.head()
```

Out[17]:

	Surname	Geography	Gender
0	0	0	0
1	1	1	0
2	2	0	0
3	3	0	0
4	4	1	0

Splitthedatainto dependentandindependent

Loading[MathJax]/extensions/Safe.js

variables.

DependentVariable:A dependentvariableisavariablewhosevaluedepends onanothervariable.

IndependentVariable:AnIndependentvariableisavariablewhosevalueneverdependsonanothervariable.

In[6]:

```
print("TheMinimumvalueofDataset:\n",data1.min(numeric_only=True))print("\n"
)print("TheMaximumvalueofDataset:\n",data1.max(numeric_only=True))print("\n
")print("TheMeanvalueofDataset:\n",data1.mean(numeric_only=True))print("
\n")


print(data1.count(0))
print(data1.shape)pr
int(data1.size)
```

```
TheMinimumvalueofDataset:Ser
 ies([],dtype:float64)


TheMaximumvalueofDataset:Se
 ries([],dtype:float64)


TheMeanvalueofDataset:Ser
 ies([],dtype:float64)


Surname      10000
Geography    10000
Genderdtype  10000
:int64
(10000,3)
30000
```

In[7]:

```
y =
data1["Surname"]x=data1.drop(columns=["
Surname"],axis=1)x.head()
```

Out[7]:

	Geography	Gender
0	France	Female
1	Spain	Female
2	France	Female
3	France	Female
4	Spain	Female

Scaletheindependentvariables

In[8]:
```
names=x.columnsnam
es
```

Out[8]:Index(['Geography','Gender'],dtype='object')In

[12]:
```
fromsklearn.preprocessingimportscale
x=scale(x)
```

In[16]:
```
x
```

Out[16]:

	Geography	Gender
0	France	Female
1	Spain	Female
2	France	Female
3	France	Female
4	Spain	Female
...	...	...
9995	France	Male
9996	France	Male
9997	France	Female
9998	Germany	Male
9999	France	Female

10000rows ×2 columns

Splitthedata intotrainingandtesting

The train-test split is used to estimate the performance of machine learning algorithms that are applicablefor prediction-based Algorithms/Applications. By default, the Test set is split into 30 % of actual data and thetrainingsetissplitinto70%oftheactual data.

In[18]:
```
fromsklearn.model_selectionimporttrain_test_split
```

In[19]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In[20]:
```
x_train.head()
```

Out[20]:

	Geography	Gender
7389	Spain	Female
9275	Germany	Male
2995	France	Female
5316	Spain	Male
356	Spain	Female

Loading[MathJax]/extensions/Safe.js

```
,y_train.shape,x_test.shape,y_test.shape
```

Out[21]: ((8000, 2), (8000,), (2000, 2), (2000,))