In [1]:

```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_csv('Churn_Modelling.csv')
df.shape
```

Out[2]:

```
(10000, 14)
```

In [3]:

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

In [4]:

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

Out[4]:

```
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 [5]:

```
df.dtypes
```

Out[5]:

```
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 [6]:

```
df['Geography'].unique()
```

Out[6]:

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

In [7]:

```
#one hot encoding
df = pd.get_dummies(data = df, columns=['Geography'])
df.dtypes
```

Out[7]:

```
CreditScore            int64
Gender                object
Age                    int64
Tenure                 int64
Balance              float64
NumOfProducts          int64
HasCrCard              int64
HasCrCard              int64
IsActiveMember         int64
EstimatedSalary      float64
Exited                 int64
Geography_France       uint8
Geography_Germany      uint8
Geography_Spain        uint8
dtype: object
```

In [8]:

```
df['Gender'].unique()
```

Out[8]:

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

In [9]:

```
df['Gender'].replace(['Male', 'Female'],[1, 0], inplace= True)
```

In [10]:

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

Out[10]:

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

In [11]:

```
#separate outcome or target col
X = df.drop(['Exited'], axis=1)
y = df['Exited']
```

In [12]:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

In [13]:

```
from sklearn.preprocessing import StandardScaler

# feature scaling

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [14]:

```
import tensorflow as tf
from tensorflow import keras
```

In [15]:

```
model = keras.Sequential([
    keras.layers.Dense(12, input_shape=(12,),activation='relu'),
    keras.layers.Dense(15, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])
```

In [16]:

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

In [17]:

```
model.fit(X_train, y_train, epochs=100)
```

```
Epoch 1/100
250/250 [==============================] - 2s 2ms/step - loss: 0.5976 - accuracy: 0.6628
Epoch 2/100
250/250 [==============================] - 0s 2ms/step - loss: 0.4396 - accuracy: 0.8119
Epoch 3/100
250/250 [==============================] - 0s 2ms/step - loss: 0.4111 - accuracy: 0.8267
Epoch 4/100
```

```
250/250 [==============================] - 1s 2ms/step - loss: 0.3898 - accuracy: 0.8382
Epoch 5/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3703 - accuracy: 0.8470
Epoch 6/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3564 - accuracy: 0.8537
Epoch 7/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3481 - accuracy: 0.8586
Epoch 8/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3434 - accuracy: 0.8587
Epoch 9/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3396 - accuracy: 0.8610
Epoch 10/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3372 - accuracy: 0.8625
Epoch 11/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3361 - accuracy: 0.8624
Epoch 12/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3341 - accuracy: 0.8637
Epoch 13/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3332 - accuracy: 0.8625
Epoch 14/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3322 - accuracy: 0.8641
Epoch 15/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3315 - accuracy: 0.8614
Epoch 16/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3305 - accuracy: 0.8631
Epoch 17/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3302 - accuracy: 0.8646
Epoch 18/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3295 - accuracy: 0.8634
Epoch 19/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3292 - accuracy: 0.8635
Epoch 20/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3282 - accuracy: 0.8648
Epoch 21/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3276 - accuracy: 0.8643
Epoch 22/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3279 - accuracy: 0.8634
Epoch 23/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3271 - accuracy: 0.8676
Epoch 24/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3275 - accuracy: 0.8649
Epoch 25/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3267 - accuracy: 0.8644
Epoch 26/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3261 - accuracy: 0.8656
Epoch 27/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3253 - accuracy: 0.8660
Epoch 28/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3257 - accuracy: 0.8673
Epoch 29/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3248 - accuracy: 0.8648
Epoch 30/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3253 - accuracy: 0.8658
Epoch 31/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3247 - accuracy: 0.8658
Epoch 32/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3242 - accuracy: 0.8668
Epoch 33/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3242 - accuracy: 0.8666
Epoch 34/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3240 - accuracy: 0.8652
Epoch 35/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3239 - accuracy: 0.8655
Epoch 36/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3239 - accuracy: 0.8679
Epoch 37/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3231 - accuracy: 0.8670
Epoch 38/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3230 - accuracy: 0.8671
Epoch 39/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3225 - accuracy: 0.8674
Epoch 40/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3223 - accuracy: 0.8668
Epoch 41/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3225 - accuracy: 0.8684
Epoch 42/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3217 - accuracy: 0.8656
Epoch 43/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3220 - accuracy: 0.8679
Epoch 44/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3221 - accuracy: 0.8680
Epoch 45/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3221 - accuracy: 0.8665
```

```
Epoch 46/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3219 - accuracy: 0.8679
Epoch 47/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3220 - accuracy: 0.8700
Epoch 48/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3211 - accuracy: 0.8692
Epoch 49/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3209 - accuracy: 0.8690
Epoch 50/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3211 - accuracy: 0.8695
Epoch 51/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3205 - accuracy: 0.8694
Epoch 52/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3212 - accuracy: 0.8658
Epoch 53/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3210 - accuracy: 0.8684
Epoch 54/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3205 - accuracy: 0.8695
Epoch 55/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3206 - accuracy: 0.8687
Epoch 56/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3206 - accuracy: 0.8674
Epoch 57/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3198 - accuracy: 0.8694
Epoch 58/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3199 - accuracy: 0.8686
Epoch 59/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3200 - accuracy: 0.8677
Epoch 60/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3193 - accuracy: 0.8695
Epoch 61/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3199 - accuracy: 0.8673
Epoch 62/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3193 - accuracy: 0.8696
Epoch 63/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3194 - accuracy: 0.8685
Epoch 64/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3194 - accuracy: 0.8683
Epoch 65/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3193 - accuracy: 0.8675
Epoch 66/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3191 - accuracy: 0.8696
Epoch 67/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3193 - accuracy: 0.8685
Epoch 68/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3188 - accuracy: 0.8676
Epoch 69/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3186 - accuracy: 0.8696
Epoch 70/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3190 - accuracy: 0.8681
Epoch 71/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3182 - accuracy: 0.8698
Epoch 72/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3187 - accuracy: 0.8690
Epoch 73/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3187 - accuracy: 0.8683
Epoch 74/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3184 - accuracy: 0.8698
Epoch 75/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3183 - accuracy: 0.8686
Epoch 76/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3184 - accuracy: 0.8687
Epoch 77/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3181 - accuracy: 0.8701
Epoch 78/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3185 - accuracy: 0.8695
Epoch 79/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3179 - accuracy: 0.8684
Epoch 80/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3183 - accuracy: 0.8711
Epoch 81/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3180 - accuracy: 0.8705
Epoch 82/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3179 - accuracy: 0.8702
Epoch 83/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3177 - accuracy: 0.8701
Epoch 84/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3175 - accuracy: 0.8692
Epoch 85/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3177 - accuracy: 0.8710
Epoch 86/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3172 - accuracy: 0.8696
Epoch 87/100
```

```
250/250 [==============================] - 0s 2ms/step - loss: 0.3173 - accuracy: 0.8699
Epoch 88/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3175 - accuracy: 0.8687
Epoch 89/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3169 - accuracy: 0.8701
Epoch 90/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3173 - accuracy: 0.8699
Epoch 91/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3166 - accuracy: 0.8706
Epoch 92/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3166 - accuracy: 0.8705
Epoch 93/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3169 - accuracy: 0.8696
Epoch 94/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3165 - accuracy: 0.8704
Epoch 95/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3159 - accuracy: 0.8684
Epoch 96/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3162 - accuracy: 0.8704
Epoch 97/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3154 - accuracy: 0.8706
Epoch 98/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3168 - accuracy: 0.8704
Epoch 99/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3161 - accuracy: 0.8691
Epoch 100/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3169 - accuracy: 0.8709
```

Out[17]:

```
<keras.callbacks.History at 0x7fab6ed52d50>
```

In [18]:

```
model.evaluate(X_test, y_test)
```

```
63/63 [==============================] - 1s 4ms/step - loss: 0.3311 - accuracy: 0.8625
```

Out[18]:

```
[0.3311230540275574, 0.862500011920929]
```

In [19]:

```
yp = model.predict(X_test)
```

```
63/63 [==============================] - 0s 1ms/step
```

In [23]:

```
y_pred = []
for element in yp:
    if element > 0.5:
        y_pred.append(1)
    else:
        y_pred.append(0)
```

In [24]:

```
from sklearn.metrics import confusion_matrix , classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.95      0.92      1595
           1       0.72      0.52      0.61       405

    accuracy                           0.86      2000
   macro avg       0.80      0.74      0.76      2000
weighted avg       0.85      0.86      0.85      2000
```

In [25]:

```
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_pred)
```

In [26]:

```
cm
```

Out[26]:

```
<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[1514,   81],
       [ 194,  211]], dtype=int32)>
       [ 194,   211]], dtype=int32)>
```

In [27]:

```
tf.math.confusion_matrix(labels=y_test,predictions=y_pred)
```

Out[27]:

```
<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[1514,   81],
       [ 194,  211]], dtype=int32)>
```