```
import tensorflow as tf
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
# Feature Scaling
from sklearn.preprocessing import StandardScaler
# Deep learning Lib
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout,LeakyReLU,PReLU,ELU
```

```
# Importing the dataset
dataset = pd.read_csv('/content/Churn_Modelling.csv')
X = dataset.iloc[:, 3:13]
y = dataset.iloc[:, 13]


#Create dummy variables
geography=pd.get_dummies(X["Geography"],drop_first=True)
gender=pd.get_dummies(X['Gender'],drop_first=True)


## Concatenate the Data Frames

X=pd.concat([X,geography,gender],axis=1)

## Drop Unnecessary columns
X=X.drop(['Geography','Gender'],axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

```
# Feature Scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
#  Now let's make the ANN!


# Initialising the ANN
classifier = Sequential()


# Adding the input layer and the first hidden layer
classifier.add(Dense(units=11,activation='relu'))


# Adding the input layer and the first hidden layer
classifier.add(Dense(units=6,activation='relu'))
```

```
# Adding the input layer and the first hidden layer
classifier.add(Dense(units=1,activation='relu'))
```

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

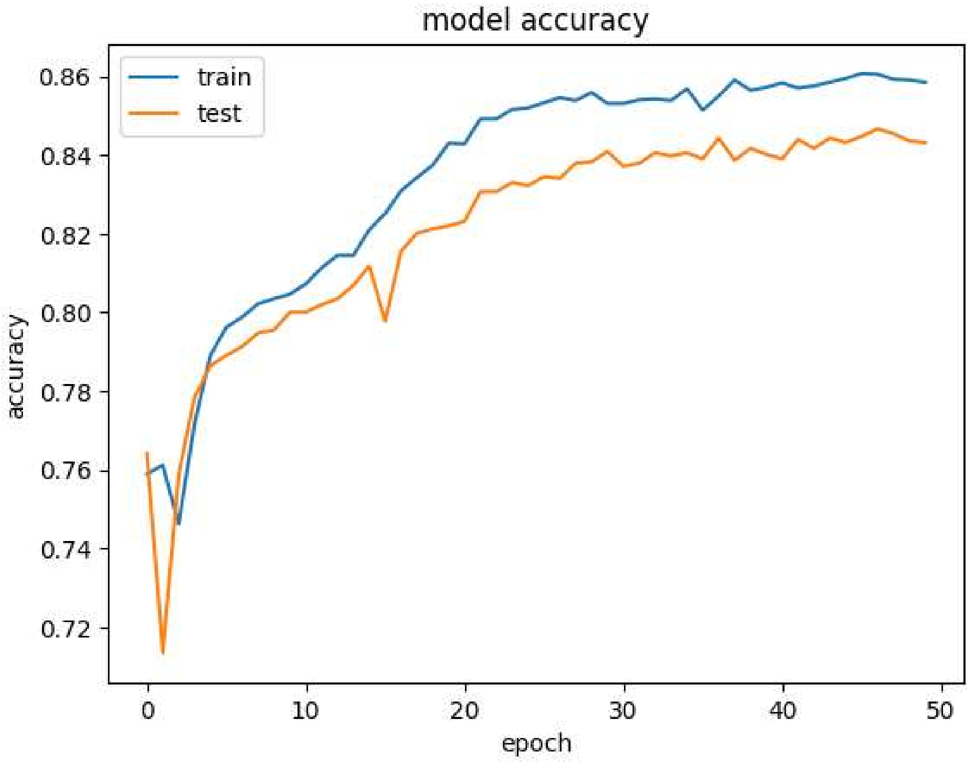
```
model_history=classifier.fit(X_train,y_train,validation_split=0.33,batch_size=10,epochs=50)
```

```
536/536 [==============================] - 2s 3ms/step - loss: 0.3748 - accuracy: 0.8492 - val_loss: 0.4079 - va
Epoch 23/50
536/536 [==============================] - 2s 3ms/step - loss: 0.3738 - accuracy: 0.8492 - val_loss: 0.4031 - va
Epoch 24/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3677 - accuracy: 0.8517 - val_loss: 0.4063 - va
Epoch 25/50
536/536 [==============================] - 2s 3ms/step - loss: 0.3621 - accuracy: 0.8520 - val_loss: 0.3949 - va
Epoch 26/50
536/536 [==============================] - 2s 3ms/step - loss: 0.3563 - accuracy: 0.8533 - val_loss: 0.3984 - va
Epoch 27/50
536/536 [==============================] - 2s 4ms/step - loss: 0.3570 - accuracy: 0.8546 - val_loss: 0.4040 - va
Epoch 28/50
536/536 [==============================] - 2s 3ms/step - loss: 0.3568 - accuracy: 0.8539 - val_loss: 0.4083 - va
Epoch 29/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3558 - accuracy: 0.8559 - val_loss: 0.3975 - va
Epoch 30/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3553 - accuracy: 0.8531 - val_loss: 0.3992 - va
Epoch 31/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3553 - accuracy: 0.8531 - val_loss: 0.4017 - va
Epoch 32/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3533 - accuracy: 0.8541 - val_loss: 0.4289 - va
Epoch 33/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3571 - accuracy: 0.8543 - val_loss: 0.4132 - va
Epoch 34/50
536/536 [==============================] - 2s 3ms/step - loss: 0.3530 - accuracy: 0.8539 - val_loss: 0.4000 - va
Epoch 35/50
536/536 [==============================] - 2s 4ms/step - loss: 0.3462 - accuracy: 0.8569 - val_loss: 0.4052 - va
Epoch 36/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3619 - accuracy: 0.8515 - val_loss: 0.4011 - va
Epoch 37/50
536/536 [==============================] - 2s 3ms/step - loss: 0.3485 - accuracy: 0.8550 - val_loss: 0.4062 - va
Epoch 38/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3495 - accuracy: 0.8591 - val_loss: 0.4094 - va
Epoch 39/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3527 - accuracy: 0.8565 - val_loss: 0.4257 - va
Epoch 40/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3493 - accuracy: 0.8572 - val_loss: 0.3915 - va
Epoch 41/50
536/536 [==============================] - 2s 3ms/step - loss: 0.3442 - accuracy: 0.8584 - val_loss: 0.3924 - va
Epoch 42/50
536/536 [==============================] - 2s 3ms/step - loss: 0.3440 - accuracy: 0.8571 - val_loss: 0.3860 - va
Epoch 43/50
536/536 [==============================] - 2s 4ms/step - loss: 0.3395 - accuracy: 0.8576 - val_loss: 0.4045 - va
Epoch 44/50
536/536 [==============================] - 2s 3ms/step - loss: 0.3412 - accuracy: 0.8586 - val_loss: 0.4186 - va
Epoch 45/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3417 - accuracy: 0.8595 - val_loss: 0.3878 - va
Epoch 46/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3420 - accuracy: 0.8608 - val_loss: 0.4047 - va
Epoch 47/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3479 - accuracy: 0.8606 - val_loss: 0.3950 - va
Epoch 48/50
536/536 [==============================] - 2s 3ms/step - loss: 0.3456 - accuracy: 0.8593 - val_loss: 0.4073 - va
Epoch 49/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3463 - accuracy: 0.8591 - val_loss: 0.3947 - va
Epoch 50/50
536/536 [==============================] - 1s 2ms/step - loss: 0.3468 - accuracy: 0.8586 - val_loss: 0.4138 - va
```

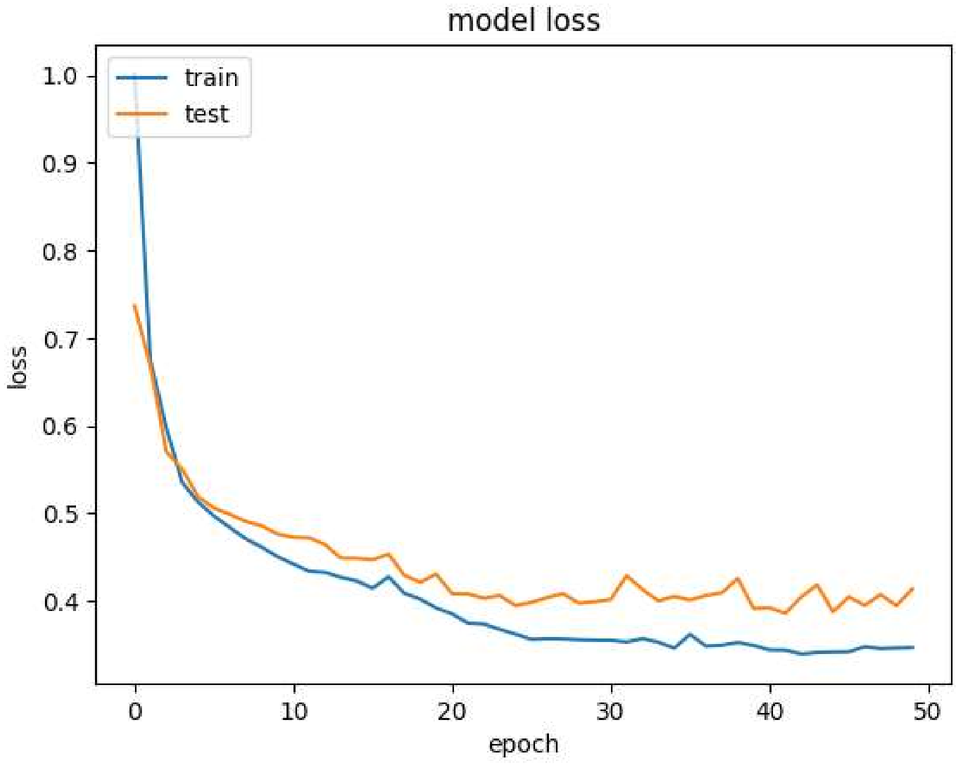
```
# list all data in history

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

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

```
# summarize history for accuracy
plt.plot(model_history.history['accuracy'])
plt.plot(model_history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



```
# summarize history for loss
plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



```
# Making the predictions and evaluating the model

# Predicting the Test set results
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
```

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

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
    array([[1504,   91],
           [ 207,  198]])
```

```
# Calculate the Accuracy
from sklearn.metrics import accuracy_score
score=accuracy_score(y_pred,y_test)


score
```

```
    0.851
```

+ Code + Text