```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import io
from datetime import datetime
import time
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473189

Enter your authorization code:
··········
Mounted at /content/drive
```

```
df_raw = pd.read_csv('/content/drive/My Drive/Colab Notebooks/Churn.csv')
```

```
df_raw.shape
```

```
(10000, 14)
```

```
df_raw.head(3)
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balanc
0	1	15634602	Hargrave	619	France	Female	42	2	0.0
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8
2	3	15619304	Onio	502	France	Female	42	8	159660.8

```
df = df_raw.copy(deep=True)
```

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

```
df.head(2)
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsAc
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	

```
#deal with categorical data --> encode them
```

```
from sklearn.preprocessing import LabelEncoder
labelencoder_x = LabelEncoder()
df.iloc[:, 1] = labelencoder_x.fit_transform(df.iloc[:, 1]) #applying on Geography
df.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsA
0	619	0	Female	42	2	0.00	1	1	
1	608	2	Female	41	1	83807.86	1	0	
2	502	0	Female	42	8	159660.80	3	1	
3	699	0	Female	39	1	0.00	2	0	
4	850	2	Female	43	2	125510.82	1	1	

```
#apply encoder on Gender as well
labelencoder_x_2 = LabelEncoder()
df.iloc[:, 2] = labelencoder_x_2.fit_transform(df.iloc[:, 2]) #applying on Gender
df.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsA
0	619	0	0	42	2	0.00	1	1	
1	608	2	0	41	1	83807.86	1	0	
2	502	0	0	42	8	159660.80	3	1	
3	699	0	0	39	1	0.00	2	0	
4	850	2	0	43	2	125510.82	1	1	

```
#One hot encoding.

from keras.utils import to_categorical
encoded = pd.DataFrame(to_categorical(df.iloc[:, 1]))
#no need to encode Gender, as there are only two categories

df = pd.concat([encoded, df], axis = 1)
df.head()
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the %tensc

	0	1	2	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	H
0	1.0	0.0	0.0	619	0	0	42	2	0.00	1	
1	0.0	0.0	1.0	608	2	0	41	1	83807.86	1	
2	1.0	0.0	0.0	502	0	0	42	8	159660.80	3	
3	1.0	0.0	0.0	699	0	0	39	1	0.00	2	
4	0.0	0.0	1.0	850	2	0	43	2	125510.82	1	

```
#Dropping the existing "geography" category, and one of the onehotcoded columns.

df = df.drop(['Geography', 0], axis = 1)
df.head()
```

	1	2	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsAct
0	0.0	0.0	619	0	42	2	0.00	1	1	
1	0.0	1.0	608	0	41	1	83807.86	1	0	
2	0.0	0.0	502	0	42	8	159660.80	3	1	
3	0.0	0.0	699	0	39	1	0.00	2	0	
4	0.0	1.0	850	0	43	2	125510.82	1	1	

```
x= np.array(df.drop(columns=(['Exited'])))
y = np.array(df['Exited'])
```

```
#train and test set split, and feature scaling

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
import keras
from keras.models import Sequential
from keras.layers import Dense #to add layers
```

```
#there is no rule on how many nodes each hidden layer should have
classifier = Sequential()
```

```
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 11))
#init --> initialize weights according to uniform distribution
#input_dim is required for the first hidden layer, as it is the first starting point. --> num
#output_dim --> number of nodes of the hidden layer
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))
#input_dim --> remove it as it already knows what to expect.

#the output layer
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
#output_dim should be 1, as output is binary outcome, and activation should be 'sigmoid'
#If dependent variables have more than two categories, use activation = 'softmax'

#compile the model --> backpropagation -> gradient descent
classifier.compile(optimizer = 'SGD', loss = "binary_crossentropy", metrics = ['accuracy'])
#optimizer = algorithm to find the optimal set of weights in ANN
#loss = functions that should be optimized. if more than two categories, use "categorical_cro
#metrics = criterion used to calculate the performance of the model.
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793:

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/op
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: UserWarning: Update your

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6: UserWarning: Update your

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:10: UserWarning: Update you
  # Remove the CWD from sys.path while we load stuff.
```

```
classifier.fit(X_train, Y_train, batch_size = 10, nb_epoch = 20)
#batch_size = the number of observations after which you want to update the weights
#           batch size and epochs should be tuned through experiments.
#epoch = going through the whole dataset
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: UserWarning: The `nb_epo
  """Entry point for launching an IPython kernel.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

Epoch 1/20
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

8000/8000 [==============================] - 2s 222us/step - loss: 0.5575 - acc: 0.7949
Epoch 2/20
8000/8000 [==============================] - 1s 126us/step - loss: 0.5083 - acc: 0.7960
Epoch 3/20
8000/8000 [==============================] - 1s 129us/step - loss: 0.5061 - acc: 0.7960
Epoch 4/20
8000/8000 [==============================] - 1s 124us/step - loss: 0.5059 - acc: 0.7960
Epoch 5/20
8000/8000 [==============================] - 1s 125us/step - loss: 0.5059 - acc: 0.7960
Epoch 6/20
8000/8000 [==============================] - 1s 128us/step - loss: 0.5059 - acc: 0.7960
Epoch 7/20
8000/8000 [==============================] - 1s 125us/step - loss: 0.5059 - acc: 0.7960
Epoch 8/20
8000/8000 [==============================] - 1s 124us/step - loss: 0.5059 - acc: 0.7960
Epoch 9/20
8000/8000 [==============================] - 1s 123us/step - loss: 0.5058 - acc: 0.7960
Epoch 10/20
8000/8000 [==============================] - 1s 125us/step - loss: 0.5058 - acc: 0.7960
Epoch 11/20
8000/8000 [==============================] - 1s 126us/step - loss: 0.5057 - acc: 0.7960
Epoch 12/20
8000/8000 [==============================] - 1s 126us/step - loss: 0.5056 - acc: 0.7960
Epoch 13/20
8000/8000 [==============================] - 1s 121us/step - loss: 0.5054 - acc: 0.7960
Epoch 14/20
8000/8000 [==============================] - 1s 132us/step - loss: 0.5050 - acc: 0.7960
Epoch 15/20
8000/8000 [==============================] - 1s 125us/step - loss: 0.5043 - acc: 0.7960
Epoch 16/20
8000/8000 [==============================] - 1s 125us/step - loss: 0.5025 - acc: 0.7960
Epoch 17/20
8000/8000 [==============================] - 1s 125us/step - loss: 0.4971 - acc: 0.7960
Epoch 18/20
8000/8000 [==============================] - 1s 120us/step - loss: 0.4800 - acc: 0.7961
Epoch 19/20
8000/8000 [==============================] - 1s 124us/step - loss: 0.4532 - acc: 0.8031
Epoch 20/20
```

```
8000/8000 [==============================] - 1s 125us/step - loss: 0.4384 - acc: 0.8106
<keras.callbacks.History at 0x7f935a387940>
```

```
#predicting the results

y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5) #to classify each probability into True or False

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, y_pred)
print (cm, '\n\n', y_pred[:5, :])
```

```
[[1548   47]
 [ 321   84]]

 [[False]
 [False]
 [False]
 [False]
 [False]]
```

```
#accuracy
print ((cm[0][0] + cm[1][1])/(cm[0][0] +cm[0][1] + cm[1][0]+cm[1][1]))
```

```
0.816
```