Assignment 04

Supriya Bachal

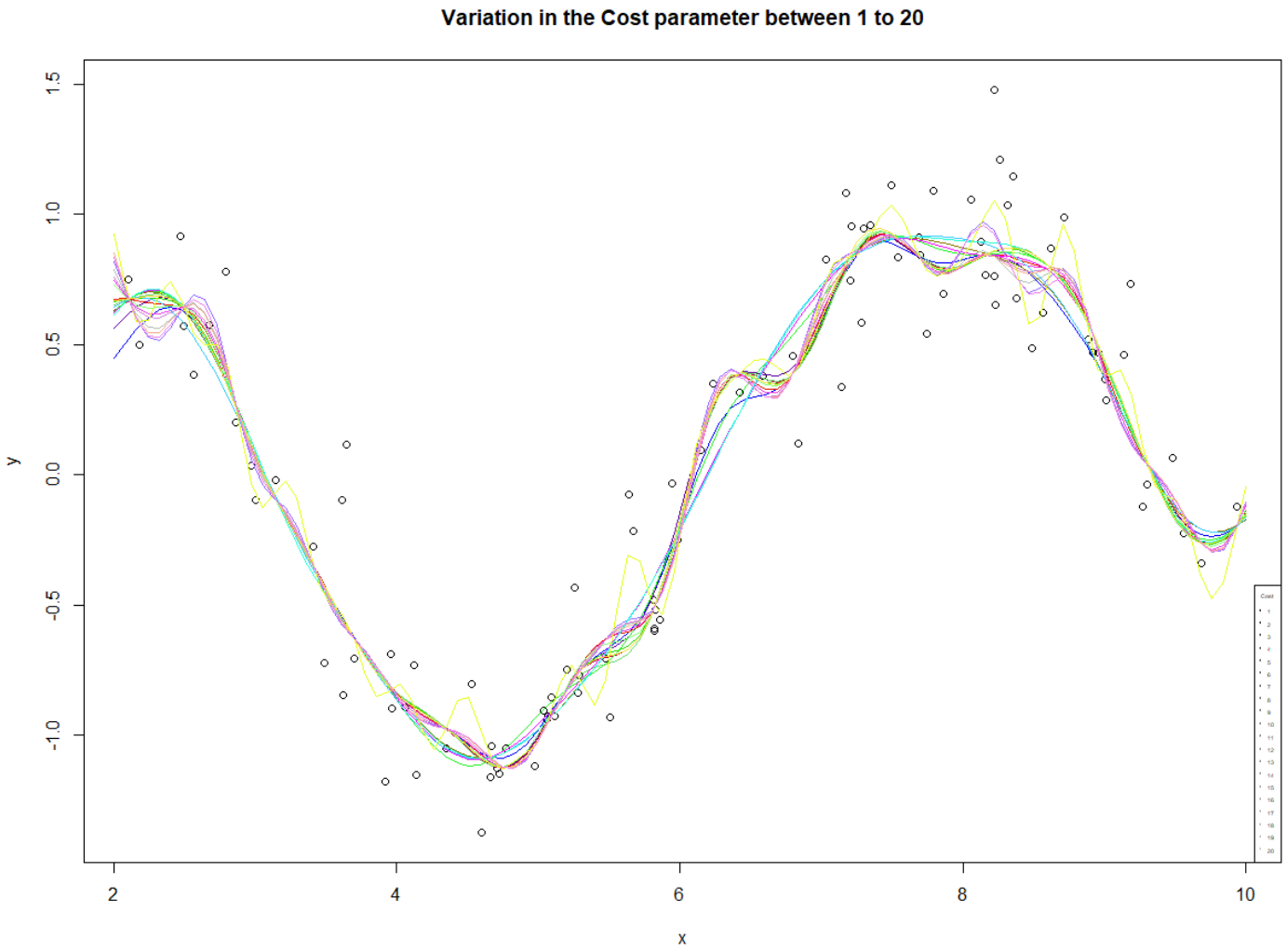
7.1. Simulate a single predictor and a nonlinear relationship, such as a sin wave shown in Fig. 7.7, and investigate the relationship between the cost, , and kernel parameters for a support vector machine model:

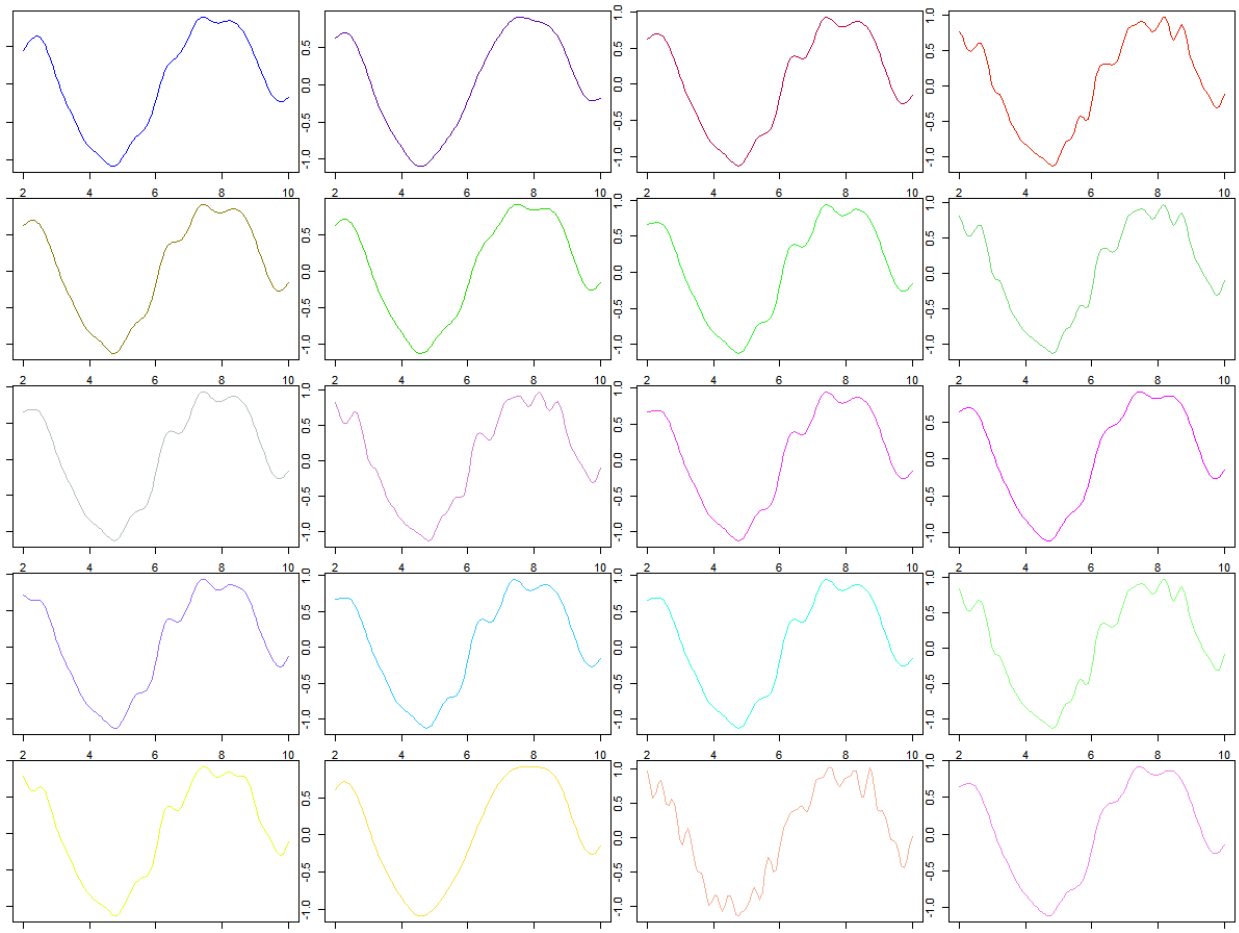
```
> set.seed(100)
> x <- runif(100, min = 2, max = 10)
> y <- sin(x) + rnorm(length(x)) * .25
> sinData <- data.frame(x = x, y = y)
> plot(x, y)
> ## Create a grid of x values to use for prediction
> dataGrid <- data.frame(x = seq(2, 10, length = 100))
```

(a) Fit different models using a radial basis function and different values of the cost (the C parameter) and . Plot the fitted curve.

Solution:

This methodology tries each unique Cost an incentive from 1 to 20. The Cost esteem is the penalizing element to the huge residual. The Cost parameter is communicated as 1/lambda.

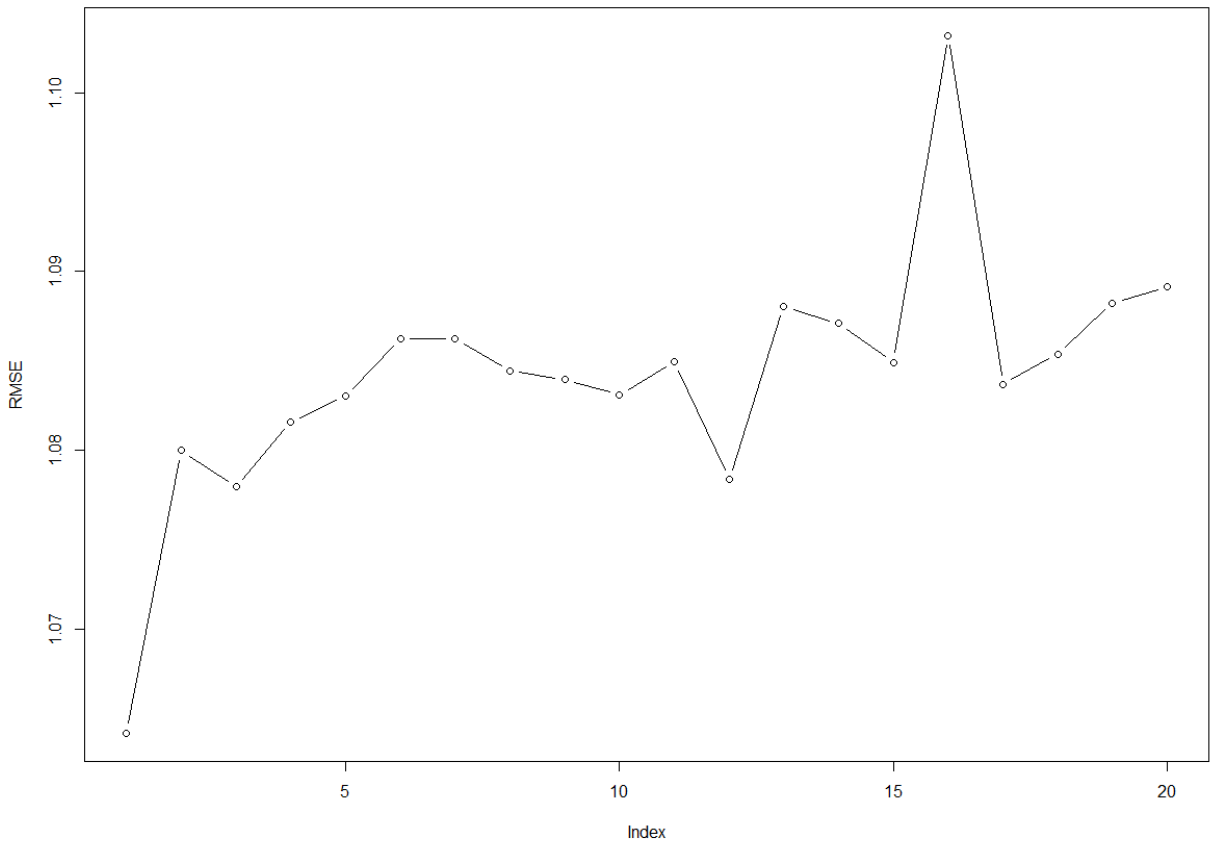

Variation in the Cost parameter between 1 to 20
y
x
1.5
1.0
0.5
0.0
-0.5
-1.0
2
4
6
8
10



We observe that:

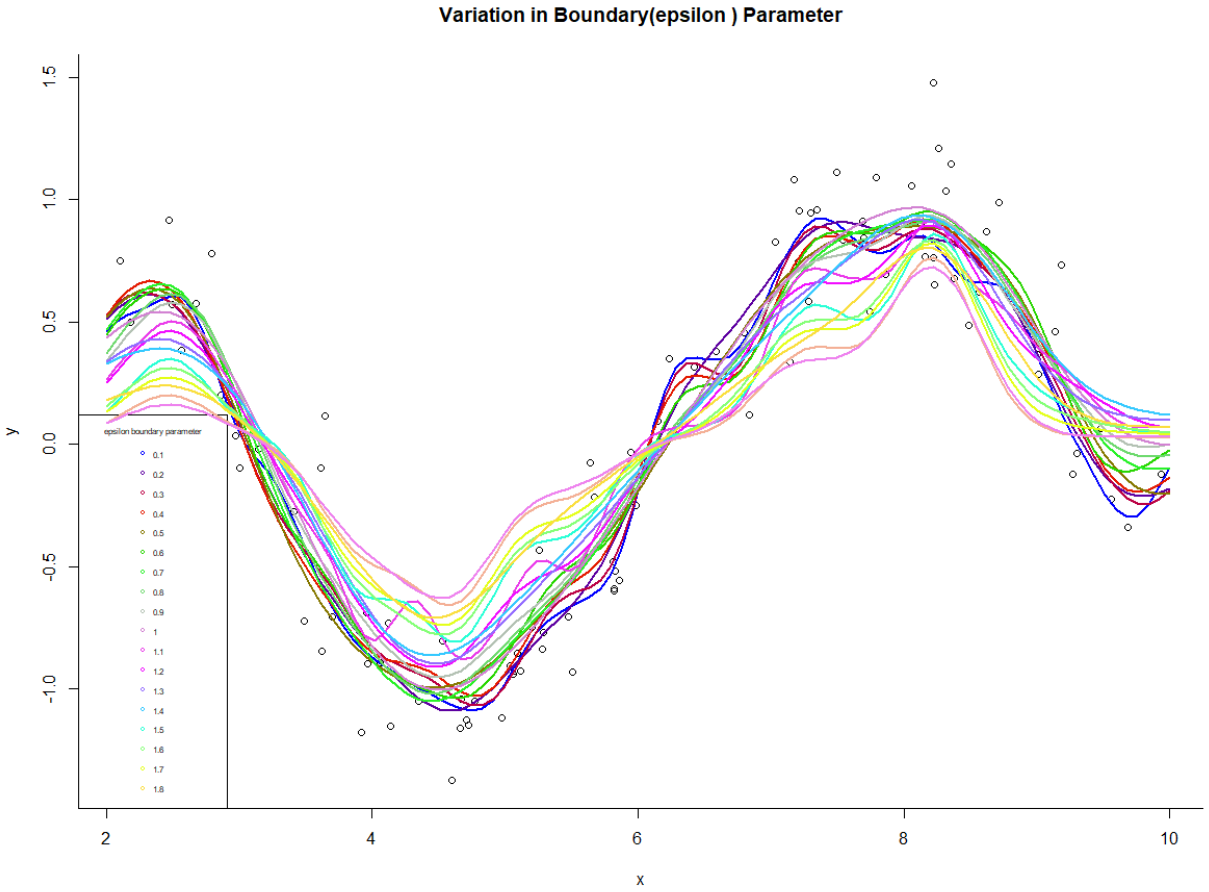
As the cost value increases the graph become more spiky and wavering.

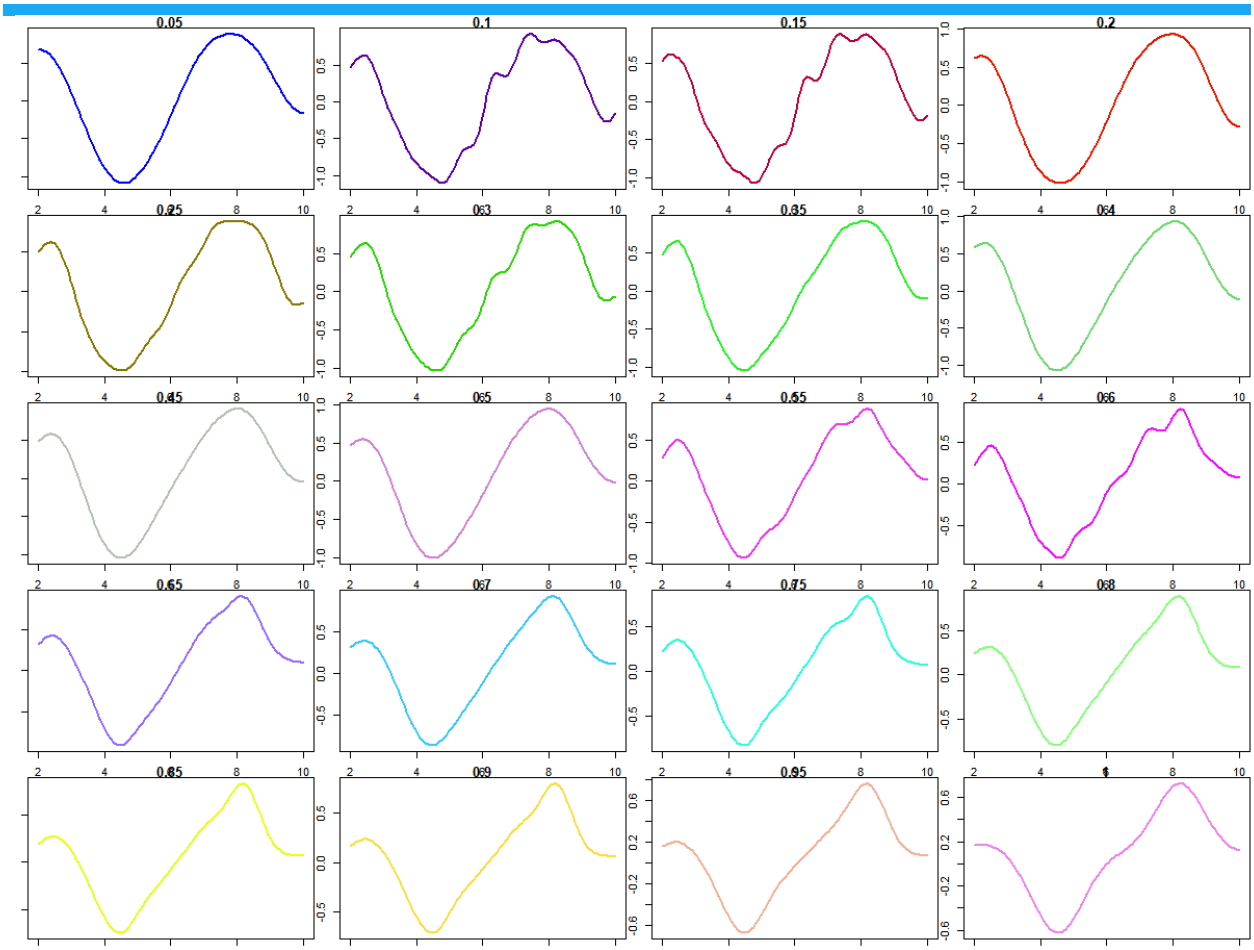
cost impacts the error of model and leads to overfitting.



Cost of 1 has the minimum RMSE .

We then vary the Boundary parameter, to produce the following graph.

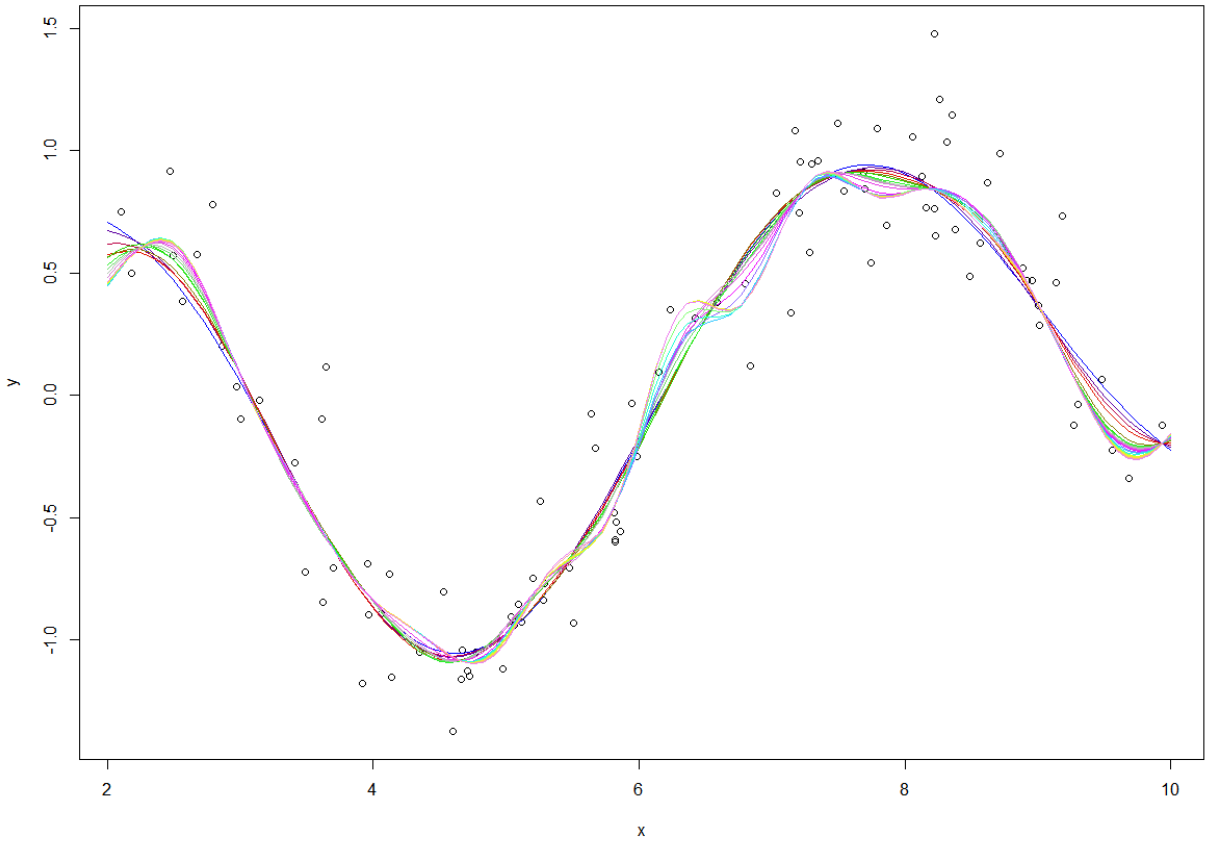

Variation in Boundary(epsilon) Parameter
y
x
epsilon boundary parameter
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8

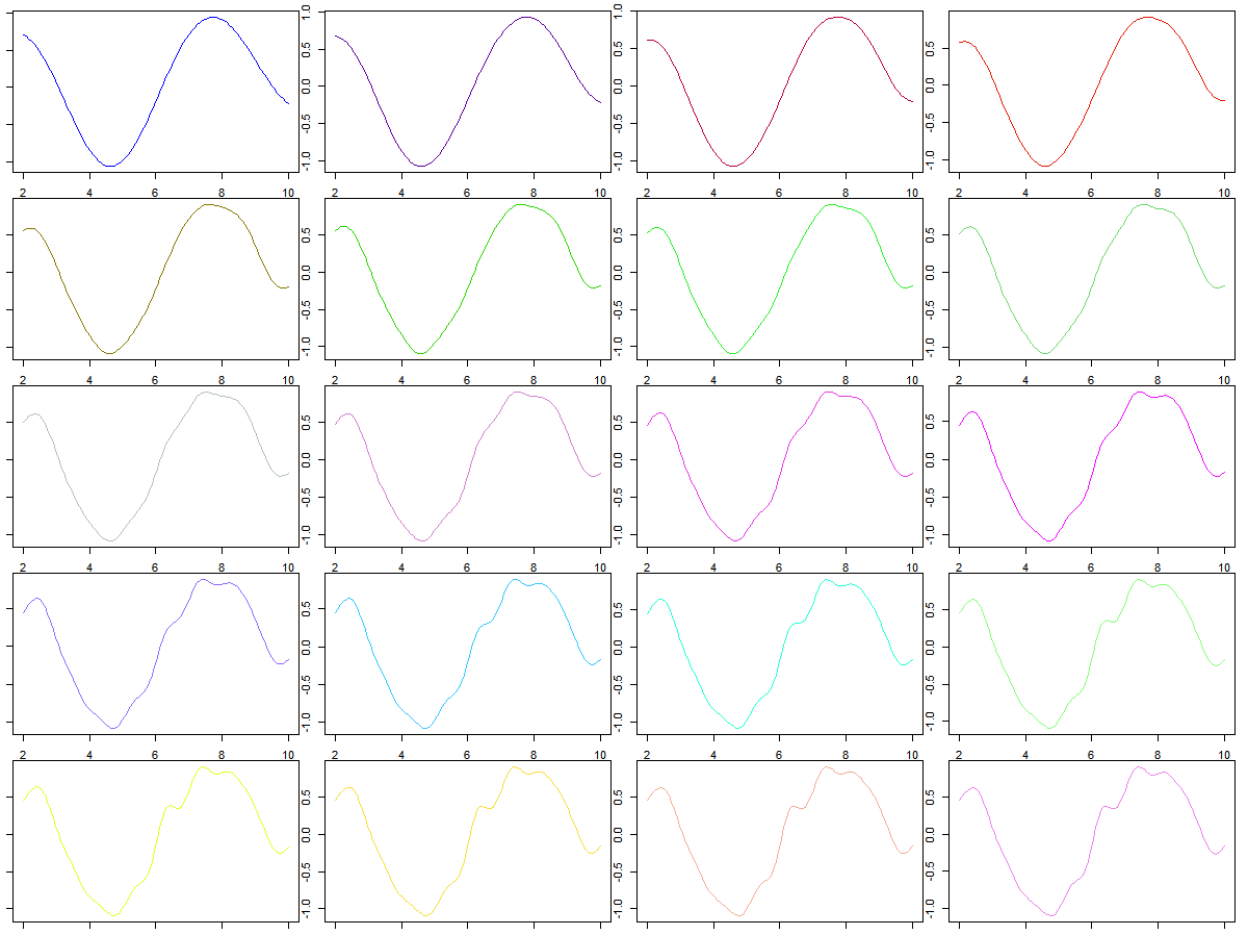


We observe:

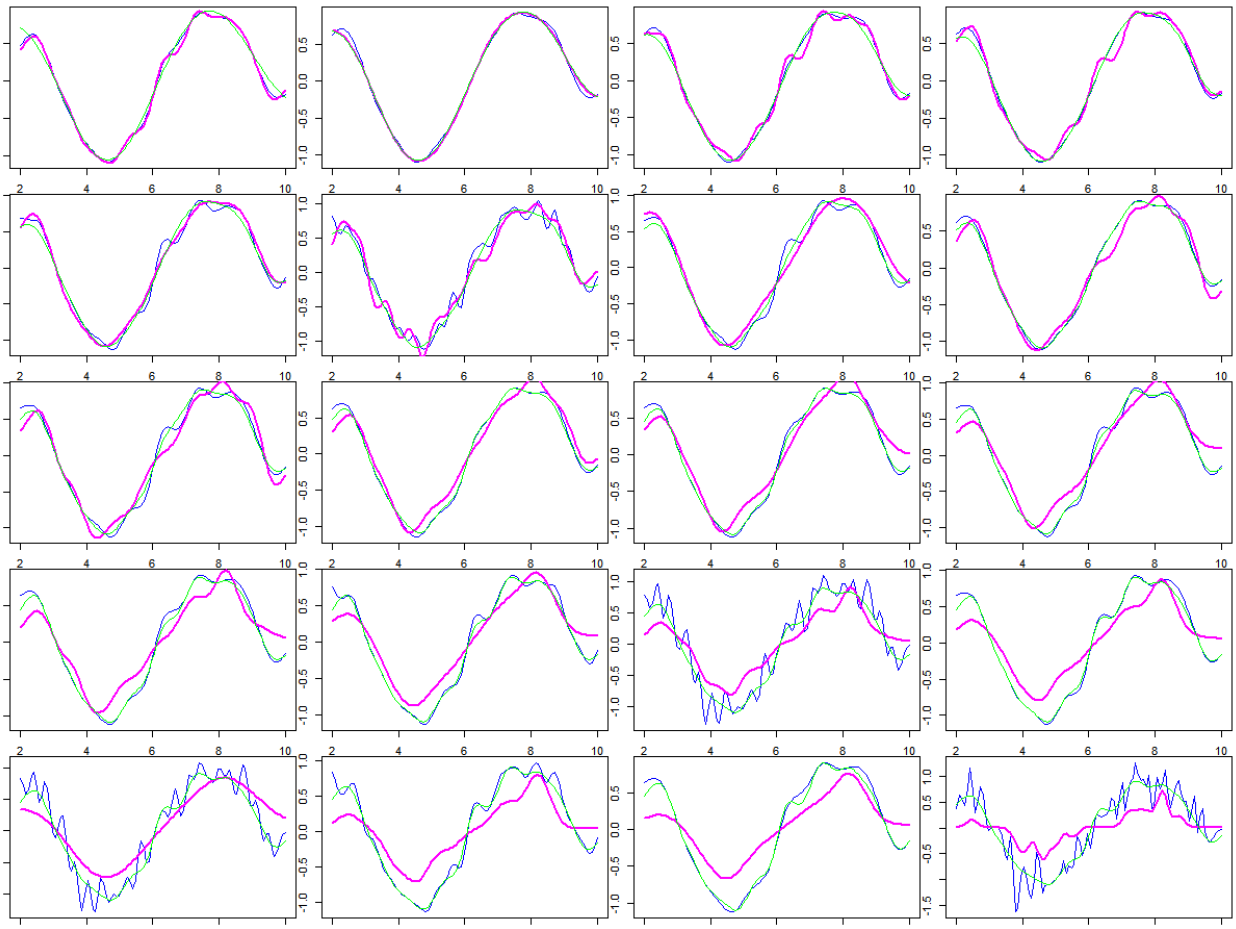
If we compare the wavering nature of the graphs then the wavering and edgy nature of the cost variation is more than the epsilon value,even though the epsilon value also shows similar behaviour of wavering as the e decreases the effect is not as profound as cost.

(b) The σ parameter can be adjusted using the kpar argument, such as kpar = list(sigma = 1). Try different values of σ to understand how this parameter changes the model fit. How do the cost, , and σ values affect the model?





The changing of the sigma value gives a smoother plot as compared to the cost function.Initially we can see that we get smooth plots however for the mid range values and higher values we get wavering plots .



The Blue line represents cost,the Magenta line represents e values and the green line represents sigma values.

The sigma values are related to cost such that for higher cost values the data is overfitted.

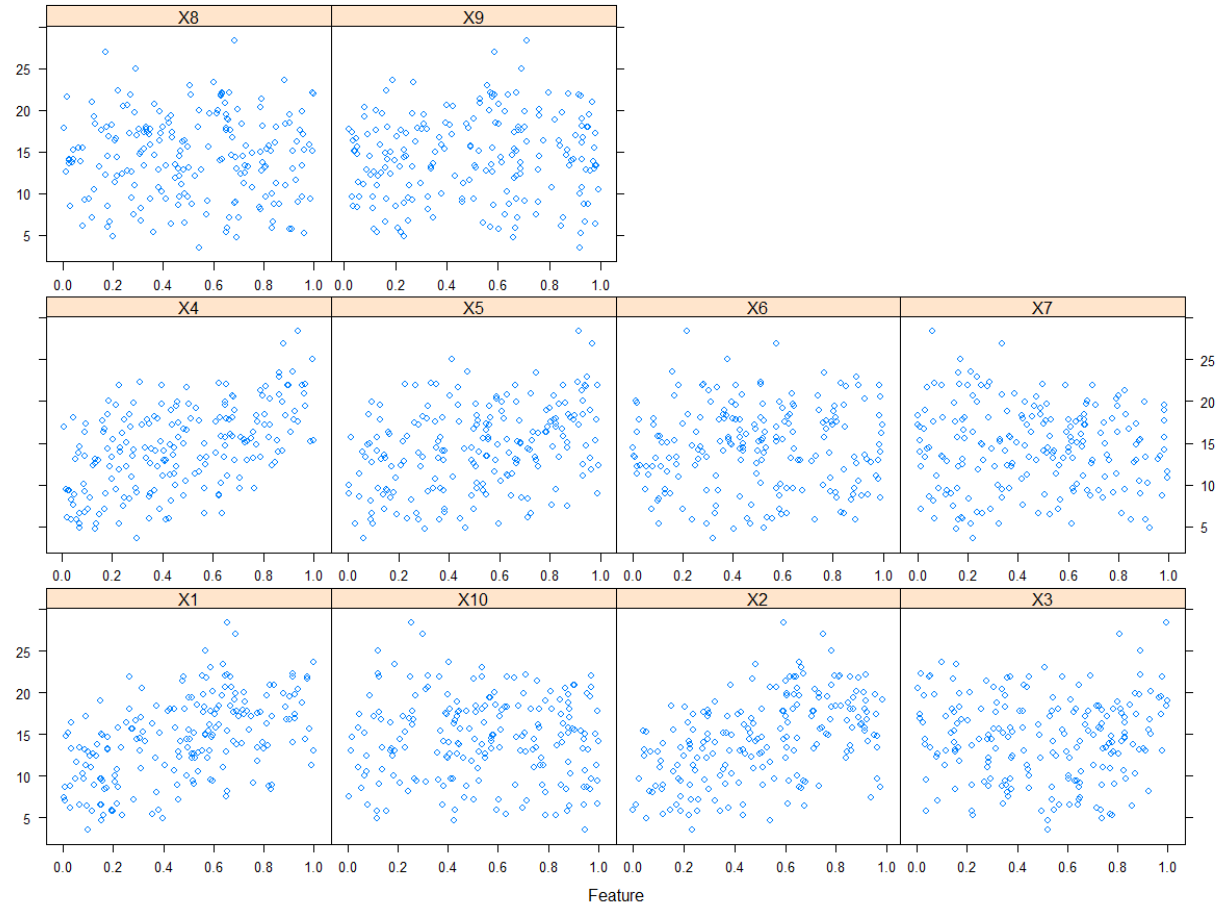
Cost value shows the highest influence on overfitting,the epsilon value shows moderate effect and the sigma value shows least effect on overfitting.

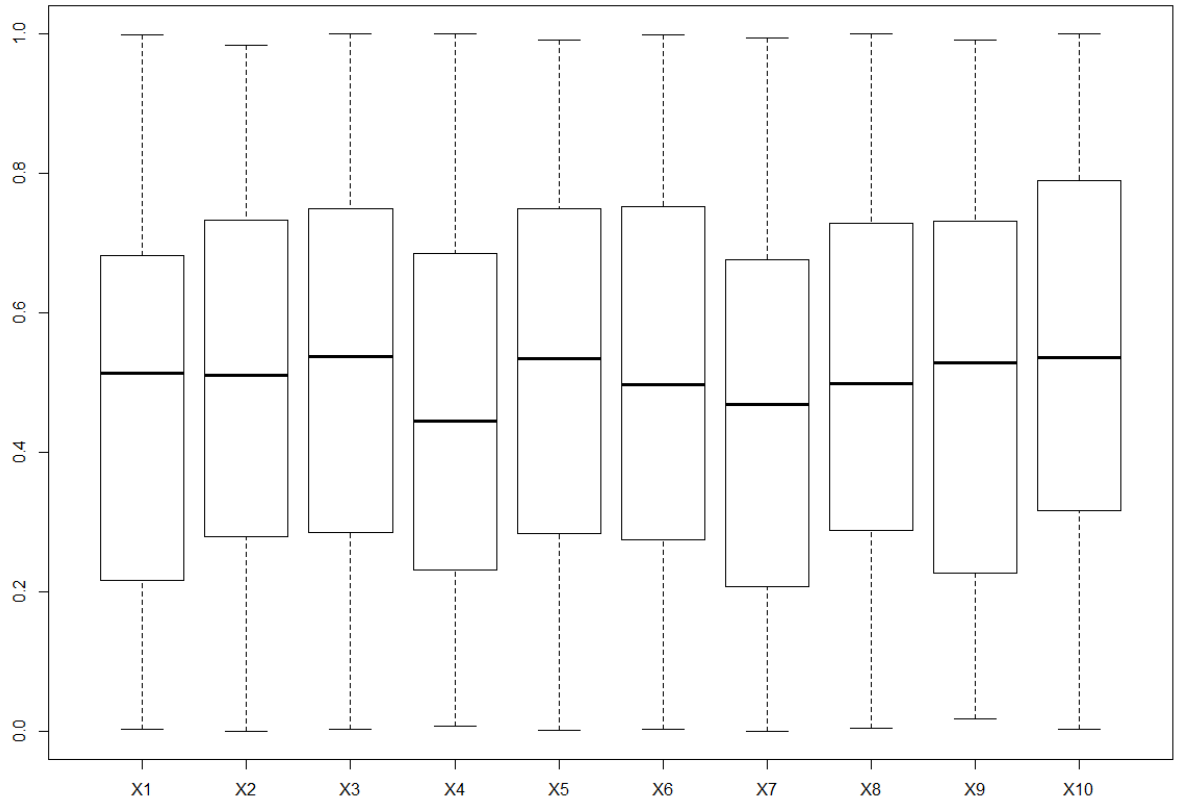
There is a relationship between cost and parameter e.

We can conclude that SVM is prone to overfitting and hence should be tuned carefully.

Question 7.2

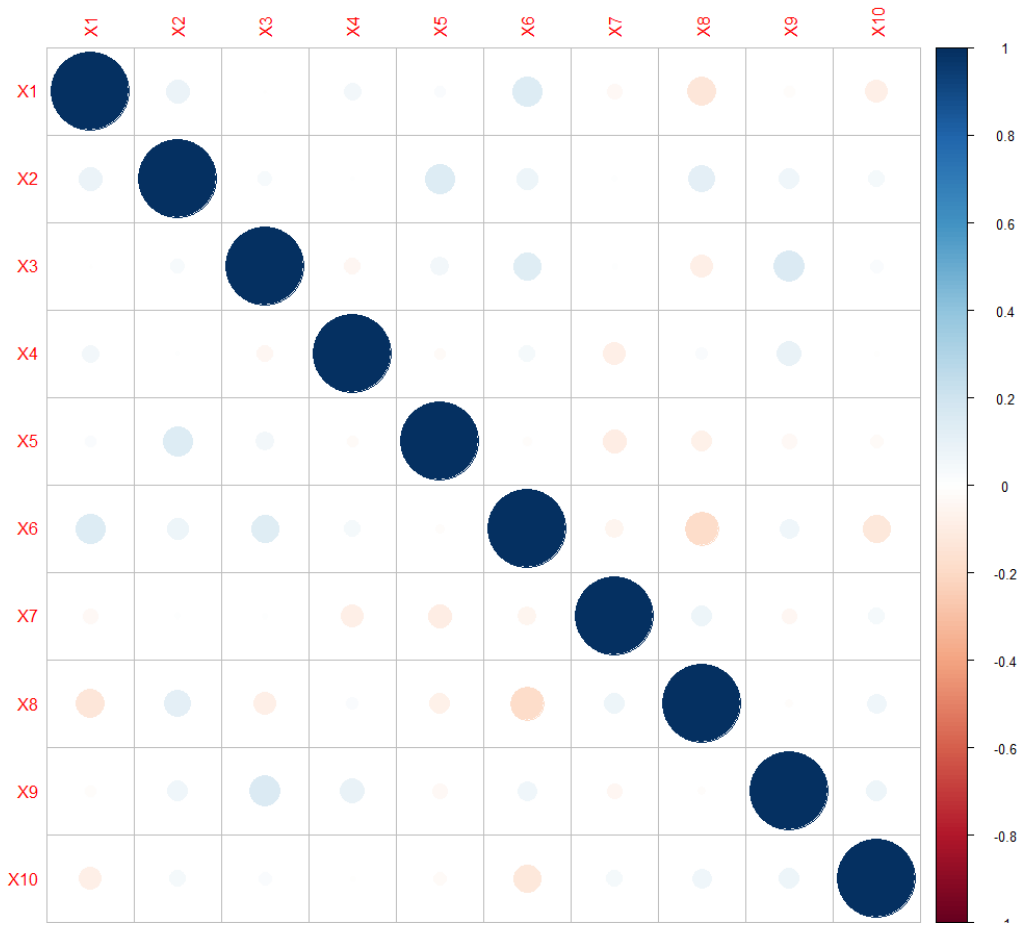
To simulate the data we set seed to 1.





This is the feature plot of the data variables from 1 to 10. The median values for each of the variables is between 0.4 to 0.6.There are no outliers in the data as expected as we have simulated the data. Further we can see from the feature plot that there is no specific pattern in the data in X, for predictors X1,X2,X5 and X4 we can see an upward trend showing a high correlation between X and Y.

To see the correlation between predictors we make a correlation plot.The correlation plot shows that there are no prominent correlations between the predictors. Which means that there is no need to eliminate any predictors to overcome collinearity issues.



We use bootstrapping method for resampling as shown in the example provided.

k-Nearest Neighbors

k-Nearest Neighbors

200 samples

10 predictor

Pre-processing: centered (10), scaled (10)

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 200, 200, 200, 200, 200, 200, ...

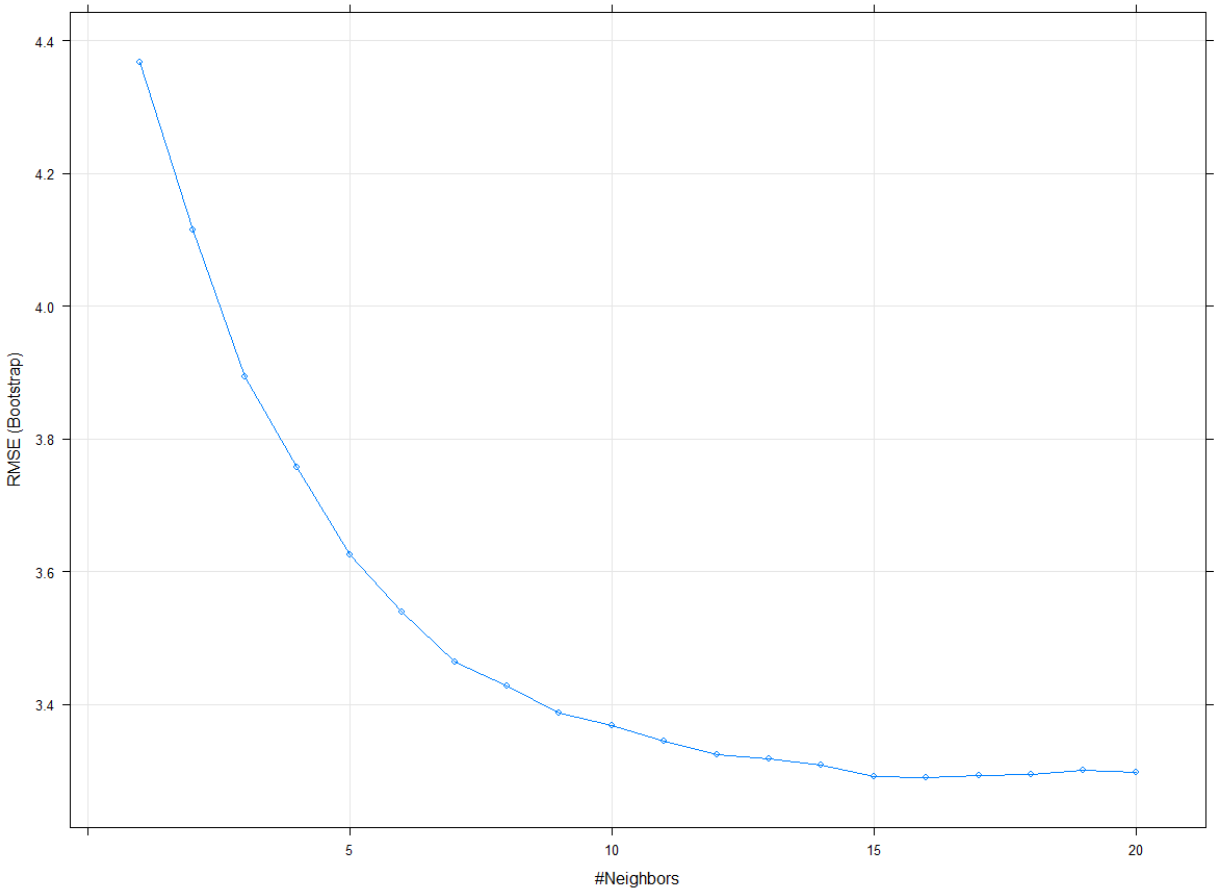
Resampling results across tuning parameters:

k	RMSE	Rsquared	MAE

1	4.367389	0.3492875	3.548289
2	4.115987	0.3837610	3.326009
3	3.894449	0.4176441	3.167881
4	3.757936	0.4431541	3.060567
5	3.626071	0.4766532	2.954620
6	3.539399	0.5002079	2.885096
7	3.463866	0.5231142	2.823065
8	3.428684	0.5377567	2.779065
9	3.387872	0.5552706	2.744969
10	3.368165	0.5650245	2.726986
11	3.345212	0.5754355	2.706971
12	3.324509	0.5887061	2.701639
13	3.317962	0.5964912	2.696927
14	3.308552	0.6016250	2.685018
15	3.291614	0.6123880	2.668077
16	3.290561	0.6188366	2.657986
17	3.293991	0.6238074	2.665458
18	3.294257	0.6319652	2.662786
19	3.300870	0.6342793	2.668902
20	3.297968	0.6414318	2.675613

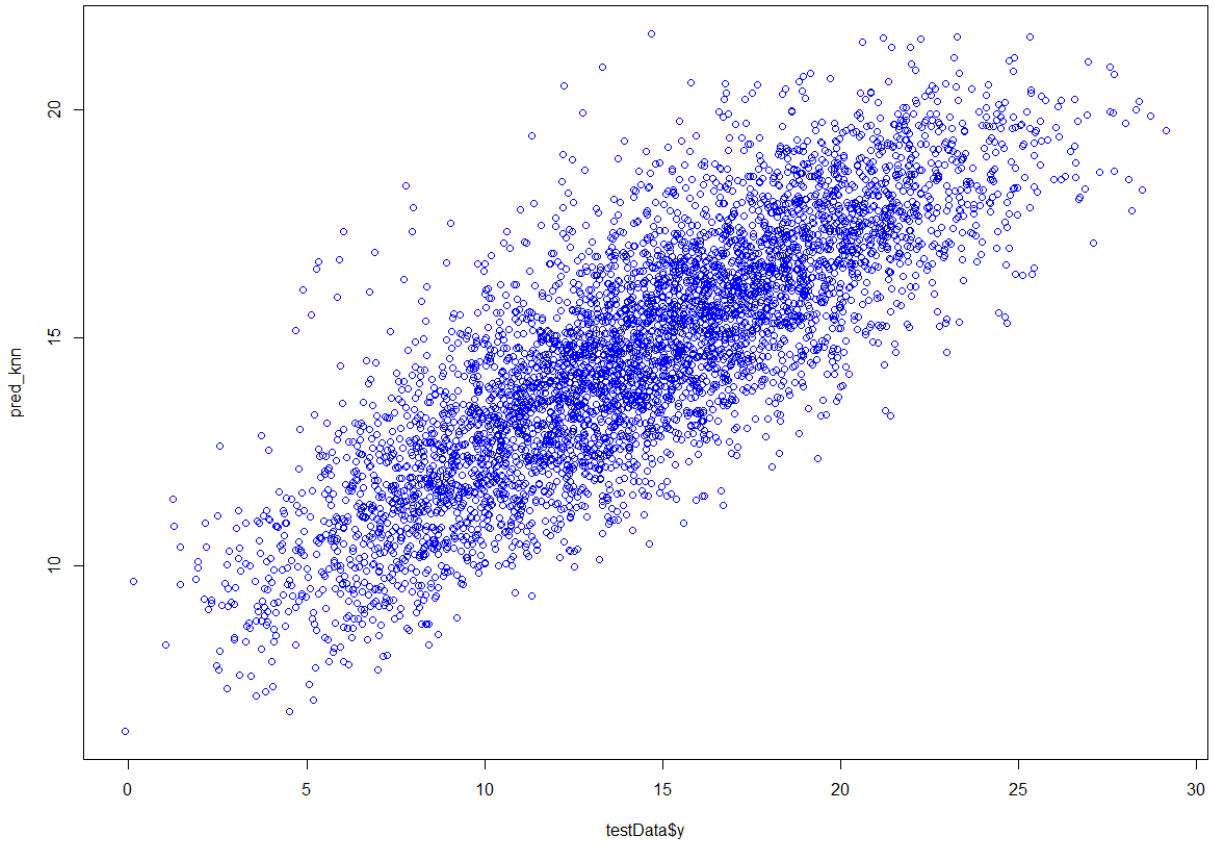
RMSE was used to select the optimal model using the smallest value.

The final value used for the model was k = 16.



From the plot we can see that the best tuning parameter for k is 16. Which means that the model uses 16 nearest neighbours. The RMSE value is 3.290561 and the R squared value is 0.6188366 for test cases.

The following plot is of observed vs predicted values



The R squared value for the test cases when Knn is applied is 0.680605 and the RMSE is 3.187461. We see that there is a slight performance increment from Training cases to test cases.

Neural Net model:

```
nnetTune <- train(trainingData$x, trainingData$y,method = 'avNNet',tuneGrid = nnetGrid,preProc = c('center', 'scale'),linout = TRUE,trace = FALSE, maxit = 500)
```

We use the AVNnet technique (averaged models) to train the neural net. We do not use bagging and hence put bagging =False. Additionally There are two parameters namely Maxit and Weight .Maxit defines the number of iteration we set the iterations to 500 and the weights were chosen at random.

We designed the Grid for the Neural network to choose values for the tuning we used decay as 0, 0.01, and 0.1 for size we used 3 to 7 hidden layers .

Model Averaged Neural Network

200 samples

10 predictor

Pre-processing: centered (10), scaled (10)

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 200, 200, 200, 200, 200, 200, ...

Resampling results across tuning parameters:

decay	size	RMSE	Rsquared	MAE
0.00	3	2.393049	0.7738388	1.840818
0.00	4	3.022083	0.7063824	2.168363
0.00	5	3.387445	0.6362126	2.402634
0.00	6	4.727621	0.4948182	3.151349
0.00	7	6.124268	0.3644477	3.885588
0.01	3	2.316956	0.7883652	1.816846
0.01	4	2.331992	0.7864001	1.838038
0.01	5	2.595034	0.7411920	2.040327
0.01	6	2.647717	0.7329345	2.079126
0.01	7	2.784524	0.7089808	2.199673
0.10	3	2.376122	0.7754506	1.859854
0.10	4	2.337420	0.7848291	1.842584
0.10	5	2.437573	0.7663824	1.892939

0.10 6 2.505001 0.7585523 1.981565

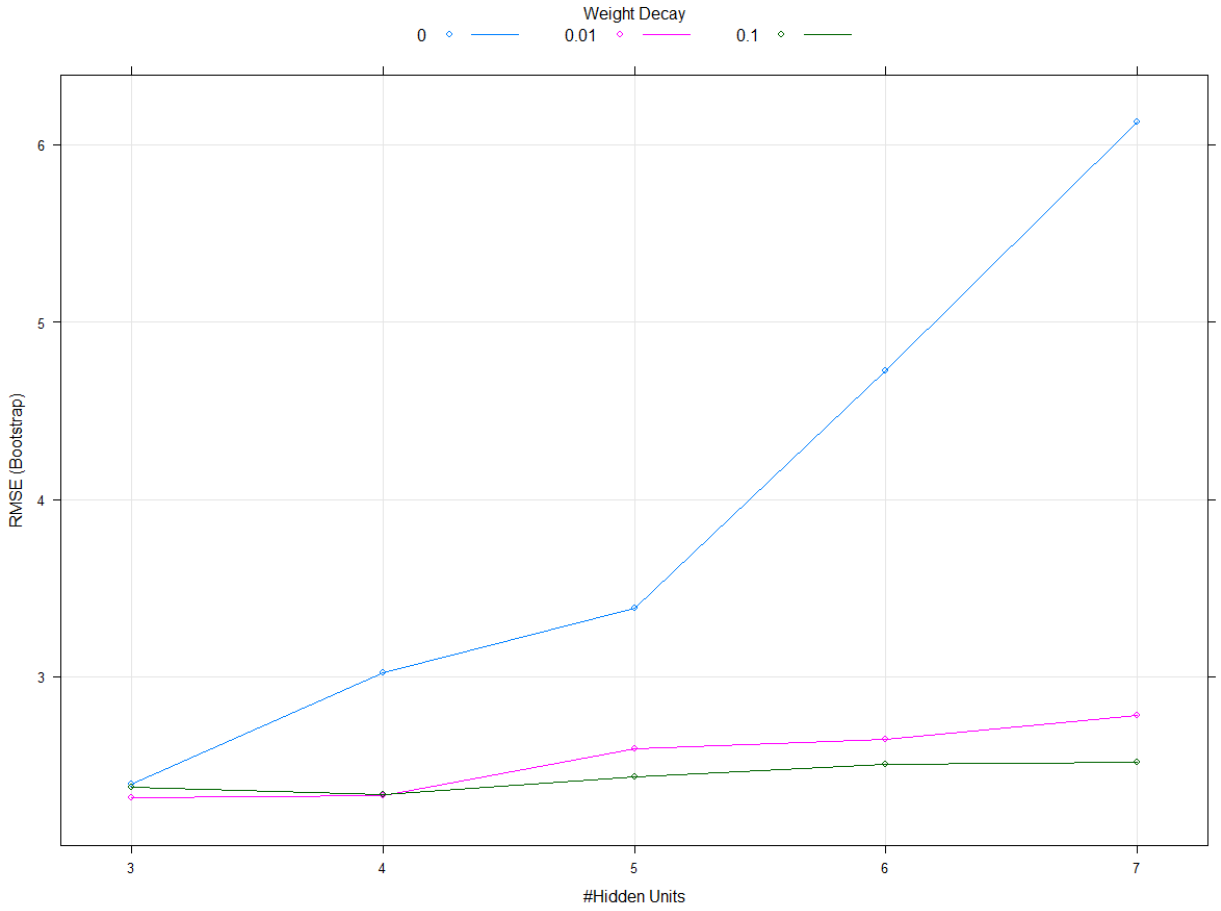
0.10 7 2.519906 0.7505244 1.983915

Tuning parameter 'bag' was held constant at a value of FALSE

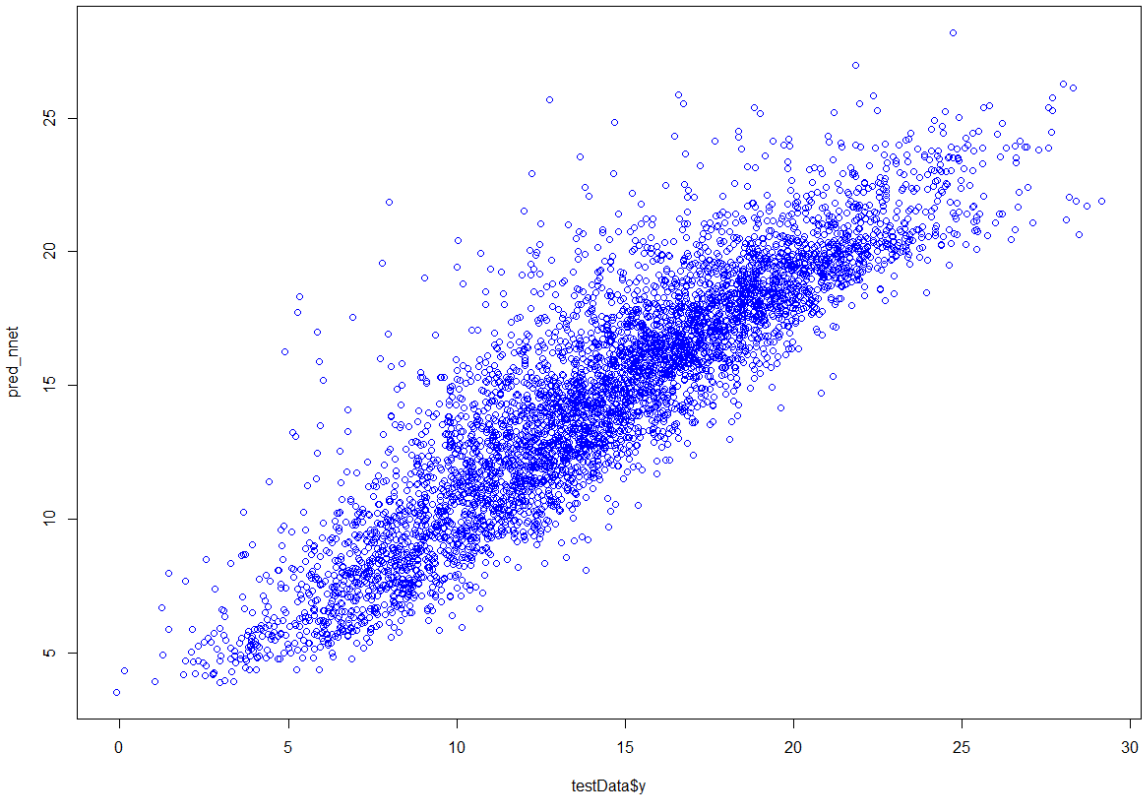
RMSE was used to select the optimal model using the smallest value.

The final values used for the model were size = 3, decay = 0.01 and bag = FALSE.

The RMSE value was 2.3 and the R squared value was 0.78 for the Training set.



The weight Decay graph shows that the min RMSE value is for weight decay at 0.01 and 3 hidden layers.



The RMSE value for the test set is 2.255836 and the R squared value is 0.7981351. We observe that even this model performs better for the test set as compared to the training set.

MARS

```
mars <- train(trainingData$x, trainingData$y,  method = 'earth', tuneGrid = mGrid,)
```

The mars model is Tuned using the mgrid which defines degree between 1 to 2 and maximum number of terms between 2 to 10.

Multivariate Adaptive Regression Spline

200 samples

10 predictor

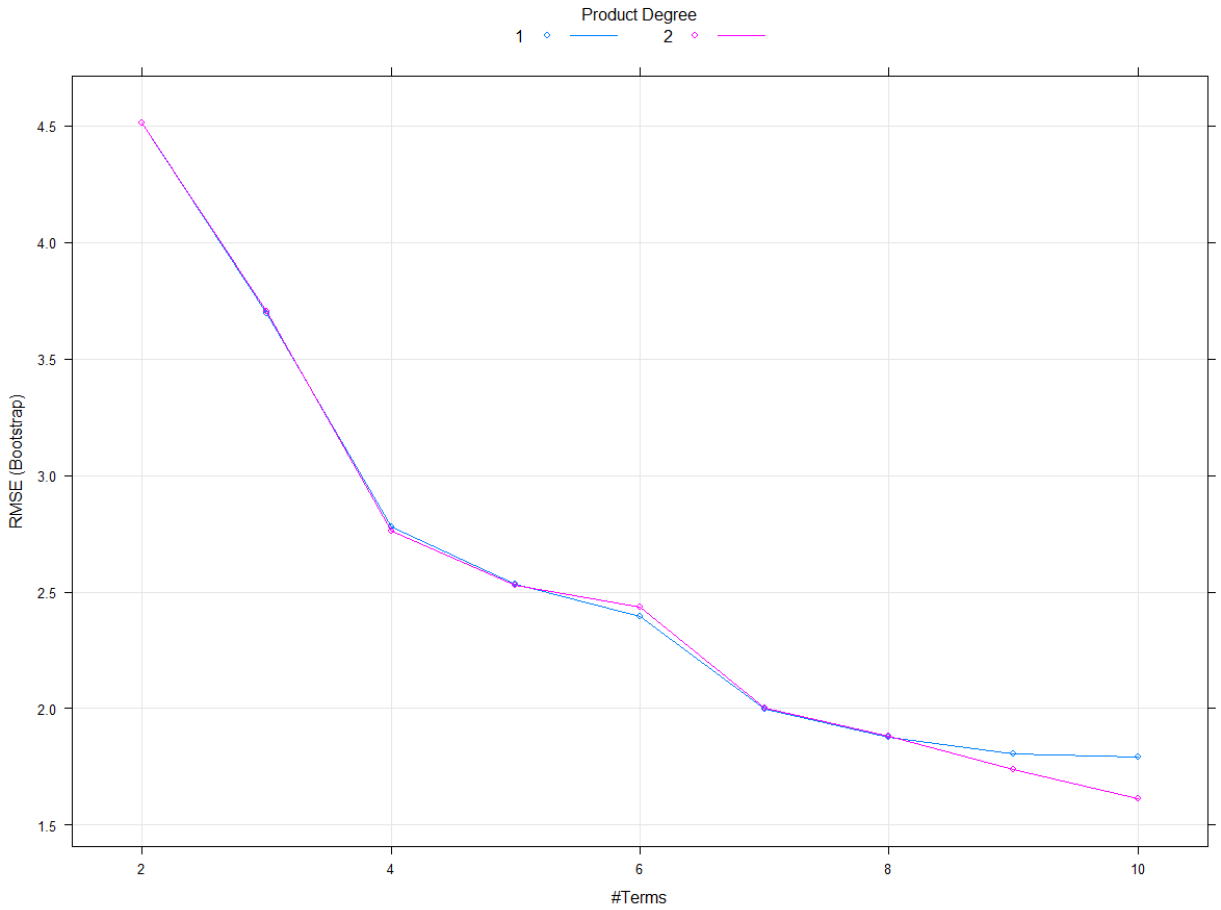
No pre-processing

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 200, 200, 200, 200, 200, 200, ...

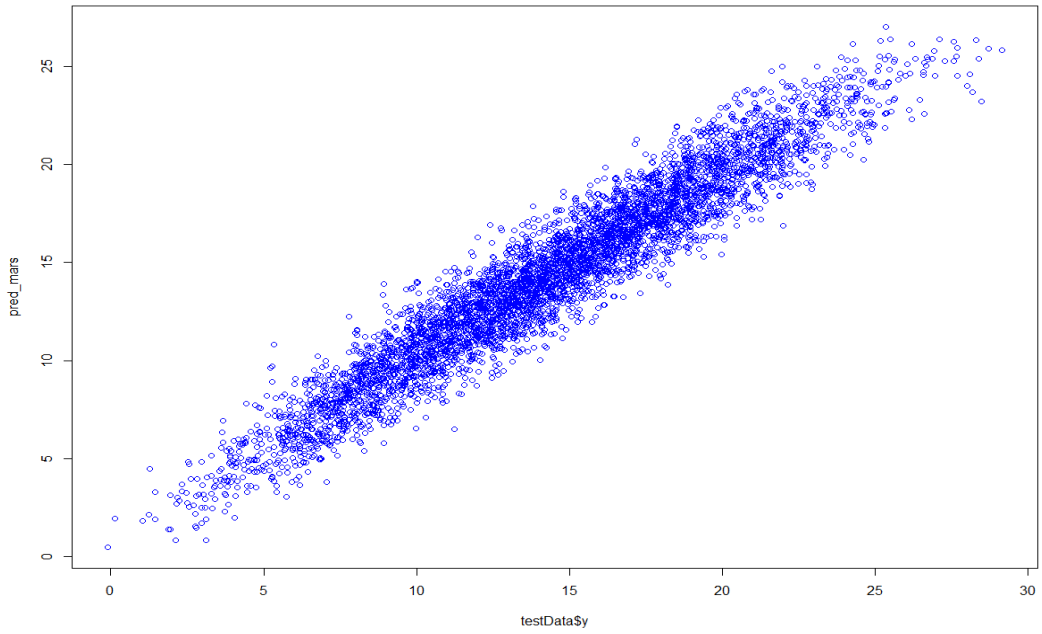
Resampling results across tuning parameters:

degree	nprune	RMSE	Rsquared	MAE
1	2	4.510221	0.2017622	3.719596
1	3	3.696575	0.4621782	2.992075
1	4	2.776976	0.6965425	2.226214
1	5	2.535576	0.7442072	2.021430
1	6	2.396059	0.7748952	1.910341
1	7	1.997451	0.8427290	1.588618
1	8	1.878071	0.8607824	1.486225
1	9	1.806903	0.8709898	1.408215
1	10	1.794167	0.8728956	1.387328
2	2	4.510138	0.2017189	3.719805
2	3	3.705720	0.4595804	2.997697
2	4	2.761341	0.7004236	2.205055
2	5	2.527292	0.7476576	2.003029
2	6	2.435956	0.7666567	1.918751
2	7	2.002104	0.8394419	1.592260
2	8	1.883270	0.8591193	1.467498
2	9	1.737783	0.8800273	1.352830
2	10	1.611688	0.8960040	1.263020



The final values used for the model were nprune = 10 and degree = 2 were selected for the Test data with RMSE =1.611688 and R squared value 0.8960040.

The following plot shows the observed vs the expected values the performance of the model can be assessed as good. The RMSE value for the test set is 1.406448 and the R squared value is 0.9197345. The Mars model performs exceptionally well for this dataset.



SVM

The SVM model was implemented using

```
set.seed(1)

svm<- train(trainingData$x, trainingData$y, method = 'svmRadial',preProc = c('center', 'scale'),tuneLength = 20)
```

The Model was tuned using the tune length parameter of 20.

Support Vector Machines with Radial Basis Function Kernel

200 samples

10 predictor

Pre-processing: centered (10), scaled (10)

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 200, 200, 200, 200, 200, 200, ...

Resampling results across tuning parameters:

C	RMSE	Rsquared	MAE
0.25	4.978256	0.1218695	4.085782
0.50	4.948892	0.1339104	4.059287

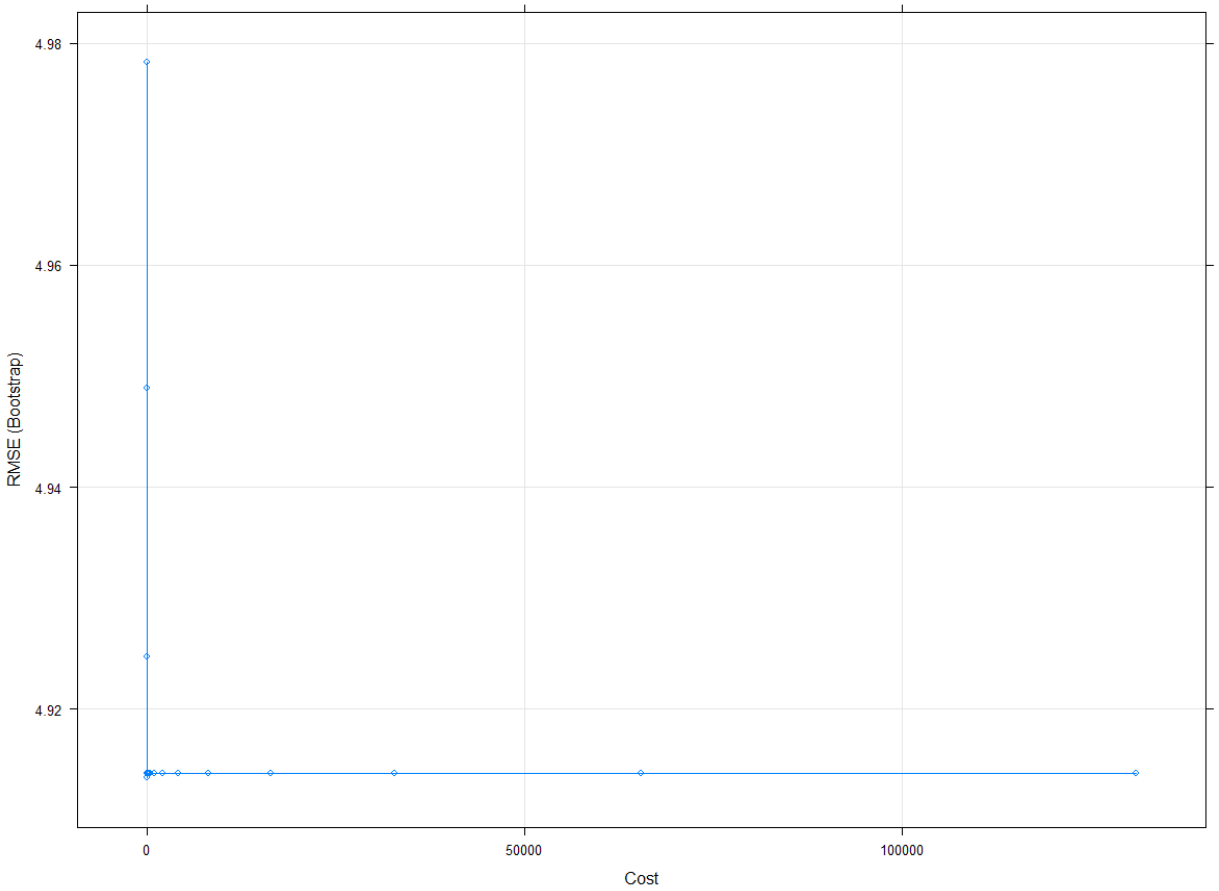
1.00	4.924717	0.1424149	4.040261
2.00	4.913855	0.1525388	4.031108
4.00	4.914246	0.1529141	4.030790
8.00	4.914246	0.1529141	4.030790
16.00	4.914246	0.1529141	4.030790
32.00	4.914246	0.1529141	4.030790
64.00	4.914246	0.1529141	4.030790
128.00	4.914246	0.1529141	4.030790
256.00	4.914246	0.1529141	4.030790
512.00	4.914246	0.1529141	4.030790
1024.00	4.914246	0.1529141	4.030790
2048.00	4.914246	0.1529141	4.030790
4096.00	4.914246	0.1529141	4.030790
8192.00	4.914246	0.1529141	4.030790
16384.00	4.914246	0.1529141	4.030790
32768.00	4.914246	0.1529141	4.030790
65536.00	4.914246	0.1529141	4.030790
131072.00	4.914246	0.1529141	4.030790

Tuning parameter 'sigma' was held constant at a value of 0.06444911

RMSE was used to select the optimal model using the smallest value.

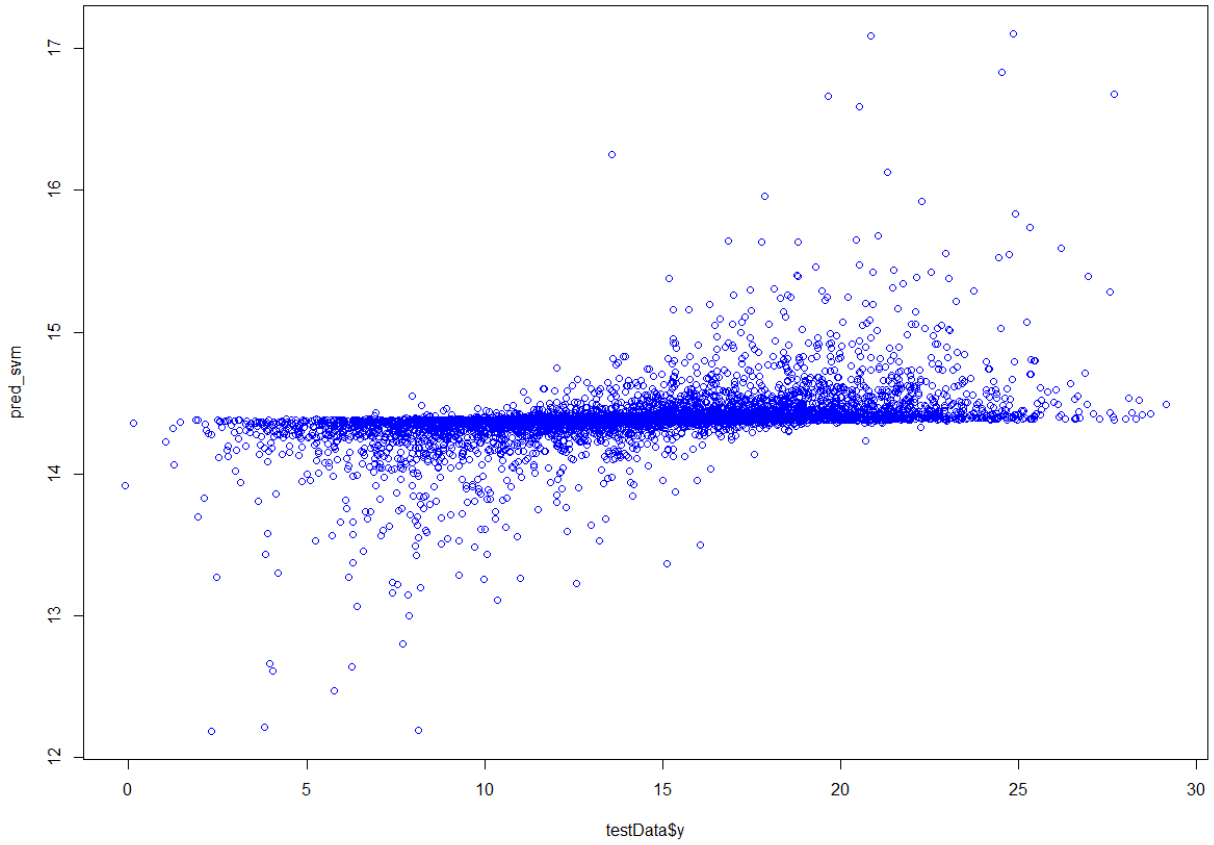
The final values used for the model were sigma = 0.06444911 and C = 2.

For training set The value for RMSE is 4.913855 and R squared is 0.1525388



THe below graph shows the expected vs derived values.

For the test set the Value for the RMSE is 4.863214and the R squared value is 0.2051648



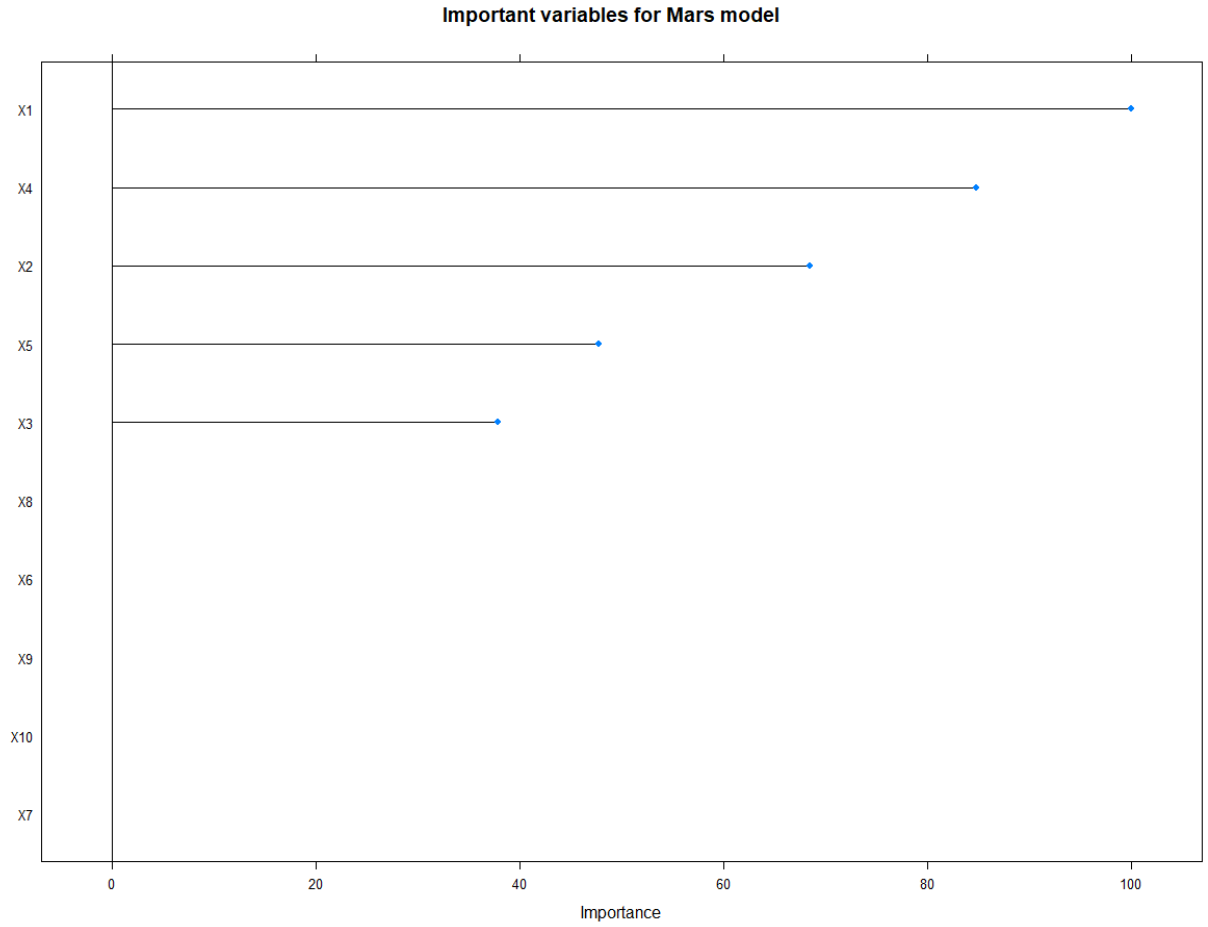
This model is the most poorly performing model until now. The performance is upgraded than the training set but the performance is still poor.

Which models appear to give the best performance? Does MARS select the informative predictors (those named X1–X5)?

From the Above observation we summarize the RMSE and R squared values of each of the models on the test data.

	RMSE	R squared
SVM	4.863214	0.2051648
MARS	1.406448	0.9197345.
NNet	2.255836	0.7981351
KNN	3.187461	0.680605

WHich shows that the Mars model performs the best for the given data set. Neural net performs the next best



Importance: X1, X4, X2, X5, X3, X6-unused, X7-unused, X8-unused, X9-unused, X10-unused.

Hence we can Conclude that Mars model selects the important informative predictors X1 to X5 and drops the non informative ones.

7.3.For the Tecator data described in the last chapter, build SVM, neural network, MARS, and KNN models. Since neural networks are especially sensitive to highly correlated predictors, does pre-processing using PCA help the model?

SVM

The Support Vector Machines models is implemented on the dataset. Cross validation is used as a method of resampling. The cost is varied between 0.25 to 2048.

Support Vector Machines with Radial Basis Function Kernel

215 samples

100 predictors

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 193, 195, 194, 191, 195, 194, ...

Resampling results across tuning parameters:

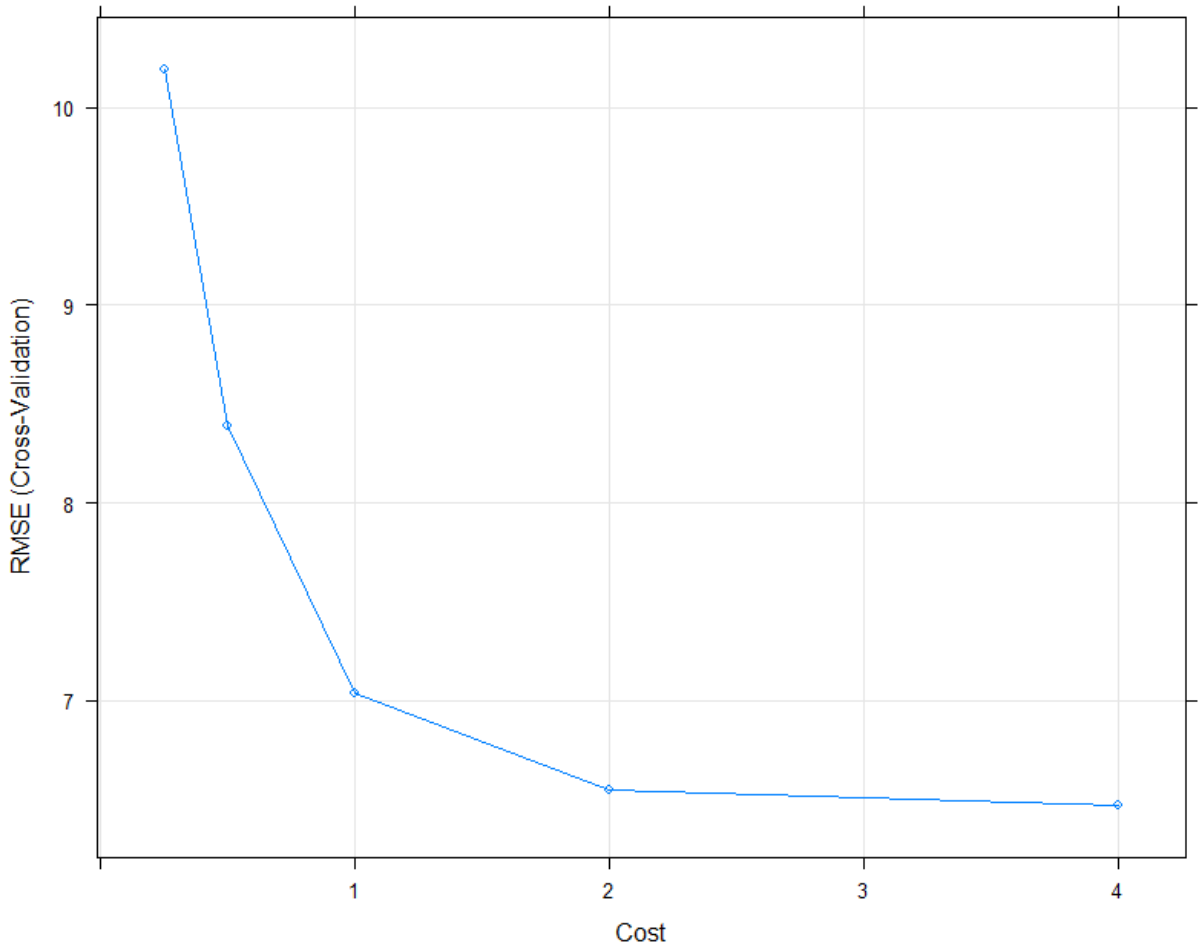
C	RMSE	Rsquared	MAE
0.25	10.193099	0.5580493	7.771910
0.50	8.387471	0.6529221	6.389346
1.00	7.037856	0.7219266	5.132887
2.00	6.545076	0.7565520	4.654613
4.00	6.467957	0.7612467	4.533934
8.00	6.468033	0.7610850	4.530353

16.00 6.468033 0.7610850 4.530353

32.00 6.468033 0.7610850 4.530353

64.00 6.468033 0.7610850 4.530353

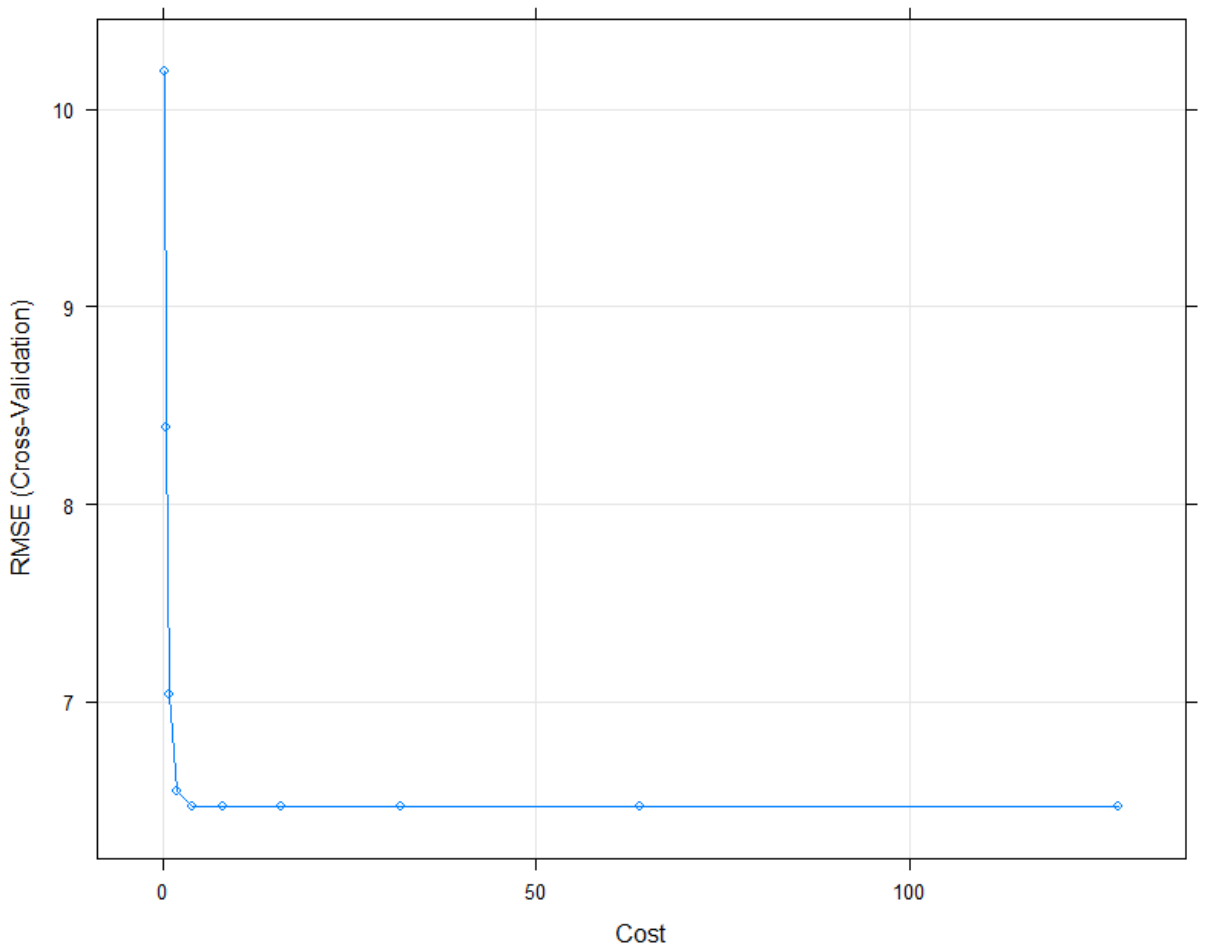
128.00 6.468033 0.7610850 4.530353



Tuning parameter 'sigma' was held constant at a value of 0.07551704

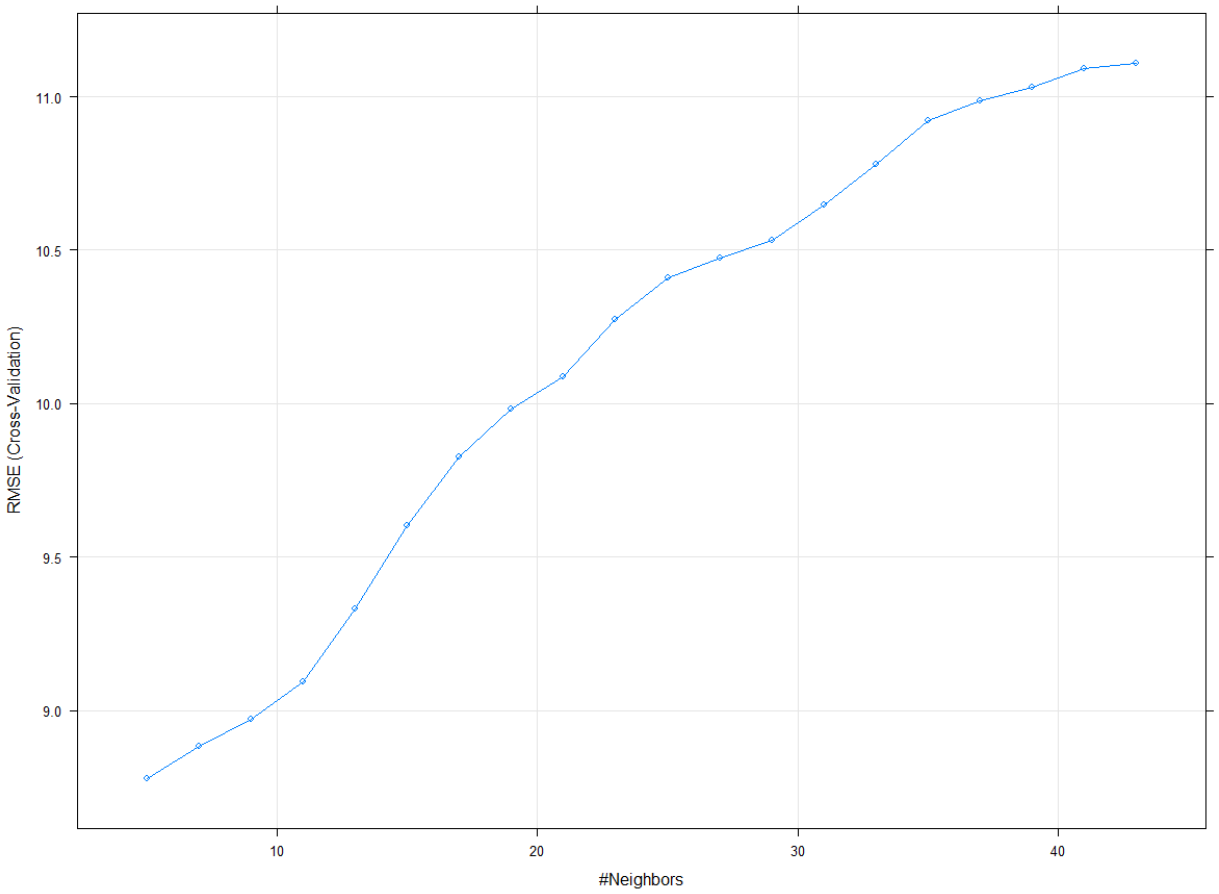
RMSE was used to select the optimal model using the smallest value.

The final values used for the model were sigma = 0.07551704 and C = 4.



knnTune

The KNN model gives an RMSE of 8.780210 and an R squared value 0.5333011,while choosing 5 nearest neighbours to build the model .



k	RMSE	Rsquared	MAE
5	8.780210	0.5333011	6.698787
7	8.884787	0.5383953	6.991618
9	8.974267	0.5366581	7.219121
11	9.095232	0.5343345	7.341312
13	9.331446	0.5117094	7.515659
15	9.601812	0.4766180	7.708154
17	9.828624	0.4468177	7.955859
19	9.982033	0.4273794	8.114174

21	10.087219	0.4107715	8.230006
23	10.274056	0.3872874	8.412787
25	10.409013	0.3704257	8.521308
27	10.475082	0.3602809	8.577461
29	10.532058	0.3531452	8.618004
31	10.647141	0.3375268	8.718320
33	10.778974	0.3171899	8.815088
35	10.922960	0.2972822	8.906369
37	10.986611	0.2866261	8.982137
39	11.029712	0.2818986	9.001255
41	11.091218	0.2708245	9.050966
43	11.107703	0.2682348	9.076985

RMSE was used to select the optimal model using the smallest value.

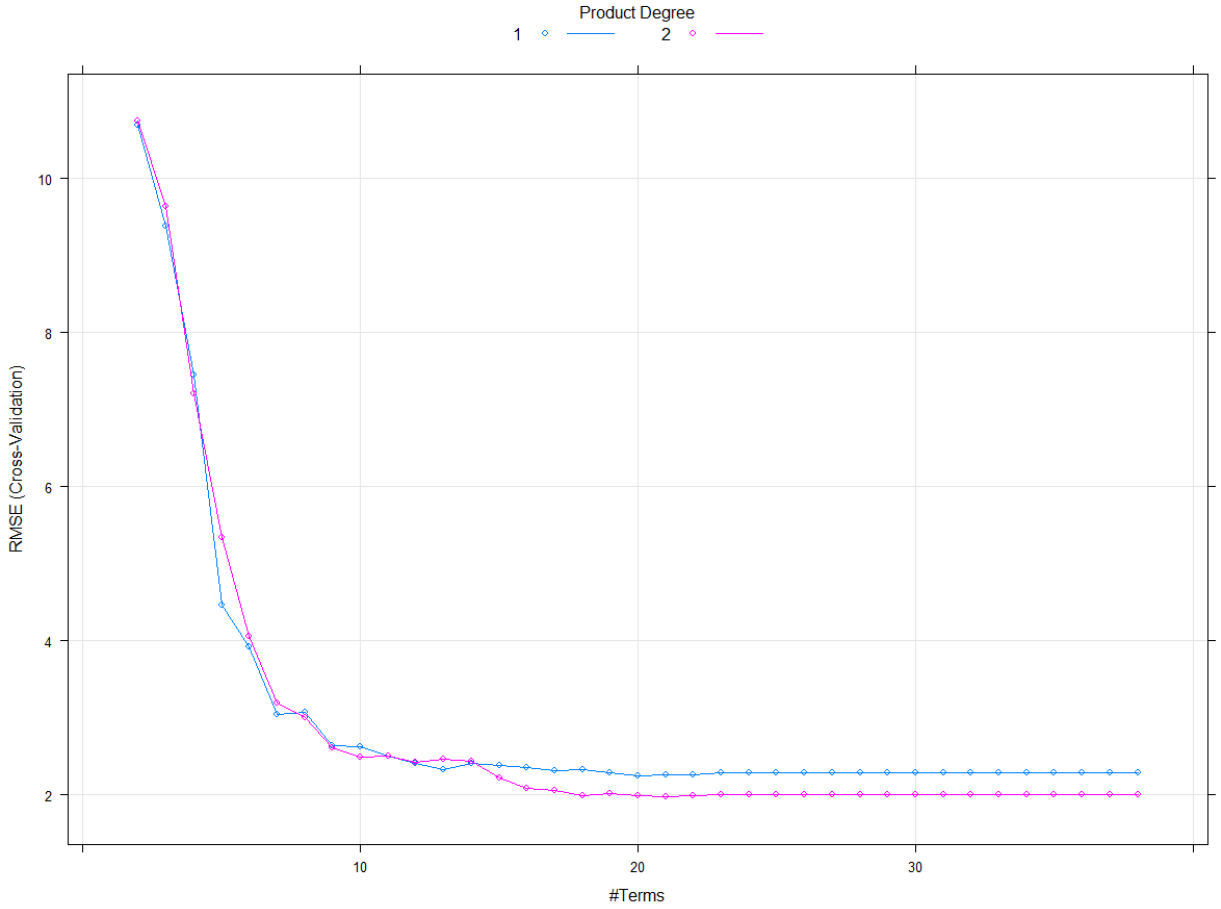
The final value used for the model was k = 5.

MARS

The mars model selects only 20 most important variables with varying importance however after the first 9 variables the importance declines to zero. X41 is the most important predictor and the nprune = 21 and degree = 2. Were selected for building the model.

The RMSE AND R Squared values are: 1.972876 , 0.9782536

The Mars model was selected by varying the degree between 1 and 2 and the nprune



parameters between 2 and 38.

Multivariate Adaptive Regression Spline

215 samples

2	21	1.972876	0.9782536	1.460436
2	22	1.993099	0.9775624	1.463526
2	23	2.006776	0.9772025	1.469213
2	24	2.009126	0.9770505	1.475065
2	25	2.009126	0.9770505	1.475065

2	26	2.009126	0.9770505	1.475065
2	27	2.009126	0.9770505	1.475065
2	28	2.009126	0.9770505	1.475065
2	29	2.009126	0.9770505	1.475065
2	30	2.009126	0.9770505	1.475065
2	31	2.009126	0.9770505	1.475065
2	32	2.009126	0.9770505	1.475065
2	33	2.009126	0.9770505	1.475065
2	34	2.009126	0.9770505	1.475065
2	35	2.009126	0.9770505	1.475065
2	36	2.009126	0.9770505	1.475065
2	37	2.009126	0.9770505	1.475065
2	38	2.009126	0.9770505	1.475065

RMSE was used to select the optimal model using the smallest value.

The final values used for the model were nprune = 21 and degree = 2.

earth variable importance

only 20 most important variables shown (out of 100)

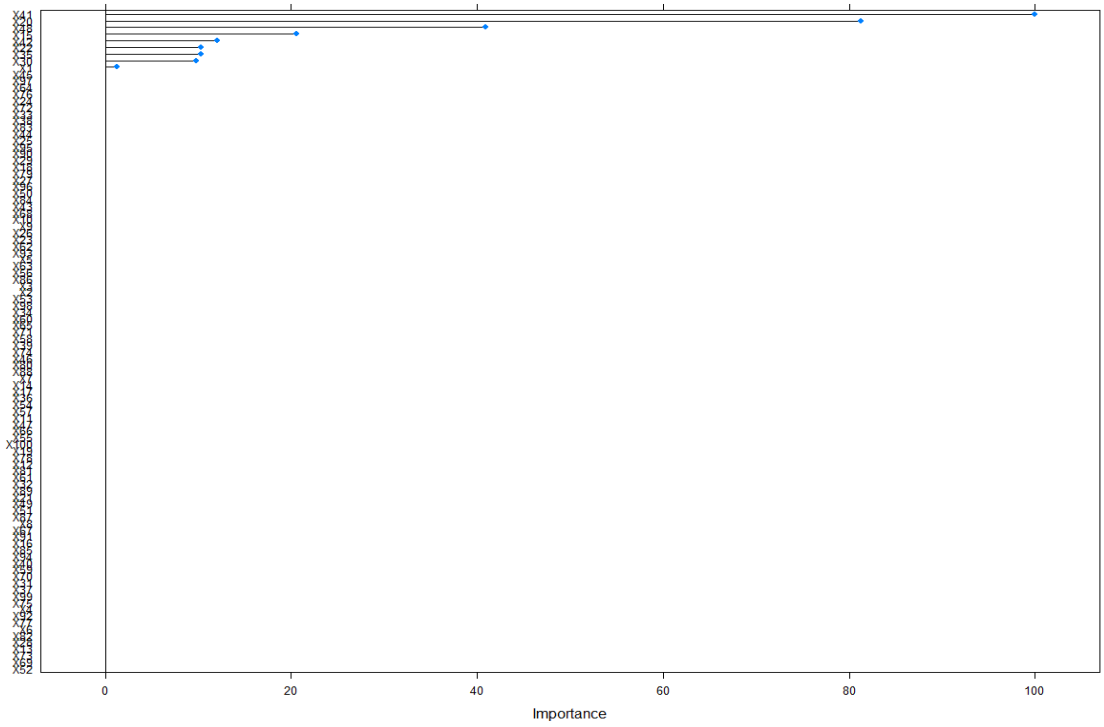
	Overall
X41	100.000
X20	81.242
X48	40.917
X15	20.519
X42	12.047

X35 10.315

X22 10.315

X30 9.813

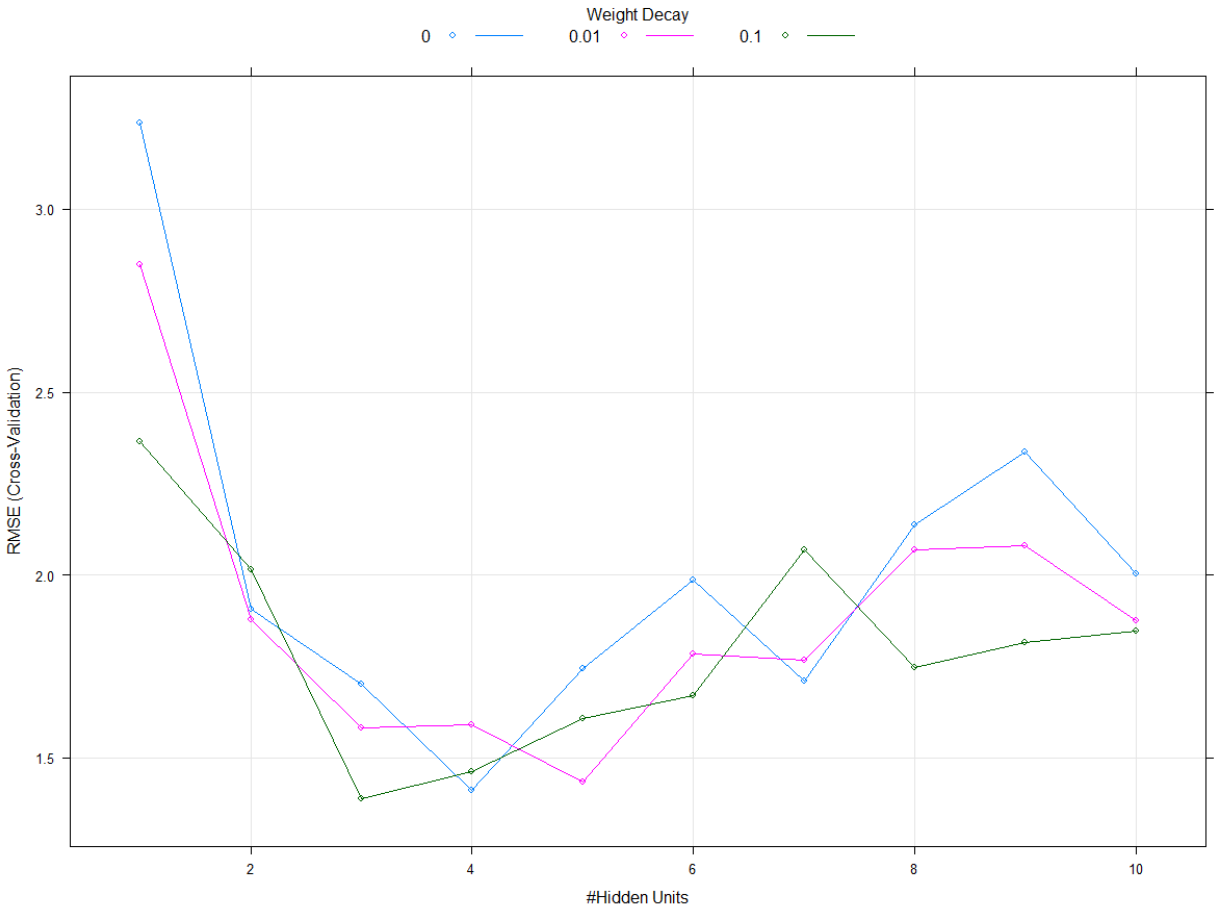
X1 1.281



Model Averaged Neural Network

The model for Neural nets was built by selecting decay 0.01 and hidden layers as 3.

The RMSE AND RSQUARED VALUES ARE 1.581738 0.9868916



215 samples

100 predictors

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 193, 194, 192, 192, 195, 195, ...

Resampling results across tuning parameters:

decay	size	RMSE	Rsquared	MAE
0.00	1	3.234945	0.9488162	2.586226
0.00	2	1.907007	0.9790177	1.548773
0.00	3	1.701299	0.9842933	1.342814

0.00	4	1.411713	0.9885381	1.084004
0.00	5	1.744212	0.9822038	1.276075
0.00	6	1.987953	0.9780812	1.470259
0.00	7	1.709982	0.9826391	1.309020
0.00	8	2.137641	0.9746444	1.691708
0.00	9	2.338168	0.9712784	1.742958
0.00	10	2.003491	0.9786260	1.497123
0.01	1	2.850271	0.9544328	2.295549
0.01	2	1.878823	0.9798539	1.464271
0.01	3	1.581738	0.9868916	1.252734
0.01	4	1.592006	0.9864053	1.207308
0.01	5	1.435123	0.9874195	1.057463
0.01	6	1.783559	0.9806659	1.383798
0.01	7	1.768570	0.9829623	1.406348
0.01	8	2.069736	0.9765132	1.565754
0.01	9	2.080322	0.9761301	1.589405
0.01	10	1.875235	0.9804005	1.431009
0.10	1	2.364841	0.9685247	1.887708
0.10	2	2.015915	0.9779321	1.566647
0.10	3	1.387862	0.9893655	1.083958
0.10	4	1.464168	0.9875389	1.117683
0.10	5	1.607660	0.9858702	1.221889
0.10	6	1.671668	0.9848870	1.283428
0.10	7	2.069226	0.9752655	1.560308
0.10	8	1.747452	0.9831352	1.376434
0.10	9	1.815825	0.9817304	1.387558

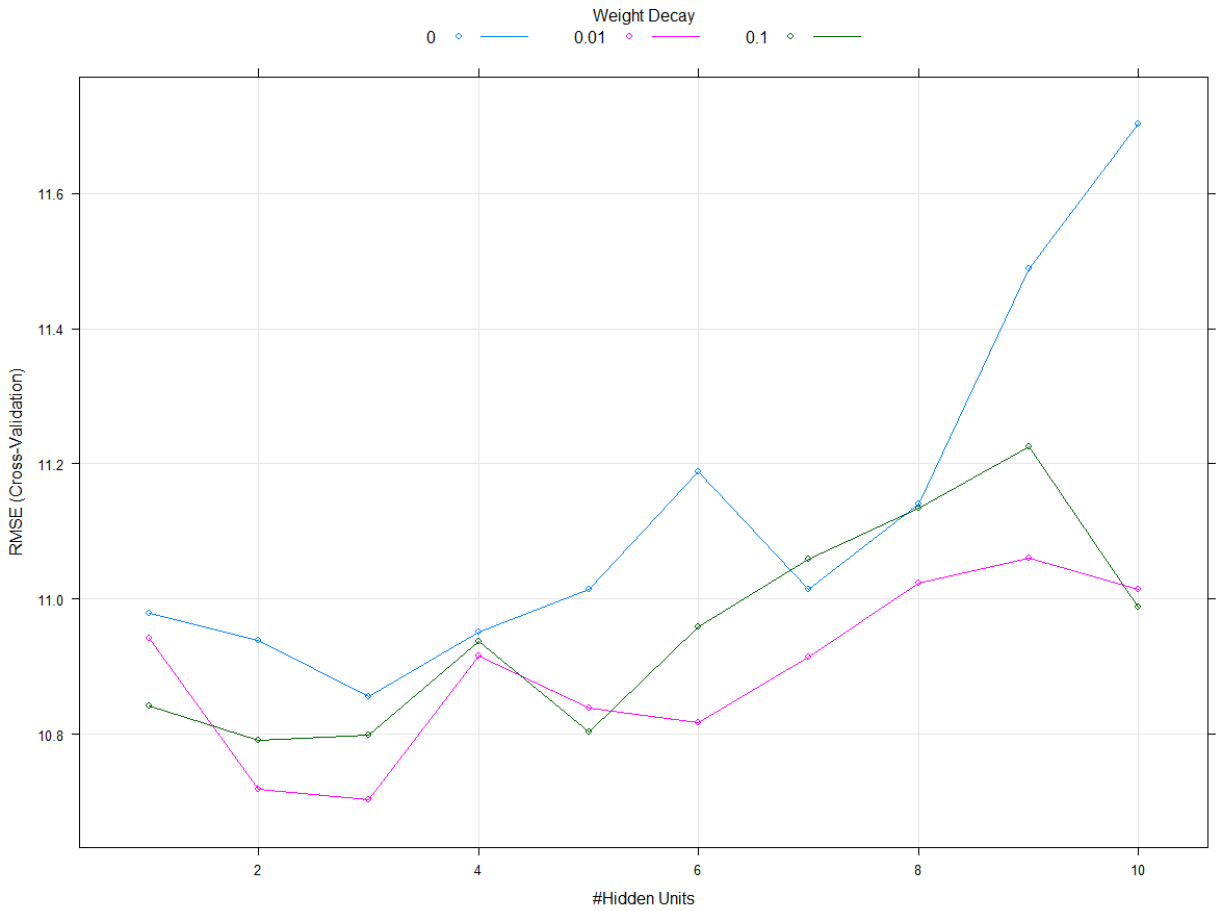
0.10 10 1.847947 0.9807612 1.417470

Tuning parameter 'bag' was held constant at a value of FALSE

RMSE was used to select the optimal model using the smallest value.

The final values used for the model were size = 3, decay = 0.1 and bag = FALSE.

Neural Net with PCA



Neural nets were used by preprocessing the data using PCA after applying PCA we saw that only two components were selected

```
nnetPCATune$preProcess$numComp
```

[1] 2.

The best model was selected for decay wt =0.01 and hidden layers = 3.

The RMSE and R squared value is : 10.70285 0.3008829

Model Averaged Neural Network

215 samples

100 predictors

Pre-processing: principal component signal extraction (100), centered (100), scaled (100)

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 193, 194, 192, 192, 195, 195, ...

Resampling results across tuning parameters:

decay	size	RMSE	Rsquared	MAE
0.00	1	10.97841	0.2640081	8.749077
0.00	2	10.93861	0.2691193	8.601411
0.00	3	10.85509	0.2809977	8.527072
0.00	4	10.95173	0.2679497	8.629795
0.00	5	11.01439	0.2762486	8.494912
0.00	6	11.18816	0.2451196	8.828184
0.00	7	11.01374	0.2673472	8.613260
0.00	8	11.14131	0.2557707	8.785943

0.00	9	11.48798	0.2262443	9.082101
0.00	10	11.70180	0.2118392	9.189857
0.01	1	10.94164	0.2736227	8.655760
0.01	2	10.71882	0.3028835	8.375490
0.01	3	10.70285	0.3008829	8.351246
0.01	4	10.91638	0.2748910	8.568254
0.01	5	10.83887	0.2816737	8.415262
0.01	6	10.81754	0.2874612	8.550622
0.01	7	10.91449	0.2706922	8.673745
0.01	8	11.02391	0.2709411	8.666594
0.01	9	11.06126	0.2662776	8.727002
0.01	10	11.01473	0.2720034	8.762959
0.10	1	10.84238	0.2833585	8.557095
0.10	2	10.79188	0.2936743	8.506744
0.10	3	10.79825	0.2868583	8.426278
0.10	4	10.93781	0.2724513	8.658717
0.10	5	10.80374	0.2865971	8.514491
0.10	6	10.95971	0.2745448	8.580460
0.10	7	11.05831	0.2613470	8.705831
0.10	8	11.13480	0.2609395	8.700019
0.10	9	11.22595	0.2440265	8.881080
0.10	10	10.98895	0.2699233	8.606101

Tuning parameter 'bag' was held constant at a value of FALSE

RMSE was used to select the optimal model using the smallest value.

The final values used for the model were size = 3, decay = 0.01 and bag = FALSE.

Before implementing the models we preprocessed each of the models using centering and scaling to treat the data.

Then we implemented various models on the data:

The table below shows the RMSE and R squared performances of each of the models .

SVM: 6.467957 0.7612467

KNN: 8.780210 .5333011

MARS 1.972876 , 0.9782536

NNET 1.387862 0.9893655

NNET-PCA: 10.70285 0.3008829

From the above table we can see that the NNEt model with pCA does not perform better than the model used without preprocessing. MARS selects 9 important predictors and the rest of the predictors are not used as they show zero importance.

The Mars and the Neural Net models are the best performing models that we can implement however we also have to note that the linear models that we used in the previous exercise for this dataset performed better.

Additionally we can also conclude that dimension reduction using PCA preprocessing does not help get good results for this dataset.

7.4 Return to the permeability problem outlined in Exercise 6.2. Train several non-linear regression models and evaluate the resampling and test set performance.

(a) Which non-linear regression model gives the optimal resampling and test set performance?

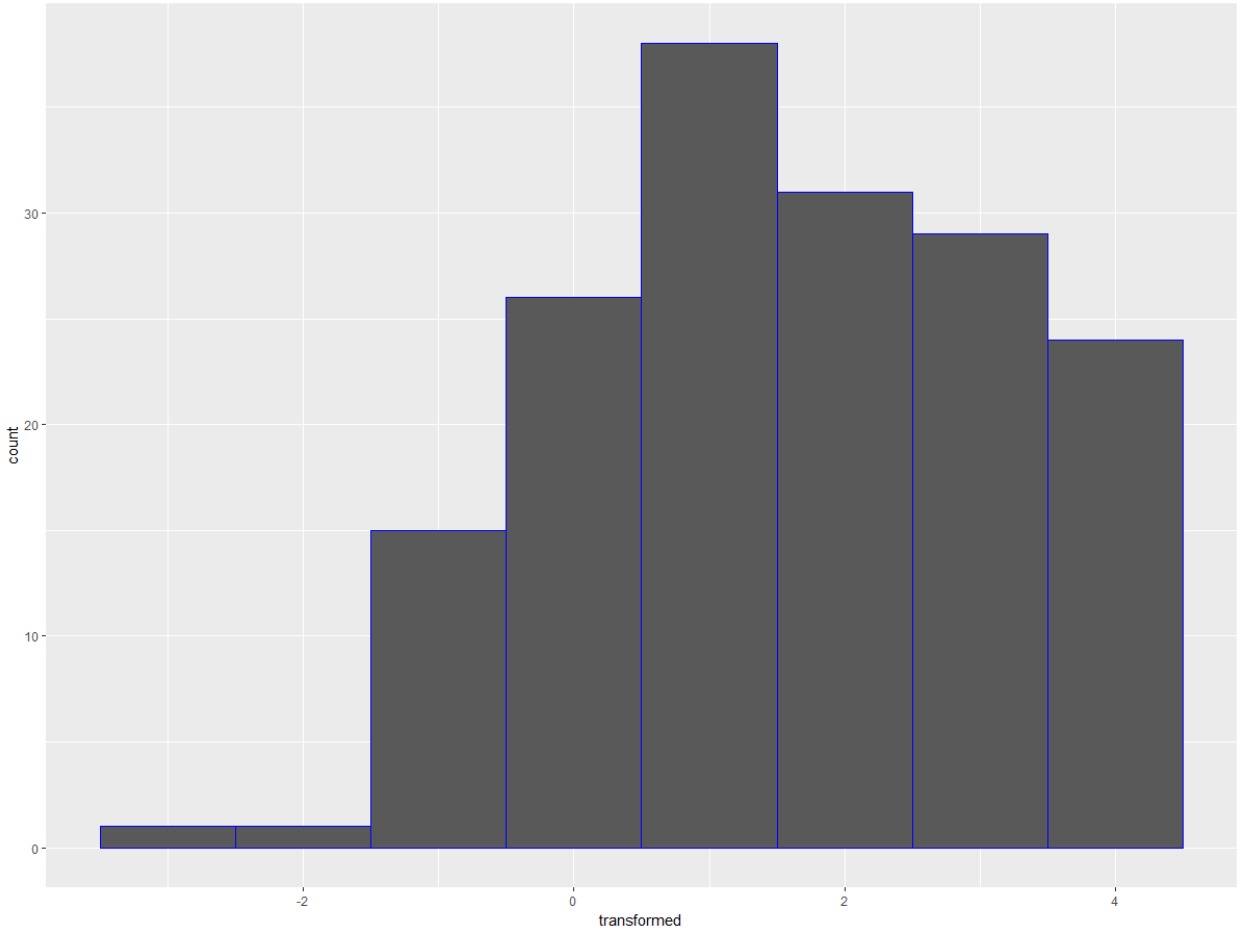
(b) Do any of the non{linear models outperform the optimal linear model you previously

**developed in Exercise 6.2? If so, what might this tell you about the underlying relationship
between the predictors and the response?**

(c) Would you recommend any of the models you have developed to replace the permeability Laboratory experiment?

Solution:

The data for this is much skewed and hence we have to preprocess it. We need to eliminate the near zero variance and zero variance variables as we did previously. Additionally e also have to eliminate the skewness of the data we do this by using box cox transformation.
Which selects the log 10 transformation for the data. Here is a histogram of the data after preprocessing.



SVM model:

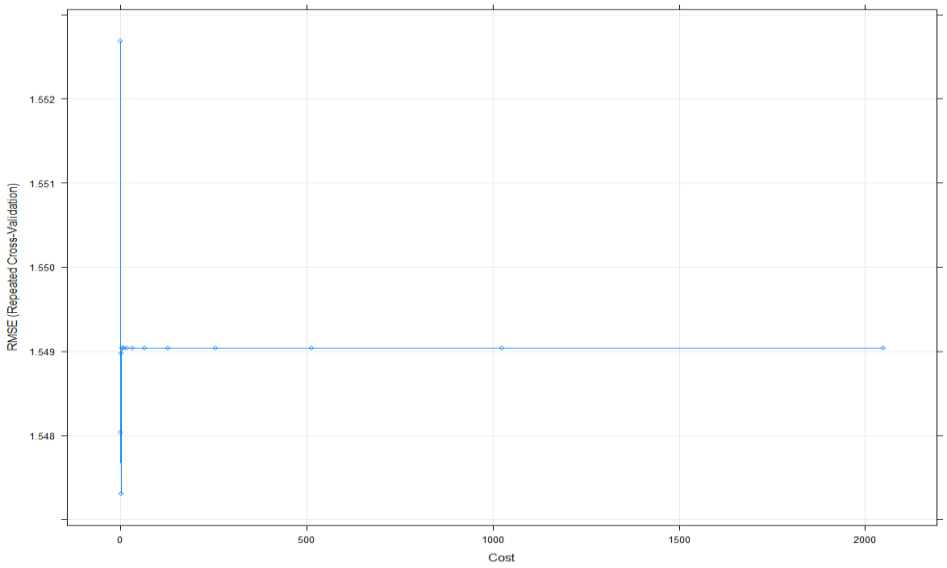
```
Support Vector Machines with Radial Basis Function Kernel

133 samples
388 predictors

No pre-processing
```

```
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 105, 106, 107, 108, 106, 107, ...
Resampling results across tuning parameters:

  C        RMSE      Rsquared    MAE
     0.25  1.552682  0.03659310  1.314273
     0.50  1.548037  0.03739932  1.305731
     1.00  1.547310  0.03857735  1.308094
     2.00  1.548972  0.03935241  1.311279
     4.00  1.549036  0.03907852  1.311291
     8.00  1.549036  0.03907852  1.311291
    16.00  1.549036  0.03907928  1.311291
    32.00  1.549036  0.03907852  1.311291
    64.00  1.549036  0.03907852  1.311291
   128.00  1.549036  0.03907852  1.311291
   256.00  1.549036  0.03907928  1.311291
   512.00  1.549036  0.03907852  1.311291
  1024.00  1.549036  0.03907928  1.311291
  2048.00  1.549036  0.03907852  1.311291

Tuning parameter 'sigma' was held constant at a value of 0.001487247
RMSE was used to select the optimal model using  the smallest value.
The final values used for the model were sigma = 0.001487247 and C = 1.
```

The SVM models selects as the optimal value and the RMSE value is 1.547310 the r squared value is 0.03857735.

Below is the graph for the RMSE values against the cost. It is evident that the RMSE values do not vary significantly and lie in the same range for most of the costs. Additionally the R squared values are also not very convincing so we probe further to find a more optimal model to justify the data.

We had used 5 fold cross validation and repeated it 5 times to get to this result since this result seems a little awry we try other methods we will try svm with varying sigma values.



The next model we use svm tune to tune the values for sigma and cost

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.00252465 and C = 1.

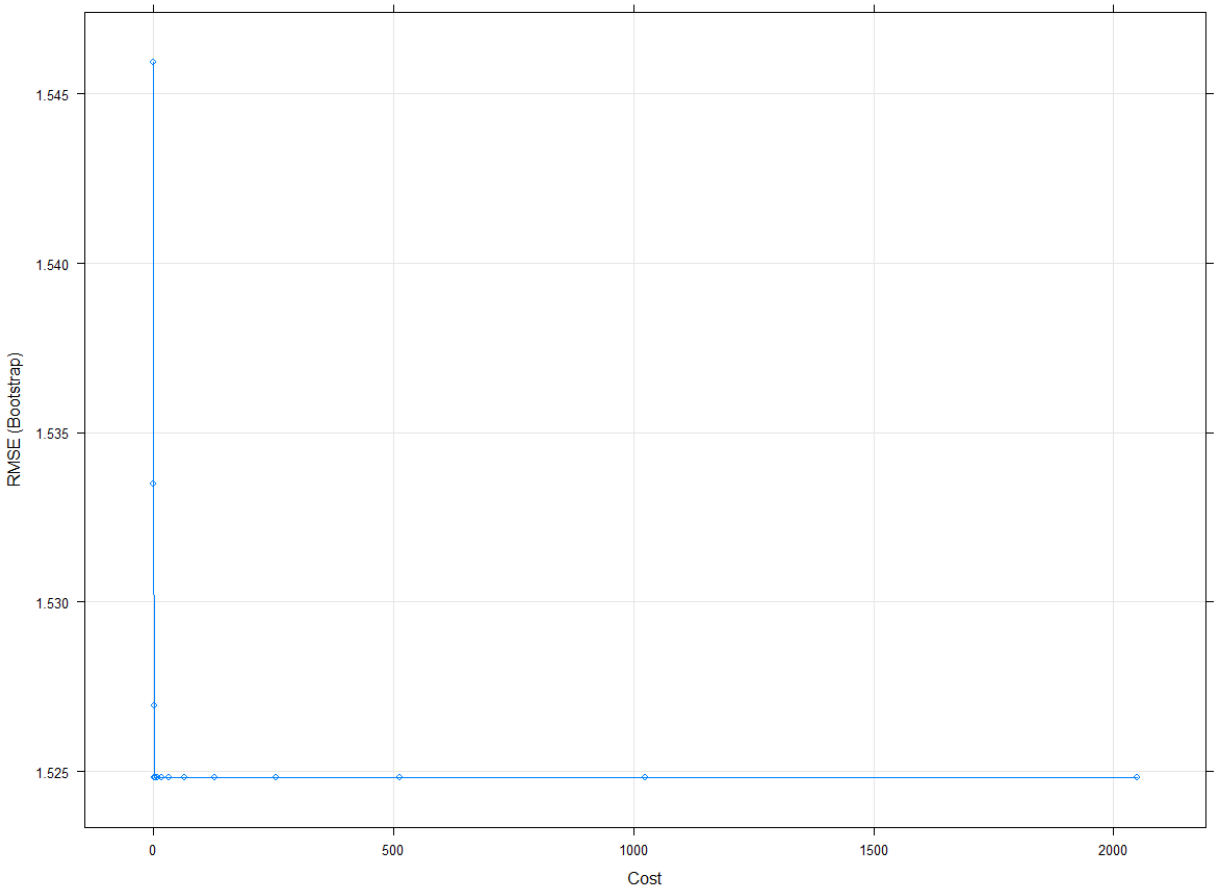
The best value selected for RMSE and R squared based on the variation's is
1.547283 and 0.06347970.

We tried the bootstrapping method for SVM on this dataset we relized that the R squared value was affected negatively hence this might not be the best case scenario for this model. The cost R MSE and R squared values were:
16.00 1.524828 0.01898360.

```
Next we tried more variations with the SVM model:
133 samples
388 predictors

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 108, 106, 107, 105, 106, 107, ...
Resampling results across tuning parameters:

  sigma   C   RMSE      Rsquared    MAE
  0.0005   1  1.549613  0.04078199  1.312911
  0.0005   7  1.551497  0.04225438  1.316190
  0.0005  13  1.551497  0.04225438  1.316190
  0.0005  19  1.551497  0.04225438  1.316190
  0.0005  25  1.551497  0.04225438  1.316190
  0.0005  31  1.551497  0.04225438  1.316190
  0.0005  37  1.551497  0.04225439  1.316190
  0.0005  43  1.551497  0.04225439  1.316190
```

0.0005	49	1.551497	0.04225439	1.316190
0.0010	1	1.549613	0.04078199	1.312911
0.0010	7	1.551497	0.04225438	1.316190
0.0010	13	1.551497	0.04225438	1.316190
0.0010	19	1.551497	0.04225438	1.316190
0.0010	25	1.551497	0.04225438	1.316190
0.0010	31	1.551497	0.04225438	1.316190
0.0010	37	1.551497	0.04225439	1.316190
0.0010	43	1.551497	0.04225439	1.316190
0.0010	49	1.551497	0.04225439	1.316190
0.0015	1	1.549613	0.04078199	1.312911
0.0015	7	1.551497	0.04225438	1.316190
0.0015	13	1.551497	0.04225438	1.316190
0.0015	19	1.551497	0.04225438	1.316190
0.0015	25	1.551497	0.04225438	1.316190
0.0015	31	1.551497	0.04225438	1.316190
0.0015	37	1.551497	0.04225439	1.316190
0.0015	43	1.551497	0.04225439	1.316190
0.0015	49	1.551497	0.04225439	1.316190

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.0015 and C = 1.



We observe that by varying the parameters the RMSE value of the model the RMS
E values was varied only a little bit and there were no drastic changes.

Next we tried another variation in the SVM model using LGOVC as a resampling method and go t the following result:

```
Support Vector Machines with Radial Basis Function Kernel

133 samples
388 predictors

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 101, 101, 101, 101, 101, 101, ...
Resampling results across tuning parameters:

  sigma   C   RMSE      Rsquared    MAE
  0.0005   1  1.538781  0.02696913  1.300105
  0.0005   7  1.540738  0.02729205  1.303368
  0.0005  13  1.540738  0.02729211  1.303368
  0.0005  19  1.540738  0.02729209  1.303368
  0.0005  25  1.540738  0.02729209  1.303368
  0.0005  31  1.540738  0.02729209  1.303368
  0.0005  37  1.540738  0.02729211  1.303368
  0.0005  43  1.540738  0.02729211  1.303368
  0.0005  49  1.540738  0.02729211  1.303368
  0.0010   1  1.538781  0.02696913  1.300105
  0.0010   7  1.540738  0.02729205  1.303368
  0.0010  13  1.540738  0.02729211  1.303368
  0.0010  19  1.540738  0.02729209  1.303368
  0.0010  25  1.540738  0.02729209  1.303368
  0.0010  31  1.540738  0.02729209  1.303368
  0.0010  37  1.540738  0.02729211  1.303368
  0.0010  43  1.540738  0.02729211  1.303368
  0.0010  49  1.540738  0.02729211  1.303368
  0.0015   1  1.538781  0.02696913  1.300105
  0.0015   7  1.540738  0.02729205  1.303368
  0.0015  13  1.540738  0.02729211  1.303368
  0.0015  19  1.540738  0.02729209  1.303368
  0.0015  25  1.540738  0.02729209  1.303368
  0.0015  31  1.540738  0.02729209  1.303368
  0.0015  37  1.540738  0.02729211  1.303368
  0.0015  43  1.540738  0.02729211  1.303368
  0.0015  49  1.540738  0.02729211  1.303368
```

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.0015 and C = 1.
After using another method for resampling we still get the same results for the sigma values as w ell as the cost function.
RMSE is 1.538781 and R squared value is 0.02696913.

In conclusion we can inference that the RMSE value for the various variations in the resampling methods as well as tuning parameters result in the SVM models whose RMSE values are betwee n 1.5 and 1.6 and the R squared values are between 0.02 to 0.06. The r squared value for these m odels is very very small and hence is cannot justify the data adequately.

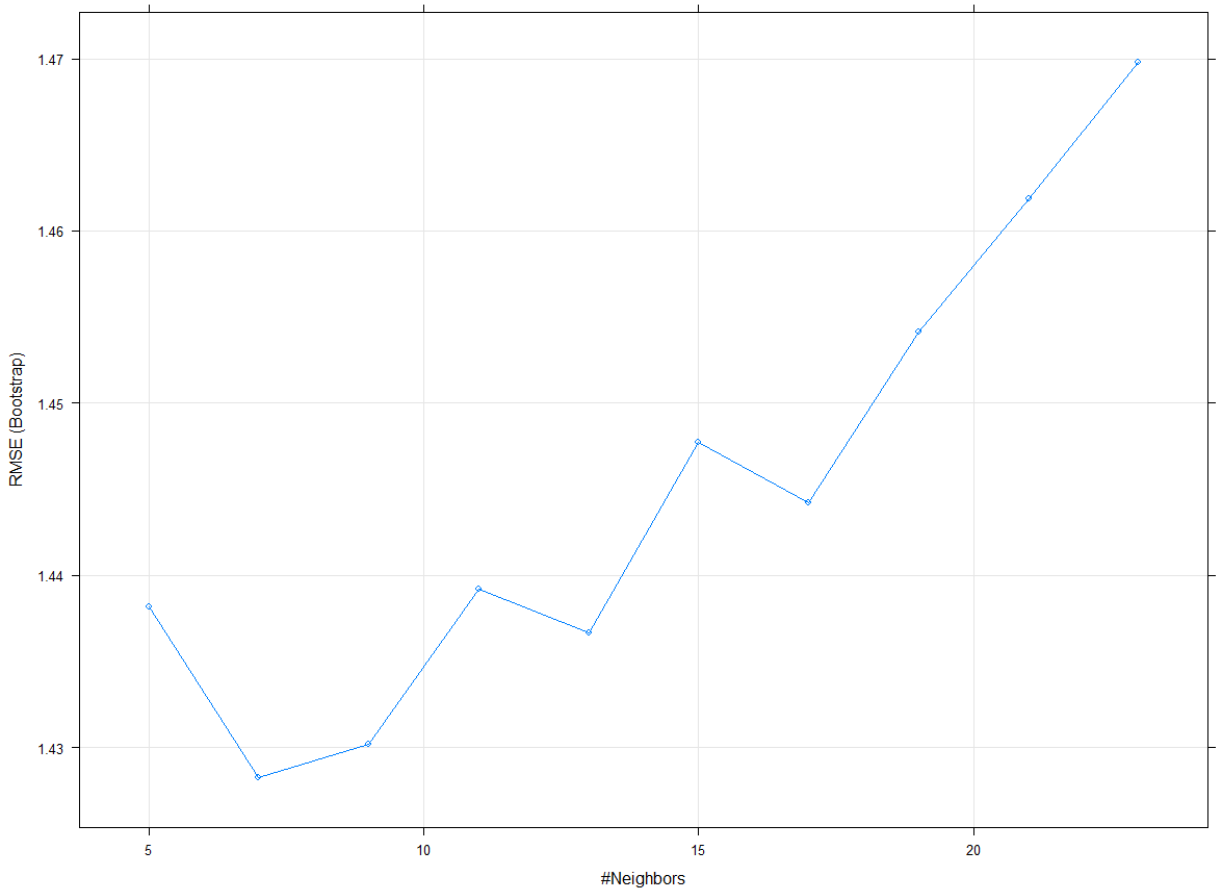
We can also see that by varying the resampling methods the results do not change drastically.

KNN model:

```
k-Nearest Neighbors

133 samples
388 predictors

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 133, 133, 133, 133, 133, 133, ...
Resampling results across tuning parameters:

  k   RMSE      Rsquared   MAE
   5  1.438197  0.2481221  1.080424
   7  1.428266  0.2373852  1.086164
   9  1.430195  0.2294311  1.096410
  11  1.439193  0.2215671  1.107282
  13  1.436679  0.2161555  1.121260
  15  1.447742  0.2044226  1.137517
  17  1.444226  0.2066426  1.138589
  19  1.454114  0.1984478  1.158368
  21  1.461851  0.1903182  1.170021
  23  1.469779  0.1833534  1.179617

RMSE was used to select the optimal model using  the smallest value.
The final value used for the model was k = 7
```

The RMSE and R squared values for the knn model are 1.428266 0.2373852

We see that the performance for r squared has increased considerably over the SVM model and the RMSE is smaller as well.

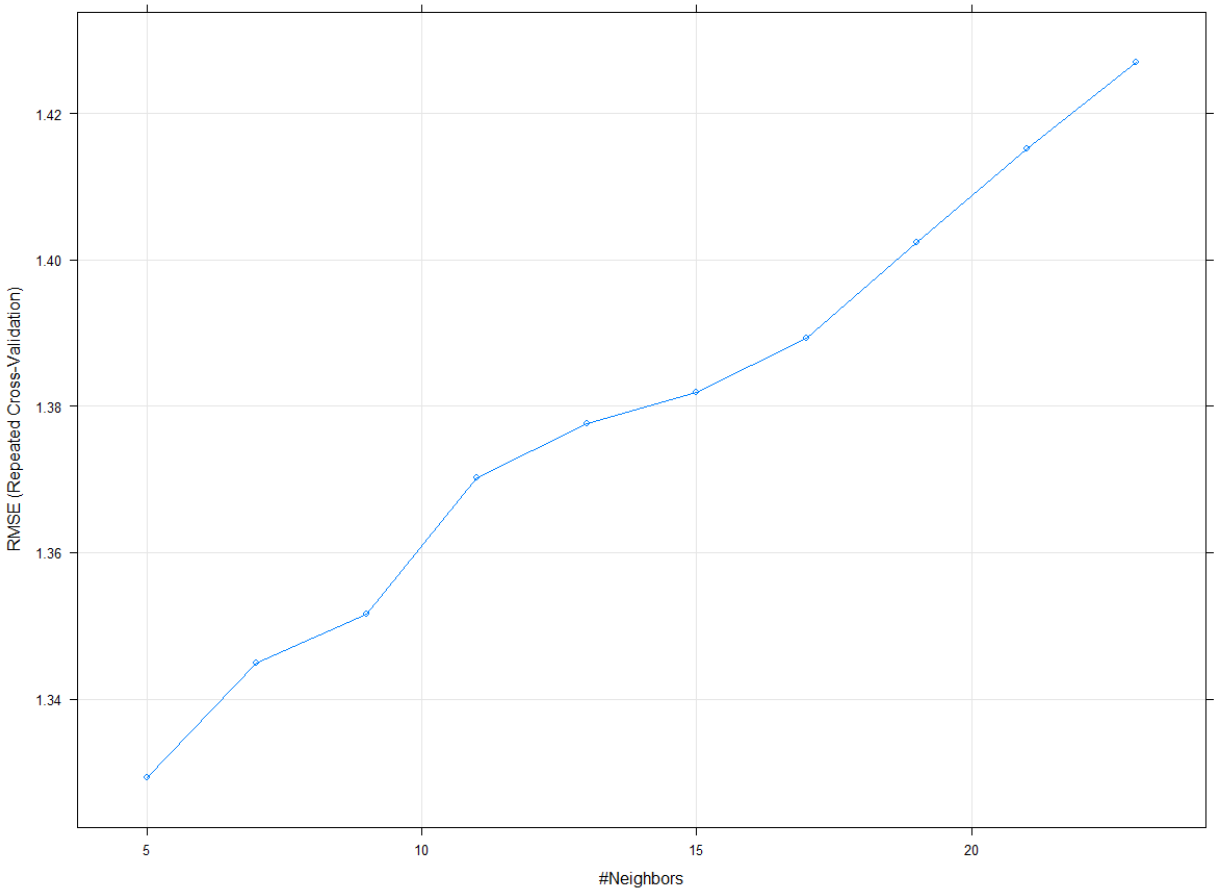


Next we select the 5 fold sampling with 5 repeats method for resampling and we perform KNN again.

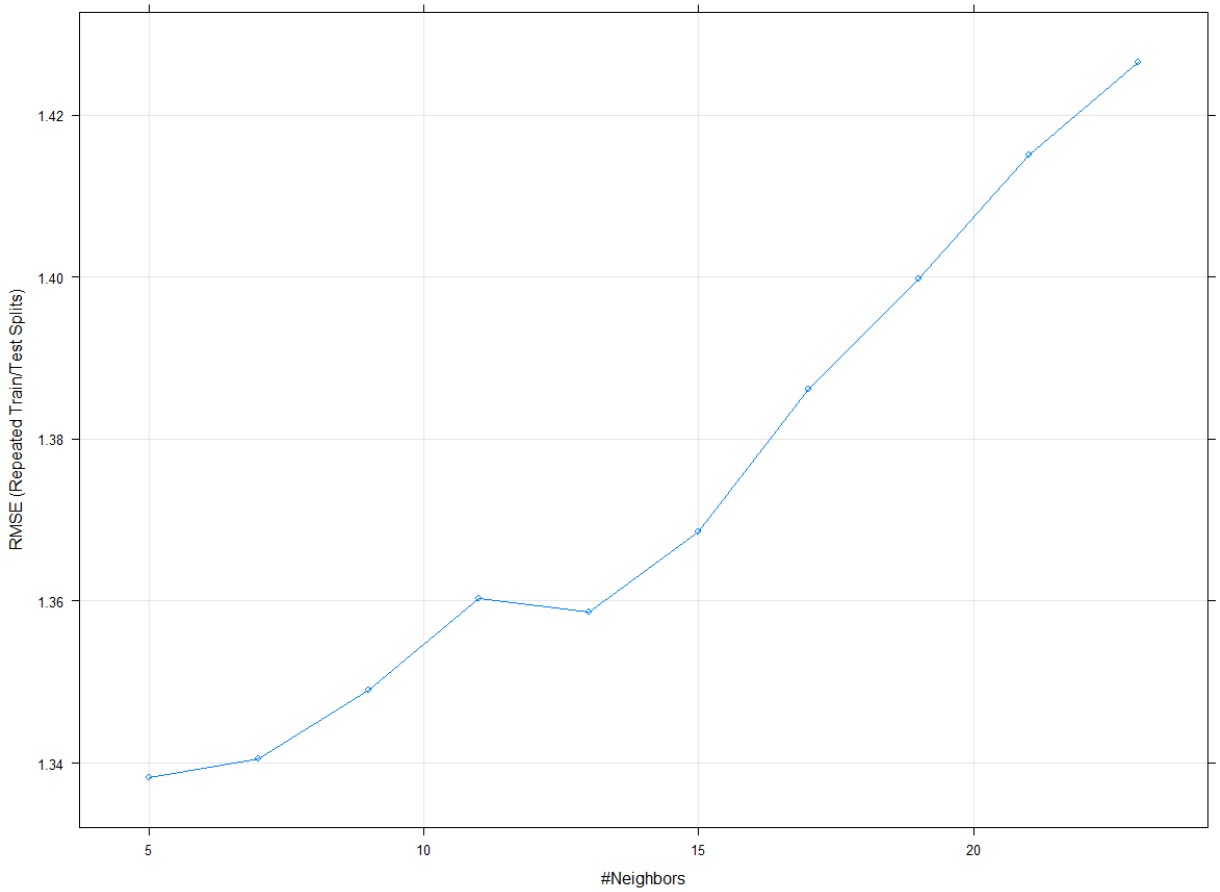
```
k-Nearest Neighbors

133 samples
388 predictors

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 107, 106, 107, 106, 106, 106, ...
Resampling results across tuning parameters:

  k   RMSE      Rsquared   MAE
   5  1.329302  0.3052201  1.011487
   7  1.344869  0.2893435  1.037369
   9  1.351613  0.2843094  1.059152
  11  1.370232  0.2638195  1.086140
  13  1.377584  0.2516275  1.104364
  15  1.381865  0.2427193  1.118744
  17  1.389291  0.2370800  1.128237
  19  1.402358  0.2258147  1.142561
  21  1.415196  0.2131469  1.159673
  23  1.426940  0.2015378  1.174784
```

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 5.

We see that the optimal parameters of the model do not change. K is still held at 5 . The RMSE value and the R squared value is: 1.329302 0.3052201 . This method of resampling provides a incremented performance for the model . This is further explained by the following graph.



The last resampling method we try is LGOCV:



The optimal parameters for this model is: that k=5 RMSE = 1.329302 and Rsquared= 0.3052201. Additionally we also notice that again the no of neighbors remains the same the RMSE and R squared values are also not drastically changed.

MARS model:

```
No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 106, 107, 107, 105, 107, 108, ...
Resampling results across tuning parameters:

  degree  nprune  RMSE      Rsquared   MAE
  1         2     1.326798  0.3085217  1.0309458
  1         7     1.159462  0.4851740  0.8980548
  1        12     1.240587  0.4427963  0.9708008
  1        17     1.257895  0.4369475  0.9731238
  1        22     1.271198  0.4464796  0.9877274
  1        27     1.274884  0.4456157  0.9889643
  1        32     1.274884  0.4456157  0.9889643
  1        37     1.274884  0.4456157  0.9889643
  1        42     1.274884  0.4456157  0.9889643
  1        47     1.274884  0.4456157  0.9889643
  1        52     1.274884  0.4456157  0.9889643
```

1	57	1.274884	0.4456157	0.9889643
1	62	1.274884	0.4456157	0.9889643
1	67	1.274884	0.4456157	0.9889643
1	72	1.274884	0.4456157	0.9889643
1	77	1.274884	0.4456157	0.9889643
1	82	1.274884	0.4456157	0.9889643
1	87	1.274884	0.4456157	0.9889643
1	92	1.274884	0.4456157	0.9889643
1	97	1.274884	0.4456157	0.9889643
1	102	1.274884	0.4456157	0.9889643
1	107	1.274884	0.4456157	0.9889643
1	112	1.274884	0.4456157	0.9889643
1	117	1.274884	0.4456157	0.9889643
1	122	1.274884	0.4456157	0.9889643
1	127	1.274884	0.4456157	0.9889643
2	2	1.326798	0.3085217	1.0309458
2	7	1.219006	0.4503619	0.9307615
2	12	1.323777	0.4053369	1.0067797
2	17	1.370705	0.3909310	1.0326870
2	22	1.411833	0.3798394	1.0679189
2	27	1.422288	0.3773070	1.0784232
2	32	1.422288	0.3773070	1.0784232
2	37	1.422288	0.3773070	1.0784232
2	42	1.422288	0.3773070	1.0784232
2	47	1.422288	0.3773070	1.0784232
2	52	1.422288	0.3773070	1.0784232
2	57	1.422288	0.3773070	1.0784232
2	62	1.422288	0.3773070	1.0784232
2	67	1.422288	0.3773070	1.0784232
2	72	1.422288	0.3773070	1.0784232
2	77	1.422288	0.3773070	1.0784232
2	82	1.422288	0.3773070	1.0784232
2	87	1.422288	0.3773070	1.0784232
2	92	1.422288	0.3773070	1.0784232
2	97	1.422288	0.3773070	1.0784232
2	102	1.422288	0.3773070	1.0784232
2	107	1.422288	0.3773070	1.0784232
2	112	1.422288	0.3773070	1.0784232
2	117	1.422288	0.3773070	1.0784232
2	122	1.422288	0.3773070	1.0784232
2	127	1.422288	0.3773070	1.0784232

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nprune = 7 and degree = 1.

The predictors selected for the mars model are :

Overall
X6 100.000
X157 62.054
X93 43.810
X698 30.267
X340 17.023
X592 4.606

The rest of the predictors have 0 importance.

The optimal RMSE and R squared are as follows : 1.159462 0.4851740 . We tried bootstrapping as well as LGOCV methods for resampling however these methods also showed similar results.

RMSE and R squared comparison for various models:

SVM 1.549613 0.04078199

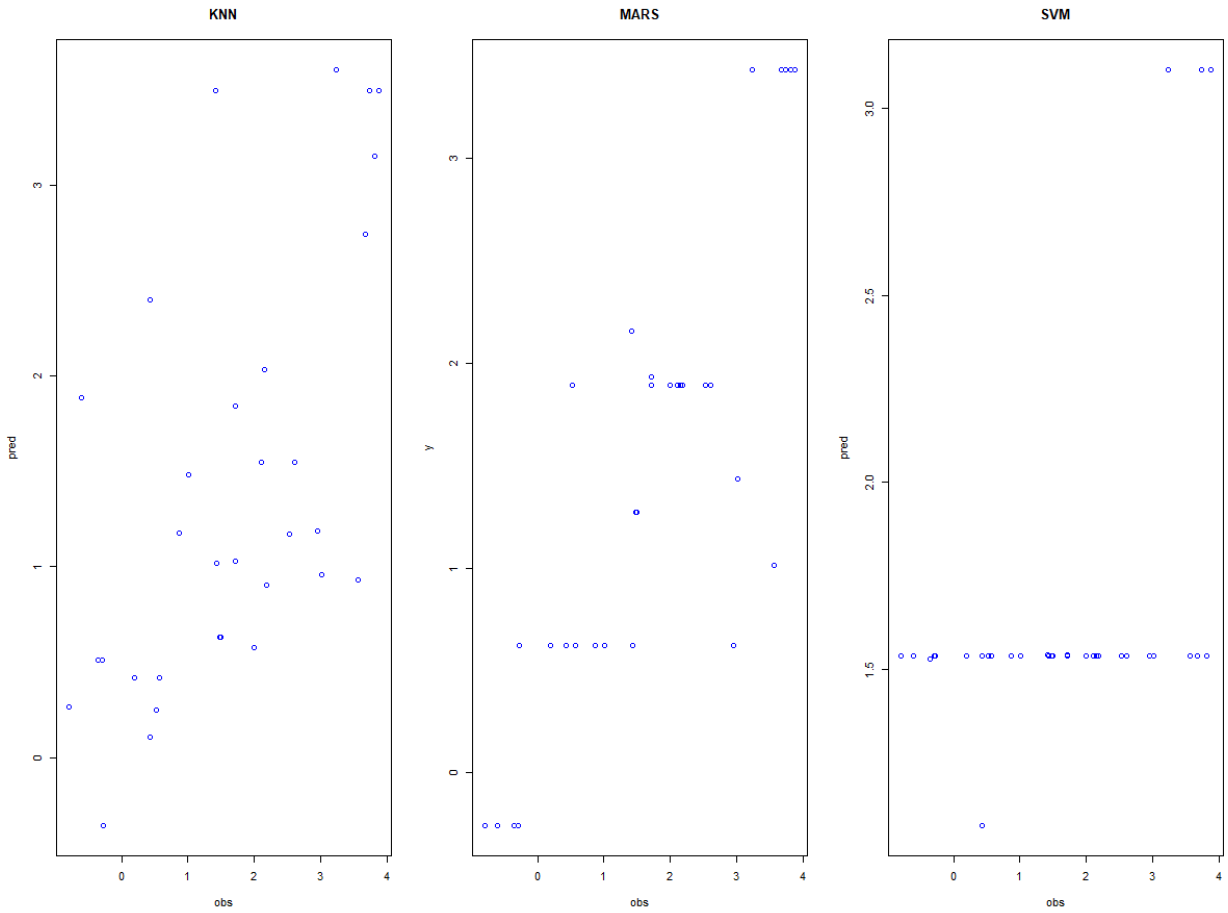
KNN 1.329302 0.3052201

MARS 1.159462 0.4851740

Based on model building until now the MARS model seems to be performing the best for this set of data let us on test it out on the data set to see if the test set is justified by this model.

	RMSE	Rsquared
KNN	1.1533646	0.3716735
SVM	1.2382182	0.2246186
MARS	0.8126192	0.67962

The above table shows that even on the test set Mars is the best performing model. The graphs below show the results for observed vs expected. In addition, none of t he models seems to be performing very for this dataset.



(a) Which non-linear regression model gives the optimal resampling and test set performance?

As the test data shows the MARS model gives the most optimal results with r squared being 0.67 and RMSE being 0.81 using 5 fold cross validation.The observed vs expected graph for the Mars model also looks finer than the other models.

(b) Do any of the non{linear models outperform the optimal linear model you previously developed in Exercise 6.2? If so, what might this tell you about the underlying relationship between the predictors and the response?

ENET: R square: 0.7440922 RMSE: 0.7467911 performed really well for the last homework, how ever comparing tboth the RMSE as well as the R squared values the current model(MARS) does not outperform the previous model.

(c) Would you recommend any of the models you have developed to replace the permeability Laboratory experiment?

None of the methods were very accurate to predict the outcome. Most methods have R squared value in the 0.3 to 0.6 range which means about 40 percent of the times the model will not predict properly additionally the residuals of the models indicate overfitting. I would not recommend any of these models to replace the Laboratory methods.

Appendix:

Question 5

```
rm(list = ls())


#(a) Start R and use these commands to load the data:

library(AppliedPredictiveModeling)

data(permeability)

#The fingerprint predictors indicate the presence or absence of substructures

#of a molecule and are often sparse meaning that relatively few of the

#molecules contain each substructure. Filter out the predictors that have

#low frequencies using the nearZeroVar function from the caret package.

#How many predictors are left for modeling




ZC <- nearZeroVar(fingerprints, saveMetrics = T,freqCut = 95/5);

NZV<-ZC[ZC[, "nzv"] == TRUE, ]

ZV<-ZC[ZC[, "zeroVar"] == TRUE, ]

drop_cols<-row.names(NZV)

ZV<-row.names(ZV)

a<-fingerprints[ , !(colnames(fingerprints) %in% drop_cols)]

a<-a[ , !(colnames(a) %in% ZV)]
```

```
# positively skewed
ggplot() + geom_histogram(aes(x = permeability), binwidth = 4, col = 1) +
  labs(title = "Histogram of permeability", x = "Molecule permeability") +
  theme_bw()




Trans <- BoxCoxTrans(permeability)
trans <- preProcess(permeability, method = c("BoxCox"))
transformed <- predict(trans, permeability)
ggplot() + geom_histogram(aes(x = transformed), binwidth = 1, col = 12)


set.seed(1)
Partition <- createDataPartition(transformed, times = 1, p = 0.80,list=FALSE)



set.seed(1)
training <- a[Partition, ]
test <- a[-Partition, ]
Y_train <- transformed[Partition]
Y_test <- transformed[-Partition]
```

```
ctrl = trainControl("repeatedcv", number = 5, repeats = 5)

ctrl1 <- trainControl(method = "LGOCV")

# run relevant ch 6 lines

fingerProcessed<-training

fingerTrainY<-Y_train

fingerTestX<-test

fingerTestY<-Y_test


nnetGrid <- expand.grid(decay = c(0.01, 0.1),
                        size = 1:10,
                        bag = FALSE)


nnetTune <- train(fingerProcessed, fingerTrainY,
                  method = 'avNNet',
                  tuneGrid = nnetGrid,
                  trControl = ctrl,
                  linout = TRUE,
                  trace = FALSE,
                  MaxNWts = 10 * (ncol(absTrainX) + 1) + 10 + 1,
                  maxit = 500)


svmTune <- train(fingerProcessed, fingerTrainY,
                 method = 'svmRadial',
```

```
                tuneLength = 14,
                trControl = ctrl)
# 10.21 RMSE
svmTuneSigma <- train(fingerProcessed, fingerTrainY,
                      method = 'svmRadialSigma',
                      tuneLength = 14,
                      trControl = ctrl)
knnPermTune <- train(x = fingerProcessed,  y = fingerTrainY,method = "knn",
tuneLength = 10)

marsGrid <- expand.grid(degree = 1:2, nprune = seq(2, 130, by = 5))
marsTune <- train(fingerProcessed, fingerTrainY,
                  method = 'earth',
                  tuneGrid = marsGrid,
                  trControl = ctrl)

models <- list(
  knn = knnPermTune,
  svmSigma = svmTuneSigma,
  mars = marsTune
)
preds <- lapply(models, function(model) {
  data.frame(obs = fingerTestY,
             pred = predict(model, fingerTestX))
```

```
})

# mars is weird


preds[[4]]$pred <- predict(marsTune, fingerTestX)[, 1]

defaultSummary(preds[[1]])

defaultSummary(preds[[2]])

defaultSummary(preds)

defaultSummary(preds[[4]])


# looks like non linear does not improve

Q2:

library(mlbench)

library(caret)

library(reshape2)

set.seed(200)

trainingData <- mlbench.friedman1(200, sd = 1)

trainingData$x <- data.frame(trainingData$x)

featurePlot(trainingData$x, trainingData$y)

boxplot(trainingData$x)

testData <- mlbench.friedman1(5000, sd = 1)

testData$x <- data.frame(testData$x)
```

```
set.seed(1)
ctrl <-trainControl(method = "cv", number = 10,p=.75)



high<-cor(trainingData$x)
Trainnet<-trainingData$x[,-high]



#nnet
nnetGrid <- expand.grid(
  .decay = c(0, 0.01, 0.1),
  .size = 3:7,
  .bag = FALSE
)

set.seed(1)
nnetTune <- train(trainingData$x, trainingData$y,
                  method = 'avNNet',
                  tuneGrid = nnetGrid,
                  preProc = c('center', 'scale'),
                  linout = TRUE,
                  trace = FALSE,
                  maxit = 500)
```

```
preds <- data.frame(obs = testData$y,

pred_nnet = predict(nnetTune, testData$x))



# MARS

set.seed(1)

mGrid <- expand.grid(degree = 1:2, nprune = 2:10)

mars <- train(trainingData$x, trainingData$y,

              method = 'earth',

              tuneGrid = mGrid)

preds$pred_mars <- predict(mars, testData$x[, ])[, 1]

mars$bestTune


# SVM


set.seed(1)

svm<- train(trainingData$x, trainingData$y,method = "svmRadial",tuneLength = 10,
preProc = c("center", "scale"))


preds$pred_svm <- predict(svm, testData$x)

svm



# knn
```

```
set.seed(1)
knn <- train(trainingData$x,
             trainingData$y,
             method = 'knn',
             preProc = c('center', 'scale'),
             tuneGrid = data.frame(.k = 1:20))

preds$pred_knn <- predict(knn, testData$x)

preds <- melt(preds, id.vars = 'obs')

ggplot(preds, aes(x = value, y = obs)) +
  facet_wrap(~variable) + theme_bw() + geom_point(alpha = 0.05)

varImp(mars)
varImp(svm)
varImp(knn)
varImp(nnet)

library(dplyr)
preds %>% group_by(variable) %>%
  summarise(
    RMSE = RMSE(obs, value)
```

```
  )
Question 3
rm(list = ls())
data(tecator)
set.seed(1)
indx <- createFolds(endpoints[,2], returnTrain = TRUE)
ctrl <- trainControl('cv', index = indx)


hists <- apply(absorp, 2, function(x) {
  qplot(x = x, geom = 'histogram', bins = 50)
})
summaries <- apply(absorp, 2, function(x) {
  s <- summary(x)
  v <- var(x)
  m <- mean(x)
  list(summary = s, var = v, mean = m)
})
absorp <- data.frame(absorp)
abPreProc <- preProcess(absorp)
scaled_ab <- apply(absorp, 2, scale)
tec_cor <- cor(absorp)
```

```
absTrainX <- predict(abPreProc, absorp)

absTrainY <- endpoints[, 2]

svmTune <- train(absTrainX, absTrainY,

                 method = 'svmRadial',

                 tuneLength = 14,

                 trControl = ctrl)

set.seed(1)

knnTune <- train(absTrainX, absTrainY,

                 method = 'knn',

                 tuneLength = 20,

                 trControl = ctrl)

set.seed(1)

marsGrid <- expand.grid(degree = 1:2, nprune = 2:38)


marsTune <- train(absTrainX, absTrainY,

                  method = 'earth',

                  tuneGrid = marsGrid,

                  trControl = ctrl)


set.seed(1)

nnetGrid <- expand.grid(decay = c(0, 0.01, 0.1),

                        size = 1:10,

                        bag = FALSE)
```

```
nnetTune <- train(absTrainX, absTrainY,
                  method = 'avNNet',
                  tuneGrid = nnetGrid,
                  trControl = ctrl,
                  linout = TRUE,
                  trace = FALSE,
                  MaxNWts = 10 * (ncol(absTrainX) + 1) + 10 + 1,
                  maxit = 50)
set.seed(1)
nnetPCATune <- train(absTrainX, absTrainY,
                     method = 'avNNet',
                     preProcess = 'pca',
                     tuneGrid = nnetGrid,
                     trControl = ctrl,
                     linout = TRUE,
                     trace = FALSE,
                     MaxNWts = 10 * (ncol(absTrainX) + 1) + 10 + 1,
                     maxit = 50)

# only used two comps
nnetPCATune$preProcess$numComp
nnetTune$bestTune
```

```
nnetPCATune$results

nnetTune$results


# PCA with nnet halved the RMSE to about 10.7


svmTune

# basic svm got RMSE of 3.56

marsTune

# RMSE 2.0

knnTune

# RMSE 8.7


varImp(marsTune)

question 1:

set.seed(1)

x <- runif(100, min = 2,

          max = 10)

y <- sin(x) + rnorm(length(x)) * 0.25

sinData <- data.frame(x = x, y = y)

dataGrid <- data.frame(x=seq(2,10, length = 100))


plot(x, y)
```

```
for(i in 1:20) {
  ksvm.fit <- ksvm(x = x, y = y, data = sinData,kernel = "rbfdot", kpar = "automatic",C= i,
epsilon =  0.1 )
  pred <- predict(ksvm.fit,dataGrid)
  points(x = dataGrid$x, y = pred, type = 'l',
         col = color[i])
}


par(mar=c(1,1,1,1))
par(mfrow=c(5,4))
for( i in 1:20)
{
  ksvm.fit <- ksvm(x = x, y = y, data = sinData,
                   kernel = "rbfdot", kpar = "automatic",
                   C= i, epsilon =  0.1 )
  pred <- predict(ksvm.fit,dataGrid)
  plot(x = dataGrid$x, y = pred, type = 'l',
         col = color[i])
}



plot(x, y)
for (i in 1:20) {
```

```
 radial_SVM <- ksvm(x = x, y = y, data = sin_data,kernel ="rbfdot", kpar = "automatic", C =
1, epsilon = i/20) # move epsilon from 0.1 to 0.9

 model_preds <- predict(radial_SVM, newdata = dataGrid)

 points(x = dataGrid$x, y = model_preds[,1], type = "l", col = color[i], lwd = 2)

}


par(mar=c(1,1,1,1))

par(mfrow=c(5,4))

for( i in 1:20)

{

 radial_SVM <- ksvm(x = x, y = y, data = sin_data,kernel ="rbfdot", kpar = "automatic", C =
1, epsilon = i/20) # move epsilon from 0.1 to 0.9

 model_preds <- predict(radial_SVM, newdata = dataGrid)

 plot(x = dataGrid$x, y = model_preds[,1], type = "l", col = color[i], lwd =
2,main=print(i/20))


}


plot(x, y)

for(i in 1:20)

{

 ksvm.fit1 <- ksvm(x = x, y = y, data = sinData,kernel = "rbfdot", kpar = list(sigma = i),C=
1, epsilon =  0.1 )

 pred1 <- predict(ksvm.fit1,dataGrid)

 points(x = dataGrid$x, y = pred1, type = 'l',col = color[i])
```

```
}



par(mar=c(1,1,1,1))

par(mfrow=c(5,4))

for( i in 1:20)

{

  ksvm.fit1 <- ksvm(x = x, y = y, data = sinData,kernel = "rbfdot", kpar = list(sigma = i),C=
1, epsilon =  0.1 )

  pred1 <- predict(ksvm.fit1,dataGrid)

  plot(x = dataGrid$x, y = pred1, type = 'l',col = color[i])


}




par(mar=c(1,1,1,1))

par(mfrow=c(5,4))

for( i in 1:20)

{

  ksvm.fit <- ksvm(x = x, y = y, data = sinData,kernel = "rbfdot", kpar = "automatic", C= i,
epsilon =  0.1 )

  pred <- predict(ksvm.fit,dataGrid)

plot(x = dataGrid$x, y = pred, type = 'l', col = "Blue")
```

```
  radial_SVM <- ksvm(x = x, y = y, data = sin_data,kernel ="rbfdot", kpar = "automatic", C =
i, epsilon = i/20) # move epsilon from 0.1 to 0.9

  model_preds <- predict(radial_SVM, newdata = dataGrid)

  points(x = dataGrid$x, y = model_preds[,1], type = "l", col = "Magenta", lwd = 2)



   ksvm.fit1 <- ksvm(x = x, y = y, data = sinData,kernel = "rbfdot", kpar = list(sigma = i), C=
1, epsilon =  0.1 )

   pred1 <- predict(ksvm.fit1,dataGrid)

   points(x = dataGrid$x, y = pred1, type = 'l',

         col = "Green")

 }
```