```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")


import tensorflow as tf
from tensorflow.keras import Sequential # It is used to build ANN
from tensorflow.keras.layers import Dense # It is used to add hidden layers
from sklearn.metrics import classification_report # evaluation



# read the dataset
df=pd.read_excel("Churn_Modelling.xlsx")
df.head()
```

```
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  float64
 1   CustomerId       10000 non-null  float64
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  float64
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  float64
 7   Tenure           10000 non-null  float64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  float64
 10  HasCrCard        10000 non-null  float64
 11  IsActiveMember   10000 non-null  float64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  float64
dtypes: float64(11), object(3)
memory usage: 1.1+ MB
```

```
x=df.iloc[:,3:-1]
x
```

```
x=df.iloc[:,3:-1].values
x
```

```
array([[619.0, 'France', 'Female', ..., 1.0, 1.0, 101348.88],
       [608.0, 'Spain', 'Female', ..., 0.0, 1.0, 112542.58],
       [502.0, 'France', 'Female', ..., 1.0, 0.0, 113931.57],
       ...,
       [709.0, 'France', 'Female', ..., 0.0, 1.0, 42085.58],
       [772.0, 'Germany', 'Male', ..., 1.0, 0.0, 92888.52],
       [792.0, 'France', 'Female', ..., 1.0, 0.0, 38190.78]], dtype=object)
```

```
df['Exited']=df['Exited'].astype(int)
```

```
df.head()
```

```
y=df['Exited'].values
```

```
y
```

```
array([1, 0, 1, ..., 1, 1, 0])
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
x[:,1]=le.fit_transform(x[:,1])
```

```
le1=LabelEncoder()
x[:,2]=le1.fit_transform(x[:,2])
```

```
x
```

```
array([[619.0, 0, 0, ..., 1.0, 1.0, 101348.88],
       [608.0, 2, 0, ..., 0.0, 1.0, 112542.58],
       [502.0, 0, 0, ..., 1.0, 0.0, 113931.57],
       ...,
       [709.0, 0, 0, ..., 0.0, 1.0, 42085.58],
       [772.0, 1, 1, ..., 1.0, 0.0, 92888.52],
       [792.0, 0, 0, ..., 1.0, 0.0, 38190.78]], dtype=object)
```

```
le.classes_
```

```
array(['France', 'Germany', 'Spain'], dtype=object)
```

```
le1.classes_
```

```
array(['Female', 'Male'], dtype=object)
```

```
# spliting data into train and test
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.30,random_state=12)
```

```
xtrain
```

```
array([[575.0, 1, 1, ..., 1.0, 1.0, 63452.18],
       [436.0, 1, 1, ..., 1.0, 1.0, 183540.22],
       [658.0, 0, 0, ..., 1.0, 1.0, 189607.71],
       ...,
       [527.0, 0, 0, ..., 1.0, 1.0, 44099.75],
```

```
       [524.0, 0, 1, ..., 0.0, 0.0, 82117.2],
       [729.0, 1, 1, ..., 1.0, 0.0, 39356.38]], dtype=object)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
xtrain=sc.fit_transform(xtrain)
```

```
xtest=sc.transform(xtest)
```

```
# building the ANN model
```

```
# step 1 initialize the model
ann=Sequential()

# step 2 add layers into model
ann.add(Dense(units=20,activation='relu',)) # create one hiddel layer
ann.add(Dense(units=1,activation="sigmoid"))

# step 3 established connection between the layers
ann.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy"])

# step 4 train the model
ann.fit(xtrain,ytrain,batch_size=30,epochs=200)

# step 5 make prediction
ypred=ann.predict(xtest)
```

```
Epoch 1/200
234/234 [==============================] - 2s 4ms/step - loss: 0.5665 - accuracy: 0.7003
Epoch 2/200
234/234 [==============================] - 1s 3ms/step - loss: 0.4404 - accuracy: 0.8097
Epoch 3/200
234/234 [==============================] - 1s 2ms/step - loss: 0.4207 - accuracy: 0.8239
Epoch 4/200
234/234 [==============================] - 1s 2ms/step - loss: 0.4091 - accuracy: 0.8334
Epoch 5/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3987 - accuracy: 0.8401
Epoch 6/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3890 - accuracy: 0.8451
Epoch 7/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3799 - accuracy: 0.8506
Epoch 8/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3716 - accuracy: 0.8523
Epoch 9/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3650 - accuracy: 0.8540
Epoch 10/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3597 - accuracy: 0.8561
Epoch 11/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3556 - accuracy: 0.8579
Epoch 12/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3526 - accuracy: 0.8593
Epoch 13/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3502 - accuracy: 0.8583
Epoch 14/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3484 - accuracy: 0.8599
Epoch 15/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3469 - accuracy: 0.8601
Epoch 16/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3456 - accuracy: 0.8596
Epoch 17/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3447 - accuracy: 0.8601
Epoch 18/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3438 - accuracy: 0.8611
Epoch 19/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3434 - accuracy: 0.8601
Epoch 20/200
234/234 [==============================] - 1s 4ms/step - loss: 0.3424 - accuracy: 0.8616
Epoch 21/200
234/234 [==============================] - 1s 4ms/step - loss: 0.3420 - accuracy: 0.8614
Epoch 22/200
234/234 [==============================] - 1s 4ms/step - loss: 0.3416 - accuracy: 0.8619
Epoch 23/200
234/234 [==============================] - 1s 3ms/step - loss: 0.3411 - accuracy: 0.8614
Epoch 24/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3408 - accuracy: 0.8611
Epoch 25/200
234/234 [==============================] - 1s 2ms/step - loss: 0.3402 - accuracy: 0.8619
Epoch 26/200
234/234 [==============================] - 1s 3ms/step - loss: 0.3400 - accuracy: 0.8621
Epoch 27/200
234/234 [==============================] - 1s 3ms/step - loss: 0.3399 - accuracy: 0.8619
Epoch 28/200
234/234 [==============================] - 1s 3ms/step - loss: 0.3394 - accuracy: 0.8617
```

```
Epoch 29/200
```

```
# step 6 set the threshold
ypred=np.where(ypred<0.5,0,1)
ypred
```

```
array([[0],
       [0],
       [1],
       ...,
       [0],
       [0],
       [0]])
```

```
print(classification_report(ypred,ytest))
```

```
              precision    recall  f1-score   support

           0       0.96      0.87      0.91      2591
           1       0.48      0.76      0.59       409

    accuracy                           0.85      3000
   macro avg       0.72      0.81      0.75      3000
weighted avg       0.89      0.85      0.87      3000
```

```
# df['Exited'].value_counts()
```