```
import numpy as np
import pandas as pd
```

```
df = pd.read_csv('drive/MyDrive/Deep Learning/Churn_Modelling.csv')
```

```
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bal
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	838(
2	3	15619304	Onio	502	France	Female	42	8	1596(
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	1255

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

```
df.duplicated().sum()   # duplicate row
```

```
0
```

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

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

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

```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

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

```
Male      5457
Female    4543
Name: Gender, dtype: int64
```

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

```
df.head()
```

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

```
df = pd.get_dummies(df,columns=['Geography','Gender'],drop_first=True)  #one hot encode
```

```
df.head()
```

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

```
X = df.drop(columns=['Exited'])
y = df['Exited'].values

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,random_state=0)
```

```
X_train.shape
```

```
(8000, 11)
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
X_train_scaled
```

```
array([[ 0.16958176, -0.46460796,  0.00666099, ..., -0.5698444 ,
         1.74309049, -1.09168714],
       [-2.30455945,  0.30102557, -1.37744033, ...,  1.75486502,
        -0.57369368,  0.91601335],
       [-1.19119591, -0.94312892, -1.031415  , ..., -0.5698444 ,
        -0.57369368, -1.09168714],
       ...,
       [ 0.9015152 , -0.36890377,  0.00666099, ..., -0.5698444 ,
        -0.57369368,  0.91601335],
       [-0.62420521, -0.08179119,  1.39076231, ..., -0.5698444 ,
         1.74309049, -1.09168714],
       [-0.28401079,  0.87525072, -1.37744033, ...,  1.75486502,
        -0.57369368, -1.09168714]])
```

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

```
model = Sequential()

model.add(Dense(3,activation='sigmoid',input_dim=11))    # first hidden layer
#model.add(Dense(11,activation='relu'))# second hidden layer
model.add(Dense(1,activation='sigmoid'))   # output layer
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 3)                 36

 dense_1 (Dense)             (None, 1)                 4

=================================================================
Total params: 40
Trainable params: 40
Non-trainable params: 0
_________________________________________________________________
```

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

```
model.fit(X_train_scaled,y_train,epochs=10)  # we can increase the epoch for better accuracy
```

```
Epoch 1/10
250/250 [==============================] - 1s 2ms/step - loss: 0.7034 - accuracy: 0.5120
Epoch 2/10
250/250 [==============================] - 0s 2ms/step - loss: 0.5585 - accuracy: 0.7937
Epoch 3/10
250/250 [==============================] - 0s 2ms/step - loss: 0.4918 - accuracy: 0.7960
Epoch 4/10
250/250 [==============================] - 1s 3ms/step - loss: 0.4640 - accuracy: 0.7960
Epoch 5/10
250/250 [==============================] - 2s 6ms/step - loss: 0.4512 - accuracy: 0.7960
Epoch 6/10
250/250 [==============================] - 1s 5ms/step - loss: 0.4437 - accuracy: 0.7960
Epoch 7/10
250/250 [==============================] - 1s 5ms/step - loss: 0.4385 - accuracy: 0.7959
Epoch 8/10
250/250 [==============================] - 1s 4ms/step - loss: 0.4345 - accuracy: 0.7983
Epoch 9/10
250/250 [==============================] - 1s 3ms/step - loss: 0.4314 - accuracy: 0.8036
Epoch 10/10
250/250 [==============================] - 1s 3ms/step - loss: 0.4288 - accuracy: 0.8092
<keras.callbacks.History at 0x7ff05c0b6a90>
```

```
model.layers[0].get_weights()
```

```
[array([[-1.27011225e-01,  8.99929926e-03,  1.72307134e-01],
        [ 1.42620027e+00, -8.35663915e-01, -1.73071682e+00],
        [-5.96624333e-03,  3.57289106e-01, -1.02884544e-03],
        [ 1.88752308e-01, -3.69591236e-01, -1.82968125e-01],
        [ 5.42560220e-03,  3.96242350e-01,  1.95033878e-01],
        [ 1.03486195e-01, -1.11132868e-01,  1.34531990e-01],
        [-2.16855228e-01,  1.29684246e+00,  5.79442739e-01],
        [-9.11872610e-02, -4.55785334e-01,  1.02091573e-01],
        [ 4.92921531e-01, -7.78887630e-01, -5.18450201e-01],
        [-9.02808607e-02, -2.83870995e-01,  4.39733379e-02],
        [-4.20748919e-01,  5.41723609e-01,  4.17325824e-01]], dtype=float32),
 array([-0.38787517,  0.8069683 ,  0.63731307], dtype=float32)]
```

```
y_log = model.predict(X_test_scaled)
```

```
63/63 [==============================] - 0s 2ms/step
```

```
y_pred = np.where(y_log>0.5,1,0)
```

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

```
0.8155
```

✓ 0s completed at 9:33 AM