In [23]:
```
import pandas as pd
import pickle
import warnings
warnings.filterwarnings('ignore')
```

In [24]:
```
a=pd.read_excel("C:\\Users\\reshma_koduri\\OneDrive\\Documents\\P12-bank.xlsx")
a
```

Out[24]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	France	Female	42	2	0.0
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	0.0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8
...	...	...	...	...	...	...	...	...	..
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61
9997	9998	15584532	Liu	709	France	Female	36	7	0.0
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31
9999	10000	15628319	Walker	792	France	Female	28	4	130142.7

10000 rows × 14 columns

In [25]:
```
a.head()
```

Out[25]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	N
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	

In [26]:
```
a.tail()
```

Out[26]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61
9997	9998	15584532	Liu	709	France	Female	36	7	0.0

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79

In [27]:

```
a.describe()
```

Out[27]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfl
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	1000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	

In [28]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [29]:

```
a.isna().sum()
```

Out[29]:

```
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
```

In [30]:
```
a['Gender']=a['Gender'].map({'Male':1,'Female':0})
a
```

Out[30]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	France	0	42	2	0.0
1	2	15647311	Hill	608	Spain	0	41	1	83807.8
2	3	15619304	Onio	502	France	0	42	8	159660.8
3	4	15701354	Boni	699	France	0	39	1	0.0
4	5	15737888	Mitchell	850	Spain	0	43	2	125510.8
...	...	...	...	...	...	...	...	...	..
9995	9996	15606229	Obijiaku	771	France	1	39	5	0.0
9996	9997	15569892	Johnstone	516	France	1	35	10	57369.61
9997	9998	15584532	Liu	709	France	0	36	7	0.0
9998	9999	15682355	Sabbatini	772	Germany	1	42	3	75075.31
9999	10000	15628319	Walker	792	France	0	28	4	130142.7

10000 rows × 14 columns

In [51]:
```
a['Geography'].value_counts()
```

Out[51]:
```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

In [55]:
```
d=a.loc[(a.Geography =='France')]
d
```

Out[55]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	France	0	42	2	0.0
2	3	15619304	Onio	502	France	0	42	8	159660.8
3	4	15701354	Boni	699	France	0	39	1	0.0
6	7	15592531	Bartlett	822	France	1	50	7	0.0
8	9	15792365	He	501	France	1	44	4	142051.0
...	...	...	...	...	...	...	...	...	..
9994	9995	15719294	Wood	800	France	0	29	2	0.0

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
9995	9996	15606229	Obijiaku	771	France	1	39	5	0.0
9996	9997	15569892	Johnstone	516	France	1	35	10	57369.61
9997	9998	15584532	Liu	709	France	0	36	7	0.0
9999	10000	15628319	Walker	792	France	0	28	4	130142.7

5014 rows × 14 columns

In [31]:
```python
a.groupby(['Gender']).count()
```

Out[31]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Age	Tenure	Balance	NumO
Gender									
0	4543	4543	4543	4543	4543	4543	4543	4543	
1	5457	5457	5457	5457	5457	5457	5457	5457	

In [63]:
```python
b=d.drop(['Surname','RowNumber'],axis=1)
b
```

Out[63]:

	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCr
0	15634602	619	France	0	42	2	0.00	1	
2	15619304	502	France	0	42	8	159660.80	3	
3	15701354	699	France	0	39	1	0.00	2	
6	15592531	822	France	1	50	7	0.00	2	
8	15792365	501	France	1	44	4	142051.07	2	
...	...	...	...	...	...	...	...	...	
9994	15719294	800	France	0	29	2	0.00	2	
9995	15606229	771	France	1	39	5	0.00	2	
9996	15569892	516	France	1	35	10	57369.61	1	
9997	15584532	709	France	0	36	7	0.00	1	
9999	15628319	792	France	0	28	4	130142.79	1	

5014 rows × 12 columns

In [64]:
```python
c=pd.get_dummies(b,dtype=int)
c
```

Out[64]:

	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiv
0	15634602	619	0	42	2	0.00	1	1	

	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiv
2	15619304	502	0	42	8	159660.80	3	1	
3	15701354	699	0	39	1	0.00	2	0	
6	15592531	822	1	50	7	0.00	2	1	
8	15792365	501	1	44	4	142051.07	2	0	
...	...	...	...	...	...	...	...	...	
9994	15719294	800	0	29	2	0.00	2	0	
9995	15606229	771	1	39	5	0.00	2	1	
9996	15569892	516	1	35	10	57369.61	1	1	
9997	15584532	709	0	36	7	0.00	1	0	
9999	15628319	792	0	28	4	130142.79	1	1	

5014 rows × 12 columns

In [65]:

```
cor=d.corr()
cor
```

Out[65]:

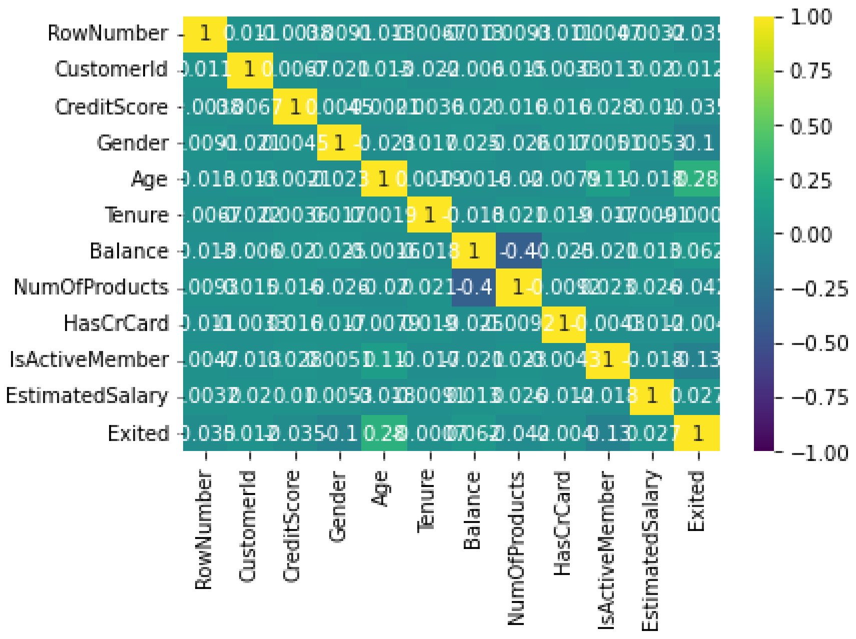
	RowNumber	CustomerId	CreditScore	Gender	Age	Tenure	Balance	N
RowNumber	1.000000	0.010675	-0.003769	0.009079	-0.013407	-0.006657	-0.012978	
CustomerId	0.010675	1.000000	0.006693	-0.020979	0.012794	-0.022154	-0.005974	
CreditScore	-0.003769	0.006693	1.000000	0.004508	-0.002055	0.003578	0.019835	
Gender	0.009079	-0.020979	0.004508	1.000000	-0.022701	0.017121	0.025013	
Age	-0.013407	0.012794	-0.002055	-0.022701	1.000000	0.001914	-0.001593	
Tenure	-0.006657	-0.022154	0.003578	0.017121	0.001914	1.000000	-0.017998	
Balance	-0.012978	-0.005974	0.019835	0.025013	-0.001593	-0.017998	1.000000	
NumOfProducts	0.009344	0.015048	0.016338	-0.026430	-0.019721	0.021043	-0.399907	
HasCrCard	-0.011132	-0.003326	0.016179	0.017317	-0.007916	0.018768	-0.024738	
IsActiveMember	0.004721	-0.013448	0.027649	0.005116	0.107284	-0.016566	-0.020674	
EstimatedSalary	0.003171	0.020026	0.010136	0.005294	-0.017982	-0.009079	0.012666	
Exited	-0.035146	0.012089	-0.035084	-0.103180	0.277646	-0.000697	0.062290	

In [87]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(cor,vmin=-1,vmax=1,annot=True,cmap='viridis')
```

Out[87]:

```
<AxesSubplot:>
```



In [67]:
```
b.groupby(['IsActiveMember']).count()
```

Out[67]:

IsActiveMember	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProdu
0	2423	2423	2423	2423	2423	2423	2423	24
1	2591	2591	2591	2591	2591	2591	2591	25

In [68]:
```
y=c['Exited']
y
```

Out[68]:
```
0       1
2       1
3       0
6       0
8       0
       ..
9994    0
9995    0
9996    0
9997    1
9999    0
Name: Exited, Length: 5014, dtype: int64
```

In [69]:
```
x=c.drop(['Exited'],axis=1)
x
```

Out[69]:

	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiv
0	15634602	619	0	42	2	0.00	1	1	
2	15619304	502	0	42	8	159660.80	3	1	
3	15701354	699	0	39	1	0.00	2	0	
6	15592531	822	1	50	7	0.00	2	1	

	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiv
8	15792365	501	1	44	4	142051.07	2	0	
...	...	...	...	...	...	...	...	...	
9994	15719294	800	0	29	2	0.00	2	0	
9995	15606229	771	1	39	5	0.00	2	1	
9996	15569892	516	1	35	10	57369.61	1	1	
9997	15584532	709	0	36	7	0.00	1	0	
9999	15628319	792	0	28	4	130142.79	1	1	

5014 rows × 11 columns

In [70]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
```

In [71]:
```
from sklearn.linear_model import LogisticRegression
cls=LogisticRegression()
cls.fit(x_train,y_train)
```

Out[71]:
```
▾ LogisticRegression

LogisticRegression()
```

In [72]:
```
ypred=cls.predict(x_test)
ypred
```

Out[72]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [76]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,ypred)
```

Out[76]:
```
array([[1245,    0],
       [ 260,    0]], dtype=int64)
```

In [77]:
```
from sklearn.metrics import accuracy_score
accuracy_score(ypred,y_test)
```

Out[77]:
```
0.8272425249169435
```

In [78]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
reg=RandomForestClassifier()
n_estimators=[25,50,75,100,125,150,175,200]
criterion=['gini','entropy']
max_depth=[3,5,10]
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_depth
rfc_reg = GridSearchCV(reg, parameters)
rfc_reg.fit(x_train,y_train)
```

Out[78]:

```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [79]:

```
rfc_reg.best_params_
```

Out[79]:

```
{'criterion': 'gini', 'max_depth': 10, 'n_estimators': 75}
```

In [80]:

```
reg=RandomForestClassifier(n_estimators=75,criterion='gini',max_depth=10)
reg.fit(x_train,y_train)
```

Out[80]:

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=10, n_estimators=75)
```

In [81]:

```
ypred=reg.predict(x_test)
ypred
```

Out[81]:

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

In [83]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,ypred)
```

Out[83]:

```
array([[1228,   17],
       [ 166,   94]], dtype=int64)
```

In [82]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,ypred)
```

Out[82]:

```
0.878405315614618
```

In [ ]: