Churn Modelling – Artificial Neural Network

Business Problem: The bank is seeing unusual churn rates i.e. customers leaving the bank. So, the bank has provided with the dataset that consists of information about 10000 randomly selected bank customers and whether they have left the bank or not to know what the problem is and to gather insights. In this dataset, "Exited" is the dependent variable that is '1' if a person has left the bank or '0' if a person has not left the bank. The banks want me to create a geo-demographic segmentation model that tells the bank which customer is at highest risk of leaving the bank.

Data Cleaning: Out of the 13 independent variables, I took out RowNumber, CustomerID and Surname from further analysis based on intuition as they will have no impact on the bottom line. Now, there were 2 categorical variables Geography and Gender which were needed to be encoded and then the whole data was standardized.

ANN Modelling: I have used Keras library and two modules within it namely Sequential module to initialize the ANN and Dense module that is required to build ANN. Within ANN, I used 2 hidden layers with RELU as an activation function in both of them. Whereas, for the output layer I used Sigmoid function, since the dependent variable is a binary variable. The method for assigning weights is Stochastic Gradient Descent. To optimize the weights using Stochastic Gradient Descent we are going to use the loss function 'binary_crossentropy'. The metrics on which the model will improve itself is accuracy.

Results:

Batch size for Stochastic Gradient Descent = 10

Number of epochs = 100

1. **Accuracy of training set**

a. Accuracy of training set during 1st and 2nd epoch

```
Epoch 1/100
2019-06-23 22:41:12.939679: I tensorflow/core/platform/cpu_feature_guard.cc:141]
Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX2 FMA
8000/8000 [==============================] - 2s 221us/step - loss: 0.4841 - acc:
0.7951
Epoch 2/100
8000/8000 [==============================] - 1s 153us/step - loss: 0.4264 - acc:
0.7960
```

b. Accuracy of training set during 99th and 100th epoch

```
Epoch 99/100
8000/8000 [==============================] - 2s 201us/step - loss: 0.3433 - acc:
0.8584
Epoch 100/100
8000/8000 [==============================] - 1s 153us/step - loss: 0.3438 - acc:
0.8587
Out[24]: <keras.callbacks.History at 0x1a2c7679b0>
```

From the results, we can see that ANN increased the accuracy from **79.5% to 85.9%** by automatically changing the weights and decreasing the cost function.

2. **Confusion Matrix**

```
In [28]: cm
Out[28]:
array([[1499,   96],
       [ 194,  211]])
```

From the confusion matrix we can see that the accuracy of the test set comes out to be (1499+211)/2000 = **85.5**, which is very close to the accuracy that we got for training set. Hence, we can say that the model is stable and can be used by the bank to predict which customers are more likely to leave the bank and the bank can take certain measures to prevent those customers from leaving the bank.

For example, if the bank wants to know that a customer with some features will leave a bank or not, above model can be used.

Features of customer:

Geography – France

Credit Score – 600

Gender – Male

Age – 40

Tenure – 30

Balance – 60000

Number of products – 2

Has credit card – Yes

Is active member – Yes

Estimated Salary – 50000

Prediction based on the model:

```
In [17]: new_pred
Out[17]: array([[False]])
```

Based on the model, the customer will not leave the bank since the result comes out to be False.