ann-model-1

September 12, 2024

1 By prisca

[3]:
```
# this bank have an issue of customers just leaving their bank and they want be␣
↪able to identify those customer even before
#the leave.
# this project is about builidng a model to predict customers who are likely to␣
↪leave than bank and vice visa with following
# features, using Artificial Neutral Network
```

[97]:
```
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import numpy
import tensorflow as tf
import seaborn as sns
```

[160]:
```
import warnings
warnings.filterwarnings('ignore')
```

[98]:
```
df=pd.read_csv(r'C:\Users\USER\Documents\Churn_Modelling.csv')
```

[99]:
```
df.head()
```

[99]:
```
   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
```

[100]:
```
df.shape
```

[100]:
```
(10000, 14)
```

[101]:
```
dfr=df.drop(['RowNumber', 'CustomerId', 'Surname',␣
 ↪'CreditScore'],axis=1,inplace=True)
```

[102]:
```
df.head()
```

[102]:
```
  Geography  Gender  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0    France  Female   42       2       0.00              1          1
1     Spain  Female   41       1   83807.86              1          0
2    France  Female   42       8  159660.80              3          1
3    France  Female   39       1       0.00              2          0
4     Spain  Female   43       2  125510.82              1          1

   IsActiveMember  EstimatedSalary  Exited
0               1        101348.88       1
1               1        112542.58       0
2               0        113931.57       1
3               0         93826.63       0
4               1         79084.10       0
```

[148]:
```
# visualization
```

[103]:
```
df.isnull().sum()
```

[103]:
```
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

[104]:
```
df.duplicated().sum()
```

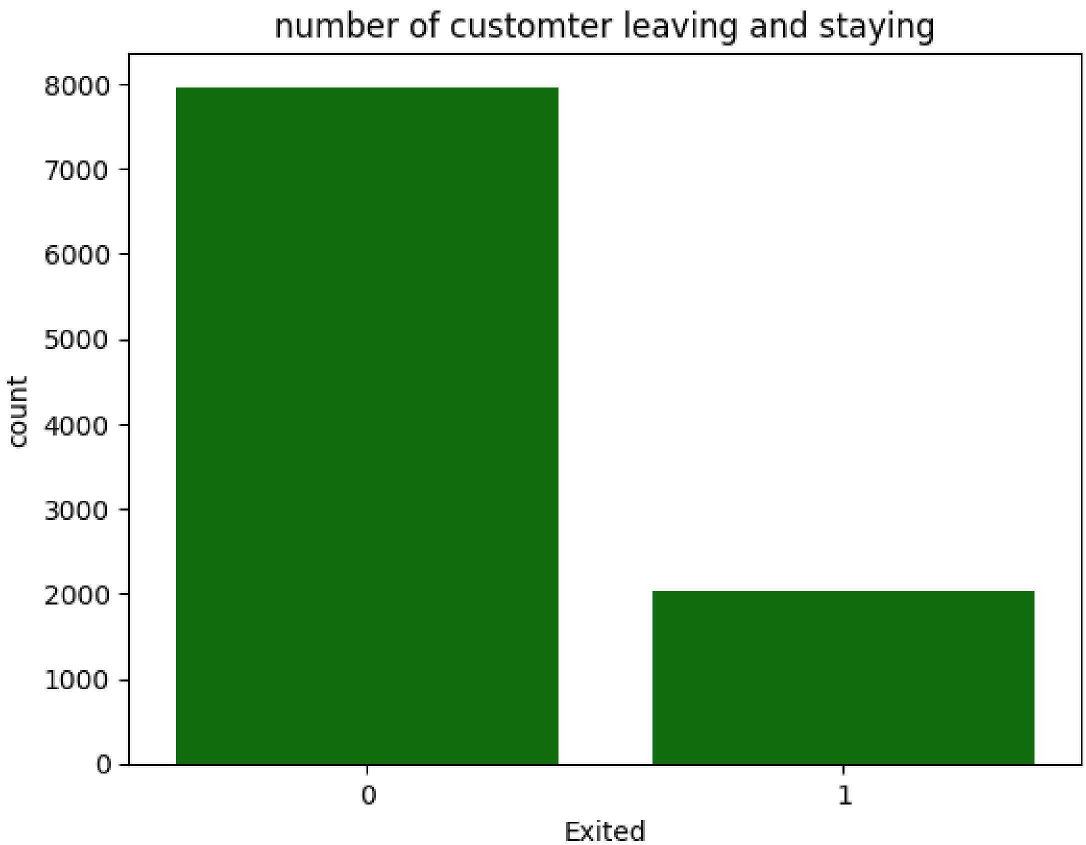
[104]: 0

2 visualization

[176]:
```
sns.countplot(x='Exited',data=df,color='green')
plt.title('number of customter leaving and staying')

# the number of customer leaving is less than the number of cutsomer staying
```

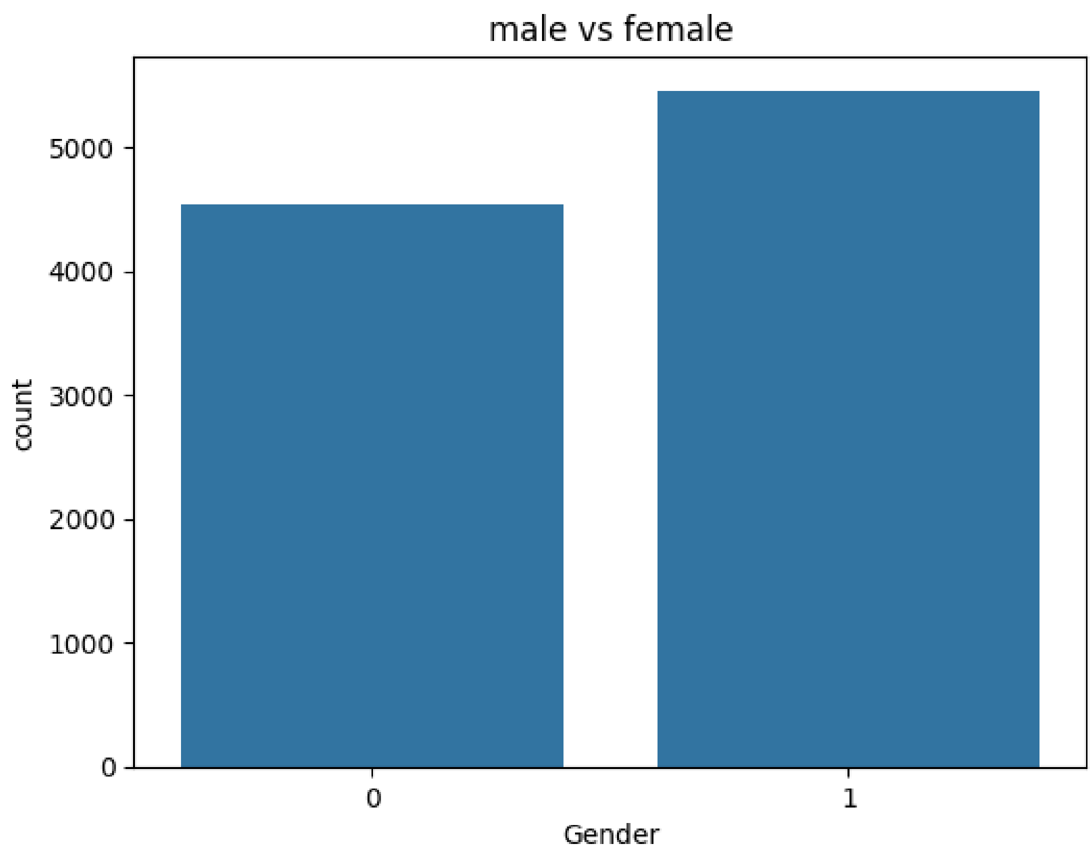
[176]: Text(0.5, 1.0, 'number of customter leaving and staying')



[155]:
```
sns.countplot(x='Gender',data=df)
plt.title('male vs female')

# the bank has more male customers than female with little differnce
```

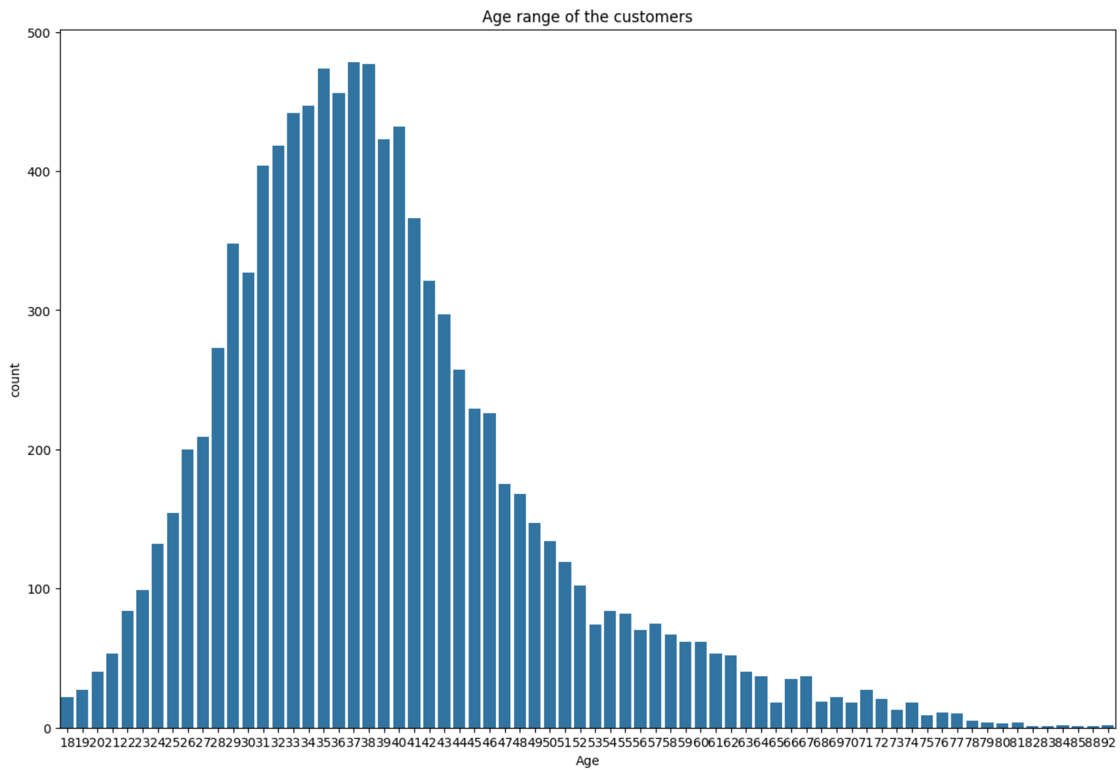
[155]: Text(0.5, 1.0, 'male vs female')



[162]:
```
fig,ax=plt.subplots(figsize=(15,10))
sns.countplot(x='Age',data=df)
plt.title('Age range of the customers')

# the customers ranges from 18 to 92 years old
# the have high amount of customers from age 29 to 47 years
```

[162]: Text(0.5, 1.0, 'Age range of the customers')

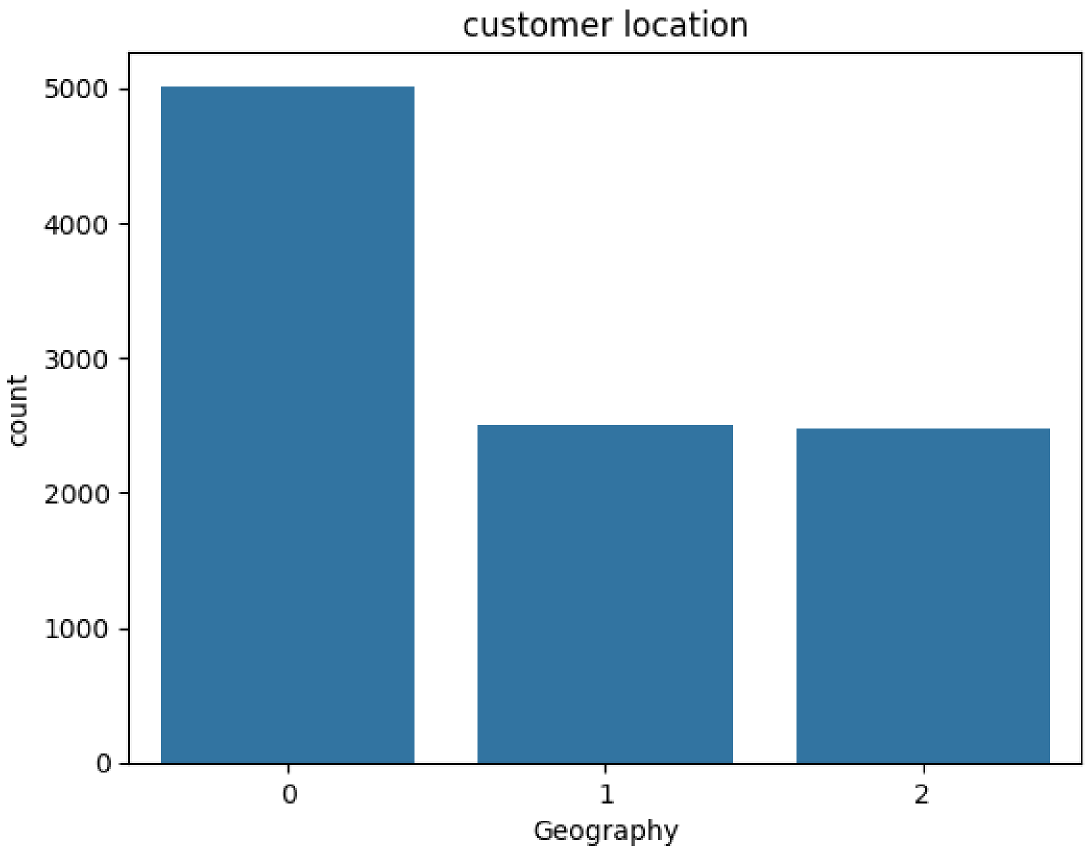


[164]:
```
df['Geography'].value_counts()
```

[164]:
```
Geography
0    5014
1    2509
2    2477
Name: count, dtype: int64
```

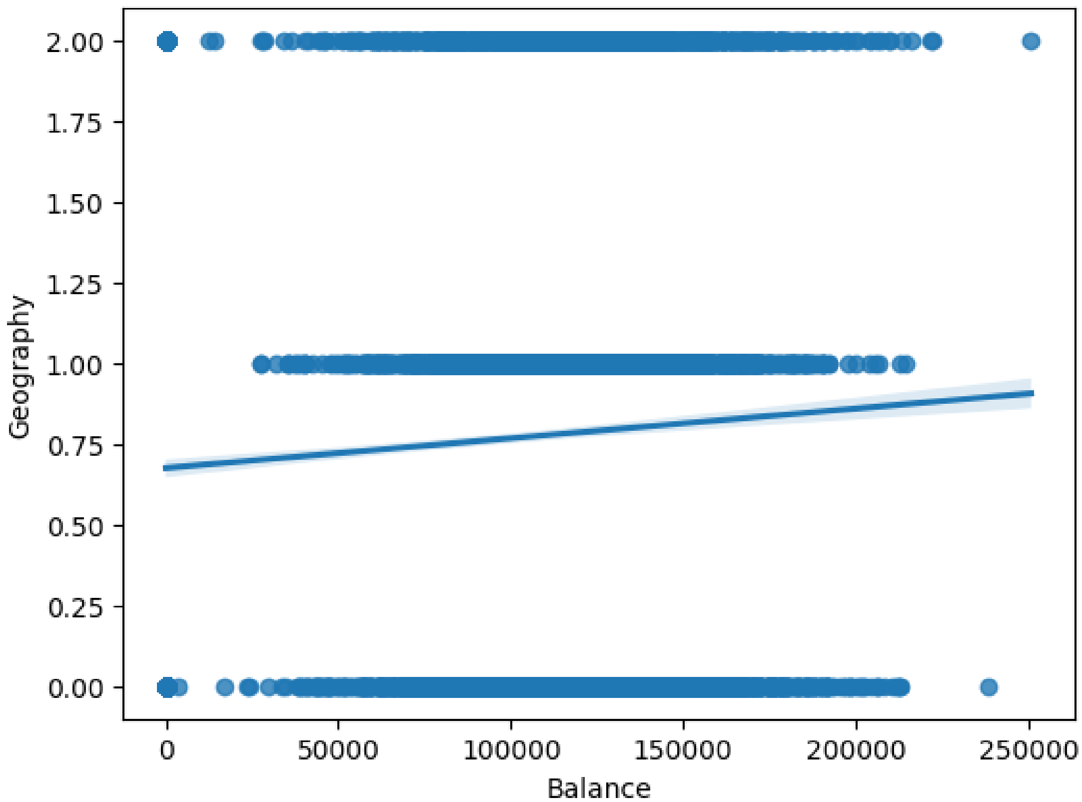
[166]:
```
sns.countplot(x='Geography',data=df)
plt.title('customer location')
```

[166]:
```
Text(0.5, 1.0, 'customer location')
```



[173]:
```
sns.regplot(x='Balance',y='Geography',data=df)
```

[173]: <Axes: xlabel='Balance', ylabel='Geography'>

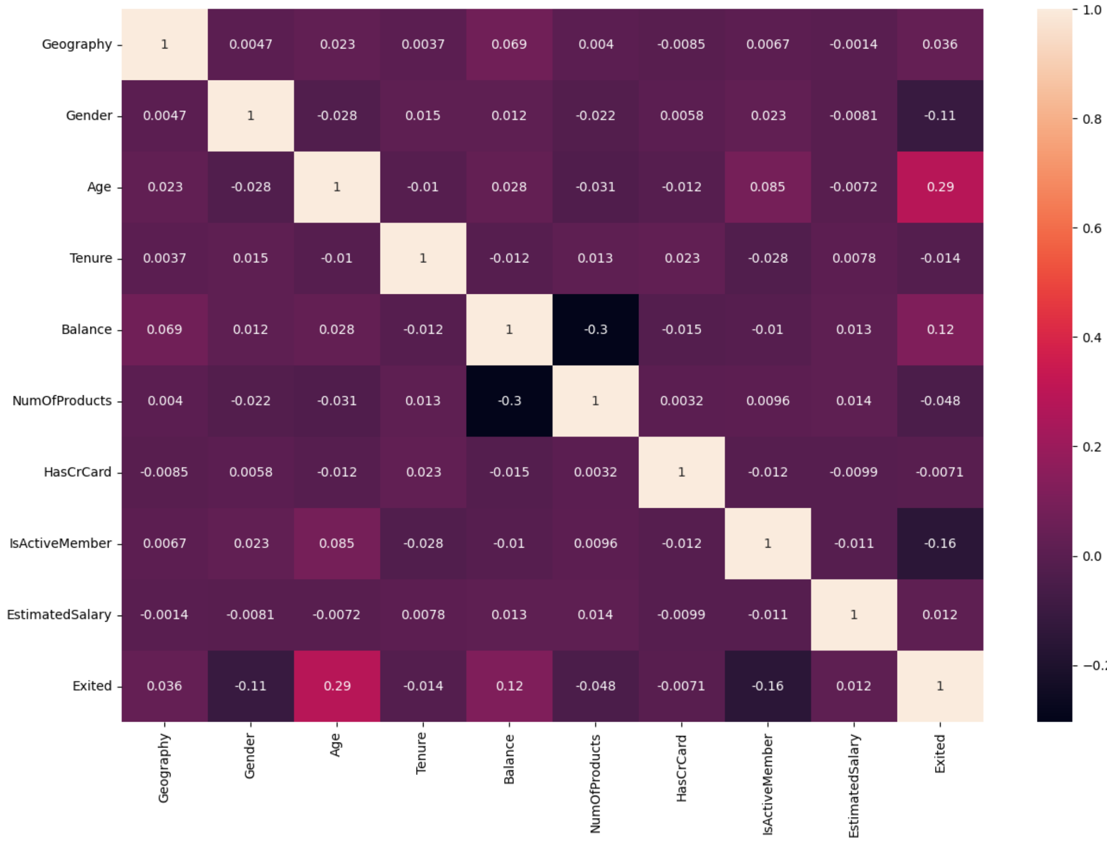


[169]:
```
df_corr=df.corr()
```

[175]:
```
fig,ax=plt.subplots(figsize=(15,10))
sns.heatmap(df_corr,annot=True)

# from the map, there is no high correlation among the variabes
```

[175]: <Axes: >



[ ]:

[ ]:

[ ]:

[105]:
```
from sklearn .preprocessing import LabelEncoder
```

[106]:
```
le=LabelEncoder()
```

[107]:
```
df['Geography']=le.fit_transform(df['Geography'])
```

[108]:
```
df['Gender']=le.fit_transform(df['Gender'])
```

[109]:
```
df.head()
```

[109]:
```
   Geography  Gender  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0          0       0   42       2       0.00              1          1
1          2       0   41       1   83807.86              1          0
2          0       0   42       8  159660.80              3          1
```

```
3          0       0   39       1        0.00              2           0
4          2       0   43       2   125510.82              1           1

   IsActiveMember  EstimatedSalary  Exited
0               1        101348.88       1
1               1        112542.58       0
2               0        113931.57       1
3               0         93826.63       0
4               1         79084.10       0
```

[110]:
```
from sklearn .preprocessing import StandardScaler
```

[111]:
```
from sklearn.model_selection import train_test_split
```

[112]:
```
df.columns
```

[112]:
```
Index(['Geography', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts',
       'HasCrCard', 'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

[113]:
```
x=df[['Geography', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts',
       'HasCrCard', 'IsActiveMember', 'EstimatedSalary']]
y=df['Exited']
```

[114]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

[115]:
```
scaler=StandardScaler()
```

[116]:
```
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

[117]:
```
x_train.shape,x_test.shape
```

[117]:
```
((7000, 9), (3000, 9))
```

[118]:
```
#building the machine brain
```

[119]:
```
ann=tf.keras.models.Sequential()
```

[120]:
```
ann.add(tf.keras.layers.Dense(units=6,activation='relu'))
```

[121]:
```
ann.add(tf.keras.layers.Dense(units=6,activation='relu'))
```

[122]:
```
ann.add(tf.keras.layers.Dense(units=1,activation='sigmoid'))
```

[123]:
```
#compile the ann
```

[124]:
```
ann.compile(optimizer='adam',loss='binary_crossentropy',metrics='accuracy')
```

[125]:
```
#train the model
```

[126]:
```
ann.fit(x_train,y_train,batch_size=32,epochs=50)
```

```
Epoch 1/50
219/219 [==============================] - 2s 3ms/step - loss: 0.5421 -
accuracy: 0.7667
Epoch 2/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4708 -
accuracy: 0.8014
Epoch 3/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4451 -
accuracy: 0.8083
Epoch 4/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4293 -
accuracy: 0.8164
Epoch 5/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4172 -
accuracy: 0.8227
Epoch 6/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4048 -
accuracy: 0.8326
Epoch 7/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3938 -
accuracy: 0.8401
Epoch 8/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3847 -
accuracy: 0.8437
Epoch 9/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3777 -
accuracy: 0.8490
Epoch 10/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3727 -
accuracy: 0.8507
Epoch 11/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3692 -
accuracy: 0.8517
Epoch 12/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3666 -
accuracy: 0.8521
Epoch 13/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3642 -
accuracy: 0.8529
Epoch 14/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3629 -
```

```
accuracy: 0.8540
Epoch 15/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3613 -
accuracy: 0.8537
Epoch 16/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3600 -
accuracy: 0.8547
Epoch 17/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3594 -
accuracy: 0.8540
Epoch 18/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3583 -
accuracy: 0.8559
Epoch 19/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3574 -
accuracy: 0.8547
Epoch 20/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3566 -
accuracy: 0.8556
Epoch 21/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3559 -
accuracy: 0.8561
Epoch 22/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3550 -
accuracy: 0.8551
Epoch 23/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3545 -
accuracy: 0.8559
Epoch 24/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3539 -
accuracy: 0.8560
Epoch 25/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3531 -
accuracy: 0.8563
Epoch 26/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3528 -
accuracy: 0.8574
Epoch 27/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3523 -
accuracy: 0.8584
Epoch 28/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3523 -
accuracy: 0.8557
Epoch 29/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3515 -
accuracy: 0.8570
Epoch 30/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3513 -
```

```
accuracy: 0.8573
Epoch 31/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3513 -
accuracy: 0.8569
Epoch 32/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3507 -
accuracy: 0.8576
Epoch 33/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3504 -
accuracy: 0.8571
Epoch 34/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3504 -
accuracy: 0.8577
Epoch 35/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3500 -
accuracy: 0.8573
Epoch 36/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3497 -
accuracy: 0.8584
Epoch 37/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3490 -
accuracy: 0.8586
Epoch 38/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3490 -
accuracy: 0.8583
Epoch 39/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3487 -
accuracy: 0.8587
Epoch 40/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3485 -
accuracy: 0.8581
Epoch 41/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3483 -
accuracy: 0.8576
Epoch 42/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3481 -
accuracy: 0.8573
Epoch 43/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3477 -
accuracy: 0.8583
Epoch 44/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3474 -
accuracy: 0.8581
Epoch 45/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3474 -
accuracy: 0.8580
Epoch 46/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3471 -
```

```
accuracy: 0.8577
Epoch 47/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3469 -
accuracy: 0.8603
Epoch 48/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3467 -
accuracy: 0.8600
Epoch 49/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3469 -
accuracy: 0.8574
Epoch 50/50
219/219 [==============================] - 1s 3ms/step - loss: 0.3463 -
accuracy: 0.8593
```

[126]:
```
<keras.src.callbacks.History at 0x223ca10d880>
```

[127]:
```
#predictions
```

[128]:
```
ypred=ann.predict(x_test)
```

```
94/94 [==============================] - 0s 2ms/step
```

[129]:
```
ypred=(ypred>0.5)
```

[130]:
```
ypred
```

[130]:
```
array([[False],
       [False],
       [False],
       …,
       [False],
       [False],
       [False]])
```

[131]:
```
y_pred= [1 if pred > 0.5 else 0 for pred in ypred]
```

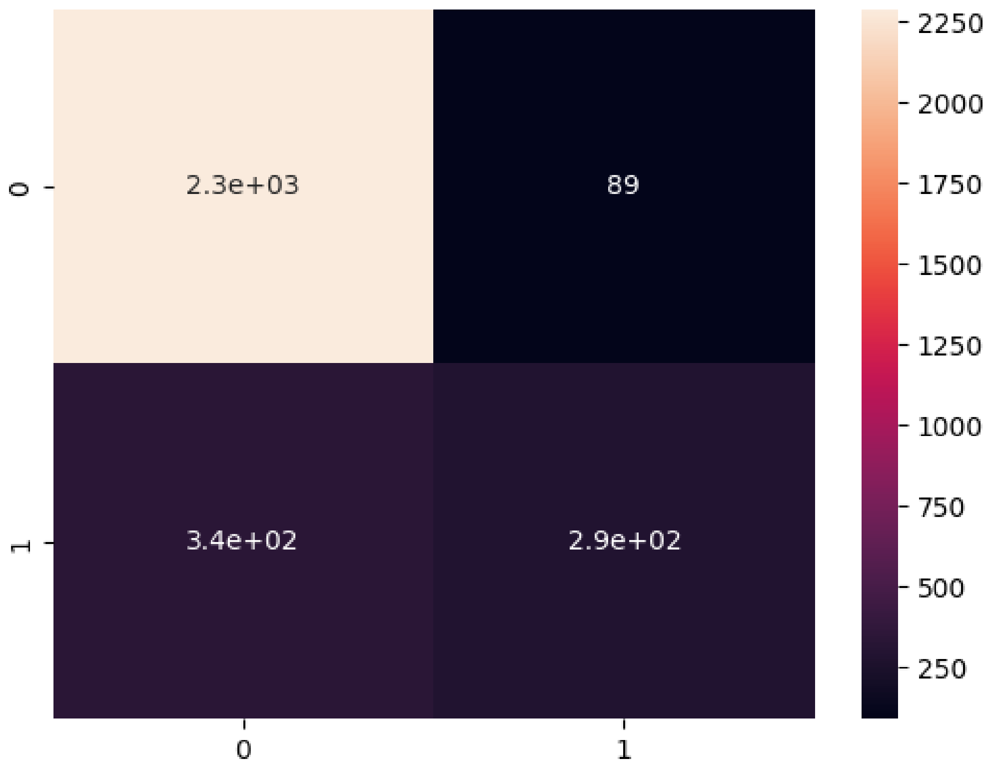
[137]:
```
#evaluating the model accuracy
```

[132]:
```
from sklearn import metrics
from sklearn.metrics import confusion_matrix
```

[133]:
```
cm=(confusion_matrix(y_test,ypred))
```

[134]:
```
sns.heatmap(cm,annot=True)
```

[134]:
```
<Axes: >
```



[135]:
```
cm
```

[135]:
```
array([[2290,   89],
       [ 335,  286]], dtype=int64)
```

[136]:
```
metrics.accuracy_score(y_pred,y_test)
```

[136]:
```
0.8586666666666667
```

3 testing the accuracy

[177]:
```
ann.predict(scaler.transform([[0,1,40,3,60000,2,1,1,50000]])) # nicely predicted
```

```
1/1 [==============================] - 0s 38ms/step
```

[177]:
```
array([[0.03612975]], dtype=float32)
```

[178]:
```
ann.predict(scaler.transform([[0,        0,        42,        8,        159660.
↪80,        3        ,1        ,0        ,113931.57        ]])) # correctly␣
↪predicted
```

```
1/1 [==============================] - 0s 38ms/step
```

[178]: array([[0.9962028]], dtype=float32)

[147]:
```
# 0.9965 means customer will leave the bank
```

[4]:
```
# with this model the bank can detect customers who are about to leave the bank␣
 ↪ and take measures to
#retain them
```

[ ]:
```

```