In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import kurtosis, skew
```

In [2]:
```
df=pd.read_csv("C:/Users/USER/Desktop/Datasets/churn_Modelling.csv")
```

In [3]:
```
df.tail()
```

Out[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balanc
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.(
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.(
9997	9998	15584532	Liu	709	France	Female	36	7	0.(
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.3
9999	10000	15628319	Walker	792	France	Female	28	4	130142.7

In [4]:
```
df.dtypes
```

Out[4]:
```
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 [5]:
```
df.shape
```

Out[5]:
```
(10000, 14)
```

In [6]:
```
df
```

Out[6]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	1	15634602	Hargrave	619	France	Female	42	2	0.(
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8
2	3	15619304	Onio	502	France	Female	42	8	159660.8
3	4	15701354	Boni	699	France	Female	39	1	0.(
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.(
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.6
9997	9998	15584532	Liu	709	France	Female	36	7	0.(
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.3
9999	10000	15628319	Walker	792	France	Female	28	4	130142.7

10000 rows × 14 columns

In [7]:
```
del df["CustomerId"]
del df["Surname"]
```

In [8]:
```
df
```

Out[8]:

	RowNumber	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	Has
0	1	619	France	Female	42	2	0.00	1	
1	2	608	Spain	Female	41	1	83807.86	1	
2	3	502	France	Female	42	8	159660.80	3	
3	4	699	France	Female	39	1	0.00	2	
4	5	850	Spain	Female	43	2	125510.82	1	
...	...	...	...	...	...	...	...	...	
9995	9996	771	France	Male	39	5	0.00	2	
9996	9997	516	France	Male	35	10	57369.61	1	
9997	9998	709	France	Female	36	7	0.00	1	
9998	9999	772	Germany	Male	42	3	75075.31	2	
9999	10000	792	France	Female	28	4	130142.79	1	

10000 rows × 12 columns

In [9]:
```
Idf=df.iloc[:,0:9]
Odf=df.iloc[:,-1]
```

In [10]:
```
Odf
```

Out[10]:
```
0       1
1       0
2       1
3       0
4       0
       ..
9995    0
9996    0
9997    1
9998    1
9999    0
Name: Exited, Length: 10000, dtype: int64
```

In [11]:
```
Idf
```

Out[11]:

	RowNumber	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	Has
0	1	619	France	Female	42	2	0.00	1	
1	2	608	Spain	Female	41	1	83807.86	1	
2	3	502	France	Female	42	8	159660.80	3	
3	4	699	France	Female	39	1	0.00	2	
4	5	850	Spain	Female	43	2	125510.82	1	
...	...	...	...	...	...	...	...	...	
9995	9996	771	France	Male	39	5	0.00	2	
9996	9997	516	France	Male	35	10	57369.61	1	
9997	9998	709	France	Female	36	7	0.00	1	
9998	9999	772	Germany	Male	42	3	75075.31	2	
9999	10000	792	France	Female	28	4	130142.79	1	

10000 rows × 9 columns

In [12]:
```
Idf.describe()
```

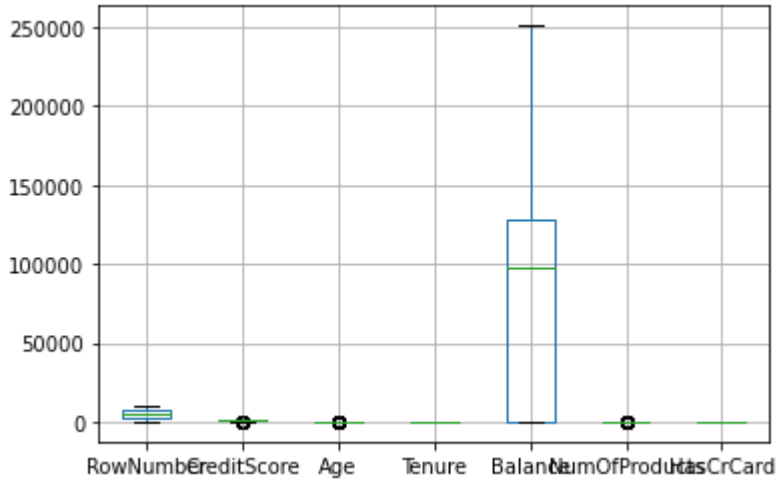
Out[12]:

	RowNumber	CreditScore	Age	Tenure	Balance	NumOfProducts	H
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10(
mean	5000.50000	650.528800	38.921800	5.012800	76485.889288	1.530200	
std	2886.89568	96.653299	10.487806	2.892174	62397.405202	0.581654	
min	1.00000	350.000000	18.000000	0.000000	0.000000	1.000000	
25%	2500.75000	584.000000	32.000000	3.000000	0.000000	1.000000	
50%	5000.50000	652.000000	37.000000	5.000000	97198.540000	1.000000	
75%	7500.25000	718.000000	44.000000	7.000000	127644.240000	2.000000	
max	10000.00000	850.000000	92.000000	10.000000	250898.090000	4.000000	

In [13]:
```
Idf.isnull().sum()
```

Out[13]:
```
RowNumber        0
CreditScore      0
Geography        0
Gender           0
Age              0
Tenure           0
Balance          0
NumOfProducts    0
HasCrCard        0
dtype: int64
```

In [14]:
```
boxplot=Idf.boxplot()
```



In [15]:
```
Newdf=pd.get_dummies(Idf)
Newdf
```

Out[15]:

	RowNumber	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	Geography_F
0	1	619	42	2	0.00	1	1	
1	2	608	41	1	83807.86	1	0	
2	3	502	42	8	159660.80	3	1	
3	4	699	39	1	0.00	2	0	
4	5	850	43	2	125510.82	1	1	
...	...	...	...	...	...	...	...	
9995	9996	771	39	5	0.00	2	1	
9996	9997	516	35	10	57369.61	1	1	
9997	9998	709	36	7	0.00	1	0	
9998	9999	772	42	3	75075.31	2	1	
9999	10000	792	28	4	130142.79	1	1	

10000 rows × 12 columns

In [16]:
```
Newdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   RowNumber          10000 non-null  int64  
 1   CreditScore        10000 non-null  int64  
 2   Age                10000 non-null  int64  
 3   Tenure             10000 non-null  int64  
 4   Balance            10000 non-null  float64
 5   NumOfProducts      10000 non-null  int64  
 6   HasCrCard          10000 non-null  int64  
 7   Geography_France   10000 non-null  uint8  
 8   Geography_Germany  10000 non-null  uint8  
 9   Geography_Spain    10000 non-null  uint8  
 10  Gender_Female      10000 non-null  uint8  
 11  Gender_Male        10000 non-null  uint8  
dtypes: float64(1), int64(6), uint8(5)
memory usage: 595.8 KB
```

In [17]:
```
from sklearn.preprocessing import scale
SNewdf=scale(Newdf)
```

In [18]:
```
names=Newdf.columns
names
```

Out[18]:
```
Index(['RowNumber', 'CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts',
       'HasCrCard', 'Geography_France', 'Geography_Germany', 'Geography_Spain',
       'Gender_Female', 'Gender_Male'],
      dtype='object')
```

In [19]:
```
S_newdf=pd.DataFrame(SNewdf, columns=names)
S_newdf
```

Out[19]:

	RowNumber	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	Geog
0	-1.731878	-0.326221	0.293517	-1.041760	-1.225848	-0.911583	0.646092	
1	-1.731531	-0.440036	0.198164	-1.387538	0.117350	-0.911583	-1.547768	
2	-1.731185	-1.536794	0.293517	1.032908	1.333053	2.527057	0.646092	
3	-1.730838	0.501521	0.007457	-1.387538	-1.225848	0.807737	-1.547768	
4	-1.730492	2.063884	0.388871	-1.041760	0.785728	-0.911583	0.646092	
...	...	...	...	...	...	...	...	
9995	1.730492	1.246488	0.007457	-0.004426	-1.225848	0.807737	0.646092	
9996	1.730838	-1.391939	-0.373958	1.724464	-0.306379	-0.911583	0.646092	
9997	1.731185	0.604988	-0.278604	0.687130	-1.225848	-0.911583	-1.547768	
9998	1.731531	1.256835	0.293517	-0.695982	-0.022608	0.807737	0.646092	
9999	1.731878	1.463771	-1.041433	-0.350204	0.859965	-0.911583	0.646092	

10000 rows × 12 columns

In [47]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(S_newdf, Odf, test_size=0.2)
```

In [48]:
```
# !pip install xgboost
from xgboost import XGBClassifier
xgb=XGBClassifier()
```

In [49]:
```
xgb.fit(x_train,y_train)
```

```
[11:18:17] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.
1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[49]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [50]:
```
y_pred=xgb.predict(x_test)
```

In [51]:
```
y_pred
```

Out[51]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [52]:
```
from sklearn.metrics import accuracy_score
```

In [54]:
```
Accuracy_Score=accuracy_score(y_test, y_pred)
Accuracy_Score
```

Out[54]:
```
0.85
```

In [ ]:
```

```