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

```
dataset = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Bank Customer Churn Prediction.csv")
dataset.head()
```

	customer_id	credit_score	country	gender	age	tenure	balance	products_n
0	15634602	619	France	Female	42	2	0.00	
1	15647311	608	Spain	Female	41	1	83807.86	
2	15619304	502	France	Female	42	8	159660.80	
3	15701354	699	France	Female	39	1	0.00	
4	15737888	850	Spain	Female	43	2	125510.82	

```
X = dataset.drop(['churn'],axis = 1).values
y = dataset['churn'].values
```

```
X.shape
```

```
(10000, 11)
```

```
y.shape
```

```
(10000,)
```

```
X
```

```
array([[15634602, 619, 'France', ..., 1, 1, 101348.88],
       [15647311, 608, 'Spain', ..., 0, 1, 112542.58],
       [15619304, 502, 'France', ..., 1, 0, 113931.57],
       ...,
       [15584532, 709, 'France', ..., 0, 1, 42085.58],
       [15682355, 772, 'Germany', ..., 1, 0, 92888.52],
       [15628319, 792, 'France', ..., 1, 0, 38190.78]], dtype=object)
```

```
X[:,2]
```

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

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoded_x1 = LabelEncoder()
X[:,2] = labelencoded_x1.fit_transform(X[:,2])
X[:,2]
```

```
array([0, 2, 0, ..., 0, 1, 0], dtype=object)
```

```
X[:,3]
```

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

```
labelencoded_x2 = LabelEncoder()
X[:,3] = labelencoded_x2.fit_transform(X[:,3])
X[:,3]
```

```
array([0, 0, 0, ..., 0, 1, 0], dtype=object)
```

```
# split data into train and test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,_y_test = train_test_split(X,y,test_size = 0.2)
```

```
# feature scaling

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# import required libraries for model creation
```

```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

```
# initialize neural network

classifier = Sequential()
```

```
classifier.add(Dense(units = 6,kernel_initializer='uniform',activation = 'relu',input_dim = 11))
```

```
classifier.add(Dense(units = 6,kernel_initializer='uniform',activation = 'relu'))
```

```
classifier.add(Dense(units = 1,kernel_initializer='uniform',activation = 'sigmoid'))
```

```
# compiling neural network

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

```
# model fitting
classifier.fit(X_train,y_train,batch_size = 10, epochs = 100)
```

```
    Epoch 38/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4030 - accuracy: 0.8363
    Epoch 39/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4032 - accuracy: 0.8356
    Epoch 40/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4028 - accuracy: 0.8356
    Epoch 41/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4029 - accuracy: 0.8357
    Epoch 42/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4024 - accuracy: 0.8356
    Epoch 43/100
    800/800 [==============================] - 1s 2ms/step - loss: 0.4024 - accuracy: 0.8360
    Epoch 44/100
    800/800 [==============================] - 1s 2ms/step - loss: 0.4025 - accuracy: 0.8354
    Epoch 45/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4026 - accuracy: 0.8349
    Epoch 46/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4028 - accuracy: 0.8354
    Epoch 47/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4022 - accuracy: 0.8355
    Epoch 48/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4020 - accuracy: 0.8360
    Epoch 49/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4026 - accuracy: 0.8364
    Epoch 50/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4022 - accuracy: 0.8351
    Epoch 51/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4020 - accuracy: 0.8372
    Epoch 52/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4021 - accuracy: 0.8370
    Epoch 53/100
    800/800 [==============================] - 2s 2ms/step - loss: 0.4019 - accuracy: 0.8349
    Epoch 54/100
    800/800 [==============================] - 3s 4ms/step - loss: 0.4018 - accuracy: 0.8355
    Epoch 55/100
    800/800 [==============================] - 2s 2ms/step - loss: 0.4019 - accuracy: 0.8361
    Epoch 56/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4008 - accuracy: 0.8371
    Epoch 57/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4018 - accuracy: 0.8351
    Epoch 58/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4012 - accuracy: 0.8375
    Epoch 59/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4016 - accuracy: 0.8350
    Epoch 60/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4012 - accuracy: 0.8359
    Epoch 61/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4020 - accuracy: 0.8350
    Epoch 62/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4014 - accuracy: 0.8355
    Epoch 63/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4014 - accuracy: 0.8351
    Epoch 64/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4012 - accuracy: 0.8360
    Epoch 65/100
    800/800 [==============================] - 1s 1ms/step - loss: 0.4013 - accuracy: 0.8346
    Epoch 66/100
    800/800 [==============================] - 1s 2ms/step - loss: 0.4010 - accuracy: 0.8350
```

```
# prediction
```

```
y_pred = classifier.predict(X_test)
y_pred = (y_pred>0.5)
y_pred
```

```
63/63 [==============================] - 0s 911us/step
array([[ True],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]])
```

```
y_test = _y_test
```

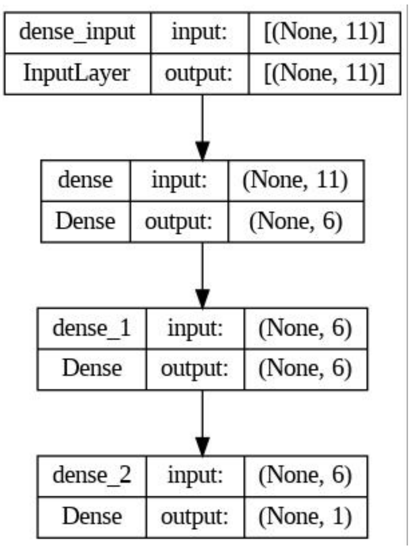
```
# confusion matrix

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

```
[[1540   67]
 [ 256  137]]
```

```
#from confusion_matrix import accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.8385
```

```
from keras.utils.vis_utils import plot_model
plot_model(classifier,to_file = 'model_plot.jpg',show_shapes = True,show_layer_names = True)
```


dense_input | input: | [(None, 11)]
InputLayer | output: | [(None, 11)]
dense | input: | (None, 11)
Dense | output: | (None, 6)
dense_1 | input: | (None, 6)
Dense | output: | (None, 6)
dense_2 | input: | (None, 6)
Dense | output: | (None, 1)

```
classifier.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 6)                 72        
                                                                 
 dense_1 (Dense)             (None, 6)                 42        
                                                                 
 dense_2 (Dense)             (None, 1)                 7         
                                                                 
=================================================================
Total params: 121
Trainable params: 121
Non-trainable params: 0
_________________________________________________________________
```

● ✕