```
#Make sure we have tensorflow 2.x
!pip3 install -U tensorflow --quiet
```

```
     |████████████████████████████████| 86.3MB 40kB/s
     |████████████████████████████████| 3.8MB 31.3MB/s
     |████████████████████████████████| 450kB 53.3MB/s
     |████████████████████████████████| 81kB 9.0MB/s
ERROR: tensorboard 2.0.2 has requirement grpcio>=1.24.3, but you'll have grpcio 1.15.0 w
ERROR: google-colab 1.0.0 has requirement google-auth~=1.4.0, but you'll have google-aut
```

```
import tensorflow as tf
tf.__version__
```

```
'2.0.0'
```

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

import os

#/content/Churn.csv
```



```
#importing the dataset
dataset = pd.read_csv('/content/Churn.csv')
X = dataset.iloc[:,3:13].values # Credit Score through Estimated Salary
y = dataset.iloc[:, 13].values # Exited


# Encoding categorical (string based) data. Country: there are 3 options: France, Spain and G
# This will convert those strings into scalar values for analysis
print(X[:8,1], '... will now become: ')
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
label_X_country_encoder = LabelEncoder()
X[:,1] = label_X_country_encoder.fit_transform(X[:,1])
print(X[:8,1])
```

```
['France' 'Spain' 'France' 'France' 'Spain' 'Spain' 'France' 'Germany'] ... will now bec
[0 2 0 0 2 2 0 1]
```

```
# We will do the same thing for gender. this will be binary in this dataset
print(X[:6,2], '... will now become: ')
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
label_X_gender_encoder = LabelEncoder()
X[:,2] = label_X_gender_encoder.fit_transform(X[:,2])
```

```
print(X[:6,2])
```

```
['Female' 'Female' 'Female' 'Female' 'Female' 'Male'] ... will now become:
[0 0 0 0 0 1]
```

```
# Converting the string features into their own dimensions. Gender doesn't matter here becaus
countryhotencoder = OneHotEncoder(categorical_features = [1]) # 1 is the country column
X = countryhotencoder.fit_transform(X).toarray()
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/preprocessing/_encoders.py:415: FutureWar
If you want the future behaviour and silence this warning, you can specify "categories='
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to i
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/preprocessing/_encoders.py:451: Deprecati
  "use the ColumnTransformer instead.", DeprecationWarning)
```

```
X
```

```
array([[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 1.0134888e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 1.1254258e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 1.1393157e+05],
       ...,
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 4.3085580e+04],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 3.8190780e+04]])
```



```
X = X[:,1:] # Got rid of Spain as a dimension. It is still there through out inferences
```

```
# Splitting the dataset into the Training and Testing set.
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)
```

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

```
X_train
```

```
array([[-0.5698444 ,  1.74309049,  0.16958176, ...,  0.64259497,
        -1.03227043,  1.10643166],
       [ 1.75486502, -0.57369368, -2.30455945, ...,  0.64259497,
         0.9687384 , -0.74866447],
       [-0.5698444 , -0.57369368, -1.19119591, ...,  0.64259497,
        -1.03227043,  1.48533467],
       ...,
       [-0.5698444 , -0.57369368,  0.9015152 , ...,  0.64259497,
        -1.03227043,  1.41231994],
       [-0.5698444 ,  1.74309049, -0.62420521, ...,  0.64259497,
         0.9687384 ,  0.84432121],
       [ 1.75486502, -0.57369368, -0.28401079, ...,  0.64259497,
        -1.03227043,  0.32472465]])
```

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

# Initializing the ANN
classifier = Sequential()

classifier.add(Dense(activation = 'relu', input_dim = 11, units=6, kernel_initializer='unifor

# Adding the second hidden layer
# Notice that we do not need to specify input dim.
```



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

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

classifier.fit(X_train, y_train, batch_size=10, epochs=5)
```

```
Train on 8000 samples
Epoch 1/5
8000/8000 [==============================] - 2s 265us/sample - loss: 0.5044 - accuracy:
Epoch 2/5
8000/8000 [==============================] - 1s 175us/sample - loss: 0.4313 - accuracy:
Epoch 3/5
8000/8000 [==============================] - 1s 169us/sample - loss: 0.4272 - accuracy:
Epoch 4/5
8000/8000 [==============================] - 1s 167us/sample - loss: 0.4225 - accuracy:
Epoch 5/5
8000/8000 [==============================] - 1s 169us/sample - loss: 0.4190 - accuracy:
<tensorflow.python.keras.callbacks.History at 0x7fb1da158ac8>
```

```
y_pred = classifier.predict(X_test)
print(y_pred)
```

```
[[0.26550376]
 [0.33042324]
 [0.20455596]
 ...
 [0.15818965]
 [0.15559867]
 [0.1560629 ]]
```

```
y_pred = (y_pred > 0.5)
print(y_pred)
```

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

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

```
[[1570   25]
 [ 313   92]]
```