<u>**INDIAN INSTITUTE OF TECHNOLOGY DELHI**</u>

ELL784 Introduction to Machine Learning
Assignment-3 Report

Topic: Regression and SVMs

Submitted To:

Prof. Sumatra Dutta Roy

By:

T A Pavan Kumar (2019BSZ7501)

Padala Suneel (2019EET2790)

Rohit Joshi (2019BSY7504)

Polynomial Curve Fitting:

- **Objective:** Learn a Polynomial function which maps the input variable **X** to continuous target variable **Y**.

Huber Cost Function:

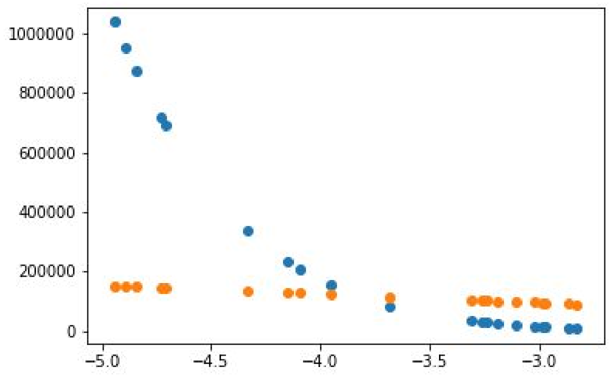
We considered Huber loss Function, which contains both mean square error and mean absolute error. Till delta = 50 (error), loss function uses absolute error function, after delta becomes less than 50, it uses mean square error because absolute error function is not not differentiable at global minimum.

$$loss = \begin{cases} \frac{1}{2} * (x-y)^2 & if\ (|x-y| \leq \delta) \\ \delta * |x-y| - \frac{1}{2} * \delta^2 & otherwise \end{cases}$$

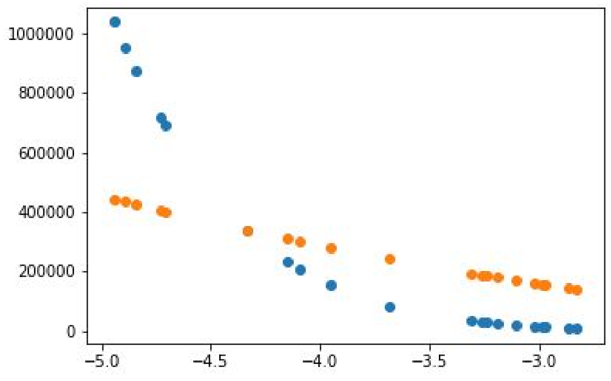
Note: Variance is Normalized

- **Curve fitting for First 20 Points:**

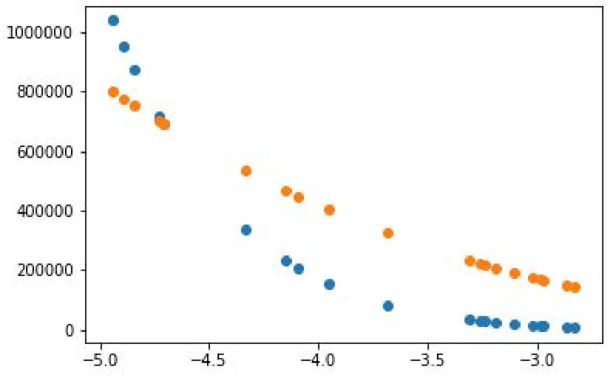
- Degree = 1, alpha = 0.01
- Noise variance = 1.1699678e-10



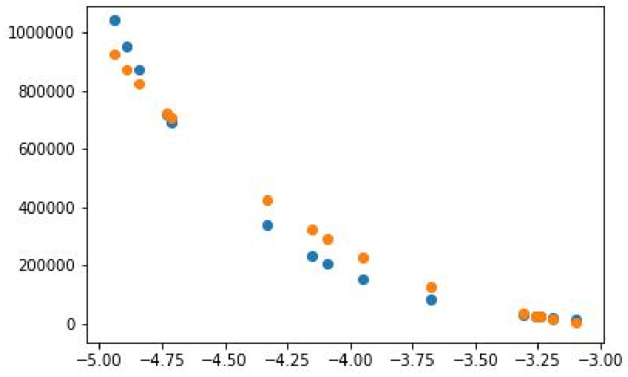
- Degree = 2 alpha = 0.01
- Noise Variance = 2.09316666e-10



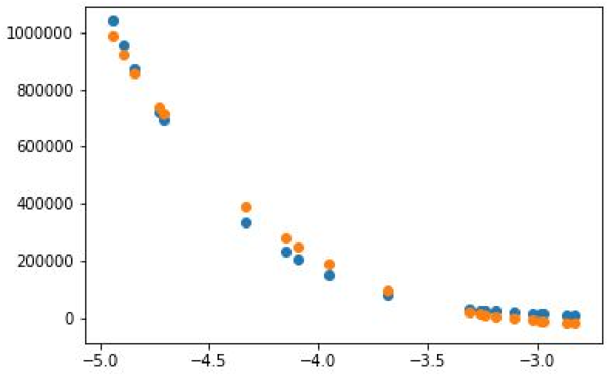
- Degree = 3 alpha = 0.01
- Noise Variance=1.0078693869e-09



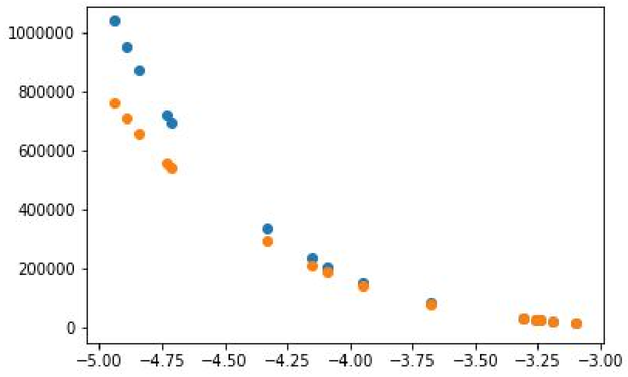
- Degree = 4 alpha = 0.001
- Noise Variance = 4.608632249e-09



- Degree = 5 alpha = 0.0001
- Noise Variance = 9.90292212e-09



- Degree = 6 alpha = 0.0001
- Noise Variance = 3.614914722e-08



- From The above results, Noise variance is increasing, so we can conclude that model is being over fitted.
- We got The Training error at 4rd degree polynomial = 0.30876 and test error is 0.27097
- We got The training error at 5th degree polynomial = 0.25809 and test error is 0.26302
- Since the test error increased at 5th degree polynomial, model started being over fitted.

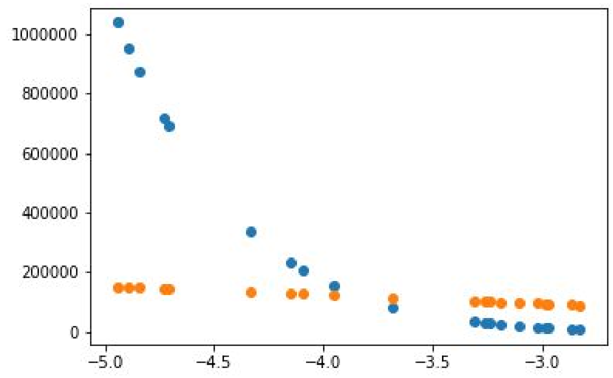
<u>Regularization:</u> Regularization in machine learning is the process of regularizing the parameters that constrain, regularizes, or shrinks the coefficient estimates towards zero. It avoids the model to become overfitted.

We took **L1 Regularization** because the target values are very large(in millions), If we take L2 regularization, then it will add more bias, so the model will go in under fitting.

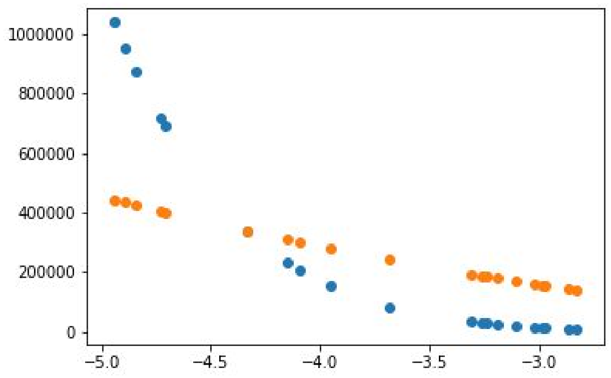
Regularization Parameter: We took **LAMBDA = 0.05,** due to the large error value. If we take greater lambda value, then it will add more bias.

Cost Function = 0.5 * (Y_TRUE-Y_PRED)^2 - delta*(Y_TRUE-Y_PRED) - 0.5*delta^2 + Lambda*Sigma(Theta)

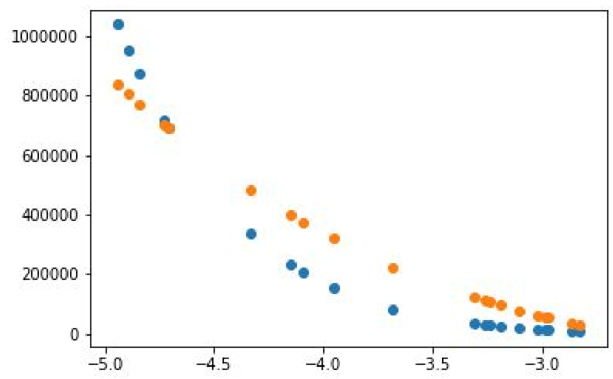
- Degree = 1, alpha = 0.01
- Noise variance =1.161222955e-10



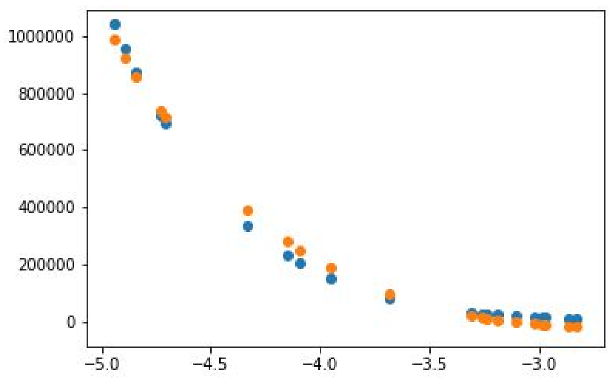
- Degree = 2, alpha = 0.01
- Noise variance = 2.092954732e-10

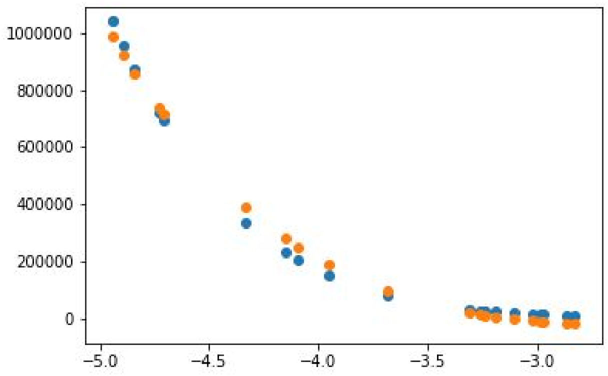
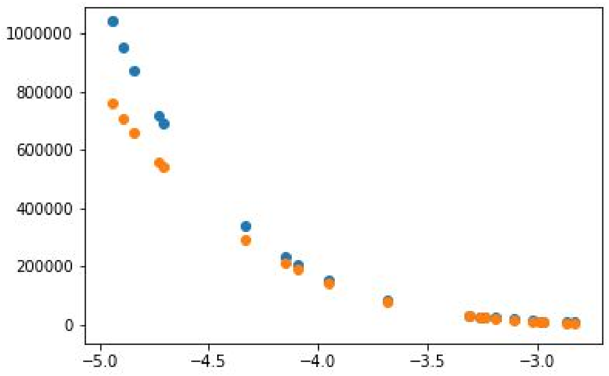


- Degree = 3, alpha = 0.01
- Noise variance =1.007569989e-09



- Degree = 4, alpha = 0.001
- Noise variance = 4.60812061e-09



• Degree = 5, alpha = 0.0001 • Noise variance = 9.902486995e-09 	• Degree = 6, alpha = 0.00001 • Noise variance =3.614904521e-09 

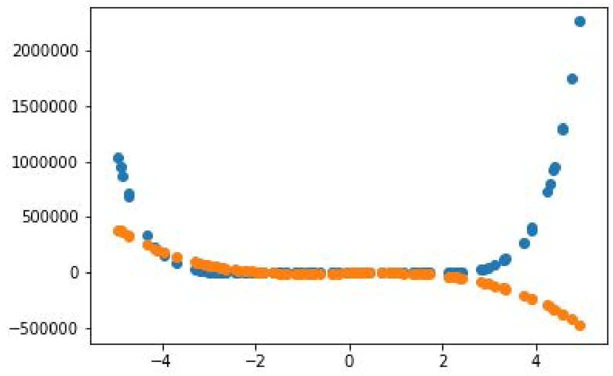
- Training data is 0 to 14 (-4.94 to -3.19) and testing data is from 15 to 19 (-3.10 to 2.83)
- The noise variance which we obtained without regularization is high compared to with regularization. SO we can confirm that regularization term helped the model to not become overfitted.

- **<u>RESULT For 1st 20 Points:-</u>**

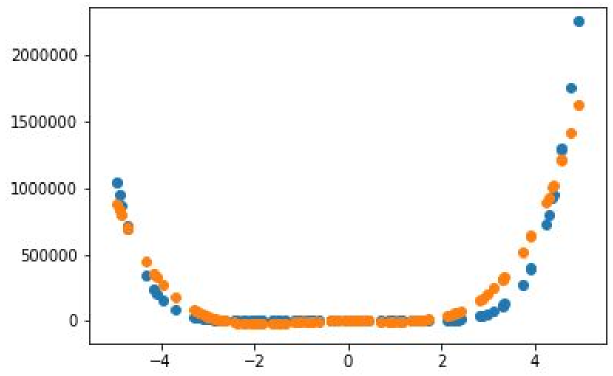
- **Therefore 4rd Degree Polynomial is Best fit**
- **Underfit model is 1st degree(linear)**
- **Overfit model is having degree greater than equal 5**

- **<u>Curve Fitting for All 100 Points:</u>**

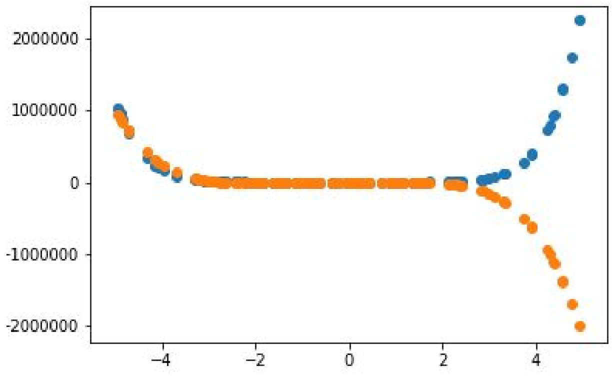
- Degree = 3, alpha = 0.001
- Noise Variance = 9.553190821e-11



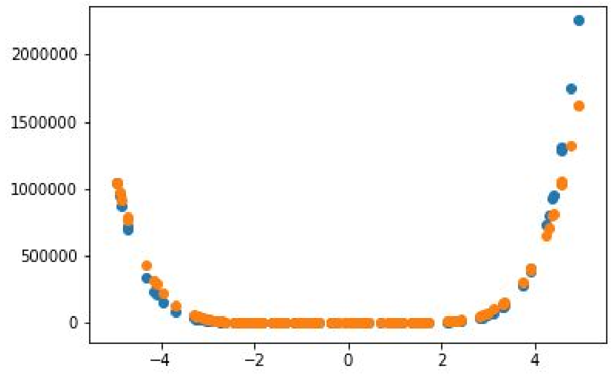
- Degree = 4, alpha = 0.001
- Noise Variance = 9.678592684e-10



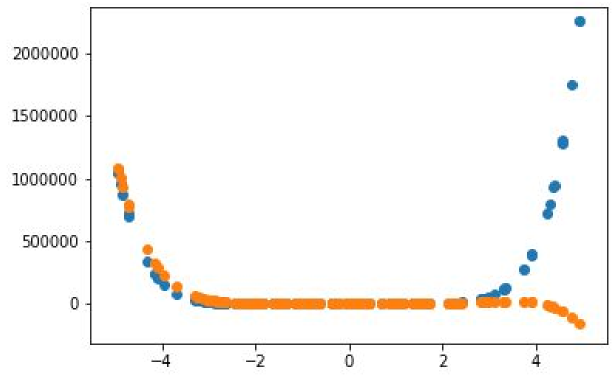
- Degree = 5, alpha = 0.0001
- Noise Variance = 1.613132593e-09



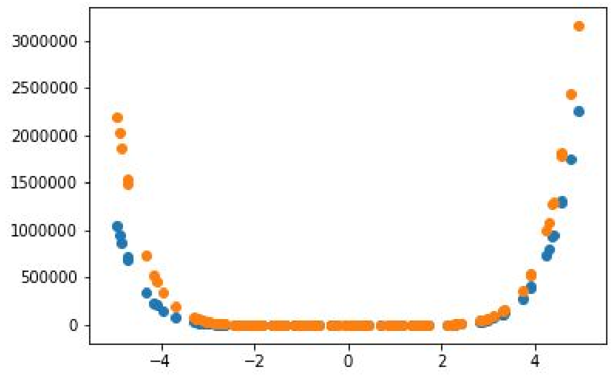
- Degree = 6, alpha = 0.00001
- Noise Variance = 2.87540784e-09



- Degree = 7, alpha = 0.00001
- Noise Variance = 9.613132593e-08

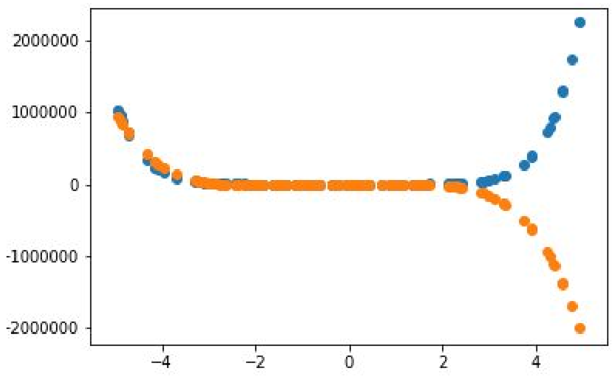


- Degree = 8, alpha = 0.00001
- Noise Variance = 2.29424178e-08

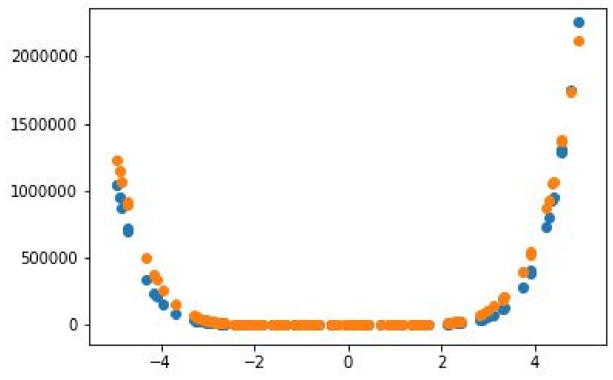


- **<u>After Adding Regularization Parameter:</u>**

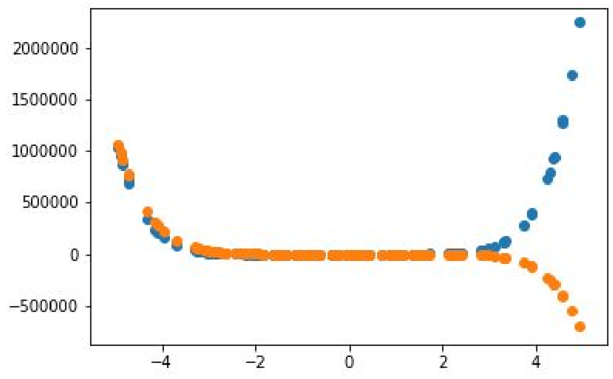
- Degree = 5, alpha = 0.0001
- Noise Variance = 0.993086395e-09



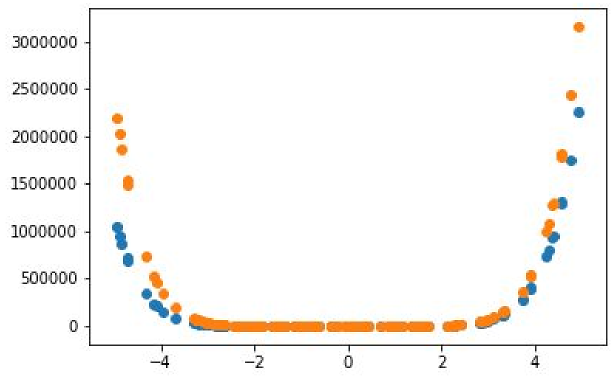
- Degree = 6, alpha = 0.00001
- Noise Variance = 2.01263485e-09



- Degree = 7, alpha = 0.00001
- Noise Variance = 7.057548211e-08



- Degree = 8, alpha =0.00001
- Noise Variance = 1.437918596e-08



- Training data is 0 to 79 and testing data is from 80 to 99
- The noise variance which we obtained without regularization is high compared to with regularization. SO we can confirm that regularization term helped model to not become overfitted.

- **<u>RESULT For All The Points:-</u>**

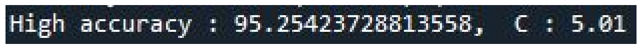
- **Therefore 6th Degree Polynomial is Best fit**
- **Underfit model is 1st to 3rd degree(linear)**
- **Overfit model is having degree greater than equal 7**

-:SVM:-

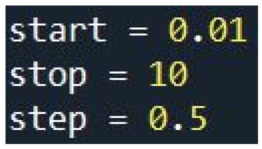
Solution 2A

- For the first part of the question a Linear kernel is used.
- The ‘libsvm’ is installed using ‘pip’ command and the libsvm version used is 3.23.0.4
- The C Value is varied from C > 0 , which is 0.01 till 10 in steps of 0.3:
  C is picked from a set 0.01, 0.31, 0.61…. 9.91 (<10)
- The number of iterations will be approximately (9.91 – 0.01) / 0.3 ~= 33
- This can be configured before running the application here :

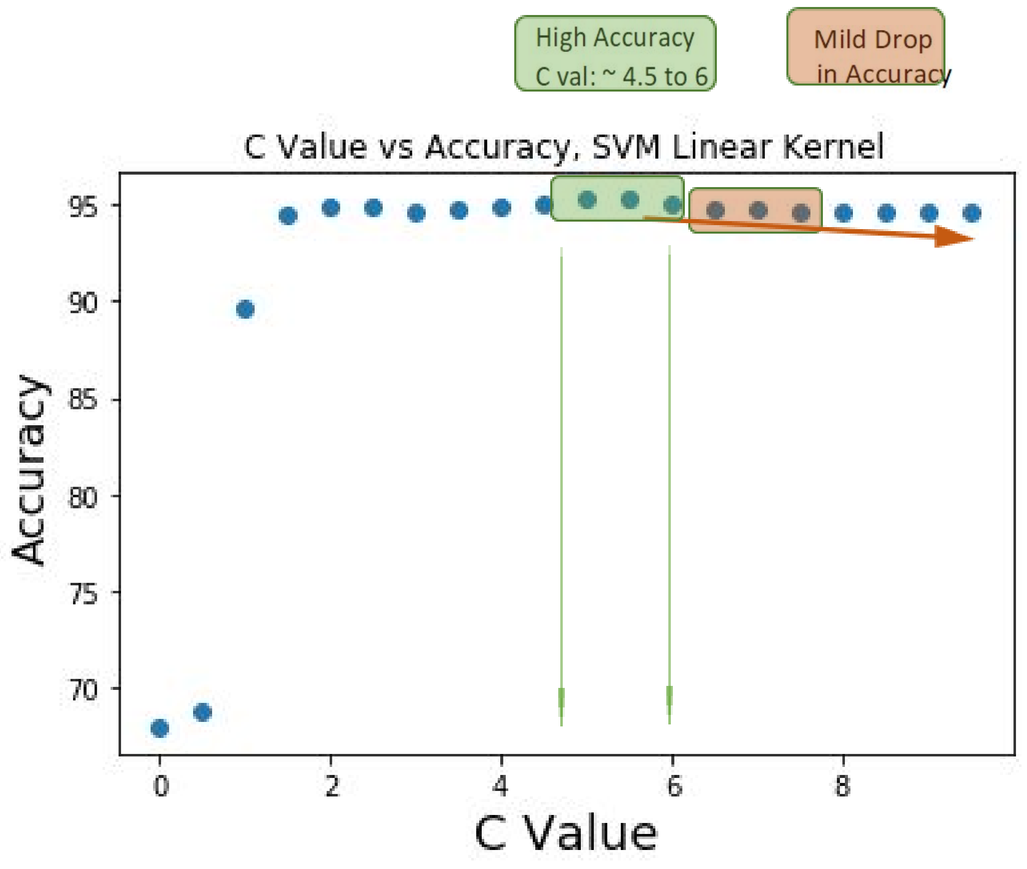
Observation 1:

- 

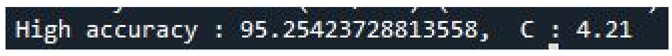
```
High accuracy : 95.25423728813558,  C : 5.01
```

- C = 5.01, Accuracy = 95.254%
- For C Values
- 

```
start = 0.01
stop = 10
step = 0.5
```



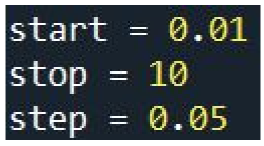
Observation 2:



```
High accuracy : 95.25423728813558,  C : 4.21
```

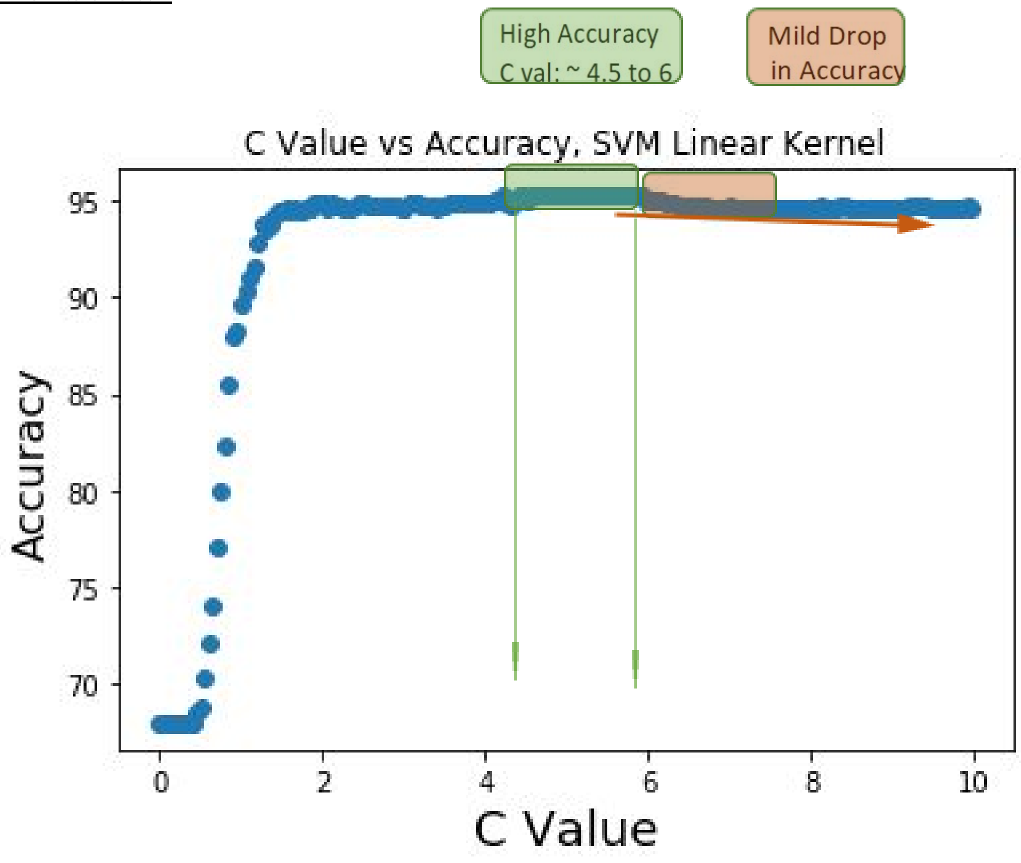
C = 4.21, Accuracy = 95.254%

For C values :



```
start = 0.01
stop = 10
step = 0.05
```

Plot below:



- The final predicted results are written to ‘SVM_Linear.txt’ that will be in the same folder. A result sheet for Observation 1 is attached here.
- **SVM_Linear.txt**

SOLUTION 2B:

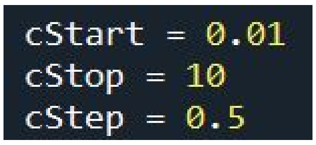
- The SVM kernel used here is a Gaussian kernel with 5 fold cross validation.
- This is achieved by using both C and Gamma parameters of the libsvm.
- Gamma here is 1 / (2 (sigma)^2

Plot for the final set of 5 fold cross validation is show below for :

- All the graphs are plotted together for C value vs Accuracy. The gamma vs Accuracy is explained on the same diagram.

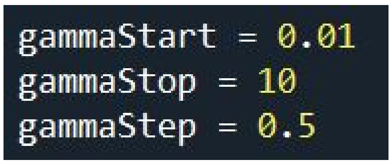
Observation 1:

C Value Range:



```
cStart = 0.01
cStop = 10
cStep = 0.5
```

Gamma Range:

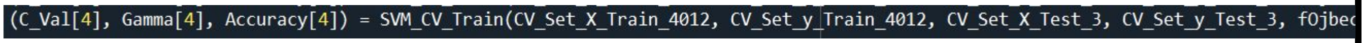


```
gammaStart = 0.01
gammaStop = 10
gammaStep = 0.5
```

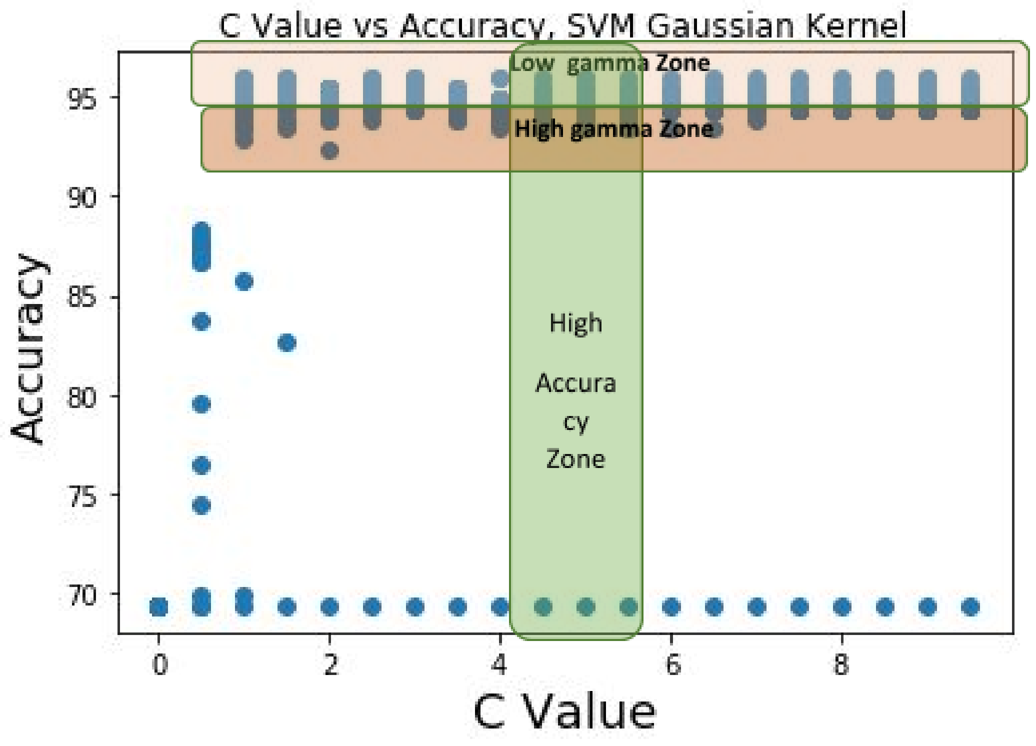
Plot for C value vs Accuracy for :

Cross validation train set Fold {4,0, 1, 2} , and Test set Fold 3

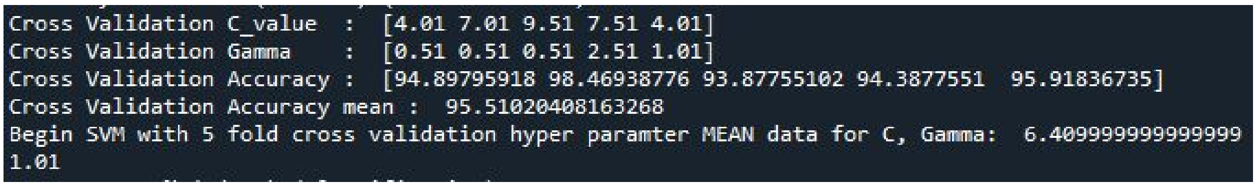
Plot for other validation sets are also provided when the code is run.



```
(C_Val[4], Gamma[4], Accuracy[4]) = SVM_CV_Train(CV_Set_X_Train_4012, CV_Set_y_Train_4012, CV_Set_X_Test_3, CV_Set_y_Test_3, fOjbec
```



Cross validation accuracy with C Value and Gamma for 5 fold cross validation:



```
Cross Validation C_value   :  [4.01 7.01 9.51 7.51 4.01]
Cross Validation Gamma     :  [0.51 0.51 0.51 2.51 1.01]
Cross Validation Accuracy :  [94.89795918 98.46938776 93.87755102 94.3877551  95.91836735]
Cross Validation Accuracy mean :  95.51020408163268
Begin SVM with 5 fold cross validation hyper paramter MEAN data for C, Gamma:  6.409999999999999
1.01
```

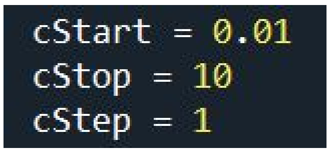
C value 6.40, Gamma value 1.01 => Sigma value = 0.7035

Accuracy = 95.510

- Accuracy is slightly improved compared to Linear kernel

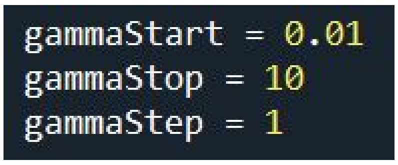
Observation 2:

For C Value range:



```
cStart = 0.01
cStop = 10
cStep = 1
```

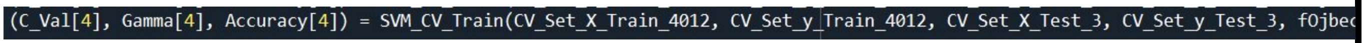
For Gamma value range:



```
gammaStart = 0.01
gammaStop = 10
gammaStep = 1
```

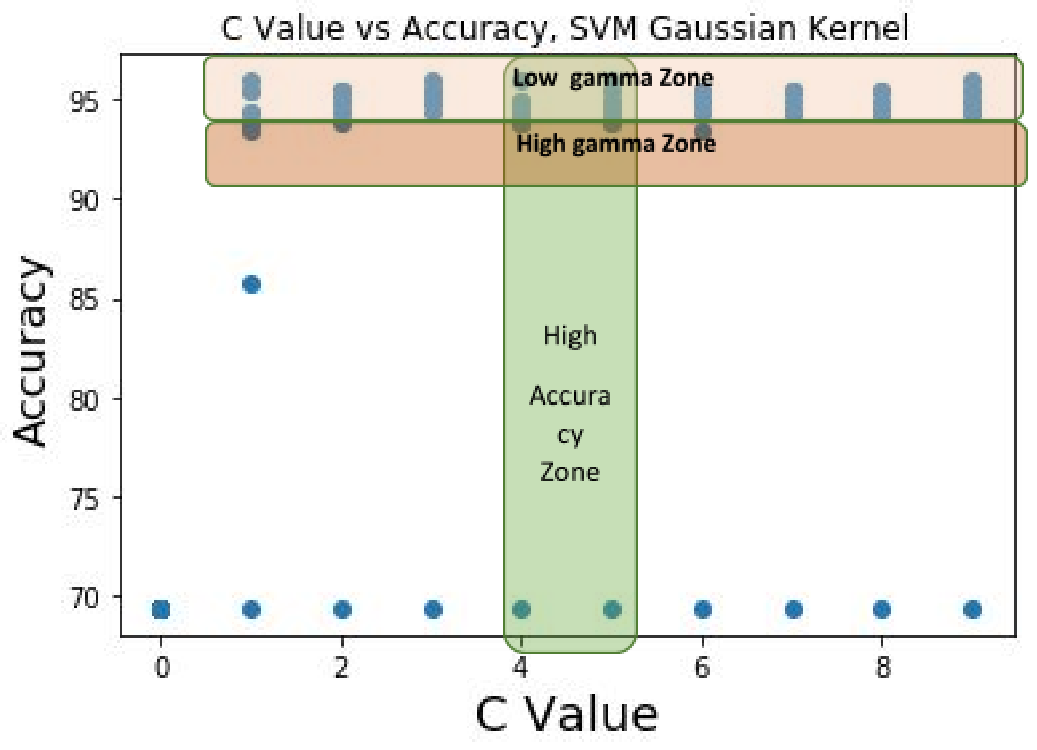
Plot for C value vs Accuracy for :

Cross validation train set Fold {4,0, 1, 2} , and Test set Fold 3



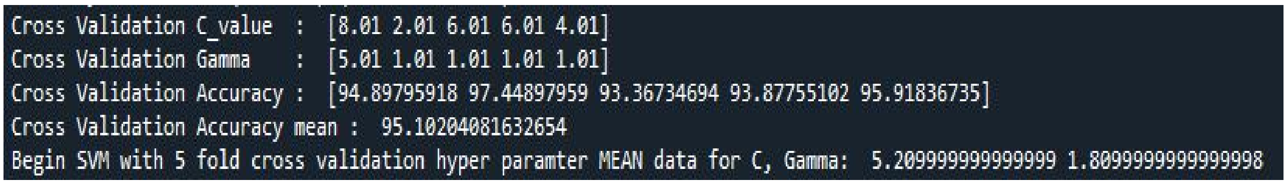
```
(C_Val[4], Gamma[4], Accuracy[4]) = SVM_CV_Train(CV_Set_X_Train_4012, CV_Set_y_Train_4012, CV_Set_X_Test_3, CV_Set_y_Test_3, fOjbec
```

Plot for other validation sets are also provided when the code is run:



C value 5.20, Gamma value 1.01 => Sigame value = 0.7035

Accuracy : 95.102



```
Cross Validation C_value  :  [8.01 2.01 6.01 6.01 4.01]
Cross Validation Gamma    :  [5.01 1.01 1.01 1.01 1.01]
Cross Validation Accuracy :  [94.89795918 97.44897959 93.36734694 93.87755102 95.91836735]
Cross Validation Accuracy mean :  95.10204081632654
Begin SVM with 5 fold cross validation hyper paramter MEAN data for C, Gamma:  5.209999999999999 1.8099999999999998
```

- The predicted values are printed to SVM_Gaussian.txt.
- A sample for observation 2 is uploaded here.
- **SVM_Gaussian.txt**