Importing Libraries

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

Loading Data

+ Code + Text

```
df = pd.read_excel("/content/Churn_Modelling.xlsx")
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1.0	15634602.0	Hargrave	619.0	France	Female	42.0	2.0	0.00	1.0	1.0	1.0	101348.88	1.0
1	2.0	15647311.0	Hill	608.0	Spain	Female	41.0	1.0	83807.86	1.0	0.0	1.0	112542.58	0.0
2	3.0	15619304.0	Onio	502.0	France	Female	42.0	8.0	159660.80	3.0	1.0	0.0	113931.57	1.0
3	4.0	15701354.0	Boni	699.0	France	Female	39.0	1.0	0.00	2.0	0.0	0.0	93826.63	0.0
4	5.0	15737888.0	Mitchell	850.0	Spain	Female	43.0	2.0	125510.82	1.0	1.0	1.0	79084.10	0.0

```
df.Exited = df.Exited.astype(int)
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1.0	15634602.0	Hargrave	619.0	France	Female	42.0	2.0	0.00	1.0	1.0	1.0	101348.88	1
1	2.0	15647311.0	Hill	608.0	Spain	Female	41.0	1.0	83807.86	1.0	0.0	1.0	112542.58	0
2	3.0	15619304.0	Onio	502.0	France	Female	42.0	8.0	159660.80	3.0	1.0	0.0	113931.57	1
3	4.0	15701354.0	Boni	699.0	France	Female	39.0	1.0	0.00	2.0	0.0	0.0	93826.63	0
4	5.0	15737888.0	Mitchell	850.0	Spain	Female	43.0	2.0	125510.82	1.0	1.0	1.0	79084.10	0

Segregating X & Y

```
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)
```

```
Y = df.Exited.values
Y
```

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

Encoding

```
from sklearn.preprocessing import LabelEncoder
LE1 = LabelEncoder()
X[:,1] = LE1.fit_transform(X[:,1])
```

```
LE2 = LabelEncoder()
X[:,2] = LE2.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)
```

Standadization

```
from sklearn.preprocessing import StandardScaler
SS = StandardScaler()
X = SS.fit_transform(X)
X
```

```
array([[-0.32622142, -0.90188624, -1.09598752, ...,  0.64609167,
         0.97024255,  0.02188649],
       [-0.44003595,  1.51506738, -1.09598752, ..., -1.54776799,
         0.97024255,  0.21653375],
       [-1.53679418, -0.90188624, -1.09598752, ...,  0.64609167,
        -1.03067011,  0.2406869 ],
       ...,
       [ 0.60498839, -0.90188624, -1.09598752, ..., -1.54776799,
         0.97024255, -1.00864308],
       [ 1.25683526,  0.30659057,  0.91241915, ...,  0.64609167,
        -1.03067011, -0.12523071],
       [ 1.46377078, -0.90188624, -1.09598752, ...,  0.64609167,
        -1.03067011, -1.07636976]])
```

Splitting

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3,random_state=1)
```

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

ANN Modelling

```
# step1 :- initialize the Model
ann = Sequential()

# step2 :- Add Layers into model
ann.add( Dense(units = 6, activation = "relu") )
ann.add( Dense(units = 1, activation = "sigmoid") )

# step3 :- Establihing connection
ann.compile(optimizer='adam', loss = 'binary_crossentropy', metrics=["accuracy"])
```

```
# step4 :- Fit the model
ann.fit(X_train, Y_train, batch_size = 30, epochs = 100)

# step5 :- Predict the model
Y_pred = ann.predict(X_test)
```

```
Epoch 1/100
234/234 [==============================] - 3s 4ms/step - loss: 0.6279 - accuracy: 0.6667
Epoch 2/100
234/234 [==============================] - 1s 3ms/step - loss: 0.4739 - accuracy: 0.7931
Epoch 3/100
234/234 [==============================] - 1s 4ms/step - loss: 0.4471 - accuracy: 0.8003
Epoch 4/100
234/234 [==============================] - 1s 4ms/step - loss: 0.4359 - accuracy: 0.8037
Epoch 5/100
234/234 [==============================] - 1s 4ms/step - loss: 0.4290 - accuracy: 0.8096
Epoch 6/100
234/234 [==============================] - 1s 4ms/step - loss: 0.4239 - accuracy: 0.8100
Epoch 7/100
234/234 [==============================] - 1s 4ms/step - loss: 0.4197 - accuracy: 0.8146
Epoch 8/100
234/234 [==============================] - 1s 4ms/step - loss: 0.4157 - accuracy: 0.8160
Epoch 9/100
234/234 [==============================] - 1s 3ms/step - loss: 0.4122 - accuracy: 0.8187
Epoch 10/100
234/234 [==============================] - 1s 4ms/step - loss: 0.4091 - accuracy: 0.8211
Epoch 11/100
234/234 [==============================] - 1s 3ms/step - loss: 0.4064 - accuracy: 0.8223
Epoch 12/100
234/234 [==============================] - 1s 6ms/step - loss: 0.4036 - accuracy: 0.8227
Epoch 13/100
234/234 [==============================] - 2s 7ms/step - loss: 0.4012 - accuracy: 0.8229
Epoch 14/100
234/234 [==============================] - 1s 5ms/step - loss: 0.3988 - accuracy: 0.8243
Epoch 15/100
234/234 [==============================] - 1s 6ms/step - loss: 0.3969 - accuracy: 0.8239
Epoch 16/100
234/234 [==============================] - 1s 4ms/step - loss: 0.3949 - accuracy: 0.8246
Epoch 17/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3929 - accuracy: 0.8263
Epoch 18/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3911 - accuracy: 0.8277
Epoch 19/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3894 - accuracy: 0.8273
Epoch 20/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3879 - accuracy: 0.8276
Epoch 21/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3865 - accuracy: 0.8277
Epoch 22/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3850 - accuracy: 0.8280
Epoch 23/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3837 - accuracy: 0.8277
Epoch 24/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3826 - accuracy: 0.8284
Epoch 25/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3816 - accuracy: 0.8291
Epoch 26/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3805 - accuracy: 0.8326
Epoch 27/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3796 - accuracy: 0.8411
Epoch 28/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3787 - accuracy: 0.8410
Epoch 29/100
234/234 [==============================] - 0s 2ms/step - loss: 0.3779 - accuracy: 0.8429
```

```
# step6 :- Set Threshold
Y_pred = np.where(Y_pred>0.5,1,0)
```

```
Y_pred
```

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

```
# evaluation
from sklearn.metrics import classification_report
print( classification_report(Y_test,Y_pred) )
```

```
              precision    recall  f1-score   support

           0       0.86      0.96      0.91      2373
           1       0.74      0.42      0.54       627

    accuracy                           0.85      3000
   macro avg       0.80      0.69      0.72      3000
weighted avg       0.84      0.85      0.83      3000
```