In [1]:

```
import tensorflow as tf
```

In [2]:

```
print(tf.__version__)
```

```
2.10.0
```

In [3]:

```
# import some basic libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:

```
df = pd.read_csv("Churn_Modelling.csv")
```

In [5]:

```
df.head()
```

Out[5]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActi
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	

In [6]:

```
# divide the dataeset independent and dependent features
x = df.drop(["RowNumber","CustomerId","Surname","Exited"], axis=1)
y = df["Exited"]
```

In [7]:

```
x.head()
```

Out[7]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	France	Female	42	2	0.00	1	1	1	101348.88
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	699	France	Female	39	1	0.00	2	0	0	93826.63
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10

In [8]:

```
y
```

Out[8]:

```
0       1
1       0
2       1
3       0
4       0
       ..
9995    0
9996    0
9997    1
9998    1
9999    0
Name: Exited, Length: 10000, dtype: int64
```

In [9]:

```
# Feature engineering
geo = pd.get_dummies(x["Geography"], drop_first= True)
gender = pd.get_dummies(x["Gender"], drop_first = True)
```

In [10]:

```
# Concatenate these variable with features
x = x.drop(["Geography", "Gender"], axis=1)
```

In [11]:

```
x
```

Out[11]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	42	2	0.00	1	1	1	101348.88
1	608	41	1	83807.86	1	0	1	112542.58
2	502	42	8	159660.80	3	1	0	113931.57
3	699	39	1	0.00	2	0	0	93826.63
4	850	43	2	125510.82	1	1	1	79084.10
...	...	...	...	...	...	...	...	...
9995	771	39	5	0.00	2	1	0	96270.64
9996	516	35	10	57369.61	1	1	1	101699.77
9997	709	36	7	0.00	1	0	1	42085.58
9998	772	42	3	75075.31	2	1	0	92888.52
9999	792	28	4	130142.79	1	1	0	38190.78

10000 rows × 8 columns

In [14]:

```
x = pd.concat([x, geo , gender], axis=1)
```

In [15]:

```
x
```

Out[15]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Germany	Spain	Male
0	619	42	2	0.00	1	1	1	101348.88	0	0	0
1	608	41	1	83807.86	1	0	1	112542.58	0	1	0
2	502	42	8	159660.80	3	1	0	113931.57	0	0	0
3	699	39	1	0.00	2	0	0	93826.63	0	0	0
4	850	43	2	125510.82	1	1	1	79084.10	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
9995	771	39	5	0.00	2	1	0	96270.64	0	0	1
9996	516	35	10	57369.61	1	1	1	101699.77	0	0	1
9997	709	36	7	0.00	1	0	1	42085.58	0	0	0
9998	772	42	3	75075.31	2	1	0	92888.52	1	0	1
9999	792	28	4	130142.79	1	1	0	38190.78	0	0	0

10000 rows × 11 columns

In [16]:

```
# splitting into train and test
from sklearn.model_selection import train_test_split
```

In [18]:

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

In [19]:

```
# feature scaling
from sklearn.preprocessing import StandardScaler
```

In [20]:

```
sc = StandardScaler()
```

In [21]:

```
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [22]:

```
x_train
```

Out[22]:

```
array([[-0.09792126, -0.55759842, -1.03635146, ..., -0.56987189,
        -0.5731713 ,  0.92295821],
       [-1.12612023,  0.01725942,  0.69700901, ..., -0.56987189,
        -0.5731713 ,  0.92295821],
       [-0.62230274,  3.5622161 ,  0.00366482, ..., -0.56987189,
        -0.5731713 , -1.08347268],
       ...,
       [ 0.89943174, -0.36597914,  0.00366482, ..., -0.56987189,
        -0.5731713 ,  0.92295821],
       [-0.62230274, -0.07855022,  1.39035319, ..., -0.56987189,
         1.74467913, -1.08347268],
       [-0.28299708,  0.87954618, -1.38302356, ...,  1.75478035,
        -0.5731713 , -1.08347268]])
```

In [23]:

```
x_test
```

Out[23]:

```
array([[-0.55032881, -0.36597914,  1.0436811 , ...,  1.75478035,
        -0.5731713 , -1.08347268],
       [-1.31119605,  0.11306906, -1.03635146, ..., -0.56987189,
        -0.5731713 , -1.08347268],
       [ 0.57040807,  0.30468834,  1.0436811 , ..., -0.56987189,
         1.74467913, -1.08347268],
       ...,
       [ 0.35448628,  0.11306906, -1.03635146, ..., -0.56987189,
        -0.5731713 ,  0.92295821],
       [ 0.42646021,  2.89154862,  1.73702529, ..., -0.56987189,
        -0.5731713 ,  0.92295821],
       [ 0.82745781,  0.97535582, -0.34300727, ...,  1.75478035,
        -0.5731713 , -1.08347268]])
```

In [24]:

```
x_train.shape
```

Out[24]:

```
(7000, 11)
```

In [25]:

```
x_test.shape
```

Out[25]:

```
(3000, 11)
```

In [60]:

```
# Creating ANN
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense  # for creating hidden layer , input layers
from tensorflow.keras.layers import LeakyReLU , PReLU , ReLU , ELU   # Activation functions
from tensorflow.keras.layers import Dropout  #reduce the overfitting
```

In [61]:

```
# lets initialize ANN
model = Sequential()
```

In [62]:

```
# Adding input layer
model.add(Dense(units=11 , activation= "relu"))
```

In [64]:

```
# first hidden layer
model.add(Dense(units=7 , activation="relu"))
model.add(Dropout(0.3))
```

In [65]:

```
# Second hidden layers
model.add(Dense(units=5 , activation="relu"))
```

In [66]:

```
# Adding the ouput layer
model.add(Dense(1, activation = "sigmoid"))
```

In [67]:

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

In [68]:

```
# Early Stopping
early_stopping = tf.keras.callbacks.EarlyStopping(
monitor = "val_loss",
    min_delta = 0.0001,
    patience = 20,
    verbose = 1,
    mode = "auto",
    baseline = None,
    restore_best_weights = False,
)
```

In [69]:

```
model_history = model.fit(x_train , y_train , validation_split=0.33 , batch_size = 10 , epochs = 100 , callbacks=ea
```

```
469/469 [==============================] - 2s 3ms/step - loss: 0.3525 - accuracy: 0.8543 - val_los
s: 0.3663 - val_accuracy: 0.8546
Epoch 35/100
469/469 [==============================] - 2s 4ms/step - loss: 0.3485 - accuracy: 0.8554 - val_los
s: 0.3665 - val_accuracy: 0.8524
Epoch 36/100
469/469 [==============================] - 2s 4ms/step - loss: 0.3508 - accuracy: 0.8560 - val_los
s: 0.3661 - val_accuracy: 0.8559
Epoch 37/100
469/469 [==============================] - 2s 4ms/step - loss: 0.3475 - accuracy: 0.8546 - val_los
s: 0.3653 - val_accuracy: 0.8537
Epoch 38/100
469/469 [==============================] - 2s 4ms/step - loss: 0.3486 - accuracy: 0.8573 - val_los
s: 0.3635 - val_accuracy: 0.8537
Epoch 39/100
469/469 [==============================] - 2s 4ms/step - loss: 0.3428 - accuracy: 0.8580 - val_los
s: 0.3621 - val_accuracy: 0.8542
Epoch 40/100
469/469 [==============================] - 2s 4ms/step - loss: 0.3503 - accuracy: 0.8580 - val_los
s: 0.3676 - val_accuracy: 0.8516
```

In [70]:

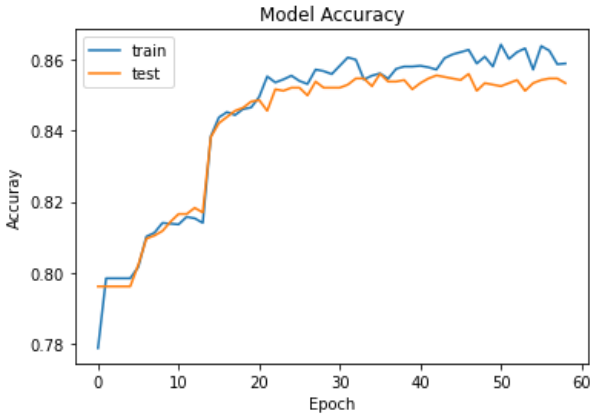
```
model_history.history.keys()
```

Out[70]:

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

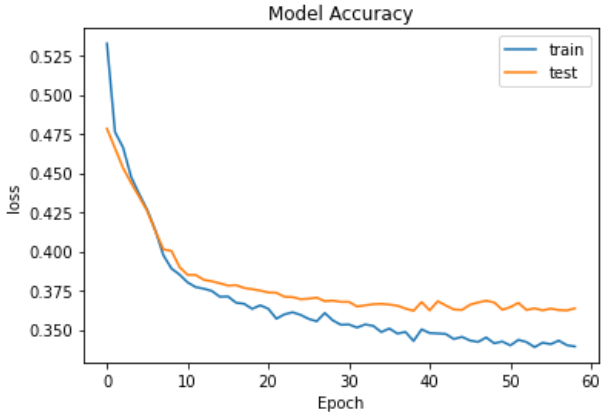
In [71]:

```
plt.plot(model_history.history["accuracy"])
plt.plot(model_history.history["val_accuracy"])
plt.title("Model Accuracy")
plt.ylabel("Accuray")
plt.xlabel("Epoch")
plt.legend(["train" , "test"])
plt.show()
```



In [72]:

```
plt.plot(model_history.history["loss"])
plt.plot(model_history.history["val_loss"])
plt.title("Model Accuracy")
plt.ylabel("loss")
plt.xlabel("Epoch")
plt.legend(["train" , "test"])
plt.show()
```



In [73]:

```
# Making the prediction
y_pred = model.predict(x_test)
y_pred = (y_pred >= 0.5)
```

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

In [74]:

```
# confusion metrix
from sklearn.metrics import confusion_matrix
```

In [75]:

```
cm = confusion_matrix(y_test , y_pred)
```

In [76]:

```
cm
```

Out[76]:

```
array([[2224,  155],
       [ 289,  332]], dtype=int64)
```

In [77]:

```
# calculate accuracy
from sklearn.metrics import accuracy_score
```

In [78]:

```
score = accuracy_score(y_test , y_pred)
```

In [79]:

```
score
```

Out[79]:

```
0.852
```

In [80]:

```
# getting the weights
model.get_weights()
```

```
          0.34677738, -0.09421454],
        [ 0.17794782,  0.22895505,  0.4130918 ,  0.38881686,  0.56160456,
          0.2705269 ,  0.18080662]], dtype=float32),
 array([ 0.1578264 , -0.07098044,  0.00422569,  0.20356254,  0.14587498,
         0.02242401, -0.04134814], dtype=float32),
 array([[ 0.46647123,  0.2634855 ,  0.44560853, -0.02916288,  0.83332103],
        [ 0.4094206 ,  0.24747613,  0.52004576, -0.12834783,  0.61817205],
        [ 0.9919129 ,  0.16972981,  0.42290705, -0.5911728 , -0.04913793],
        [ 0.7435097 , -0.07134432,  0.00527197, -0.10111122, -1.0366809 ],
        [ 0.6453634 ,  0.2536646 ,  0.5050317 , -0.13370897,  0.23280394],
        [ 0.43836018,  0.19008416,  0.5393441 ,  0.01429297,  0.63922524],
        [-0.01574454,  0.91182363,  0.97740364,  1.2852892 , -0.05037176]],
       dtype=float32),
 array([ 0.00294377, -0.6830707 , -1.2031955 , -0.1120965 ,  0.0181223 ],
       dtype=float32),
 array([[-0.7303195 ],
        [ 0.67491645],
        [ 0.5551723 ],
        [ 1.3897061 ],
        [-0.70757294]], dtype=float32)]
```

In [ ]: