```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
import os
%matplotlib inline
```

```
df= pd.read_csv('/content/Churn_Modelling.csv')
```

```
df.shape
```

```
(10000, 14)
```

```
df.sample(5)
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
5687	5688	15691840	Fraser	505	Germany	Female	37	6	159863.90	2	0	1	125307.87
3945	3946	15652789	Hancock	657	Spain	Male	40	10	0.00	2	1	1	52990.70
8235	8236	15760177	Lombardi	564	Spain	Male	37	9	100252.18	1	1	1	146033.52
4469	4470	15692443	Piccio	612	Spain	Male	33	5	69478.57	1	1	0	8973.67
1702	1703	15713644	Marshall	686	Spain	Male	22	5	0.00	2	1	0	158974.45

```
df.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
```

```
df.drop(['RowNumber','CustomerId','Surname'],axis=1,inplace=True)
```

```
df.dtypes
```

```
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

```
df.Exited.value_counts()
```

```
0    7963
1    2037
Name: Exited, dtype: int64
```

```
df.isna().sum()
```

```
CreditScore        0
Geography          0
```

```
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

```
cat_cols=['Geography','Gender']
num_cols=[col for col in df.columns if col not in cat_cols]
```

```
for col in cat_cols:
    print(f'{col} : {df[col].unique()}')
```

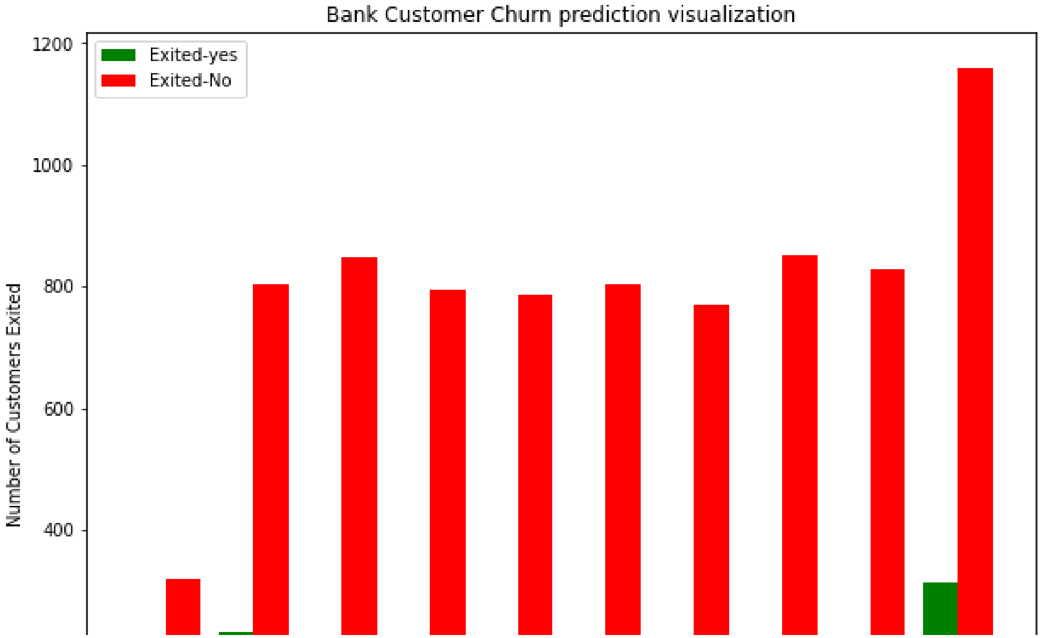
```
Geography : ['France' 'Spain' 'Germany']
Gender : ['Female' 'Male']
```

```
df['Gender'].replace({'Female':1,'Male':0},inplace=True)
df=pd.get_dummies(data=df, columns=['Geography'])
```

```
tenure_exited_0=df[df.Exited==0].Tenure
tenure_exited_1=df[df.Exited==1].Tenure
```

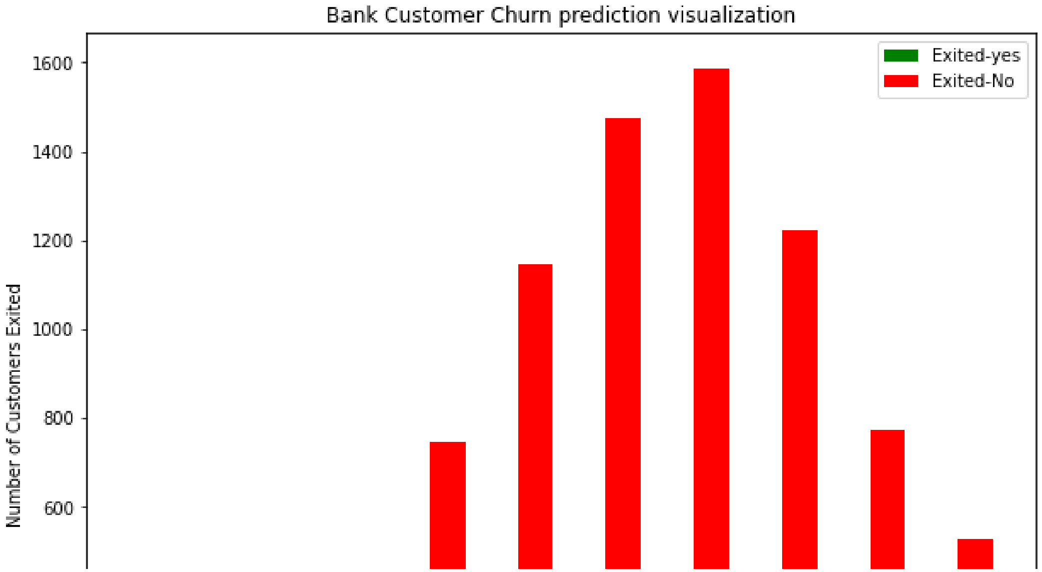
```
plt.figure(figsize=(10,8))
plt.xlabel('T enure')
plt.ylabel('Number of Customers Exited')
plt.title('Bank Customer Churn prediction visualization')
plt.hist([tenure_exited_1,tenure_exited_0], color=['green','red'], label=['Exited-yes','Exited-No'])
plt.legend()
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/fromnumeric.py:3208: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (whi
  return asarray(a).size
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:1376: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (
  X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else np.asarray(X))
<matplotlib.legend.Legend at 0x7f386975df10>
```



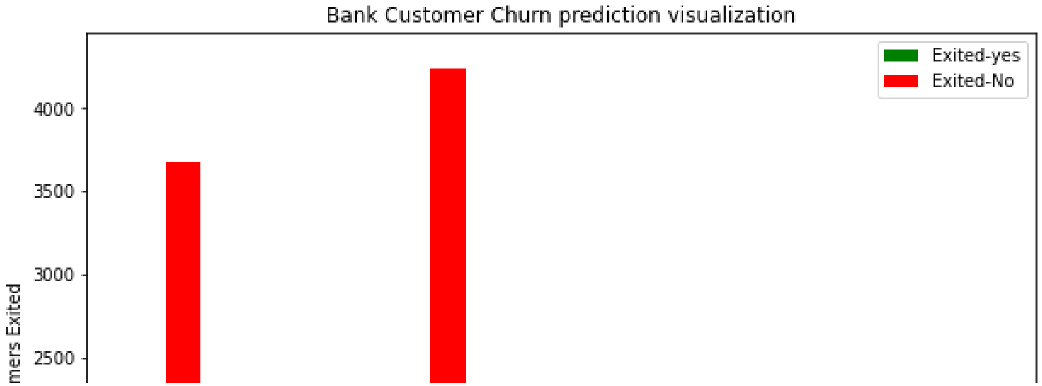

```
creditscore_exited_0=df[df.Exited==0].CreditScore
creditscore_exited_1=df[df.Exited==1].CreditScore

plt.figure(figsize=(10,8))
plt.xlabel('Credit Score')
plt.ylabel('Number of Customers Exited')
plt.title('Bank Customer Churn prediction visualization')
plt.hist([creditscore_exited_1,creditscore_exited_0], color=['green','red'], label=['Exited-yes','Exited-No'])
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f38696e6e90>
```



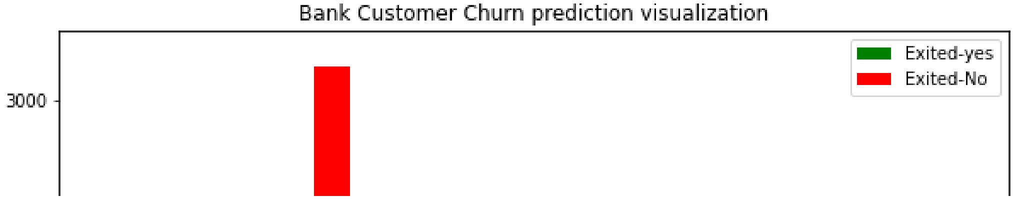

```
NumOfProducts_exited_0=df[df.Exited==0].NumOfProducts
NumOfProducts_exited_1=df[df.Exited==1].NumOfProducts

plt.figure(figsize=(10,8))
plt.xlabel('NumOfProducts')
plt.ylabel('Number of Customers Exited')
plt.title('Bank Customer Churn prediction visualization')
plt.hist([NumOfProducts_exited_1,NumOfProducts_exited_0], color=['green','red'], label=['Exited-yes','Exited-No'])
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f386960ba10>
```




```
Age_exited_0=df[df.Exited==0].Age
Age_exited_1=df[df.Exited==1].Age

plt.figure(figsize=(10,8))
plt.xlabel('Age')
plt.ylabel('Number of Customers Exited')
plt.title('Bank Customer Churn prediction visualization')
plt.hist([Age_exited_1,Age_exited_0], color=['green','red'], label=['Exited-yes','Exited-No'])
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f38695bce90>
```




```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   CreditScore        10000 non-null  int64  
 1   Gender             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   IsActiveMember     10000 non-null  int64  
 8   EstimatedSalary    10000 non-null  float64
 9   Exited             10000 non-null  int64  
 10  Geography_France   10000 non-null  uint8  
 11  Geography_Germany  10000 non-null  uint8  
 12  Geography_Spain    10000 non-null  uint8  
dtypes: float64(2), int64(8), uint8(3)
memory usage: 810.7 KB
```

```
# Scaling
cols_to_scale=['CreditScore','Tenure','Balance','NumOfProducts','EstimatedSalary','Age']

from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()

df[cols_to_scale]=scaler.fit_transform(df[cols_to_scale])


# Training
x=df.drop('Exited',axis=1)
y=df.Exited

from sklearn.model_selection import train_test_split
```

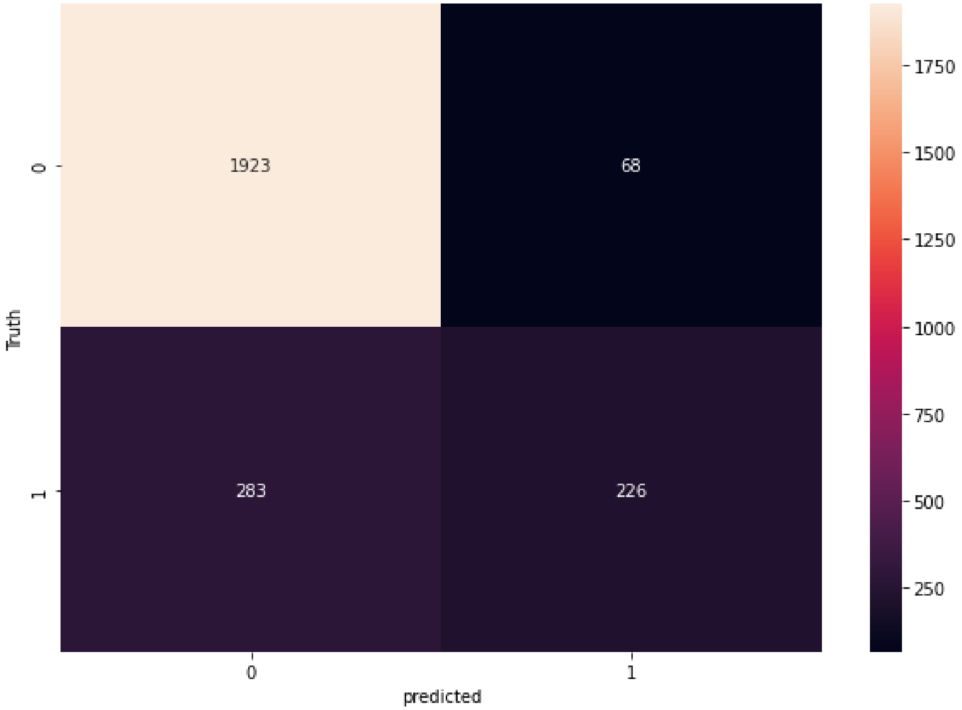
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25,random_state=15,stratify=y)
def ANN(xtrain,xtest,ytrain,ytest,loss,weight):
    model=keras.Sequential([
    keras.layers.Dense(20,input_shape=(12,),activation='relu'),
    keras.layers.Dense(1,activation='sigmoid')
    ])

    model.compile(optimizer='adam',
                  loss=loss,
                  metrics=['accuracy'])

    if weight==-1:
        model.fit(xtrain,ytrain,epochs=100)
    else:
        model.fit(xtrain,ytrain,epochs=100,class_weight=weight)
    print()
    print(model.evaluate(xtest,ytest))
    print()
    ypred= model.predict(xtest)
    ypred=np.round(ypred)
    print()
    print(classification_report(ytest,ypred))

    return ypred

ypred=ANN(xtrain,xtest,ytrain,ytest,'binary_crossentropy',-1)
```

```
235/235 [==============================] - 1s 2ms/step - loss: 0.3435 - accuracy: 0.8579
Epoch 79/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3434 - accuracy: 0.8583
Epoch 80/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3426 - accuracy: 0.8579
Epoch 81/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3430 - accuracy: 0.8583
Epoch 82/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3427 - accuracy: 0.8576
Epoch 83/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3426 - accuracy: 0.8597
Epoch 84/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3426 - accuracy: 0.8599
Epoch 85/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3424 - accuracy: 0.8589
Epoch 86/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3419 - accuracy: 0.8580
Epoch 87/100
```

```
Epoch 87/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3416 - accuracy: 0.8600
Epoch 88/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3430 - accuracy: 0.8599
Epoch 89/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3417 - accuracy: 0.8589
Epoch 90/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3424 - accuracy: 0.8585
Epoch 91/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3413 - accuracy: 0.8603
Epoch 92/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3417 - accuracy: 0.8595
Epoch 93/100
235/235 [==============================] - 1s 2ms/step - loss: 0.3418 - accuracy: 0.8588
Epoch 94/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3406 - accuracy: 0.8593
Epoch 95/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3411 - accuracy: 0.8589
Epoch 96/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3414 - accuracy: 0.8599
Epoch 97/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3409 - accuracy: 0.8603
Epoch 98/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3409 - accuracy: 0.8595
Epoch 99/100
235/235 [==============================] - 0s 2ms/step - loss: 0.3413 - accuracy: 0.8608
Epoch 100/100
235/235 [==============================] - 1s 2ms/step - loss: 0.3407 - accuracy: 0.8584

79/79 [==============================] - 0s 2ms/step - loss: 0.3347 - accuracy: 0.8596
[0.3347238004207611, 0.8596000075340271]

79/79 [==============================] - 0s 2ms/step

              precision    recall  f1-score   support

           0       0.87      0.97      0.92      1991

           1       0.77      0.44      0.56       509

    accuracy                           0.86      2500
   macro avg       0.82      0.70      0.74      2500
```

```
cm=tf.math.confusion_matrix(labels=ytest,predictions=ypred)
plt.figure(figsize=(10,7))
sns.heatmap(cm,annot=True,fmt='d')
plt.xlabel('predicted')
plt.ylabel('Truth')
```

```
Text(69.0, 0.5, 'Truth')
```




```
count_class_0, count_class_1 = df.Exited.value_counts()
```

```
df_class_0= df[df.Exited==0]
df_class_1= df[df.Exited==1]
```

```
df_class_0_under=df_class_0.sample(count_class_1)

df_test_under=pd.concat([df_class_0_under,df_class_1])

df_test_under.shape
```

```
(4074, 13)
```

```
# Training
x=df_test_under.drop('Exited',axis=1)
y=df_test_under.Exited
```

```
from sklearn.model_selection import train_test_split

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25,random_state=15,stratify=y)


ypred=ANN(xtrain,xtest,ytrain,ytest,'binary_crossentropy',-1)
```

```
96/96 [==============================] - 0s 2ms/step - loss: 0.4658 - accuracy: 0.7735
Epoch 79/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4654 - accuracy: 0.7787
Epoch 80/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4657 - accuracy: 0.7745
Epoch 81/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4652 - accuracy: 0.7787
Epoch 82/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4638 - accuracy: 0.7755
Epoch 83/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4646 - accuracy: 0.7758
Epoch 84/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4641 - accuracy: 0.7781
Epoch 85/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4637 - accuracy: 0.7728
Epoch 86/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4631 - accuracy: 0.7781
Epoch 87/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4632 - accuracy: 0.7758
Epoch 88/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4631 - accuracy: 0.7732
Epoch 89/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4624 - accuracy: 0.7771
Epoch 90/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4625 - accuracy: 0.7771
Epoch 91/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4618 - accuracy: 0.7764
Epoch 92/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4620 - accuracy: 0.7774

Epoch 93/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4618 - accuracy: 0.7768
Epoch 94/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4615 - accuracy: 0.7755
Epoch 95/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4615 - accuracy: 0.7755
Epoch 96/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4607 - accuracy: 0.7751
Epoch 97/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4608 - accuracy: 0.7768
Epoch 98/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4606 - accuracy: 0.7791
```

```
Epoch 99/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4596 - accuracy: 0.7758
Epoch 100/100
96/96 [==============================] - 0s 2ms/step - loss: 0.4612 - accuracy: 0.7745

32/32 [==============================] - 0s 2ms/step - loss: 0.4774 - accuracy: 0.7605
[0.47738170623779297, 0.7605495452880859]

32/32 [==============================] - 0s 2ms/step

              precision    recall  f1-score   support

           0       0.75      0.78      0.76       510
           1       0.77      0.74      0.76       509

    accuracy                           0.76      1019
   macro avg       0.76      0.76      0.76      1019
```

```
df_class_1_over= df_class_1.sample(count_class_0,replace=True)
```

```
df_test_over=pd.concat([df_class_0,df_class_1_over],axis=0)
```

```
df_test_over.shape
```

```
(15926, 13)
```

```
# Training
x=df_test_over.drop('Exited',axis=1)
y=df_test_over.Exited
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25,random_state=15,stratify=y)
```

```
ypred=ANN(xtrain,xtest,ytrain,ytest,'binary_crossentropy',-1)
```

```
374/374 [==============================] - 1s 2ms/step - loss: 0.4564 - accuracy: 0.7796
Epoch 79/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4561 - accuracy: 0.7806
Epoch 80/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4565 - accuracy: 0.7791
Epoch 81/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4558 - accuracy: 0.7786
Epoch 82/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4562 - accuracy: 0.7796
Epoch 83/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4549 - accuracy: 0.7811
Epoch 84/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4552 - accuracy: 0.7802
```

```
Epoch 85/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4548 - accuracy: 0.7823
Epoch 86/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4552 - accuracy: 0.7795
Epoch 87/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4551 - accuracy: 0.7808
Epoch 88/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4549 - accuracy: 0.7794
Epoch 89/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4547 - accuracy: 0.7808
Epoch 90/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4545 - accuracy: 0.7814
Epoch 91/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4539 - accuracy: 0.7844
Epoch 92/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4537 - accuracy: 0.7809
Epoch 93/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4537 - accuracy: 0.7833
Epoch 94/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4533 - accuracy: 0.7828
Epoch 95/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4531 - accuracy: 0.7827
Epoch 96/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4532 - accuracy: 0.7815
Epoch 97/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4530 - accuracy: 0.7828
Epoch 98/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4531 - accuracy: 0.7832
Epoch 99/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4525 - accuracy: 0.7846
Epoch 100/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4521 - accuracy: 0.7810

125/125 [==============================] - 0s 2ms/step - loss: 0.4561 - accuracy: 0.7788
[0.4560912549495697, 0.7787544131278992]


125/125 [==============================] - 0s 2ms/step

              precision    recall  f1-score   support

           0       0.80      0.75      0.77      1991
           1       0.76      0.81      0.79      1991

    accuracy                           0.78      3982
```

```
x=df.drop('Exited',axis=1)
```

```
y=df.Exited
```

```
from imblearn.over_sampling import SMOTE
```

```
smote= SMOTE(sampling_strategy='minority')
```

```
x_sm,y_sm=smote.fit_resample(x,y)
```

```
y_sm.value_counts()
```

```
1    7963
0    7963
Name: Exited, dtype: int64
```

```
xtrain,xtest,ytrain,ytest=train_test_split(x_sm,y_sm,test_size=0.25,random_state=15,stratify=y_sm)
ypred=ANN(xtrain,xtest,ytrain,ytest,'binary_crossentropy',-1)
```

```
Epoch 79/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4341 - accuracy: 0.7929
Epoch 80/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4335 - accuracy: 0.7950
Epoch 81/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4337 - accuracy: 0.7949
Epoch 82/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4337 - accuracy: 0.7930
Epoch 83/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4333 - accuracy: 0.7950
Epoch 84/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4334 - accuracy: 0.7943
Epoch 85/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4331 - accuracy: 0.7956
Epoch 86/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4334 - accuracy: 0.7930
Epoch 87/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4322 - accuracy: 0.7950
Epoch 88/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4326 - accuracy: 0.7959
Epoch 89/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4327 - accuracy: 0.7951
Epoch 90/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4321 - accuracy: 0.7921
Epoch 91/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4320 - accuracy: 0.7943
Epoch 92/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4322 - accuracy: 0.7935
Epoch 93/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4315 - accuracy: 0.7951
```

```
Epoch 94/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4316 - accuracy: 0.7964
Epoch 95/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4319 - accuracy: 0.7945
Epoch 96/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4310 - accuracy: 0.7966
Epoch 97/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4310 - accuracy: 0.7953
Epoch 98/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4309 - accuracy: 0.7939
Epoch 99/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4300 - accuracy: 0.7961
Epoch 100/100
374/374 [==============================] - 1s 2ms/step - loss: 0.4310 - accuracy: 0.7954

125/125 [==============================] - 0s 2ms/step - loss: 0.4332 - accuracy: 0.7943
[0.43315598368644714, 0.7943244576454163]

125/125 [==============================] - 0s 2ms/step

              precision    recall  f1-score   support

           0       0.79      0.79      0.79      1991
           1       0.79      0.79      0.79      1991

    accuracy                           0.79      3982
   macro avg       0.79      0.79      0.79      3982
weighted avg       0.79      0.79      0.79      3982
```

✓ 1m 16s completed at 11:25 AM