CS 584
MACHINE LEARNING
ASSIGNMENT 1

by
Saurabh Katkar
A20320336

Collaborators: None

Code created using Rstudio version 0.98.1062

Parametric Regression

Introduction

Regression is the process of fitting models to data. In Parametric regression, regression estimates the parameters from the data. In a linear model, estimation is based on methods from linear algebra that minimize the norm of a residual vector. In a non-linear(polynomial) model, estimation is based on search methods from optimization that minimize the norm of a residual vector.

The advantage of the parametric approach is that the model is defined up to a small number of parameters—for example, mean, variance—the sufficient statistics of the distribution. Once those parameters are estimated from the sample, the whole distribution is known. We estimate the parameters of the distribution from the given sample, plug in these estimates to the assumed model, and get an estimated distribution, which we then use to make a decision.

1. The Univariate Dataset

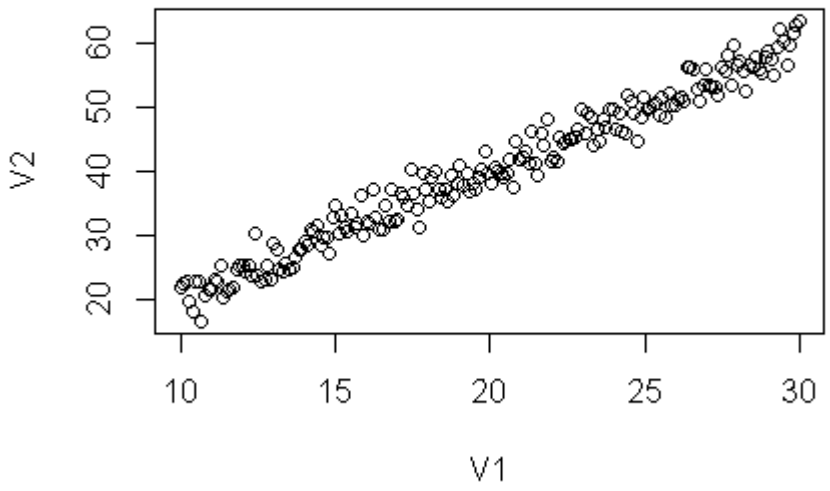
In a univariate dataset the regressed variable is influenced by only one independent variable.
Here we would be using 4 univariate datasets, namely svar-set1.dat, svar-set2.dat, svar-set3.dat and svar-set4.dat .

1.1 Loading the dataset:

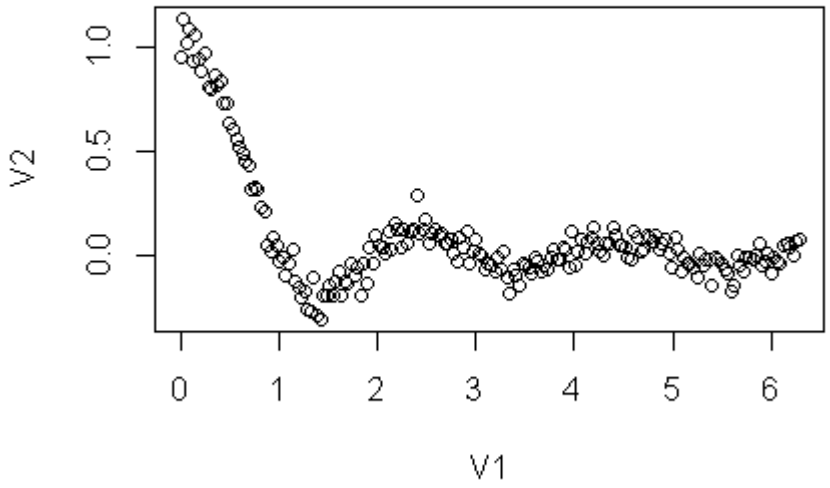
The dataset is loaded into **R** using the function **read.table()**. We can subsequently print the contents of the dataset and plot them on the graph using **print()** and **plot()** respectively.
We then split the dataset into training and testing set in order to test the performance of regression on data of different sizes.

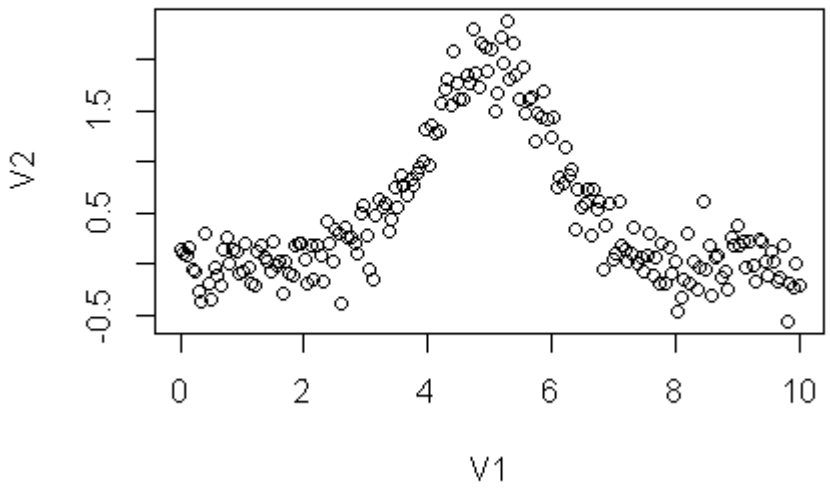
We obtain the following plots of the univariate datasets;

For svar-set1.dat

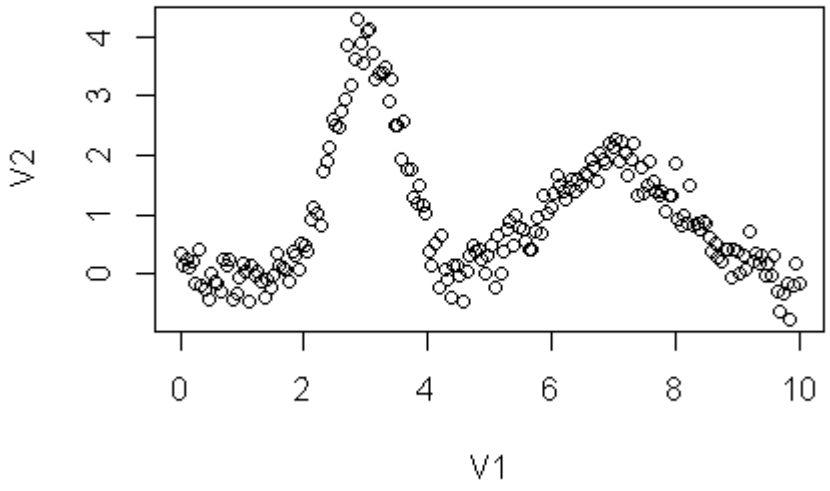


For svar-set2.dat





For svar-set3.dat



For svar-set4.dat

1.2 Performing Linear Regression:

Using linear regression on univariate data, we model the relationship between scalar dependent variable and the single independent variable that is used in the four datasets.

We apply the following linear regression equation for our fitting the training and testing set;

$$\hat{y} = \theta_0 + \theta_1 x$$

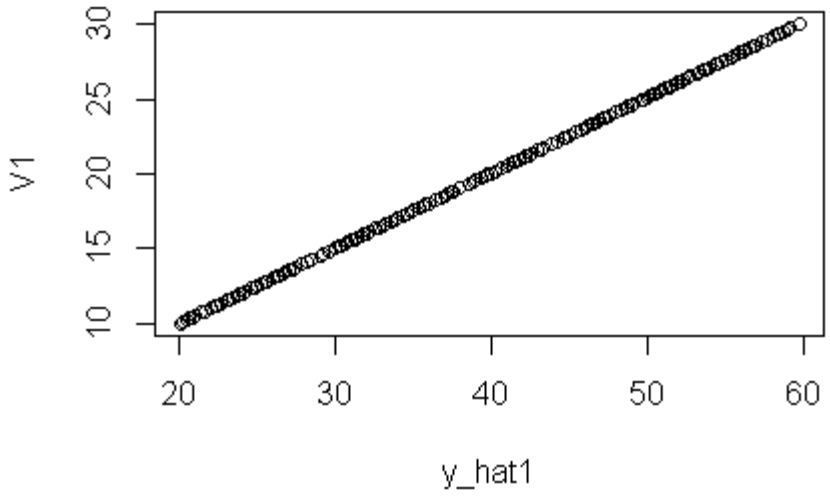
Then we formulate the mean square error;

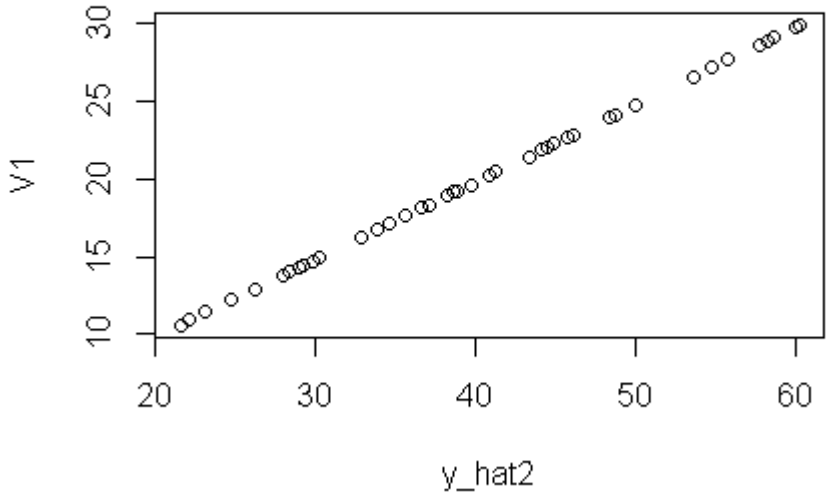
$$\text{MSE} = (\hat{y} - y)^2$$

The plots obtained after applying linear regression are as follows:

For svar-set1.dat

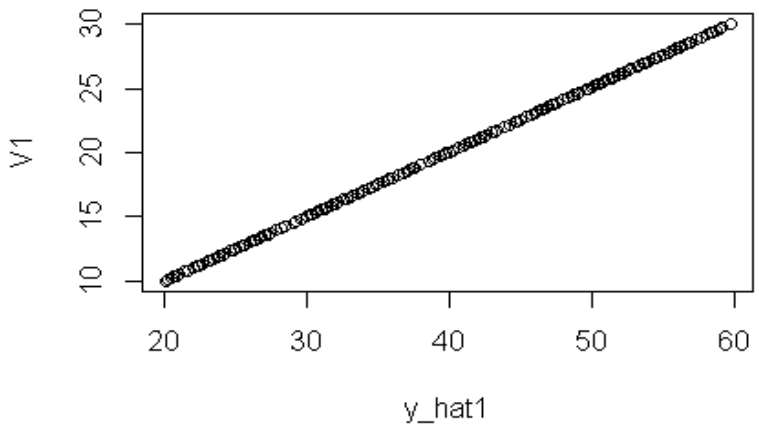
Training plot:



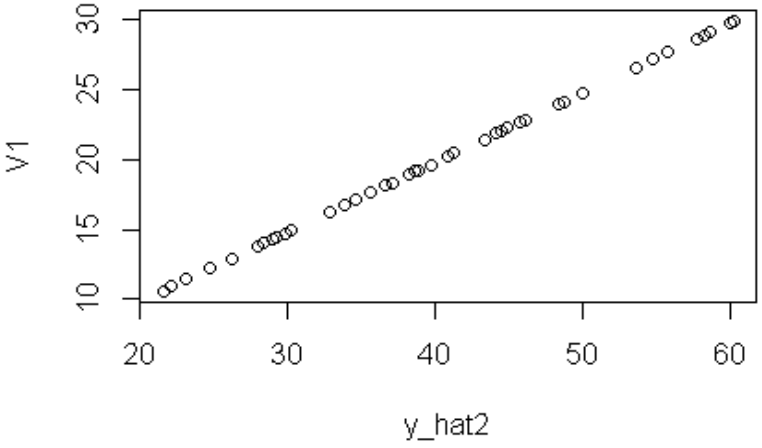


Test Plot:

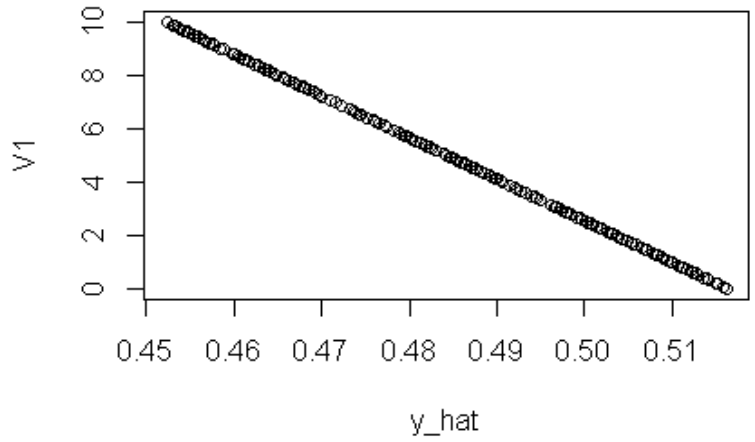
For svar-set2.dat:
Training plot:



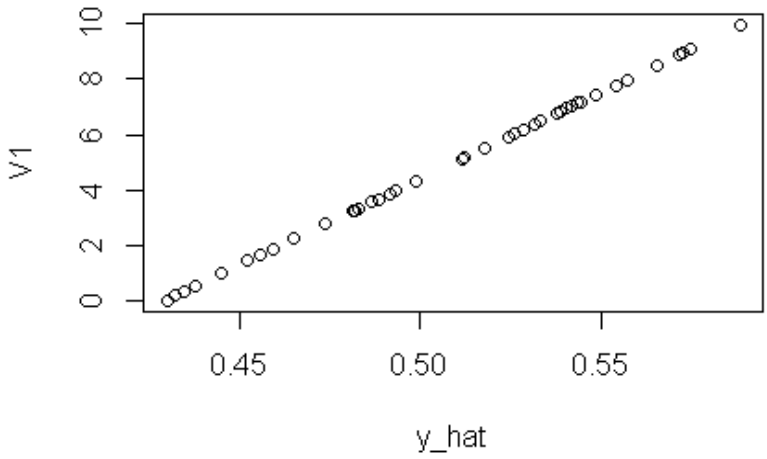
Test plot:



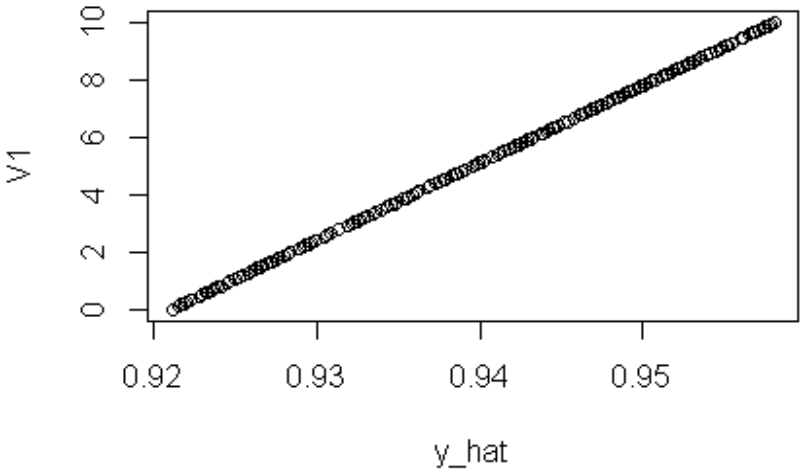
For svar-set3.dat:
Training plot:



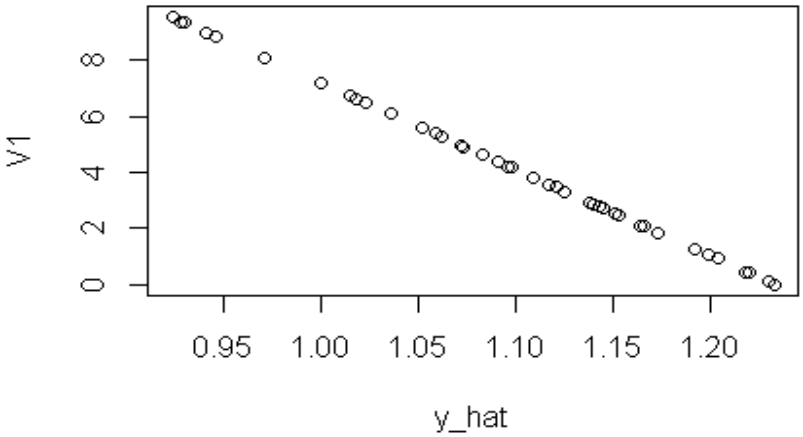
Test Plot:



For svar-set4.dat:
Training plot:



Test plot:



The results for MSE on training and test datasets are as follows:

	training error	test error
dataset1	4.239696	3.984512
dataset2	0.0583	0.0643
dataset3	0.526	0.387
dataset4	1.08	1.67

Applying 10-fold cross validation:

Cross Validation is used to determine how accurately a predictive model, in our case, a linear model, will perform in practice.
The 10-fold cross validation will be implemented as follows:

- Break data into 10 sets of size n/10.
- Train on 9 datasets and test on 1.
- Repeat 10 times and take a mean accuracy.

By applying this technique we obtain the following accuracy;

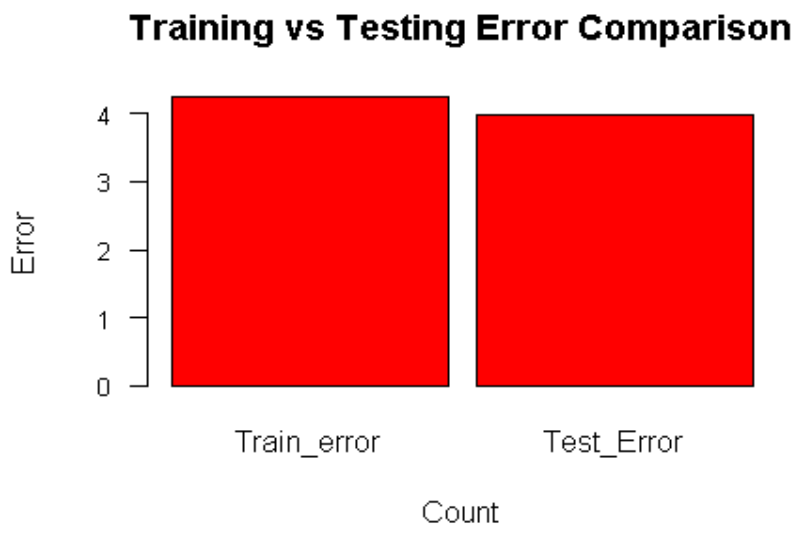
	training error	test error
dataset1	5.07	5.3
dataset2	0.07	0.07
dataset3	0.37	0.37
dataset4	1.35	1.35

Thus the MSE obtained using our linear regression model gives almost identical accuracy and thus performance as compared to the results obtained by 10-fold cross validation.

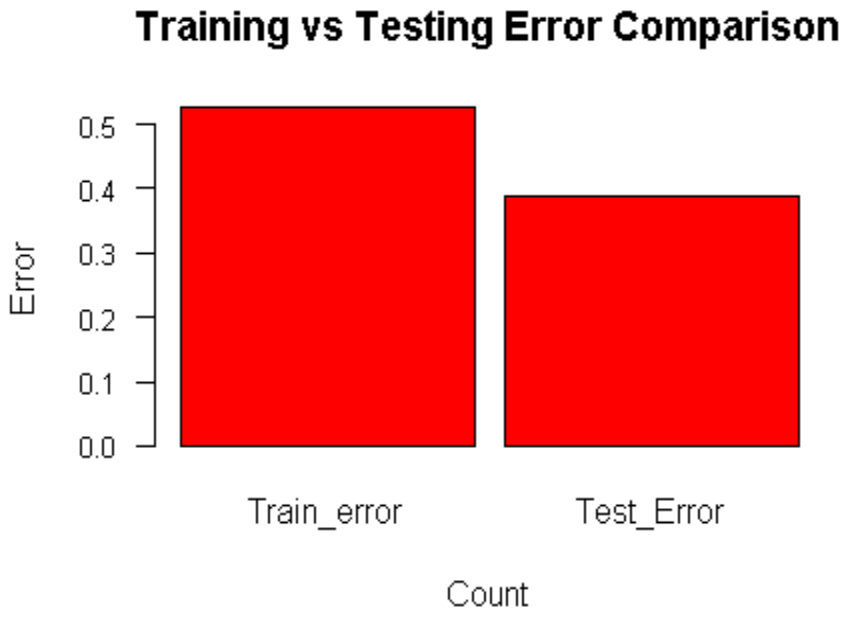
Comparing the errors for the training and test set:

The training vs test datasets error is represented in the form of a barplot as follows;

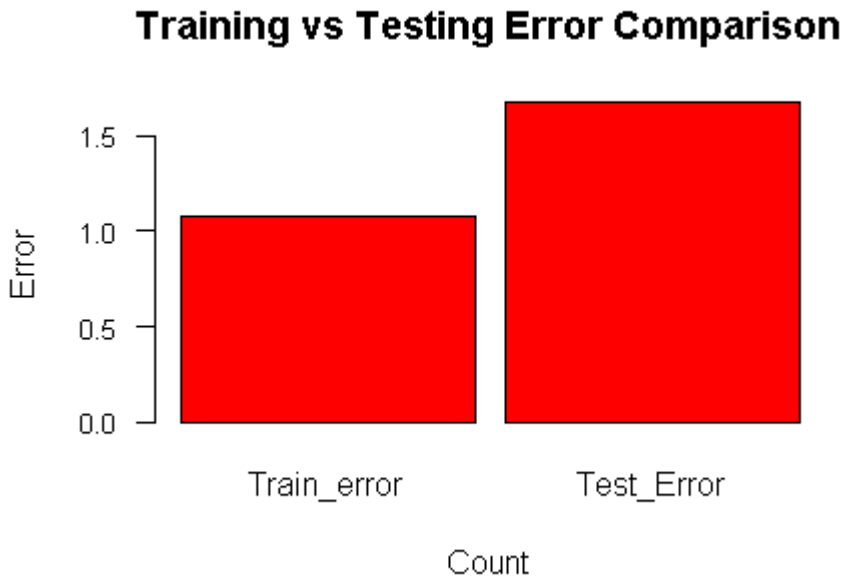
For svar-set1:

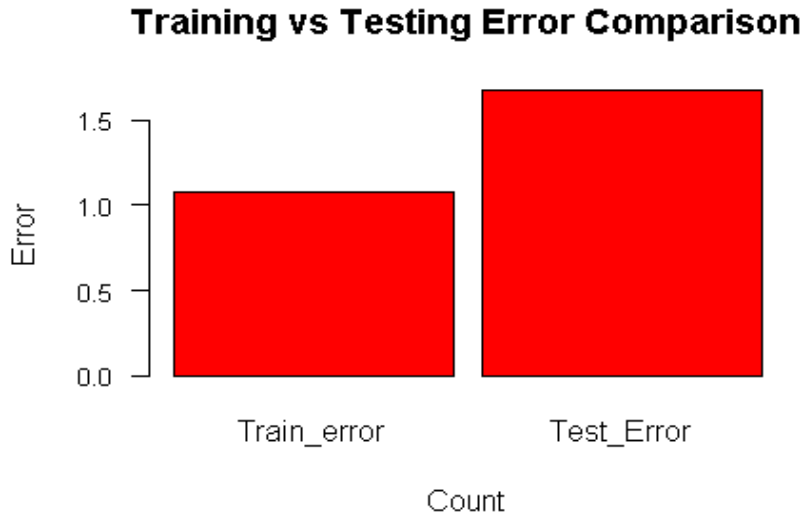


For svar-set2:



For svar-set3.dat:





For svar-set4.dat:

1.3 Performing Polynomial Regression:

Polynomial Regression is based on the same idea as basic linear regression, except that the relationship between the independent and dependent variables is non-linear. However, these regression models are still constructed by a "linear like process", which is to say the independent variables may be thought of algebraic transformations where the new variable is regressed as if it were linear.

In Polynomial Regression we apply the following equation for our fitting the training and testing set;

$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 ..... + \theta_{k-1} x^{k-1} + \theta_k x^k$$

Fitting the curve:
At what degree of the polynomial stop? Depends on the degree of precision that we seek. The greater the degree of the polynomial, the greater the accuracy of the model, but the greater the difficulty in calculating.

Also we should make sure that the we do not overfit or underfit the regression data in the model.
Overfitting results in high testing error, high generalization error and high variance.
Underfitting results in high training error and high bias.

Finding the best polynomial model:

We consider the **Multiple R-Squared** value in order to determine the polynomial model that has the best fit.

For svar-set1, we get;

	Degree	Multiple R^2 (%)
dataset1	2	96.9
	3	97
	4	97

Subsequently for the other datasets we get the following values;

For svar-set2:

	Degree	Multiple R^2 (%)
dataset2	2	51.4
	3	74.4
	4	85.6

For svar-set3:

	Degree	Multiple R^2 (%)
dataset3	2	49.2
	3	49.2
	4	74.6
	5	74.7
	6	87.3
	7	87.6
	8	91.2
	10	91.6

For svar-set4:

	Degree	Multiple R^2 (%)
dataset4	2	22.7
	3	23.5
	4	29.2
	5	33.1
	6	62.4
	7	62.5
	8	79.4
	10	85.1
	12	90.2

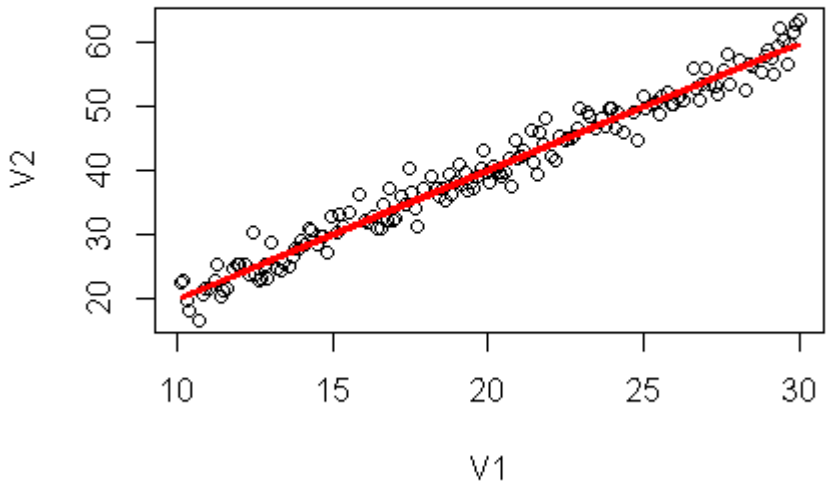
For our stopping criterion, we consider the multiple R^2 values of each polynomial model until there is **not a significant increase** in the subsequent degrees.

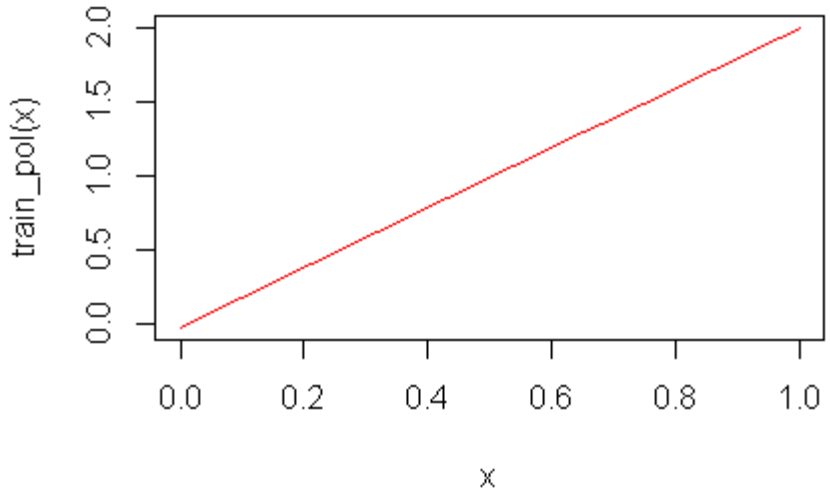
Thus, looking at the table, we take the following polynomial model for each dataset.

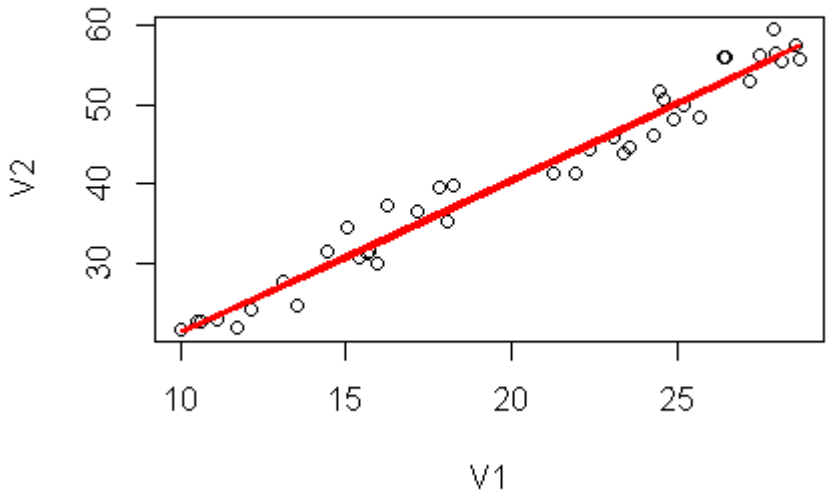
Dataset	Degree
Svar-set1	2
Svar-set2	4
Svar-set3	8
Svar-set4	12

Using **lines** to estimate the curve for the points and then using **curves** we get the following graph:

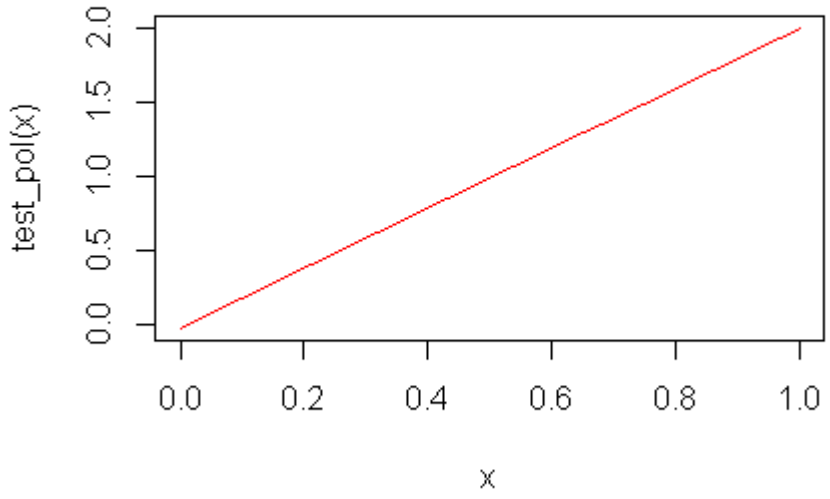
For svar-set1(training set):

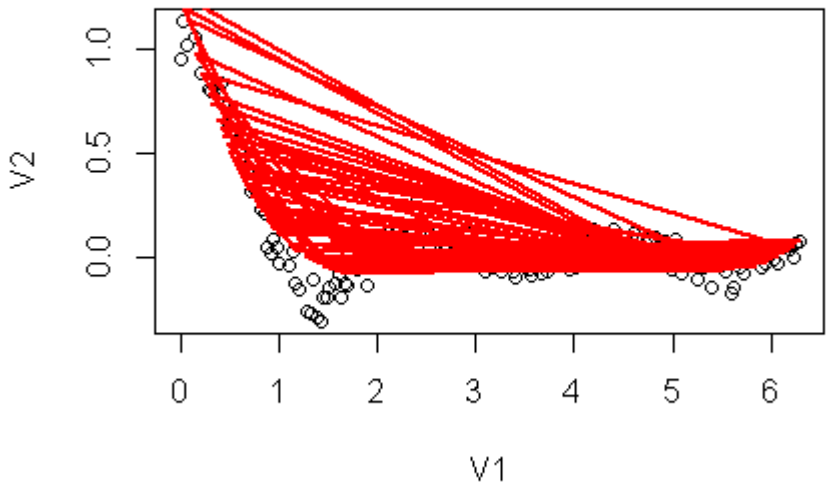




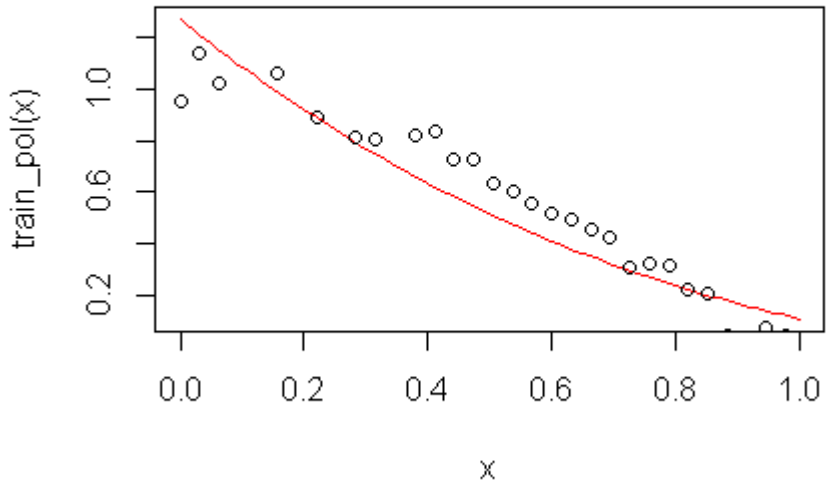


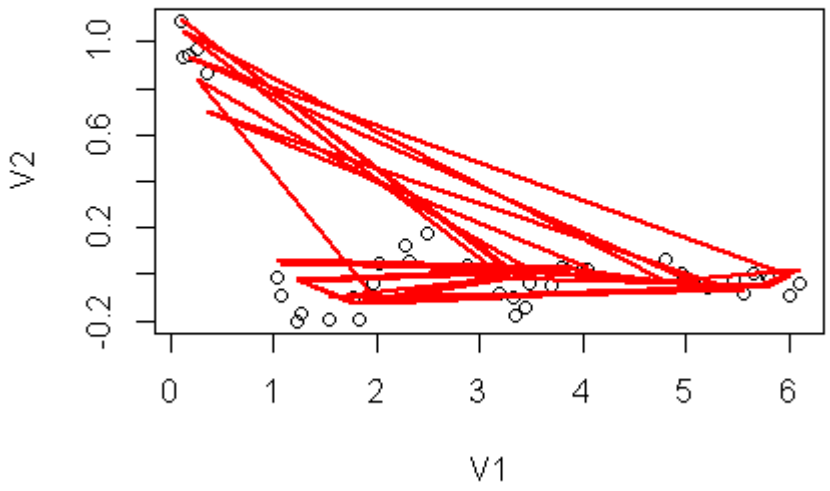
For svar-set1(test set):



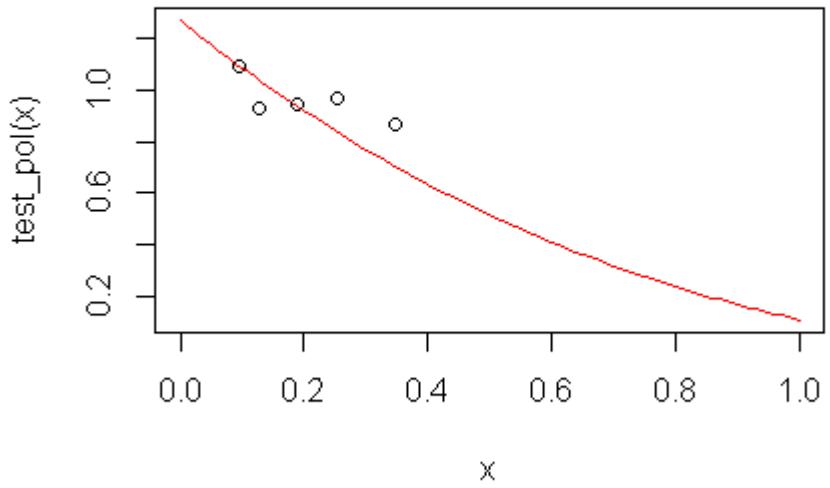


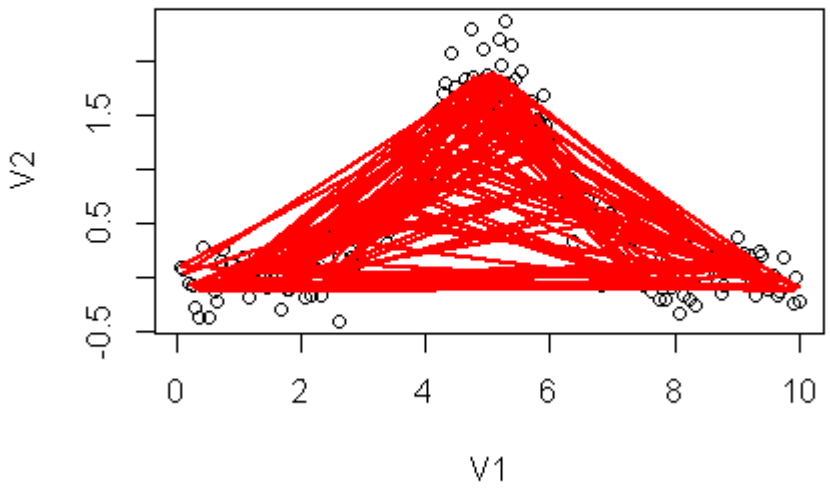
For svar-set2(training set):



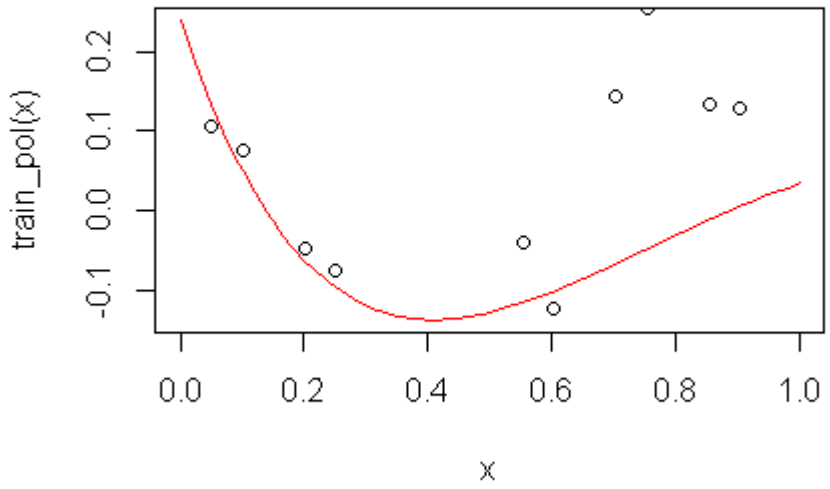


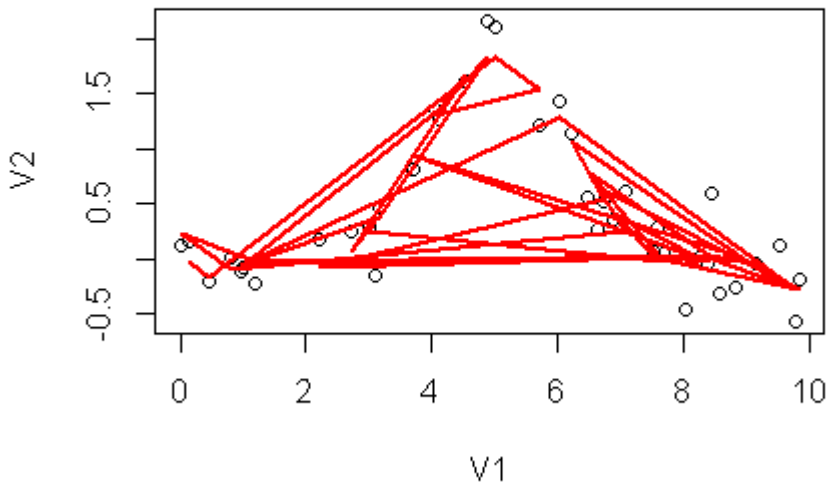
For svar-set2(test set):



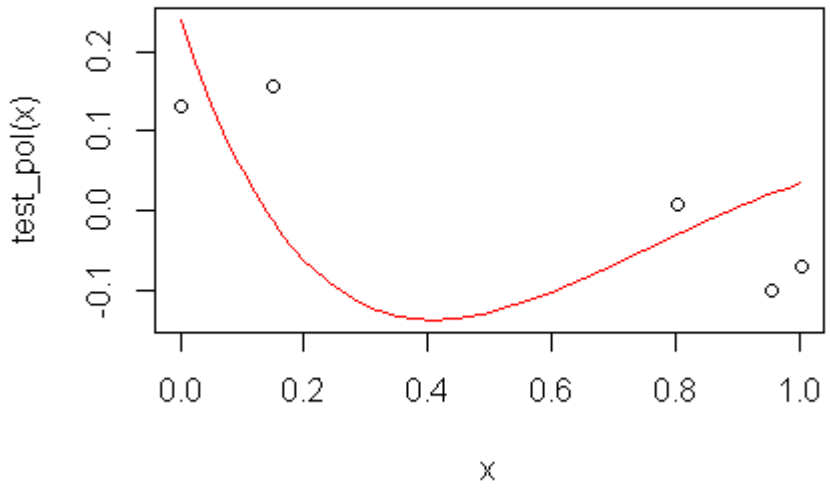


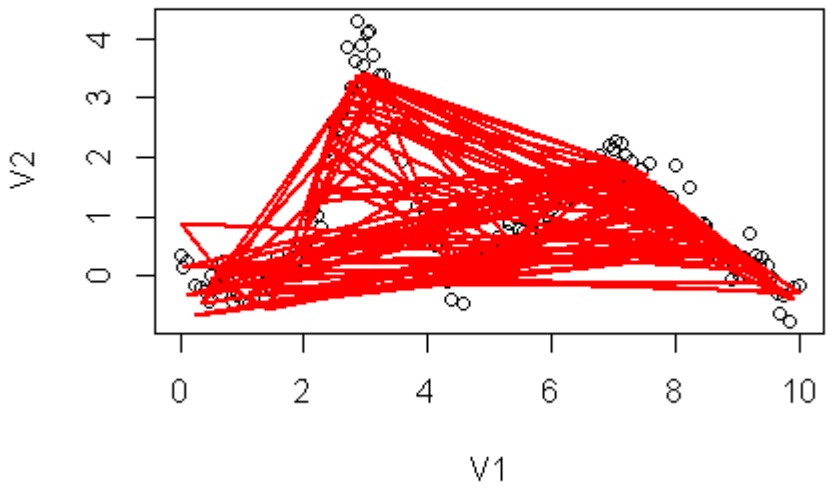
For svar-set3(training set):



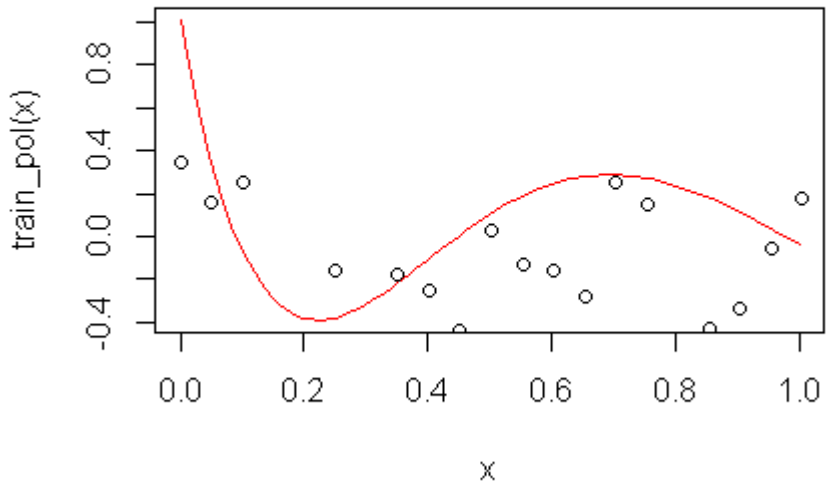


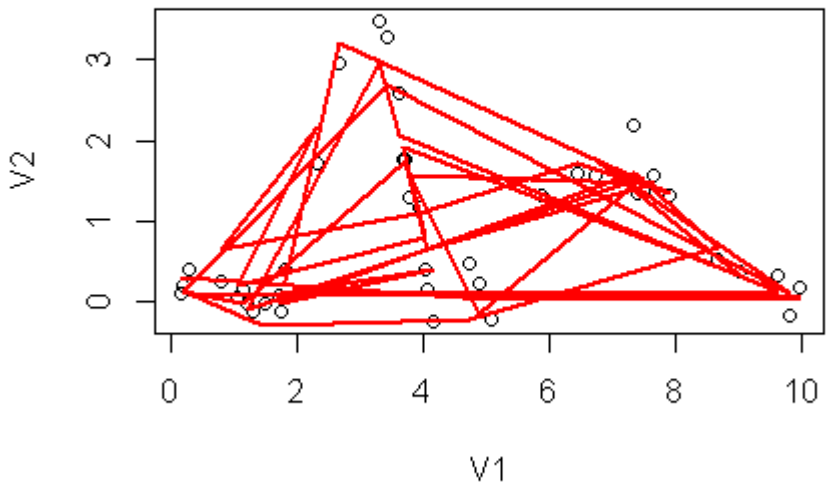
For svar-set3(test set):



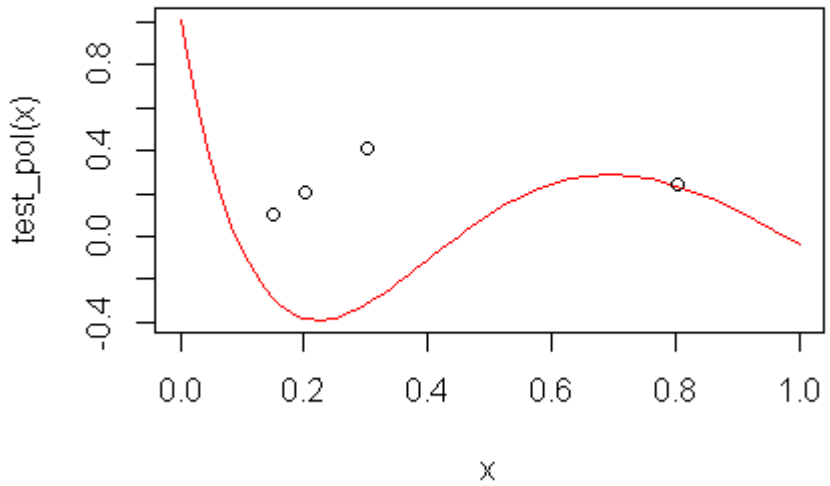


For svar-set4(training set):





For svar-set4(test set):



1.4 Reducing the size of the training set:

We reduce the size of training set to 50% of the dataset and observe the effect on the accuracy of predictions.

Thus the MSE obtained for the training and test sets are as follows:

Split = 50%	training error	test error
dataset1	4.44	4
dataset2	0.0567	0.0624
dataset3	0.569	0.405
dataset4	1.36	1.04

Conclusions for the Singlevariate Dataset:

- There is not much disparity between the errors obtained in the training and test datasets, probably owing to the small size of the dataset.

- For polynomial model, we stopped at a Multiple R^2 value that would give an efficiency of about 85-95% with only minimal increase in efficiency with an addition of degree, this would prevent unnecessary overfitting of the model.

- By reducing the training set to only 50% of the dataset the accuracy and performance difference between training and test sets become negligible.

- Design Issue: The use of lines to depict the fit in the polynomial model has been done because, I could not get the curve to be plotted with all the points in the dataset.

2. The Multivariate Dataset

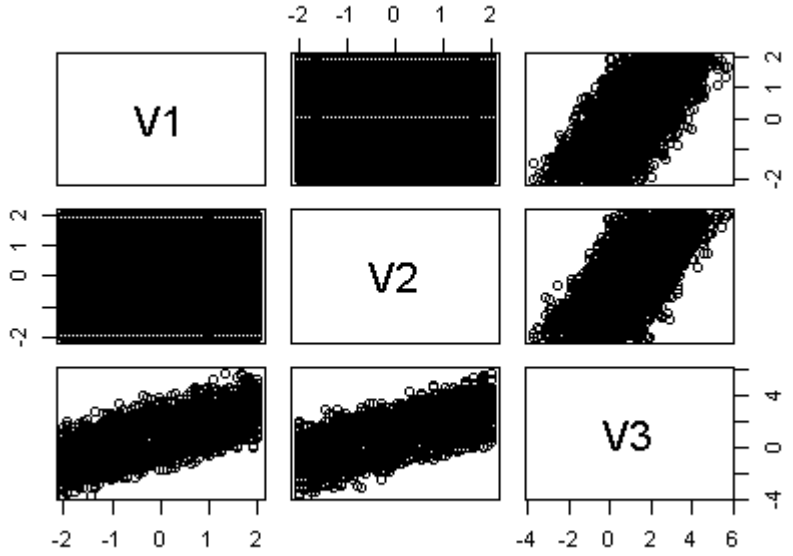
In a multivariate dataset the regressed variable is influenced by two or more independent variables. Here we would be using 4 multivariate datasets, namely mvar-set1.dat, mvar-set2.dat, mvar-set3.dat and mvar-set4.dat .

2.1 Loading the dataset:

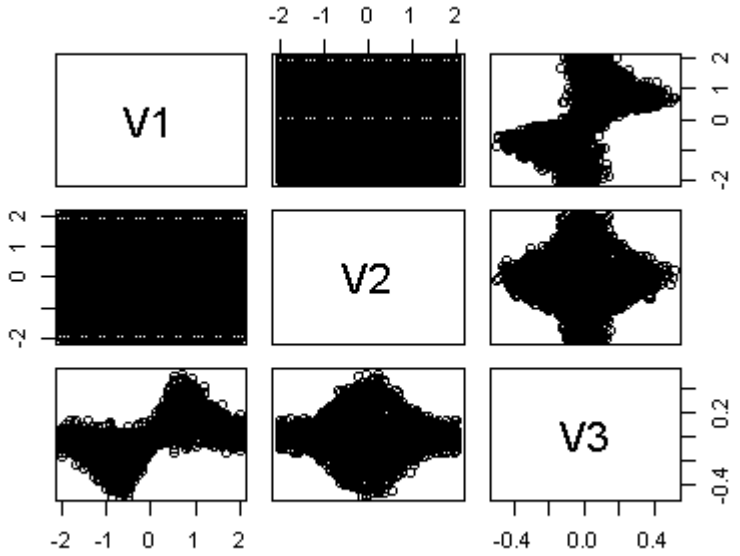
The dataset is loaded into R using the function read.table(). We can subsequently print the contents of the dataset and plot them on the graph using print() and plot() respectively.
We then split the dataset into training and testing set in order to test the performance of regression on data of different sizes.

We obtain the following plots of the multivariate datasets;

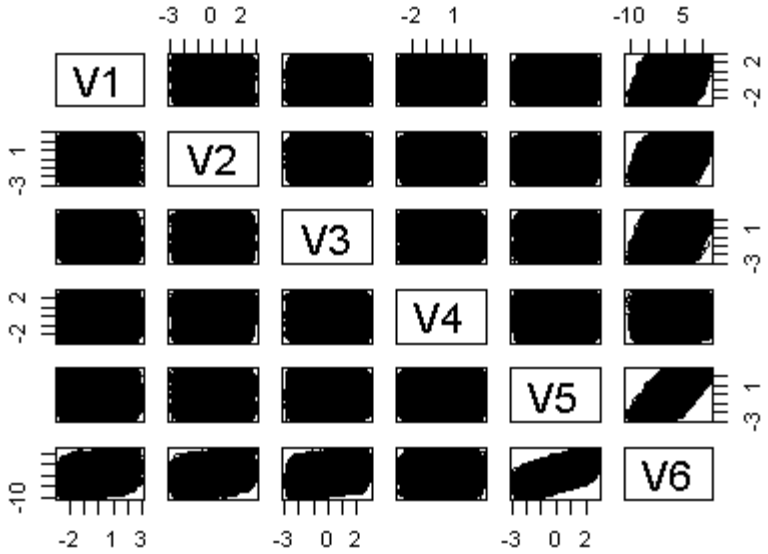
For mvar-set1:



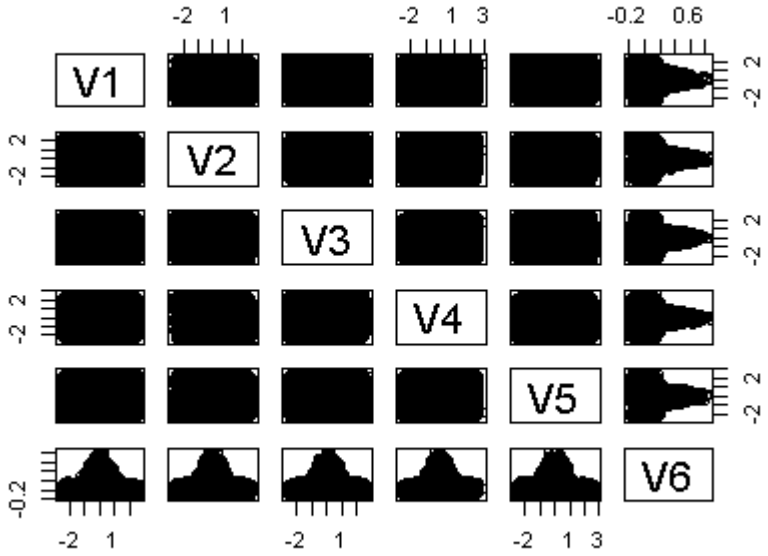
For mvar-set2:



For mvar-set3:



For mvar-set4:



2.2 Performing Linear Regression:

Using linear regression on univariate data, we model the relationship between scalar dependent variable and the single independent variable that is used in the four datasets.

We apply the following linear regression equation for our fitting the training and testing set;

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \ldots\ldots\ldots + \theta_n x_n$$

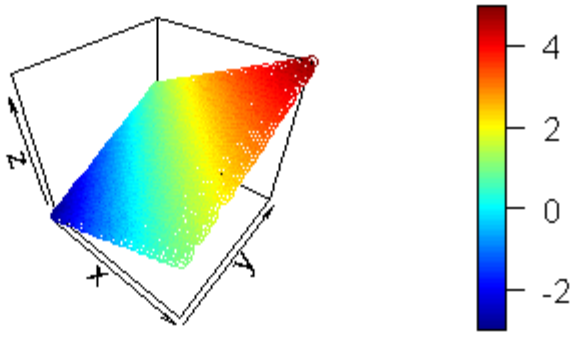
Then we formulate the mean square error;

$$\mathbf{MSE} = (\hat{y} - y)^2$$

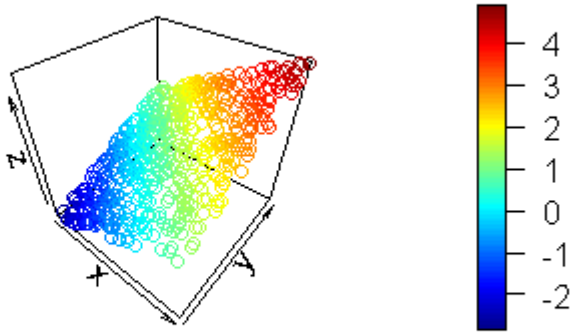
The plots obtained after applying linear regression are as follows:

For mvar-set1.dat

Training plot:

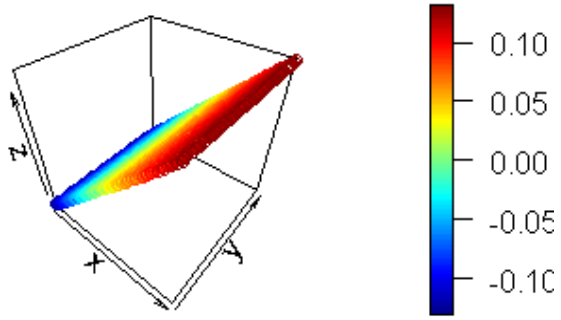


Test plot:

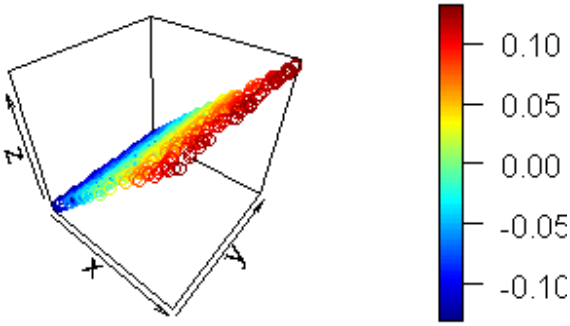


For mvar-set2.dat

Training plot

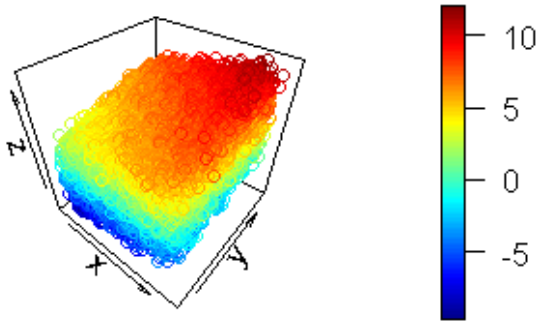


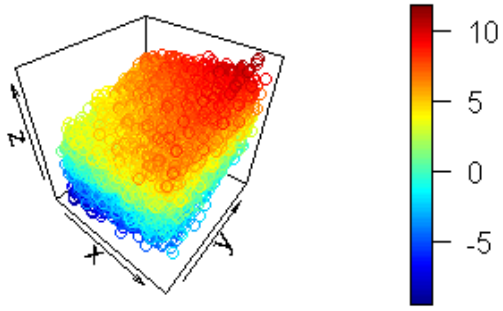
Test plot:



For mvar-set3.dat

Training plot

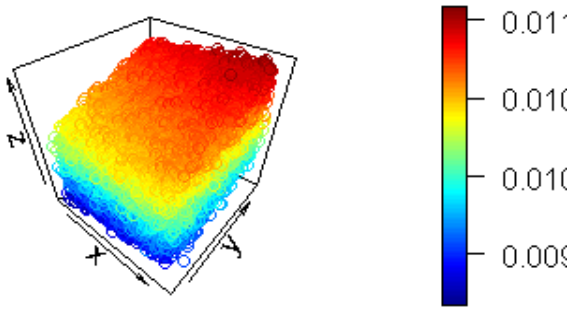




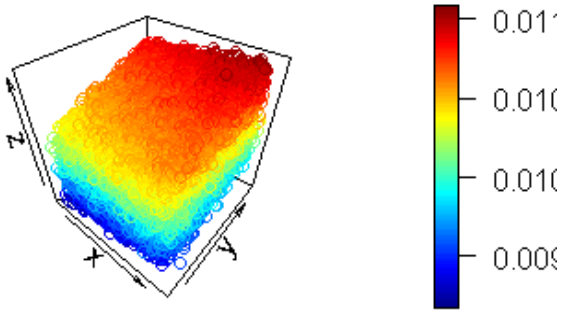
Test plot:

For mvar-set4.dat

Training plot



Test plot:



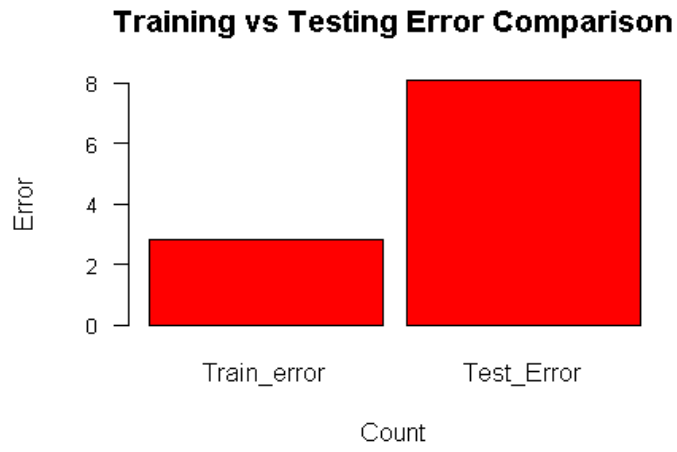
The results for MSE on training and test datasets are as follows:

	training error	test error
dataset1	3.014184	5.011399
dataset2	0.0201581	0.09134297
dataset3	13.07812	24.14468
dataset4	0.00014156	0.00027159

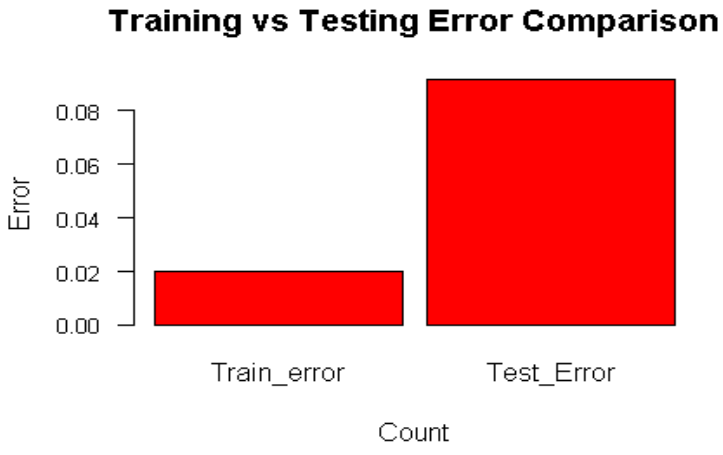
Comparing the errors for the training and test set:

The training vs test datasets error is represented in the form of a barplot as follows;

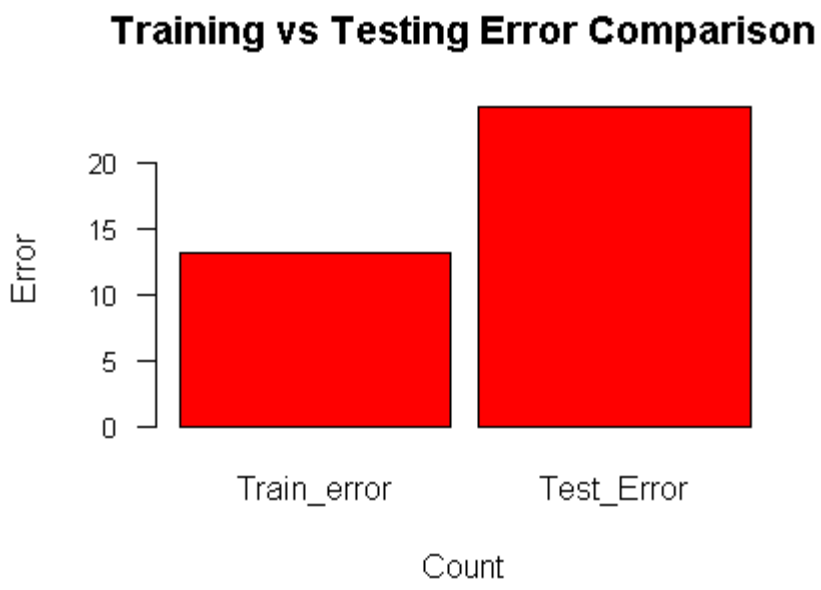
For mvar-set1.dat:



For mvar-set2.dat:

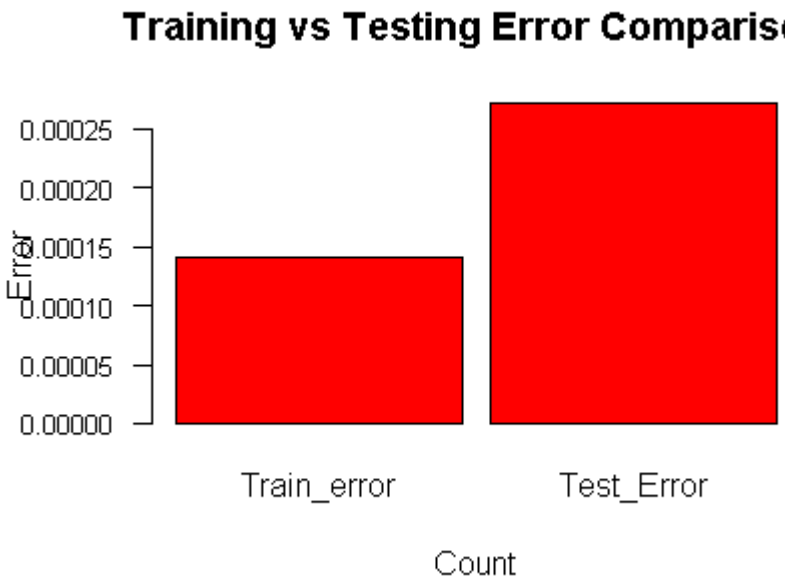


For mvar-set3.dat



:

For mvar-set4.dat

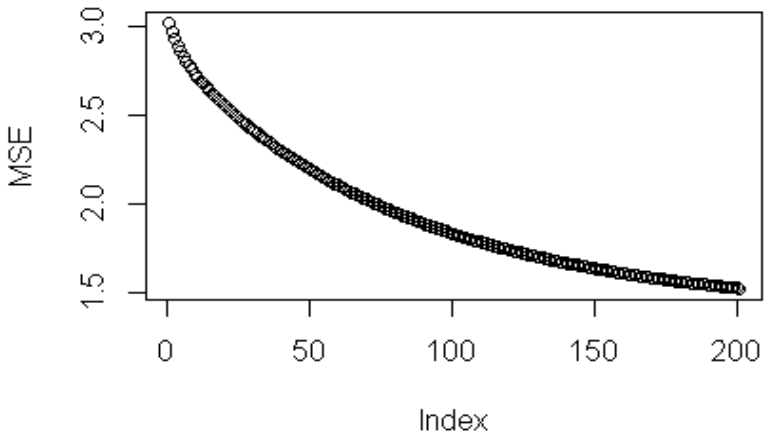


2.3 Deriving an Iterative solution for the regression problem:

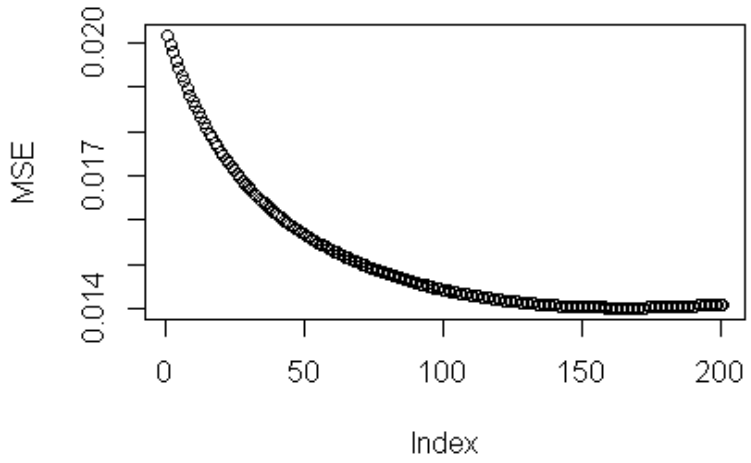
We apply a Gradient descent algorithm to optimize our regression solution. Using this algorithm, we find a local minimum of a function by taking steps proportional to the *negative* of the gradient (or of the approximate gradient) of the function at the current point.

We get the following graphs by applying gradient descent algorithm to our training set:

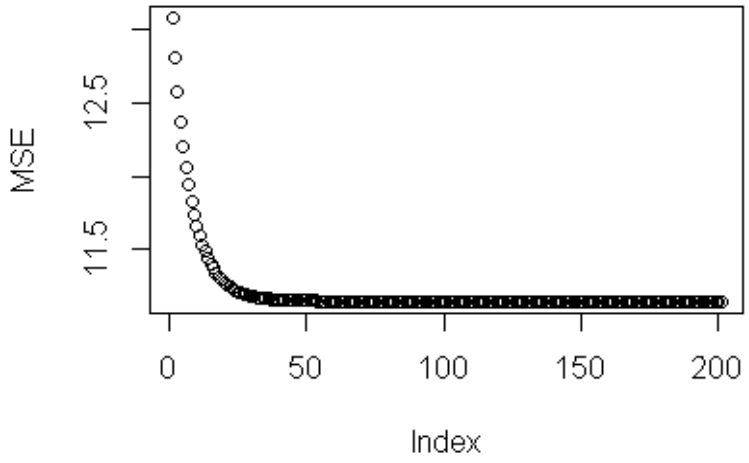
For mvar-set1:



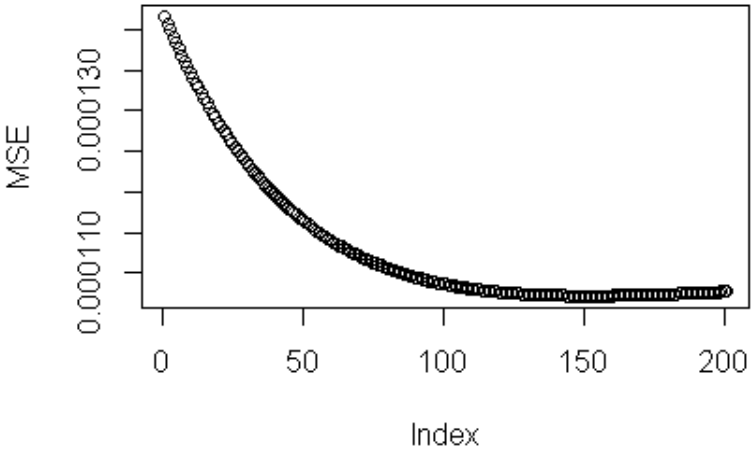
For mvar-set2:



For mvar-set3:



For mvar-set4:



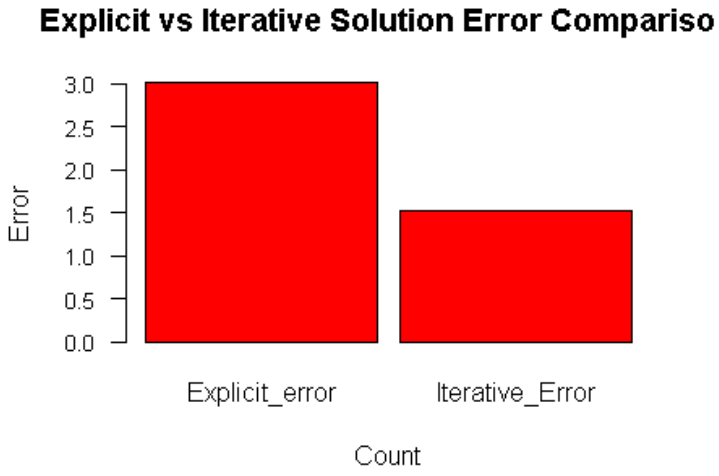
The iterative solution gives us the following MSE at the end of its function;

	training error
dataset1	1.526278
dataset2	0.0141007
dataset3	11.14602
dataset4	0.00010774

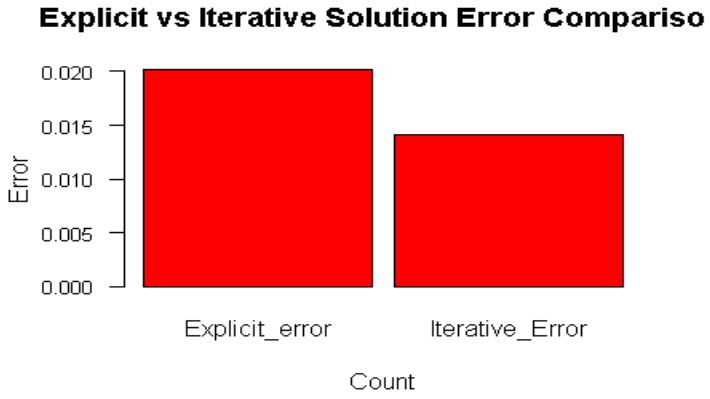
Comparing Explicit vs Iterative solution:

The explicit solution of the training set is compared with the iterative solution obtained and is represented in the form of a barplot as follows;

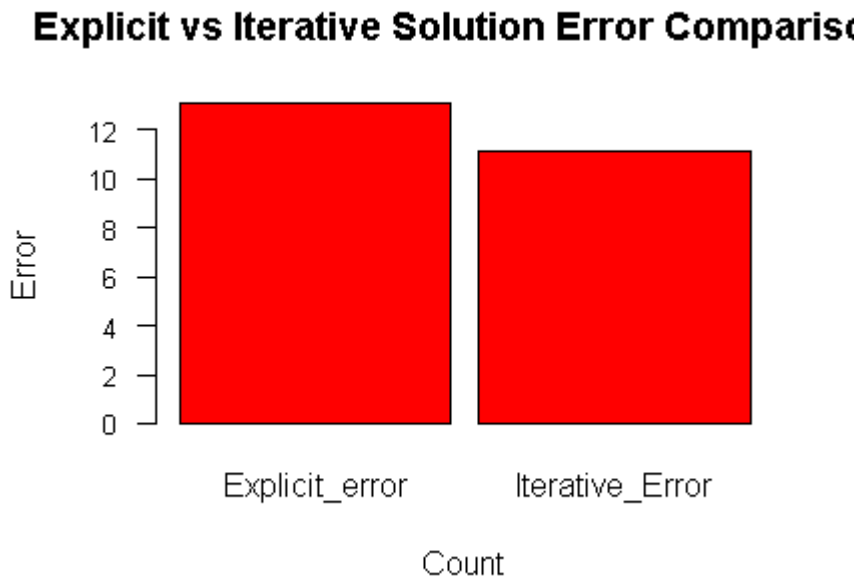
For mvar-set1.dat



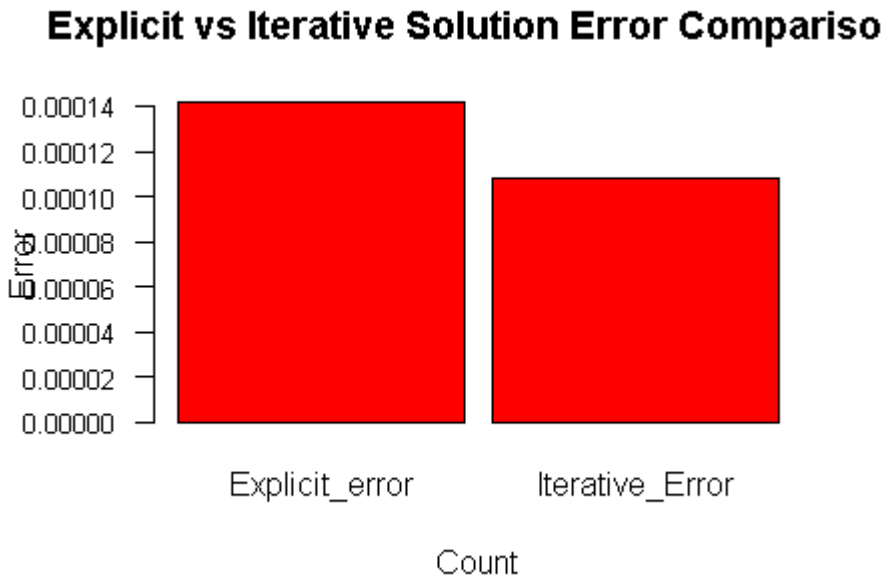
For mvar-set2.dat



For mvar-set3.dat



For mvar-set4.dat



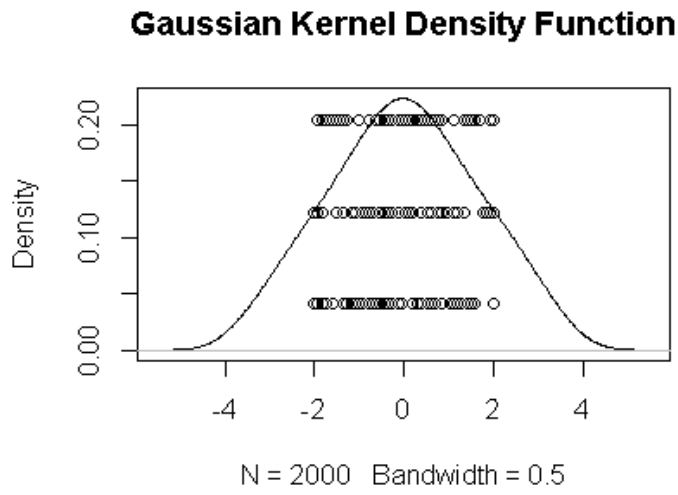
2.4 Applying the Gaussian kernel function:

The Gaussian kernel for n-dimensions is defined as:

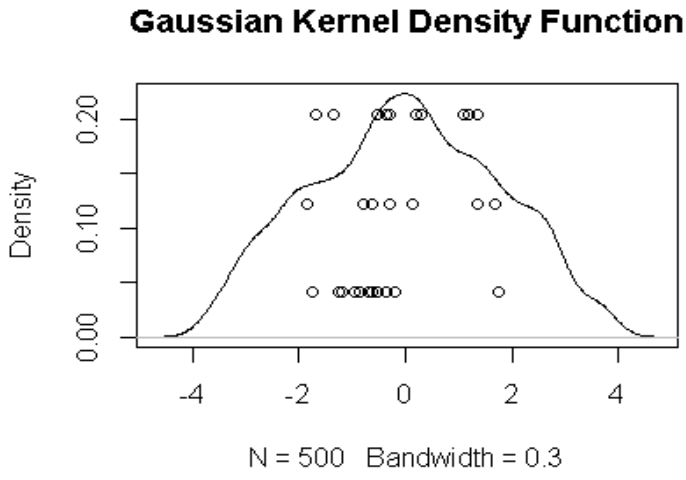
$$\frac{1}{(\sqrt{2*\Pi*\sigma})^N} * exp(\frac{x^2}{2*\sigma^2})$$

We estimate the following density of each dataset using the gaussian kernel function:

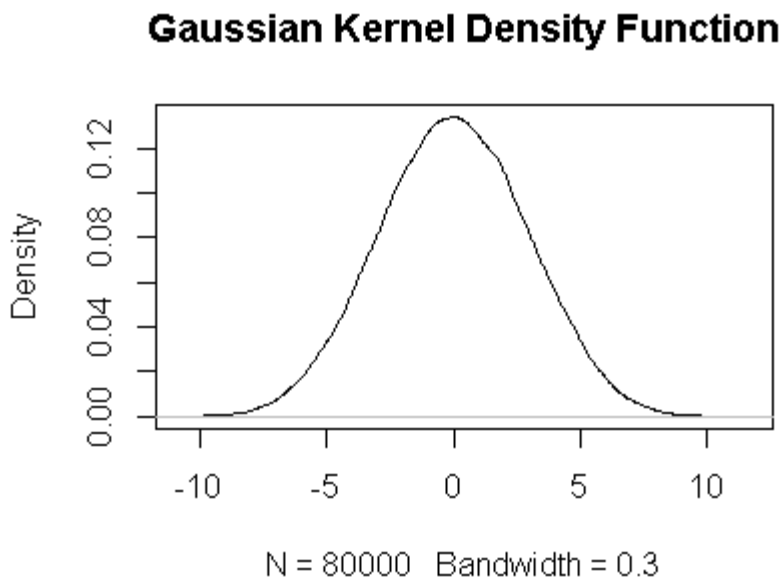
For mvar-set1.dat



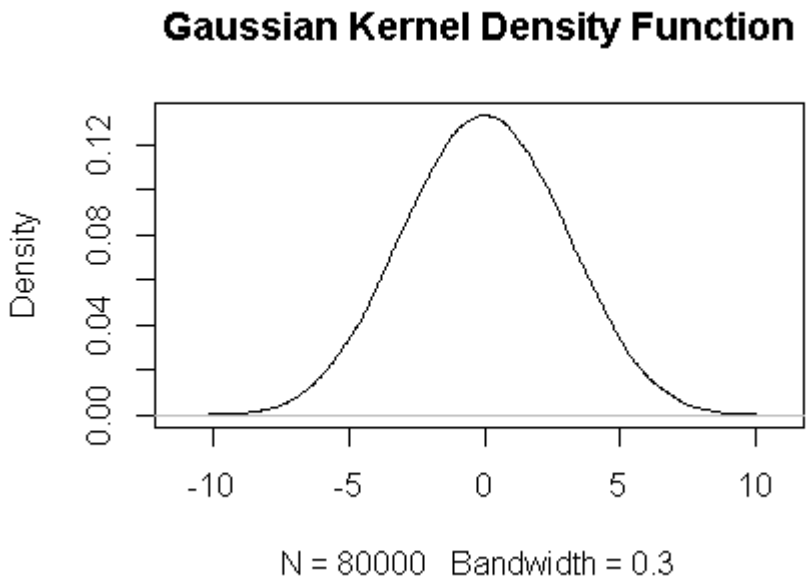
For mvar-set2.dat



For mvar-set3.dat



For mvar-set3.dat



Conclusions for the Multivariate Dataset:

- There is a significant disparity between the errors obtained in the training and test datasets, mostly due to the large size of the datasets

- Design issue: For gradient descent, the learning rate ('alpha' in the source code), which controls how big a step we take on the regression plane, has been derived manually without any computations and thus has been tweaked to get the descent approaching local minima.

- On each iteration of our gradient descent algorithm we reduce the value of our co-efficients and intercepts, thus approaching the local minima. This results in a much better accuracy of the iterative approach as compared to the explicit approach.