**A PROJECT REPORT ON**

**CUSTOMER CHURN MODELLING USING ANN**

**Submitted by**

**Pranav Kalikate**

**What is ANN?**

**Artificial Neural Networks or ANN is an information processing model that is inspired by the way the biological nervous system such as brain process information. It is composed of large number of highly interconnected processing elements (neurons) working in unison to solve a specific problem.**

**What is Activation Function?**

The activation function is the process applied to the weighted input value once it enters the neuron. The neurons are activated in a way that the impact of each neuron’s activation is limited by weights. Different activation functions are,

1. Threshold function
2. Sigmoid function
3. Rectifier function
4. Hyperbolic tangent function (tanh)

**What are Weights?**

Weights are a pivotal factor in a Neural Network’s functioning. Weights are how Neural Networks learn. Based on each weight, the Neural Network decides what information is important, and what isn’t. The weight determines which signals get passed along or not, or to what extent a signal gets passed along. The weights are what you will adjust through the process of learning.

**What is Cost Function?**

The cost function tells us the error in our prediction. Our aim is to minimize the cost function. The lower the cost function, the closer Ŷ is to Y, and hence, the closer our output value to our actual value. A lower cost function means higher accuracy for our Network.

**Steps in Building your ANN**

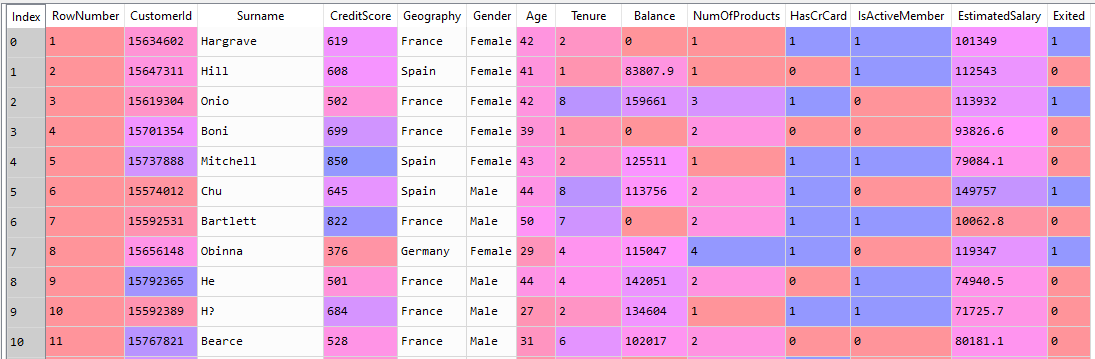
1. Data preprocessing.
2. Creating the structure of ANN.

* Add input layer and first hidden layer.
* Add more hidden layers according to the requirement.
* Select input & hidden nodes, activation function, optimizer, loss, and performance metrics.
* Compile the ANN
* Fit the ANN to training set.
* Evaluate the model performance.
* Adjust the optimization parameters or model if needed.

**Problem Description**

Here we have the first 10 rows of our sample dataset. The sample dataset we have is of a fictional bank but it is very realistic. There are 10,000 customers or rows and the columns represents the information like customers Gender, Age, Balance etc. We also have a column ‘Exited’ which gives an information about whether or not a customer leaves a bank.

First Ten rows:



Here the bank is seeing some unusual churn at higher rate and they want to assess and address this problem. My goal here is to go through the dataset, build geo demographic segmentation model to tell the bank which of the customers are at a higher risk of leaving.

**Building a model**

**Part 1 Data Preprocessing**

* **Importing the libraries**

import numpy as np

import matplotlib.pyplot as plt

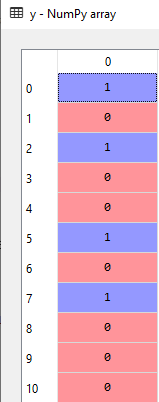
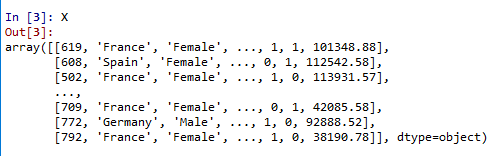
import pandas as pd

* **Importing the dataset**

dataset = pd.read\_csv('Churn\_Modelling.csv')

X = dataset.iloc[:, 3:13].values

y = dataset.iloc[:, 13].values



* **Encoding categorical data**

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

labelencoder\_X\_1 = LabelEncoder()

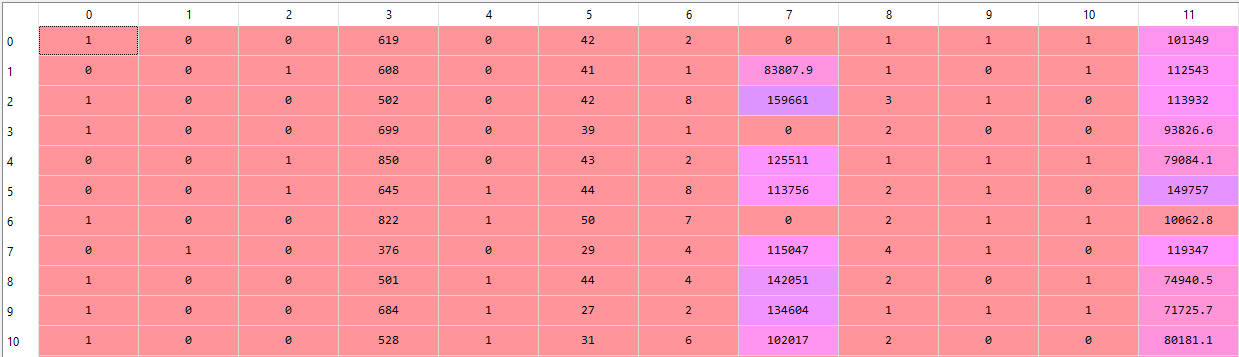
X[:, 1] = labelencoder\_X\_1.fit\_transform(X[:, 1])

labelencoder\_X\_2 = LabelEncoder()

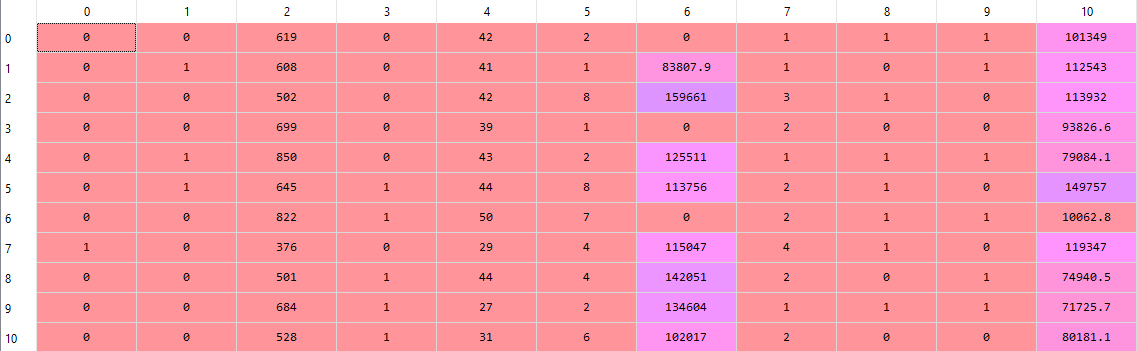
X[:, 2] = labelencoder\_X\_2.fit\_transform(X[:, 2])

onehotencoder = OneHotEncoder(categorical\_features = [1])

X = onehotencoder.fit\_transform(X).toarray()



* **Avoiding dummy variable trap**

X = X[:, 1:]

* **Splitting the dataset into the Training set and Test 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)

**Part 2 Initializing the ANN**

* **Importing the Keras libraries and packages**

import keras

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import Dropout

* **Initializing the ANN**

classifier = Sequential()

* **Adding the input layer, hidden layer and output layer**

#Adding the input layer and the first hidden layer

classifier.add(Dense(units = 6, kernel\_initializer = 'uniform', activation = 'relu', input\_dim = 11))

#Adding the second hidden layer

classifier.add(Dense(units = 6, kernel\_initializer = 'uniform', activation = 'relu'))

#Adding the third hidden layer

classifier.add(Dense(units = 6, kernel\_initializer = 'uniform', activation = 'relu'))

#Adding the output layer

classifier.add(Dense(units = 1, kernel\_initializer = 'uniform', activation = 'sigmoid'))

* **Compiling the ANN**

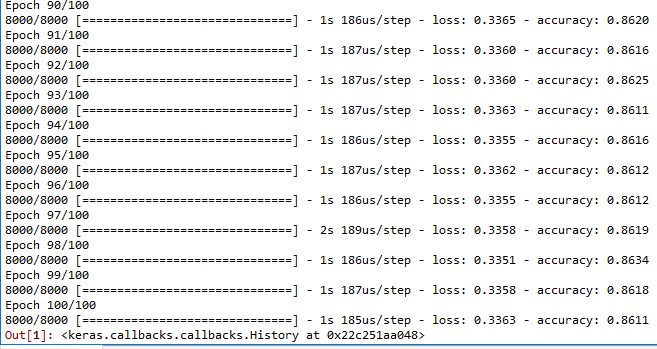
classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

* **Fitting the ANN to the Training set**

classifier.fit(X\_train, y\_train, batch\_size = 10, epochs = 100)

**Steps in Training the ANN with Stochastic Gradient Descent:-**

* 1. Randomly initialize the weights to a small numbers close to 0. (But not 0)
* 2. Input the first observation of your dataset in the input layer, each feature in one input node.
* 3. Forward Propagation: From left to right, the neurons are activated in a way that the impact of each neuron’s activation is limited by weights. Propagate the results until getting the predicted result y.
* 4. Compare the predicted result to the actual result. Measure the generated error.
* 5. Back Propagation: From right to left, the error is propagated. Update the weights according to how much they are responsible for the error. The learning rate decides by how much we update the weights.
* 6. Repeat the steps 1 to 5 and update the weights after each observation or update the weights only after a batch of observations.
* 7. When the whole training set passed through the ANN that makes an epoch. Redo more epochs.



Accuracy obtained on training set = 0.8611 = 86%

* **Predicting the Test set results**

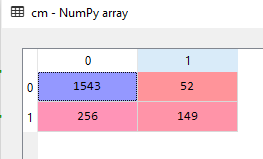
y\_pred = classifier.predict(X\_test)

y\_pred = (y\_pred > 0.5)

* **Making the Confusion Matrix**

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

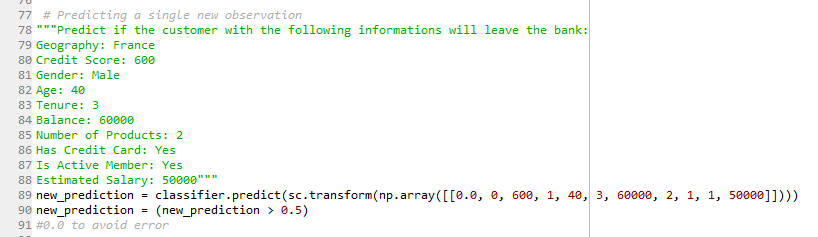


Accuracy obtained on test set = (TP+TN) / (TP+TN+FP+FN)

= 0.846= 84%

Our Model is now trained on training set and it is also performing well on test set.

* **Part 3 Obtaining a result of a single new observation**





The person with the above information will not leave a bank.