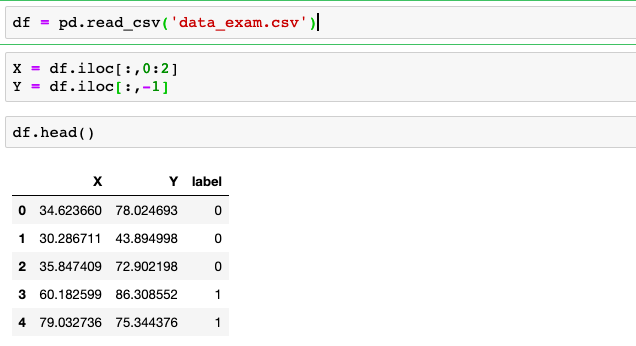
Implementing Logistic regression with delta learning rule using Newton’s method and compare the results with LOG(using gradient descent)

Solution:

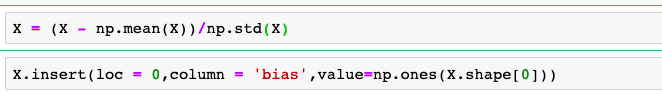
**Step 1. Reading the dataset:**

I am using pandas for reading the dataset and further classifying them into training and test set.



**Data Normalization:**

I have used mean shifting and variance scaling for normalising my dataset. After normalising the data, a bias term has been added to the data.



**Splitting into training set and test set:**

For testing the model performance, I am training my model on 70 % of the data and it’s performance has been tested on rest of the 30 % of the data.



**Weight Initialisation:**

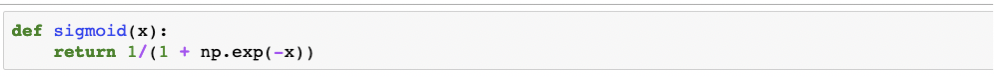
Weights has been initialised from a normal distribution of mean 0 and variance 1.



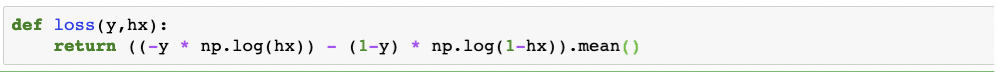
**Creating the Model.**

**Sigmoid function:**

Sigmoid function is used for converting any value in the range of 0 and 1. It is also known as activation function for the logistic regression. It can be defined as follows.



Also for logistic regression we use log loss function for calculating the cost and we minimize this log loss function by using gradient Descent. Loss can be defined as



**Newton’s method:**

In constrast to classical gradient descent algorithm, newton’s method tries to find the minima of a function by taking variable sized step towards minima of a function. Update rules for newton’s method is given as:

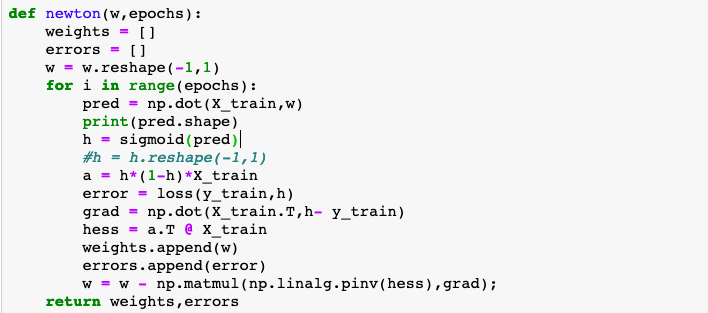
begin{displaymath}
\theta^{(t+1)}=\theta^{(t)}-H^{-1}\nabla_{\theta}J \nonumber
\end{display

In logistic regression, the gradient and hessian are given as:

begin{displaymath}
\nabla_{\theta}J = \frac{1}{m}\sum_{i=1}^{m}(h_{\theta}(x^{(i)})-y^{(i)})

begin{displaymath}
H & = & \frac{1}{m}\sum_{i=1}^{m}\left[h_{\theta}(x^{(i)})\l...
...^{(i)}

The above formula is written in the vectorised form. My implementation of the above approach in python is as follows:

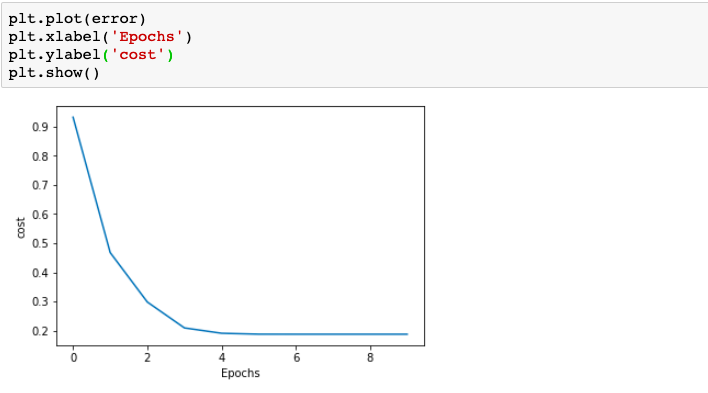


The function newton can be called as:



At every iteration, value of weight is stored in weights[] list. We can use last value of weight for our prediction.

**Cost vs. Epochs plot:**



**Observation:** It can be seen that np. of epochs required for convergence using newton’s method is quite less. It converged in only 4 iterations.

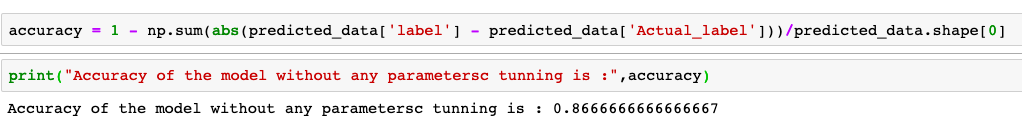
**Predicting Values:**

By taking the sigmoid of the dot product, we can calculate the predicted value.

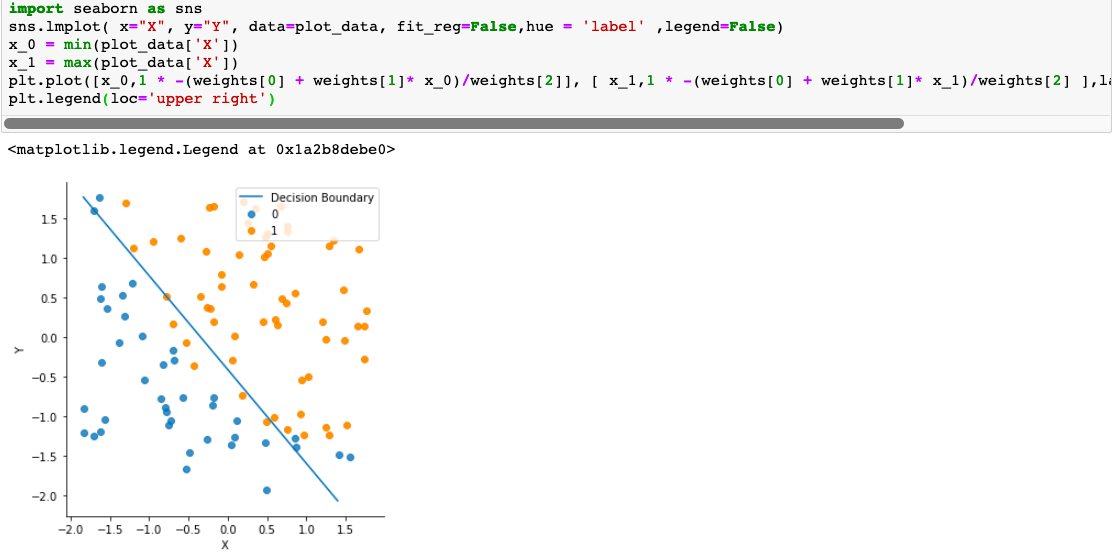


**Calculating the accuracy:**

Accuracy for logistic regression is defined as ratio of correctly classified points to the ratio of total points.



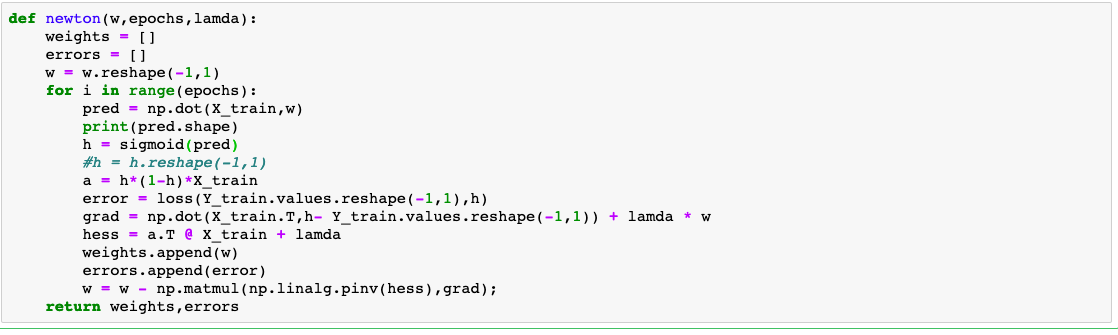
**Plotting the Decision Boundary:**

****

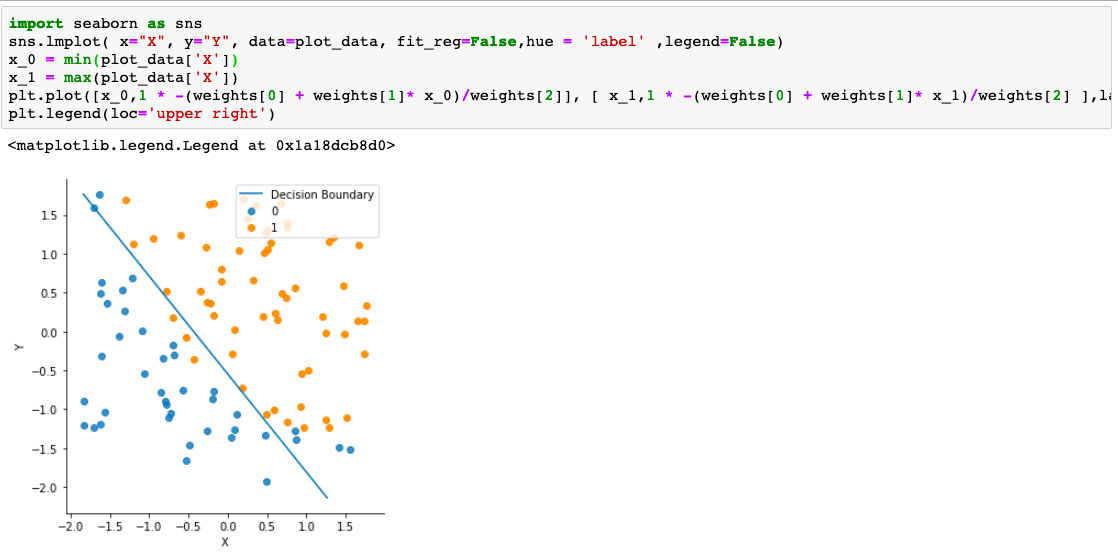
**Using regularisation:**

We can modify our newton’s method a bit to add a regulariser parameter to penalise our higher order polynomials.

Regularised Newton’s method:



Once obtaining the weight values from newton’s method, we can predict our values as per the steps defined above and calculate the accuracy.



Comparison: As compared to gradient descent algorithm, cost function converges very fast in case of Newton’s method. However it doesn’t effect much on the accuracy of the model.