# REPORT - HW1

| | |
|---|---|
| **ID** | Vikramaditya |
| **Accuracy** | 0.86 |

**Problem Statement:**
Implementation of Gradient Descent Algorithm along with Logistic Regression on Cleveland dataset and analyze the outcome.

**Implementation:**
The code has been implemented with following formulas taken under consideration:

Sigmoid Function: $$\varnothing(x) = \frac{1}{1+e^{-x}}$$

Cost Function: $$\text{Cost}(h_\theta(\mathbf{x}), y) = -ylog(h_\theta(\mathbf{x})) - (1-y)log(1-h_\theta(\mathbf{x}))$$

Gradient Formula:

$$X = X - lr * \frac{d}{dX} f(X)$$

*Where,*
$X = parameters\ to\ be\ optimized$
$f(X) = cost\ function$
$lr = learning\ rate$

Simplified after taking derivatives to:

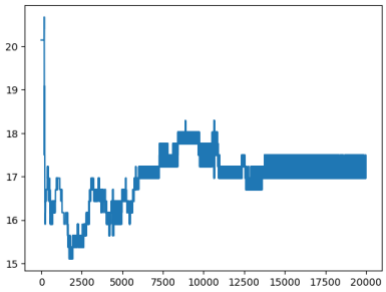
X=1/y.size * X.T.dot(sigmoid(X.dot(theta))-y)

Note: I have taken a constant 'nonZeroFactor' and assigned the value as low as 0.000001 and added it with value inside log(), so that it will not calculate log(0)which is infinite(nan in case of python).

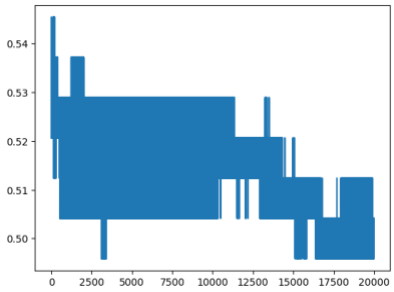
Further, I had implemented the training and prediction for training set, and implemented graphs and Accuracy, precision, recall and F1 scores for training dataset, test dataset, and overall dataset. Also, I have presented various observation based on difference in various parameters of number of iterations, learning rate and intercept values. Also observed cross entropy and classification error and time consumed in training the model for all datasets.

At the end, I have compared various parameters with the implementation done on HW1 and scaled the features.

**Observation:**
Following is the graph curve formed for cross entropy error and classification error for the iteration of 100000.



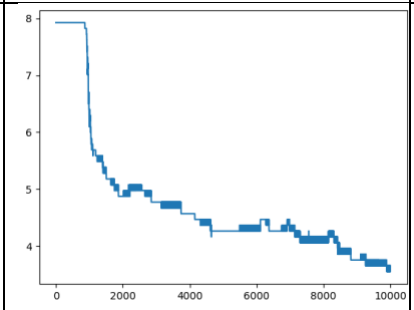
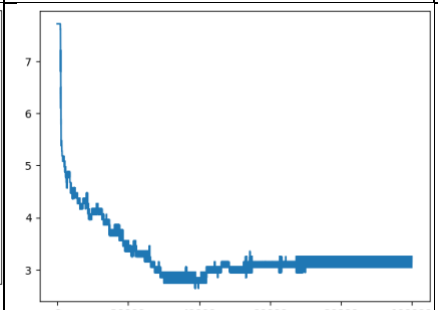
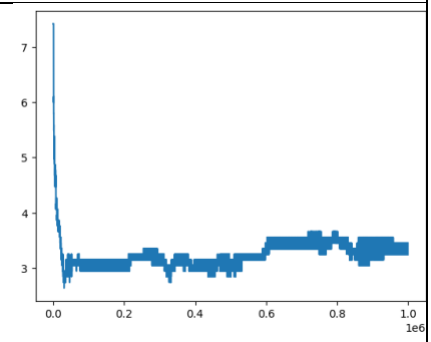
Cross Entropy Error



Classification Error

So, from above we can observe that cross entropy error is minimum around 1700 iterations and then further increased with number of iterations increased, and classification error decreased along with the number of iterations.

**Comparison based of number of iterations**.

| **Iteration** | **10000** | **100000** | **1000000** |
|---|---|---|---|
| **Cost vs Iteration Graph** |  |  |  |
| **Cross Entropy Error** | 9.275940210434563 | 8.215832757813468 | 8.480859620968742 |
| **Classification error** | 0.25735294117647056 | 0.2279411764705882 | 0.23529411764705888 |
| **Time Consumed for training the model** | 4.340889930725098 Seconds | 42.07937717437744 Seconds | 431.1345422267914 Seconds |
| **Accuracy Score of Train Data** | 0.7426470588235294 | 0.7720588235294118 | 0.7647058823529411 |
| **Precision Score of Train Data** | 0.6885245901639344 | 0.7377049180327869 | 0.746031746031746 |
| **Recall Score of Train Data** | 0.7241379310344828 | 0.75 | 0.746031746031746 |
| **F1 Score of Train Data** | 0.7058823529411765 | 0.743801652892562 | 0.746031746031746 |
| **Accuracy Score of Test Data** | 0.6875 | 0.875 | 1.0 |
| **Precision Score of Test Data** | 0.6666666666666666 | 1.0 | 1.0 |
| **Recall Score of Test Data** | 0.75 | 0.6666666666666666 | 1.0 |
| **F1 Score of Test Data** | 0.7058823529411765 | 0.8 | 1.0 |

From above it is clear that as we increased number of iterations the accuracy scores increases but the time consumed is also increased. So I decided to move further with 100 thousand iterations only. And from the graph as well we can observe that the lowest cost is achieved during lower iterations only.

**Comparison with HW1(Logistic Regression Library (Scikit)) (iterations)**

| | **HW1**(Logistic Regression Library (Scikit)) | **HW2**(Logistic Regression with gradient descent) |
|---|---|---|
| **Time Consumed for training the model** | 0.09949612617492676 Seconds | 42.07937717437744 Seconds |
| **Accuracy Score of Train Data** | 0.75 | 0.7720588235294118 |
| **Precision Score of Train Data** | 0.7142857142857143 | 0.7377049180327869 |
| **Recall Score of Train Data** | 0.7142857142857143 | 0.75 |
| **F1 Score of Train Data** | 0.7142857142857143 | 0.743801652892562 |

From above it is clear that using logistic regression with Gradient descent will have increased accuracy scores because that uses an optimization algorithm to find the optimal parameters of the model. Regular logistic regression, on the other hand, uses numerical methods such as maximum likelihood estimation to estimate the parameters.

**Comparison after scaling of training data and increased ETA of 10^-6**

| **Iteration** | **10000** | **100000** | **1000000** |
|---|---|---|---|
| **Time Consumed for training the model** | 3.5015671253204346 seconds | 34.84175181388855 Seconds | 349.6588771343231 Seconds |
| **Cross Entropy Error** | 9.540967073589835 | 9.805993936745107 | 10.071020799900381 |
| **Classification error** | 0.2647058823529411 | 0.2720588235294118 | 0.27941176470588236 |
| **Accuracy Score of Train Data** | 0.7352941176470589 | 0.7279411764705882 | 0.7205882352941176 |

From above, we can observe that as we scaled the features i.e, training data by subtracting the mean and dividing by the standard deviation for each of the features, the accuracy has been increased, but the cross entropy error is increased a little, overall time consumed to train the model is decreased and have classification error almost similar to that of ETA 10^-5.