Machine Learning Using Networks

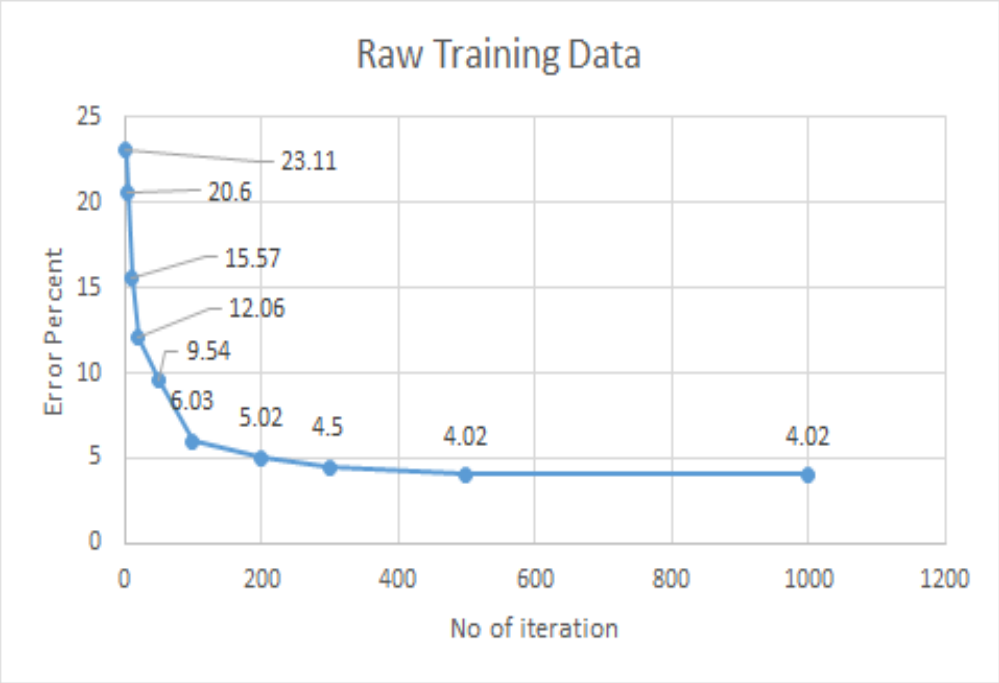
Programming Assignment 2 report

Abhijeet Sahu UIN:424009758

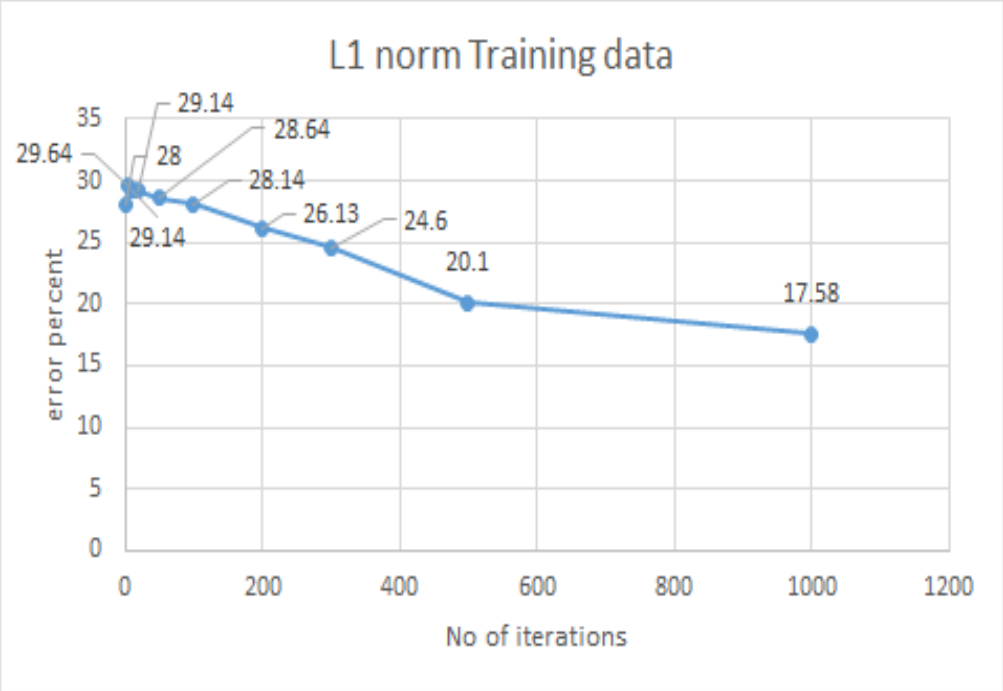
1.

Training and Testing Error using own functions without regularization:

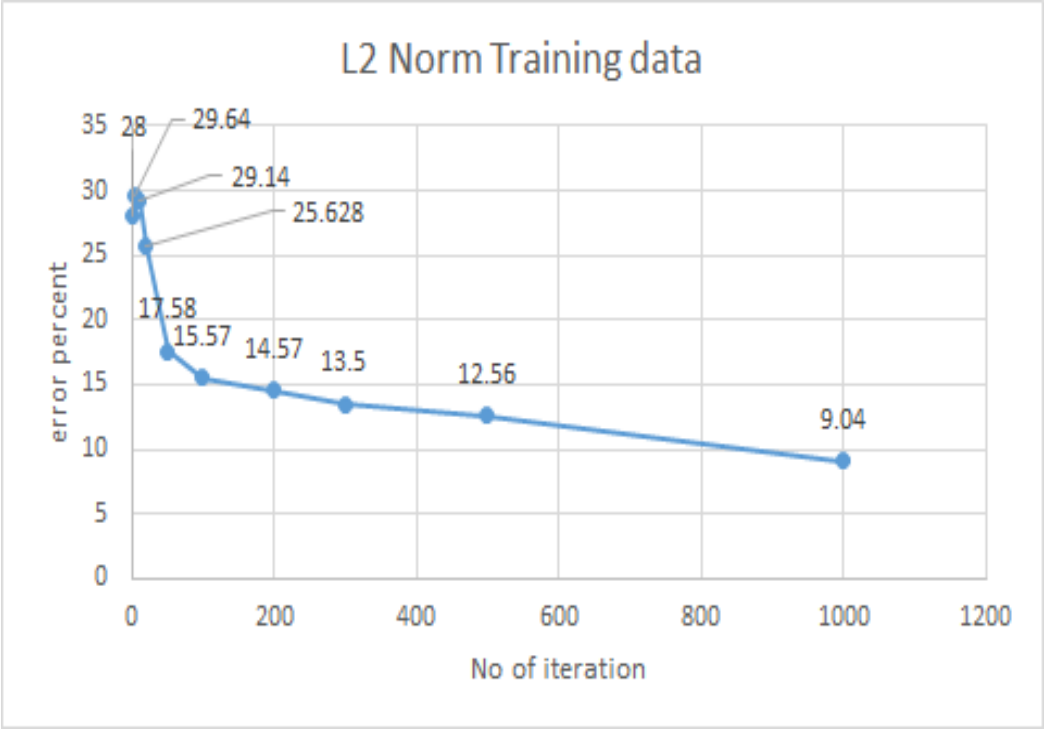
Training Error with Raw Data:



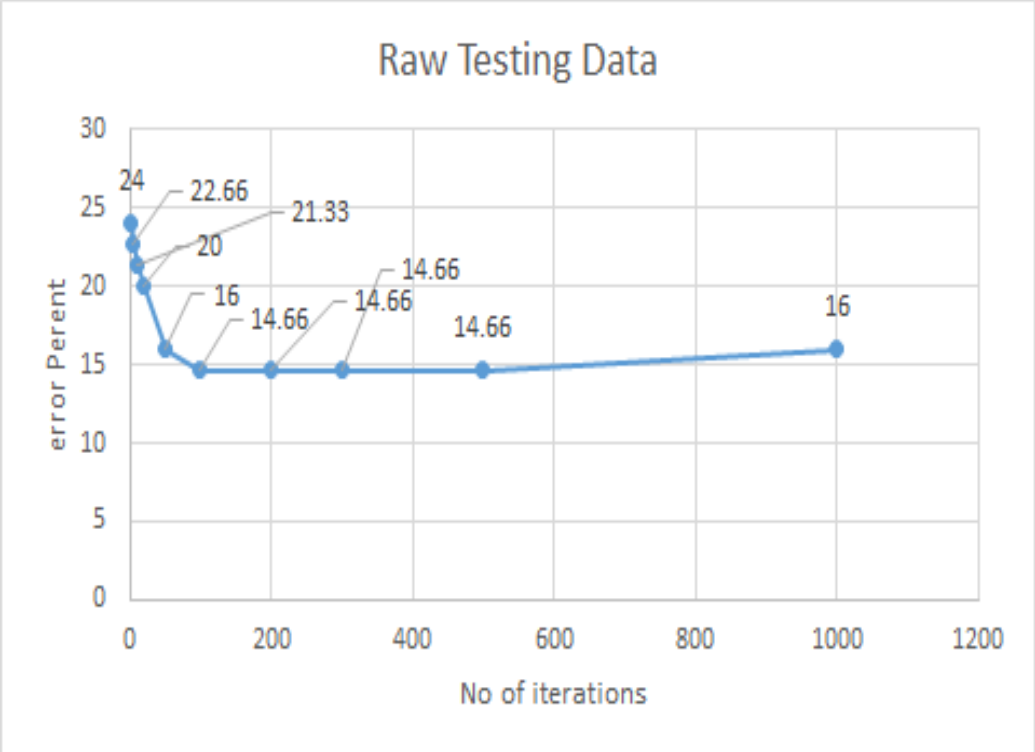
Training Error with L1 normalized Data:



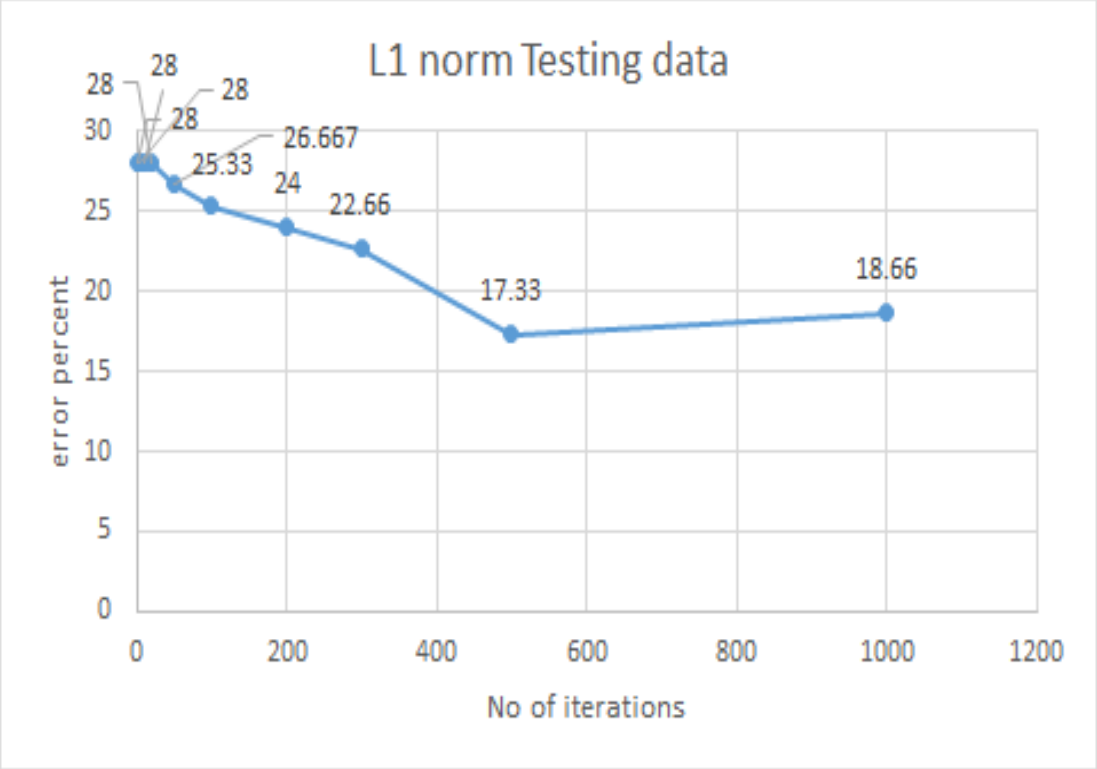
Training Error with L2 normalized Data:



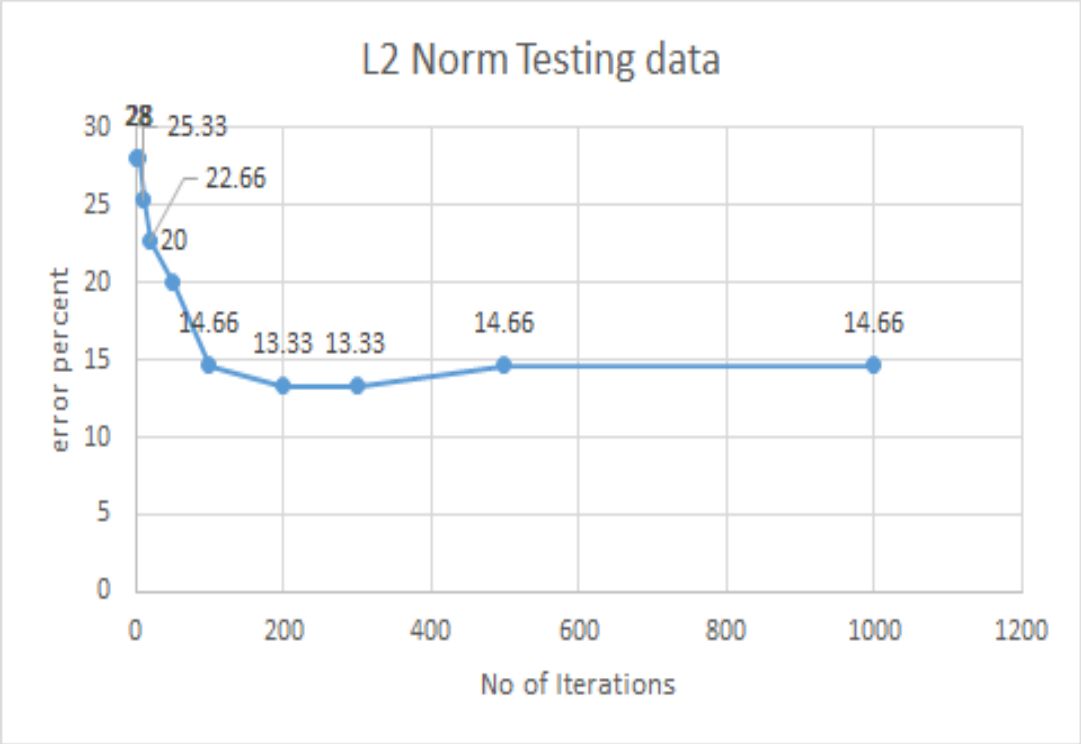
Testing Error with Raw Data



Testing Error with L1 normalized Data



Testing Error with L2 normalized Data



Answer 1

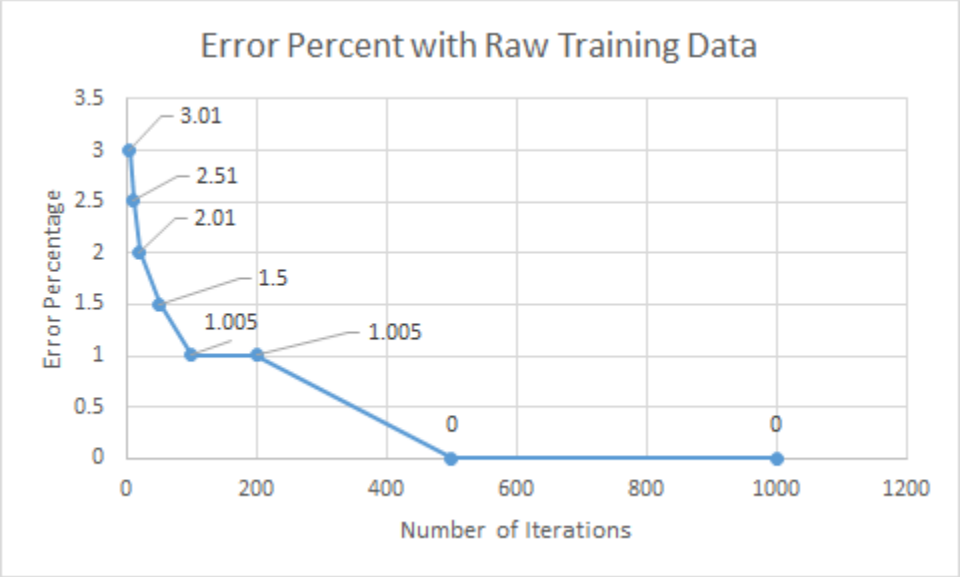
Observations:

Method used : Gradient Descent Method (for upgrading the parameter with every iteration)

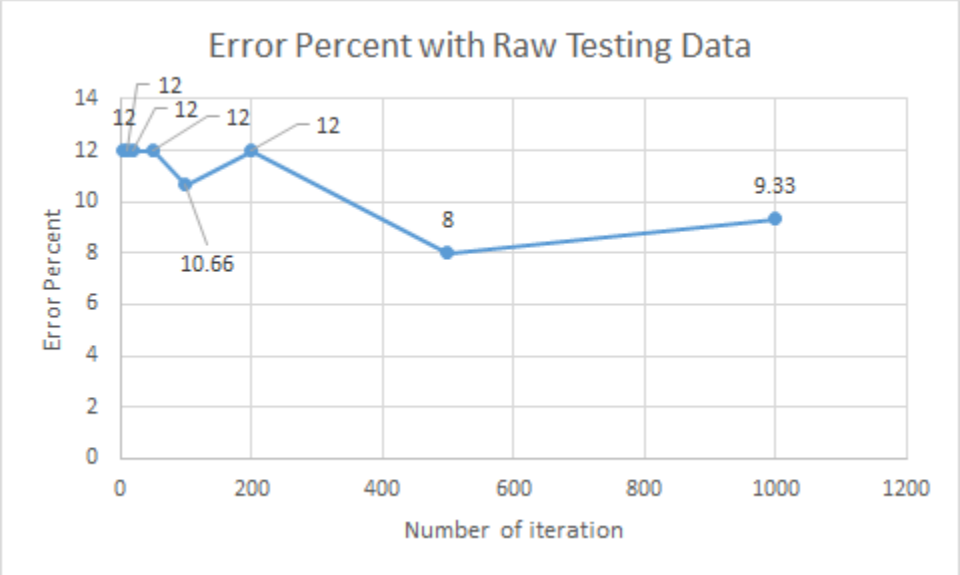
1. It was observed that after certain number of iterations the training error got stabilized for the Raw Data. And for the L1 and L2 normalized data the error was reducing till 1000 iterations.
2. Training error for the Raw data was minimum among all of them. Around 4.02percent for 1000 iteration. The training error for the L1 normalized data was around 17.58 percent for 1000 iterations. The training error for L2 normalized data was around 9.04 percent for 1000 iterations.
3. For the testing error too L2 norm data was better compared to the L1 norm data.
4. It was observed that after certain iterations the testing error was increased for all the scenarios. But the lowest error was observed in the case of L2 norm data for iteration (itr=200 and 300) i.e. 13.33%.

Now implementing using SCIKIT library which has used Regularization to avoid over fitting

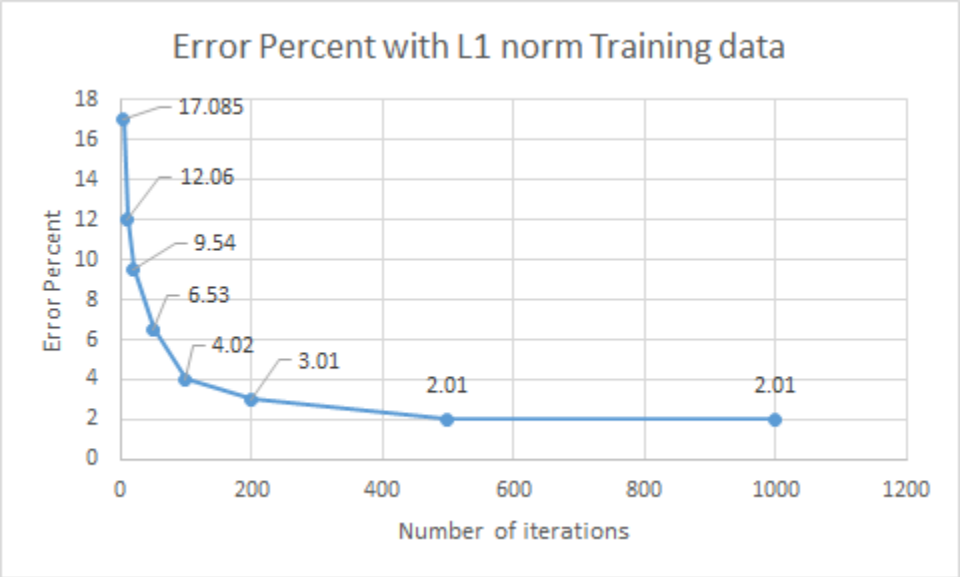
a) Training Error using Raw Data using Logistic Regression



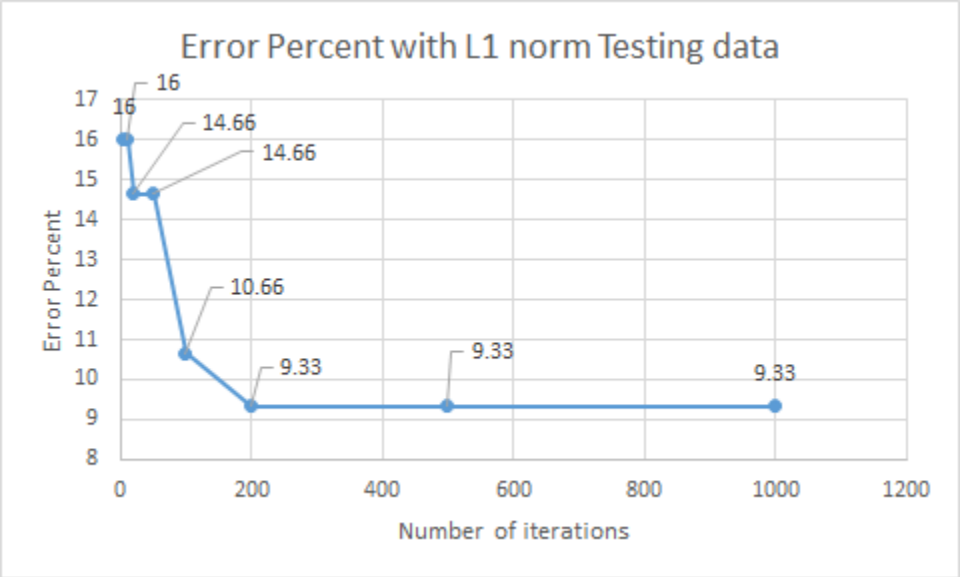
b) Testing Error using Raw Data using Logistic Regression



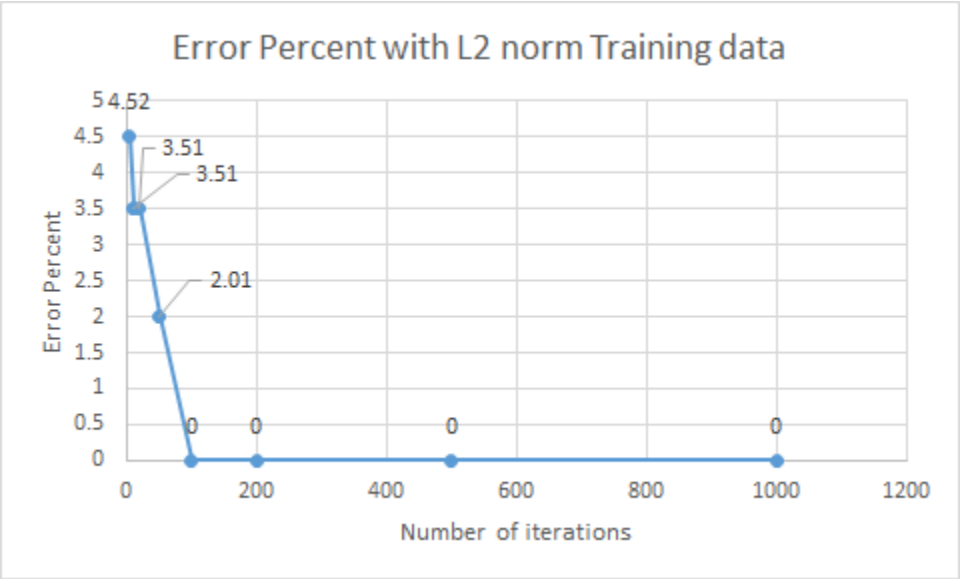
c) Training Error using L1 Normalized Data using Logistic Regression



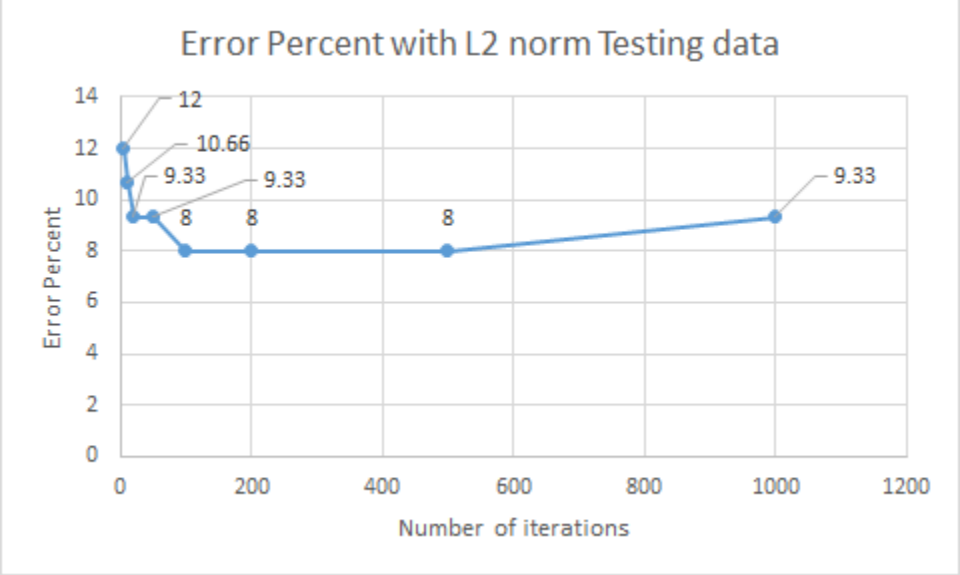
d) Testing Error using L1 Normalized Data using Logistic Regression



e) Training Error using L2 Normalized Data using Logistic Regression



f) Testing Error using L2 Normalized Data using Logistic Regression



It was observed that as the number of iterations increased the error rates reduced for both training and test raw and normalized data in the case of using the scikit tools.
Among the L1 and L2 norm, it was observed that L2 norm performed better in comparison to the L1 norm. Yet it totally depends on the data how these norm can impact the errors in training and predictions.

2.

a)Computing the Gradient and Hessian for the Locally Weighted Logistic Regression Function

2) Locally Weighted Logistic Regression

In this the problem is to maximize the undermentioned objective function.

$$l(\beta) = \sum_{i=1}^{N} w^i \{ y^i \log f_\beta(x^i) + (1-y^i) \log [1 - f_\beta(x^i)] \} - \lambda \beta^T \beta$$

The last term $\lambda \beta^T \beta$ is the regularization term

f_β - sigmoid function $\lambda = 0.001$.

$$w^i = \exp\left(-\frac{\|x - x^i\|_{l_2}^2}{2\tau^2}\right) \quad \tau: \text{Bandwidth}.$$

Compute Gradient $\nabla_\beta l(\beta)$ and the Hessian Matrix. $\nabla_\beta [\nabla_\beta l(\beta)]$.

$$f_\beta(x^i) = \frac{1}{1 + \exp^{-\beta x^i}}$$

$$\frac{d f_\beta(x^i)}{d\beta} = \frac{-d(1+\exp^{-\beta x^i})/d\beta}{(1+\exp^{-\beta x^i})^2}$$

$$= \frac{\exp^{-\beta x^i} \cdot x^i}{(1+\exp^{-\beta x^i})^2} = f_\beta(x^i) \times (1 - f_\beta(x^i)) x^i \quad \text{---(1)}$$

$$\frac{d\, l(\beta)}{d\beta} = \frac{d}{d\beta}\left[\sum_{i=1}^{N} w^i y^i \log f_\beta(x^i) + \sum_{i=1}^{N} w^i (1-y^i) \log(1 - f_\beta(x^i)) - \lambda \beta^T \beta\right]$$

$$= \sum_{i=1}^{N} w^i y^i \frac{d}{d\beta} \log f_\beta(x^i) + \sum_{i=1}^{N} w^i \frac{d \log(1 - f_\beta(x^i))}{d\beta} \sum_{i=1}^{N} y^i w^i \frac{d}{d\beta} \log(1 - f_\beta(x^i)) - 2\lambda\beta$$

$$= \sum_{i=1}^{N} \frac{w^i y^i}{f_\beta(x^i)} \frac{d}{d\beta} f_\beta(x^i) + \sum_{i=1}^{N} \frac{-w^i \frac{d}{d\beta} f_\beta(x^i)}{(1 - f_\beta(x^i))} + \sum \frac{y^i w^i \frac{d}{d\beta} f_\beta(x^i)}{(1 - f_\beta(x^i))} - 2\lambda\beta$$

Substitute eqn. ① in $\frac{d}{d\beta} f_\beta(x)$

$$= \sum_{i=1}^{N} \left(\frac{w^i y^i}{f_\beta(x^i)} f_\beta(x^i)(1-f_\beta(x^i)) x^i - \frac{w^i f_\beta(x^i)(1-f_\beta(x^i)) x^i}{(1-f_\beta(x^i))} + \frac{y^i w^i f_\beta(x^i)(1-f_\beta(x^i)) x^i}{(1-f_\beta(x^i))} \right) - 2\lambda\beta$$

$$= \sum_{i=1}^{N} \left(w^i y^i x^i - w^i y^i f_\beta(x^i) x^i - w^i f_\beta(x^i) x^i + w^i y^i f_\beta(x^i) x^i \right) - 2\lambda\beta$$

$$= \sum_{i=1}^{N} \left(w^i (y^i - f_\beta(x)) x^i \right) - 2\lambda\beta$$

= This is the gradient of the objective function.

Compute the Hessian matrix we need to differentiate the gradient w.r.t to β.

$$\frac{d^2}{\beta} l(\beta) = \frac{d}{d\beta} \left(\sum_{i=1}^{N} \left(w^i (y^i - f_\beta(x)) x^i \right) \right) - 2\lambda\beta$$

$$= - \sum_{i=1}^{N} \frac{d(w^i f_\beta(x) x^i)}{d\beta} - 2\lambda$$

$$= - \sum_{i=1}^{N} w^i (x^i)^2 f_\beta(x^i)(1-f_\beta(x^i)) - 2\lambda$$

as we are summing for all the values of i, where i ranges the no. of training data

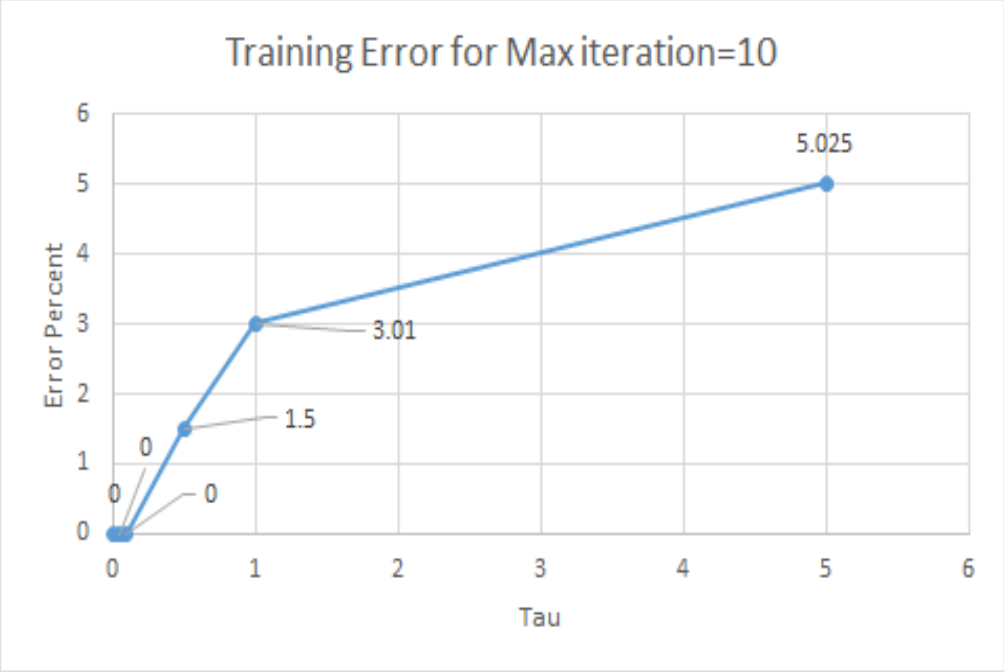
$$= X^T D_{ii} X - 2\lambda I$$

I : $N \times N$ Identity matrix.

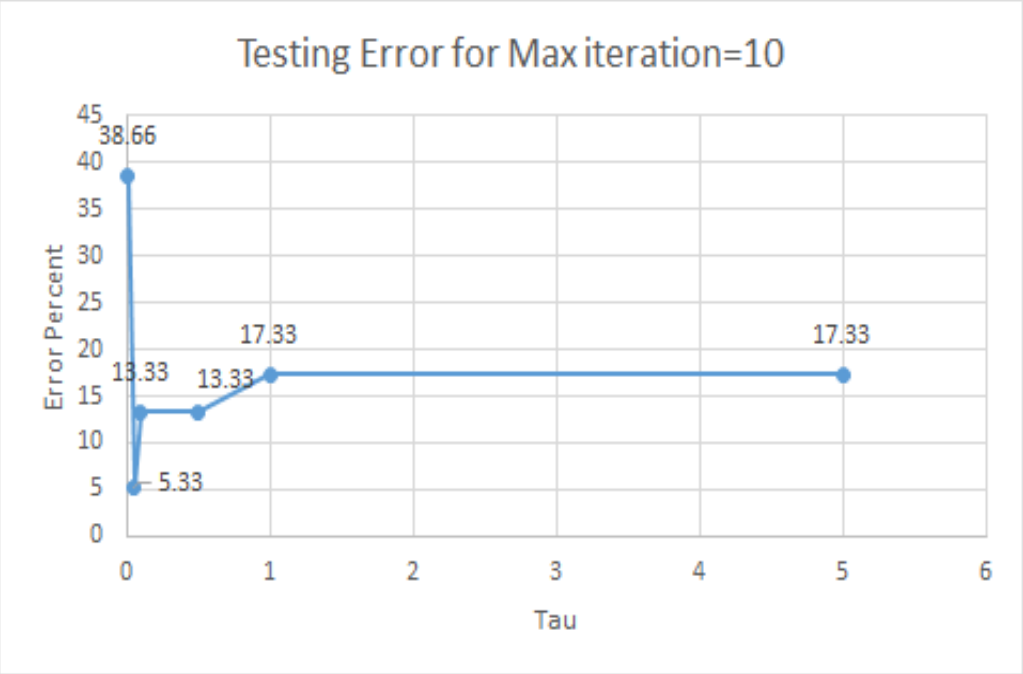
$$D_{ii} = \begin{bmatrix} w^1 f_\beta(x^1)(1-f_\beta(x^1)) & 0 & 0 & 0 \\ 0 & w^2 f_\beta(x^2)(1-f_\beta(x^2)) & 0 & 0 \\ 0 & & & w^N f_\beta(x^N)(1-f_\beta(x^N)) \end{bmatrix}$$

b) w_i were computed using the formula given in the question
The Hessian and the gradient obtained in the section one were used to compute the optimal parameter beta in the Newton's method. After the beta was computed the sigmoid function associated with each data points were calculated and compared with the label to predict error.

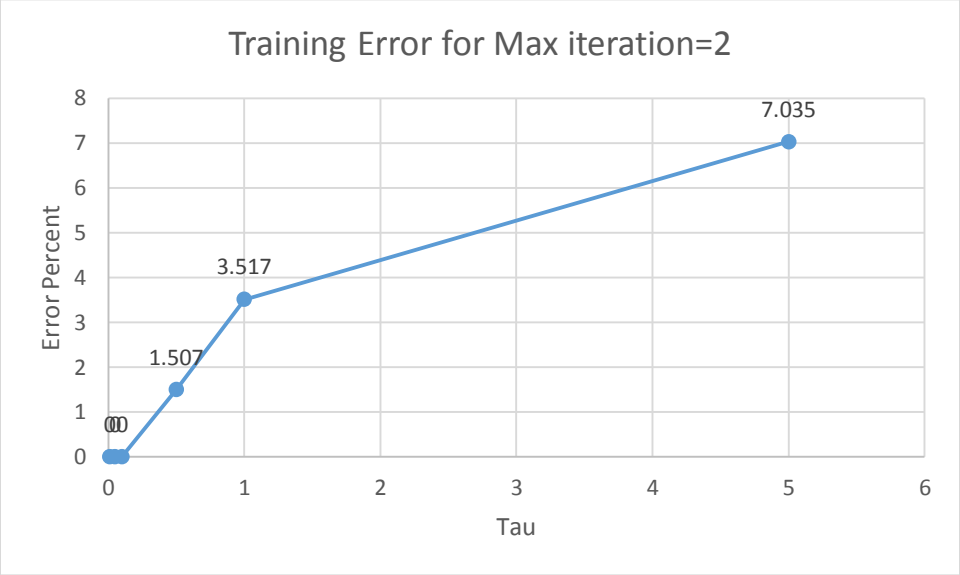
Training Error with 10 no. of iterations in the newton method



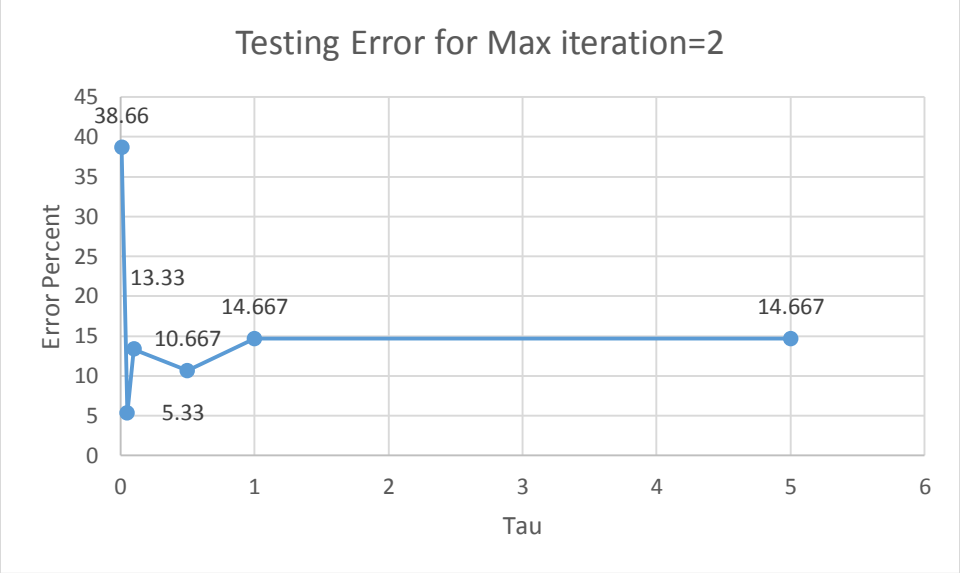
Testing Error with 10 no. of iterations in the newton method



Training Error with 2 no. of max iterations in the newton method



Testing Error with 2 no of max iterations in the newton method



It was observed that the convergence is very fast using the locally weighted logistic regression followed by newton's method. In the above two solution I used 10 number of iteration in one case and 2 number of iteration. The results obtained for both of them were almost similar. And moreover when I ran the simulation for 20 iteration it gave the exact result for the 10 iteration case.

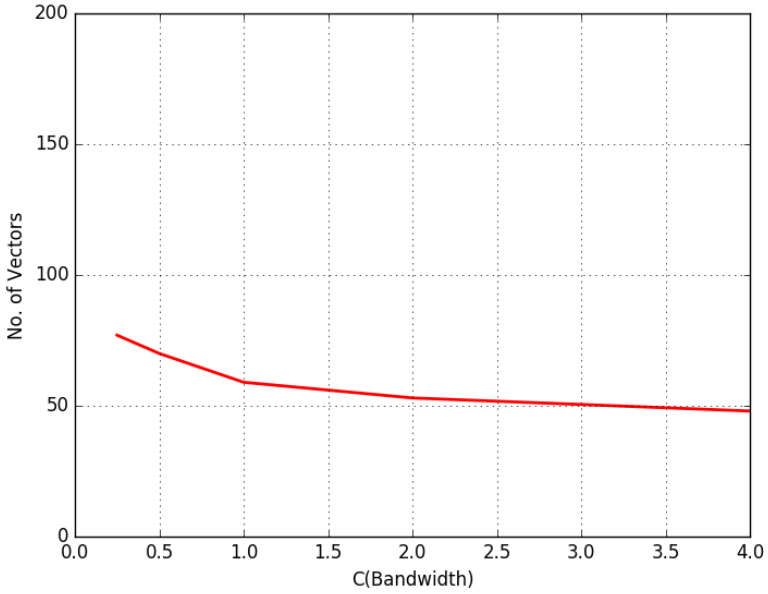
The training error found in this case ranged from 0% to 5% when the tau was changed from 0.01 to 0.5(which is very less than the simple logistic regression whose results were around 15% for 10 iteration and was 4% after 1000 iterations)

3) In this question i have used the scikit toolkit for SVM implementation available in http://scikit-learn.org/stable/modules/svm.html

Training the **Linear SVM** implementation using the binary data set. For splitting the training data into training and development data I have taken k = 5 in the k fold.

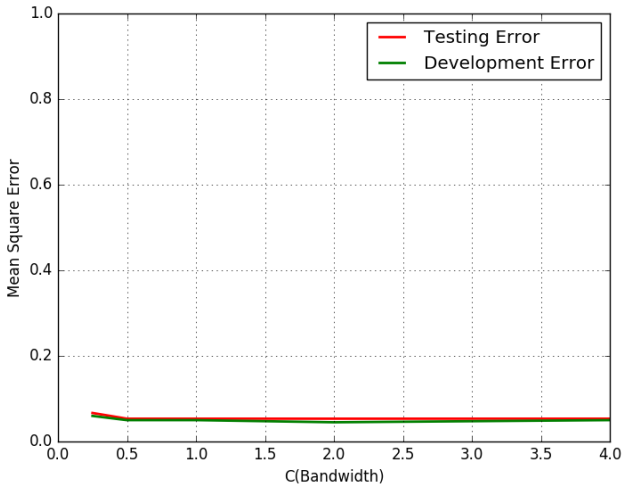
Error Rate Plot for Development and Testing Data Set with changing C values:

Number of Support Vectors used for each models:

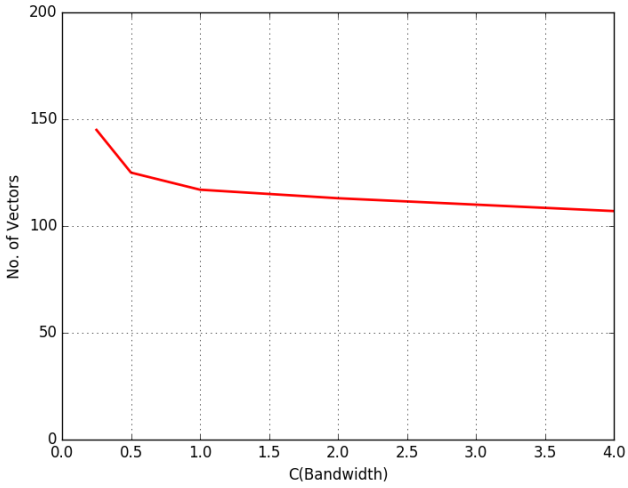


Training the **RBF SVM** implementation using the binary data set. For splitting the training data into training and development data I have taken k = 5 in the k fold.

Error Rate Plot for Development and Testing Data Set with changing C values for Tau = 0.25

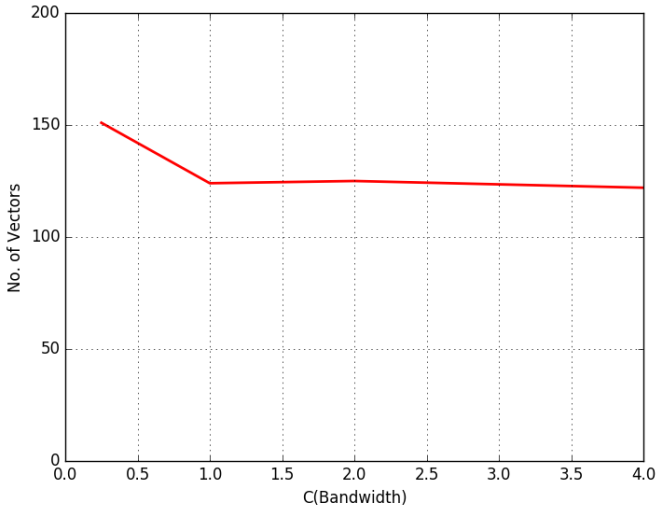


Number of Support vectors for different value of C for tau = 0.25

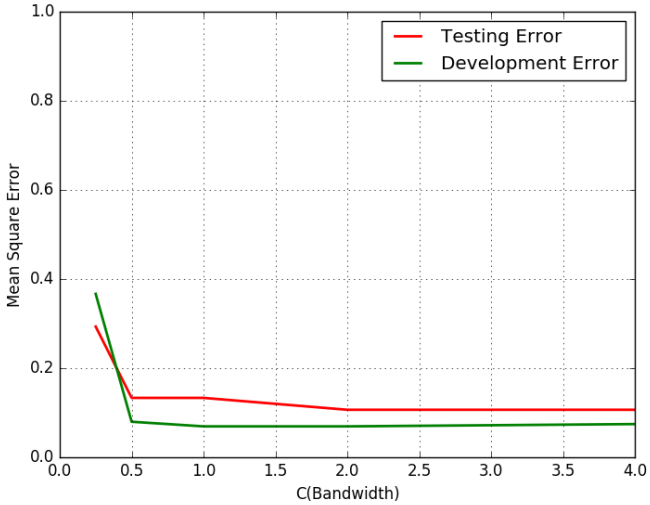


Error Rate Plot for Development and Testing Data Set with changing C values for Tau = 0.5

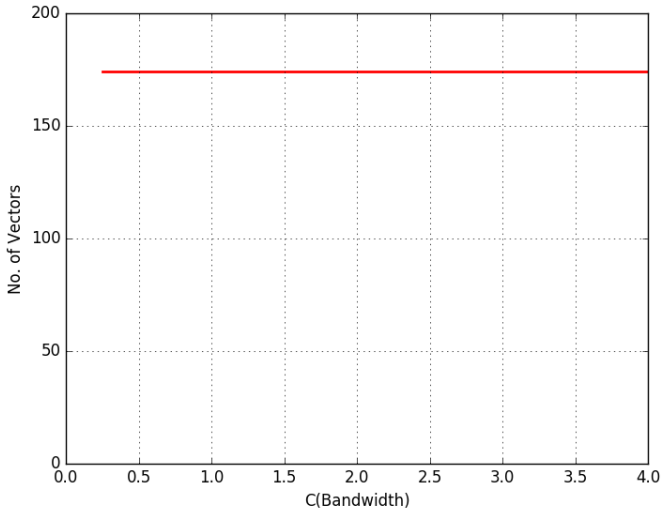
Number of Support vectors for different value of C for tau = 0.5



Error Rate Plot for Development and Testing Data Set with changing C values for Tau = 1

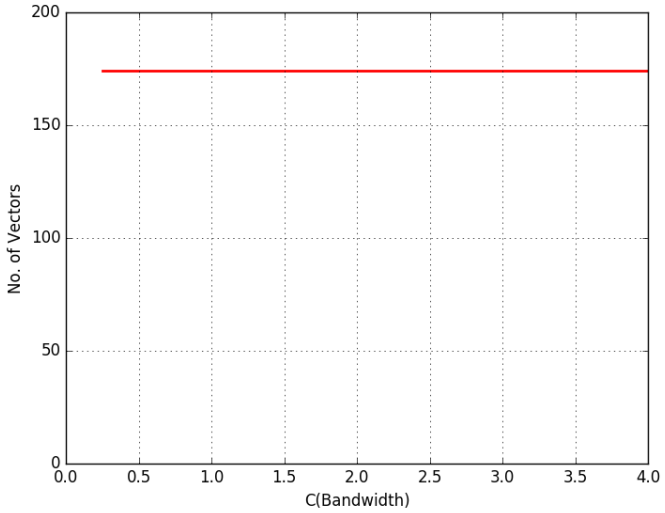


Number of Support vectors for different value of C for tau = 1

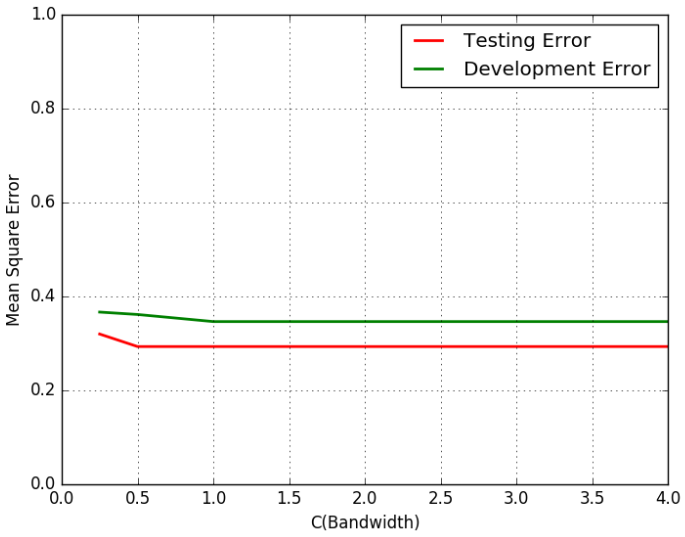


Error Rate Plot for Development and Testing Data Set with changing C values for Tau = 2

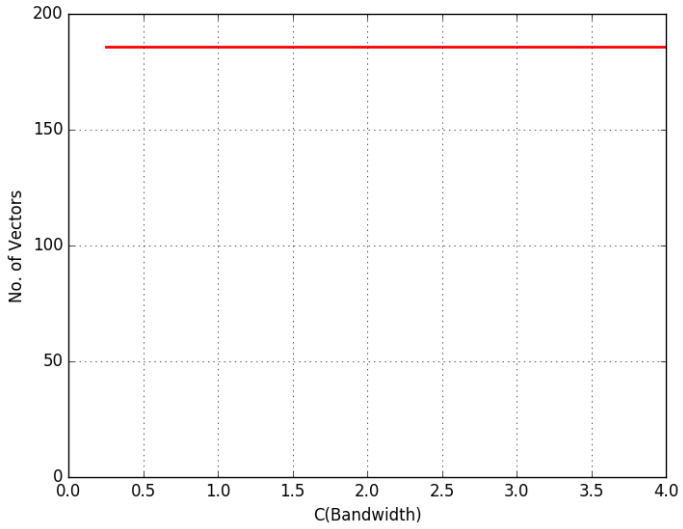
Number of Support vectors for different value of C for tau = 2



Error Rate Plot for Development and Testing Data Set with changing C values for Tau = 4



Number of Support vectors for different value of C for tau = 4



3)The C parameter tells the SVM optimization how much it want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller- margin hyperplane if that hyperplane does a better job of getting all the training point classified correctly. And a very small value of C will cause the optimizer to look for a large-margin separating hyperplane, even if that hyperplane misclassifies more points. The number of Support vectors must be decreased by increasing the value of C, as Large C results in small margin separating hyperplane. So small margin accommodates less Support vectors and vice versa.

4)Interesting Observations:

1) For very small C, there should be higher chances of misclassification even if the training data is linearly separable. It was observed in both the Linear and RBF SVM that as the C value was increased the number of Support vectors reduces.
2) In the case of RBF SVM it was observed that as the Tau value was increased the number of support vectors was less impacted by the C parameters. The behavior of the model is very sensitive to the Tau parameter. If the Tau is too large, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization i.e. C will be able to prevent overfitting.