Homework – 4

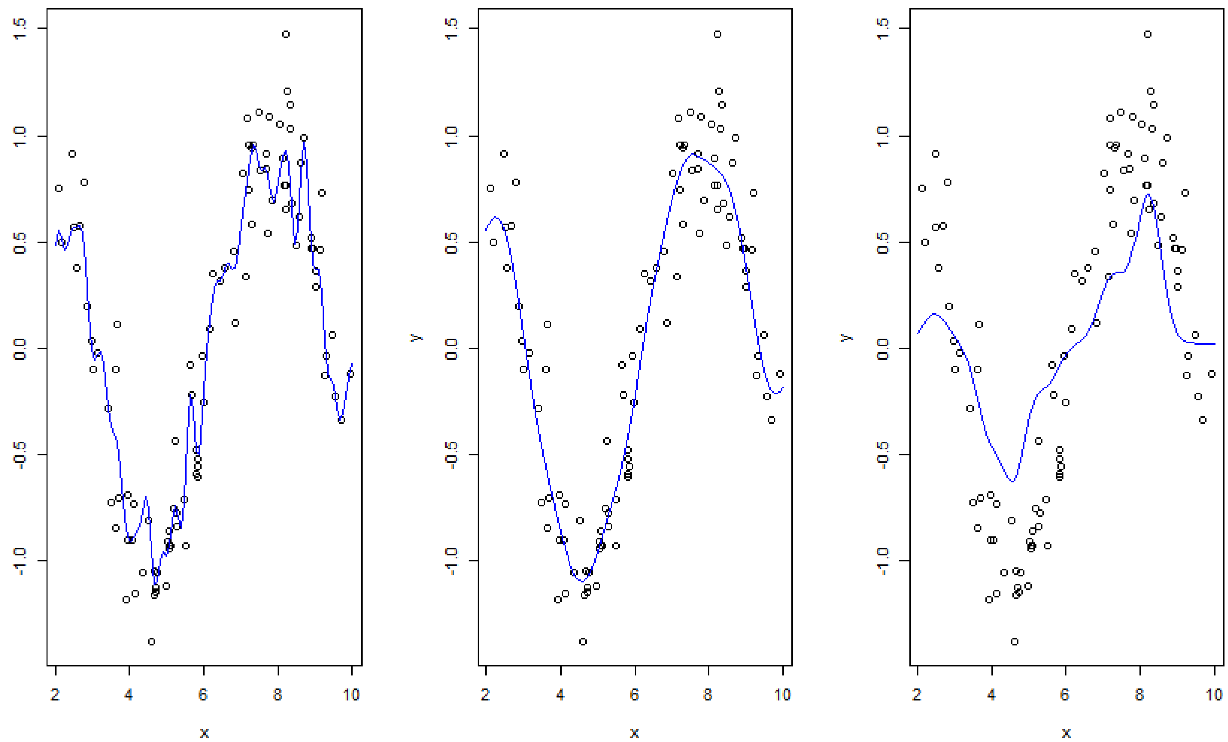
Question 7.1 (a)

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

Answer:

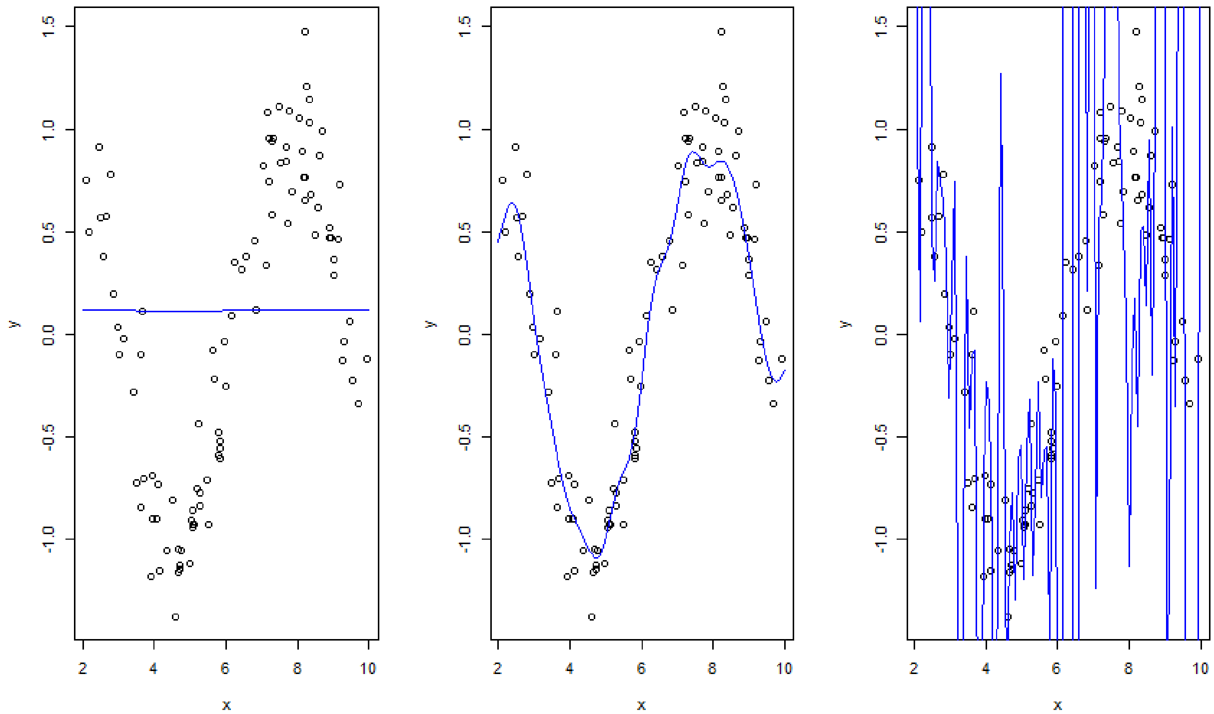
The first 3 plots show change in the epsilon value:

The first plot represents overfitting with smaller epsilon value and last graph represents underfitting with higher epsilon value. We get the below best fit curve in the middle. **The best value of epsilon = 0.1**



The below 3 plots show change in the Cost value:

The first plot represents underfitting with smaller C value and last graph represents overfitting with higher C value. We get the below best fit curve in the middle. **The best value of C = 1**

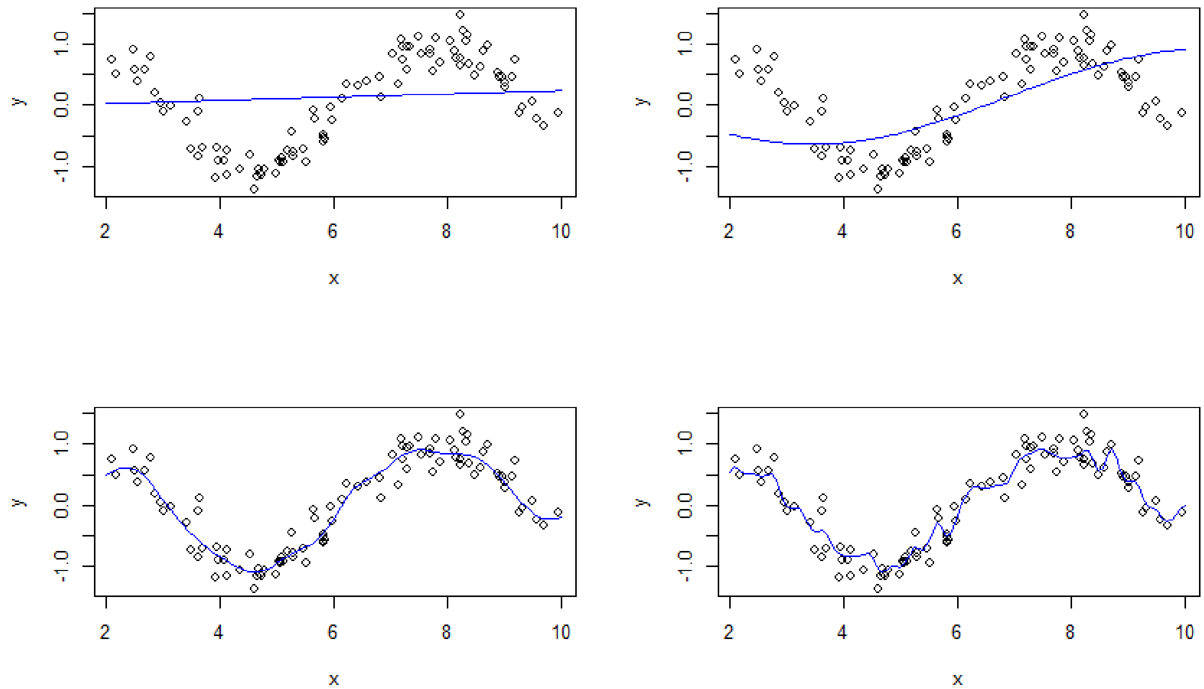


Question 7.1 (b)

Try different values of σ to understand how this parameter changes the model fit. How do the cost, epsilon, and σ values affect the model?

Answer: Using the best values of c and epsilon from 7.1(a), we try to tune sigma values and see that with **sigma = 10 , C= 1 and epsilon = 0.1** we get the below best fit curve:

With the **decrease in Cost**, the model will undergo high bias and low variability i.e**. UNDERFITTING** since very small penalty is given. Also, we observe with the decrease in epsilon value, the model begins to overfit. With the increase in sigma seems to overfit the model, however small values underfits the model.



Question 7.2 (a)

Consider KNN and MARS, which model appears to give the best performance?

Answer: Considering both KNN and MARS model, we see that Root Mean Square Error (RMSE) for KNN is 3.1954604 and for MARS model it is 1.2184795, thereby making **MARS model the best performer.**

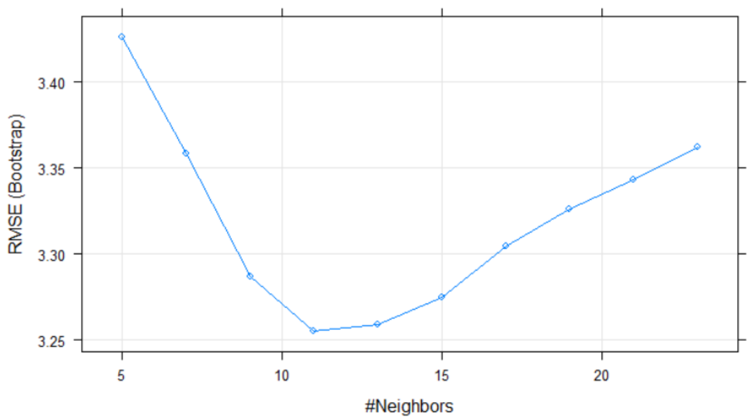
```
> knnModel
k-Nearest Neighbors

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
   5  3.556112  0.5198288  2.839161
   7  3.460457  0.5475880  2.777304
   9  3.428418  0.5655020  2.752174
  11  3.388099  0.5894854  2.717181
  13  3.378902  0.6059643  2.713234
  15  3.378760  0.6174332  2.721502
  17  3.376512  0.6296186  2.729312
  19  3.391905  0.6370843  2.754631
  21  3.404855  0.6457253  2.776341
  23  3.425590  0.6513261  2.788951
```

```
RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 17
```

RMSE and R square for KNN:

```
> postResample(pred = knnPred, obs = testData$y)
     RMSE  Rsquared       MAE 
3.1954604 0.6763216 2.5643472
```

Plot of Knn:



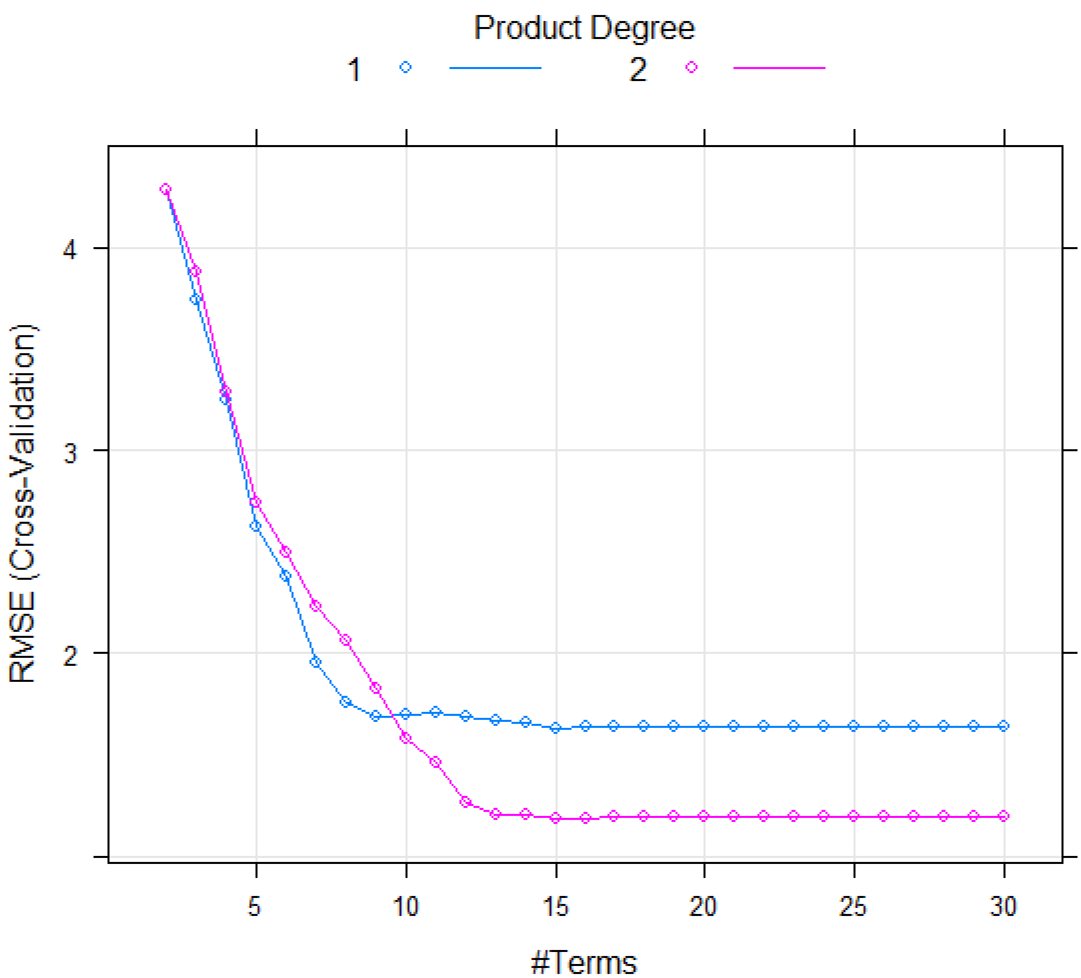
******MARS MODEL******

```
> marsTuned
Multivariate Adaptive Regression Spline 

200 samples
 10 predictor

No pre-processing
Resampling: Cross-Validated (10 fold) 
Summary of sample sizes: 180, 180, 180, 180, 180, 180, ... 
Resampling results across tuning parameters:

  degree  nprune  RMSE      Rsquared   MAE      
  1        2      4.284557  0.2907768  3.6249495
  1        3      3.745343  0.4559584  3.0619808
  1        4      3.245863  0.5891338  2.7177850
  1        5      2.629813  0.7306695  2.0789777
  1        6      2.384633  0.7778715  1.9280078
  1        7      1.958668  0.8414514  1.5928906
  1        8      1.762501  0.8781308  1.3803270
  1        9      1.685097  0.8864650  1.3278093
  1       10      1.698655  0.8828539  1.3375487
  1       11      1.712238  0.8795209  1.3438979
  1       12      1.684415  0.8807914  1.3252383
  1       13      1.672263  0.8813845  1.3142691
  1       14      1.663376  0.8827605  1.3038362
  1       15      1.634537  0.8855884  1.2820001
```

```
  1       16      1.644353  0.8824488  1.2885712
  1       17      1.644353  0.8824488  1.2885712
  1       18      1.644353  0.8824488  1.2885712
  1       19      1.644353  0.8824488  1.2885712
  1       20      1.644353  0.8824488  1.2885712
  1       21      1.644353  0.8824488  1.2885712
  1       22      1.644353  0.8824488  1.2885712
  1       23      1.644353  0.8824488  1.2885712
  1       24      1.644353  0.8824488  1.2885712
  1       25      1.644353  0.8824488  1.2885712
  1       26      1.644353  0.8824488  1.2885712
  1       27      1.644353  0.8824488  1.2885712
  1       28      1.644353  0.8824488  1.2885712
  1       29      1.644353  0.8824488  1.2885712
  1       30      1.644353  0.8824488  1.2885712
  2        2      4.284557  0.2907768  3.6249495
  2        3      3.880966  0.4241772  3.2223793
  2        4      3.284656  0.5916369  2.7133834
  2        5      2.748910  0.7124206  2.2065683
  2        6      2.502461  0.7563797  1.9966779
  2        7      2.229125  0.7989121  1.8148853
  2        8      2.067839  0.8338152  1.6268420
  2        9      1.826997  0.8652168  1.4324910
  2       10      1.584590  0.8983887  1.1902973
  2       11      1.457277  0.9085165  1.1438142
  2       12      1.265827  0.9358851  1.0372929
  2       13      1.200899  0.9421594  0.9773092
  2       14      1.209770  0.9419917  0.9915414
  2       15      1.187278  0.9425942  0.9690146
  2       16      1.188954  0.9427666  0.9598889
  2       17      1.192742  0.9419037  0.9673360
  2       18      1.194431  0.9416624  0.9670779
  2       19      1.194431  0.9416624  0.9670779
  2       20      1.194431  0.9416624  0.9670779
  2       21      1.194431  0.9416624  0.9670779
  2       22      1.194431  0.9416624  0.9670779
  2       23      1.194431  0.9416624  0.9670779
  2       24      1.194431  0.9416624  0.9670779
  2       25      1.194431  0.9416624  0.9670779
  2       26      1.194431  0.9416624  0.9670779
  2       27      1.194431  0.9416624  0.9670779
  2       28      1.194431  0.9416624  0.9670779
  2       29      1.194431  0.9416624  0.9670779
  2       30      1.194431  0.9416624  0.9670779

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nprune = 15 and degree = 2.

> summary(marsTuned)
Call: earth(x=data.frame[200,10], y=c(16.29,15.08,1...), keepxy=TRUE,
            degree=2, nprune=15)

                                coefficients
(Intercept)                        18.876238
h(0.47762-X1)                     -10.654789
h(X1-0.47762)                      11.763927
h(0.333521-X2)                    -16.042699
h(X2-0.333521)                     11.100290
h(0.450485-X3)                     10.905837
h(X3-0.450485)                      4.130150
h(X3-0.758697)                     11.318926
h(0.929743-X4)                     -9.969791
h(0.925323-X5)                     -5.595141
h(X1-0.47762) * h(X2-0.429713)    -41.248060
```

```
h(X1-0.47762) * h(0.429713-X2)    -24.788387
h(0.553036-X1) * h(0.333521-X2)    29.499372
h(0.724499-X1) * h(X2-0.333521)   -10.655523
h(X1-0.724499) * h(X2-0.333521)   -27.328765

Selected 15 of 20 terms, and 5 of 10 predictors
Termination condition: Reached nk 21
Importance: X4, X2, X1, X5, X3, X6-unused, X7-unused, X8-unused, ...
Number of terms at each degree of interaction: 1 9 5
GCV 1.510035    RSS 203.0695    GRSq 0.9422094    RSq 0.9607501
```

Plot of MARS:




```
> postResample(pred = marsPred, obs = testData$y)
     RMSE  Rsquared       MAE
1.2184795 0.9405195 0.9689160


> varImp(marsTuned)
earth variable importance

    Overall
X4   100.00
X2    83.07
```

```
X1     68.03
X5     55.51
X3     43.99
X6      0.00
X10     0.00
X8      0.00
X9      0.00
X7      0.00
```

Question 7.2 (b)

Does MARS select the informative predictors (those named X1-X5)?

Answer: Yes, MARS only selects predictors from X1- X5.

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

On checking the importance of the variables, we observe:

```
> varImp(marsTuned)
earth variable importance

    Overall
X4   100.00
X2    83.07
X1    68.03
X5    55.51
X3    43.99
X6     0.00
X10    0.00
X8     0.00
X9     0.00
X7     0.00
```

Question 7.5 (a)

Which nonlinear regression model gives the optimal resampling and test

set performance?

Answer:

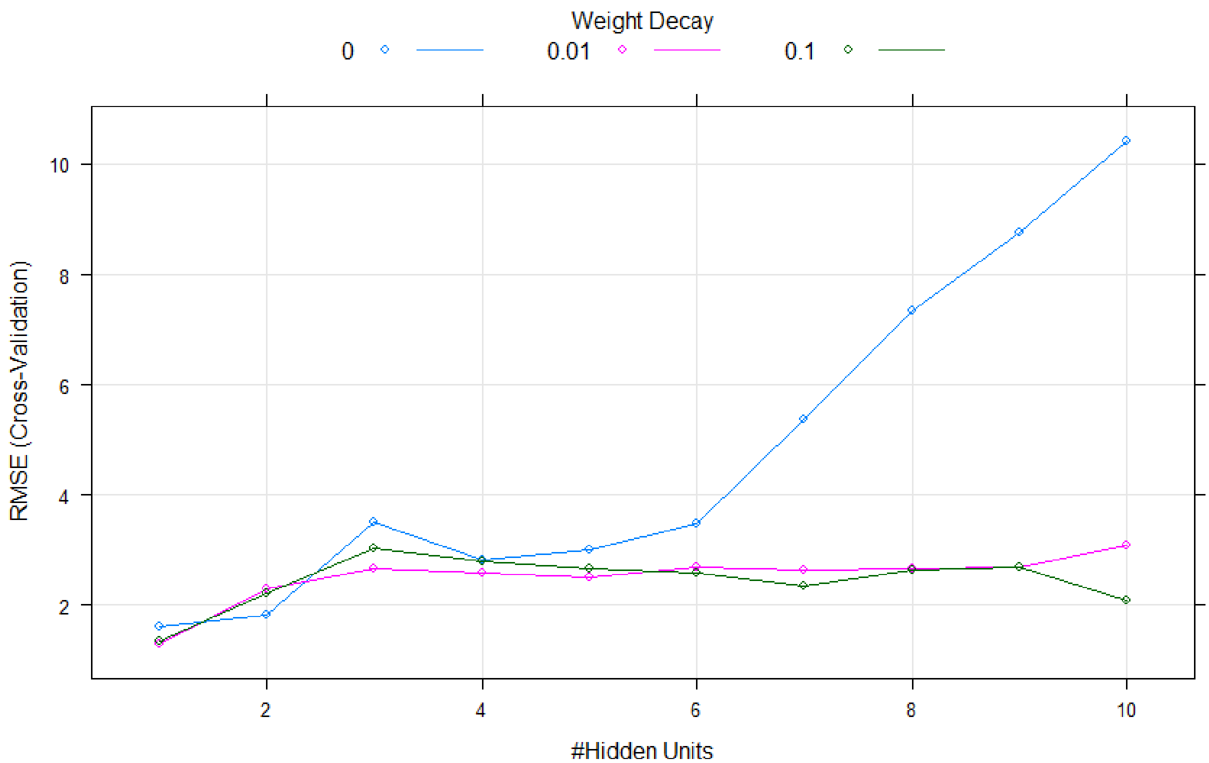
On checking the below 4 models we receive the below RMSE values and conclude that **SVM** is the best model, followed by **MARS**, **KNN** and last **Neural Network**:

```
Neural Network:  RMSE : [1] 1.481175
MARS: RMSE: [1] 0.976105
SVM: RMSE : [1] 0.8771186
KNN: RMSE : [1] 1.15631
```

*******Neural Network*******

```
> nnetTune
Neural Network

144 samples
 57 predictor

Pre-processing: centered (57), scaled (57)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 129, 130, 130, 129, 130, 131, ...
Resampling results across tuning parameters:

  decay  size  RMSE       Rsquared    MAE
  0.00    1     1.613133  0.29680365  1.274704
  0.00    2     1.829547  0.24644535  1.469571
  0.00    3     3.514748  0.19934127  2.592703
  0.00    4     2.815952  0.27272470  2.243474
  0.00    5     3.021870  0.13418450  2.356914
  0.00    6     3.478266  0.26920342  2.752366
  0.00    7     5.371975  0.03786749  3.908115
  0.00    8     7.333995  0.12922071  5.586466
  0.00    9     8.754481  0.07477464  6.172310
  0.00   10    10.411476  0.13388064  6.777419
  0.01    1     1.313547  0.57891130  1.072006
  0.01    2     2.305394  0.35672567  1.760655
  0.01    3     2.681485  0.30159334  2.060560
  0.01    4     2.580796  0.25428844  1.997650
  0.01    5     2.513486  0.23123738  1.889188
  0.01    6     2.698990  0.24594234  2.031484
  0.01    7     2.642295  0.20900749  1.932284
  0.01    8     2.672798  0.23740558  2.052299
  0.01    9     2.687185  0.30079984  2.094814
  0.01   10     3.092851  0.22843847  2.316658
  0.10    1     1.360945  0.57819231  1.082098
  0.10    2     2.208850  0.40737415  1.727699
  0.10    3     3.043843  0.33748370  2.047708
  0.10    4     2.796069  0.20667729  2.024215
  0.10    5     2.672958  0.27962276  1.888574
  0.10    6     2.599611  0.21499006  1.846923
  0.10    7     2.344680  0.35088991  1.845712
  0.10    8     2.651221  0.26366577  2.011825
  0.10    9     2.693228  0.31586094  2.085887
  0.10   10     2.093504  0.36036608  1.622409

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were size = 1 and decay = 0.01.
```

Plot of Neural Networks:



RMSE and R square values:

```
> RMSE
[1] 1.481175
> Rsquared
 [1] 0.463877
```

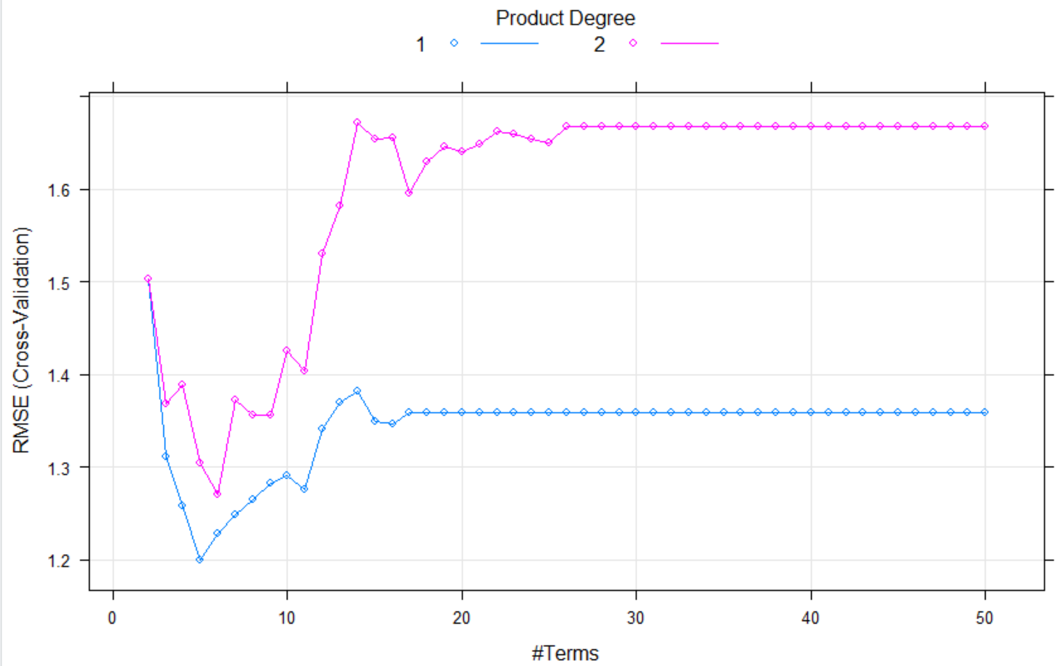
*******MARS Model*******

```
> marsTuned2
Multivariate Adaptive Regression Spline

144 samples
 57 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 129, 130, 130, 129, 130, 131, ...
Resampling results across tuning parameters:

  degree  nprune  RMSE      Rsquared   MAE
  1        2      1.502409  0.3816048  1.1739426
  1        3      1.310604  0.5317297  1.0515917
  1        4      1.257524  0.5704640  1.0105068
  1        5      1.199891  0.5909569  0.9776183
  1        6      1.228152  0.5755721  1.0115693
  1        7      1.248118  0.5582910  1.0260538
  1        8      1.264259  0.5588769  1.0283702
  1        9      1.282930  0.5441876  1.0231224
  1       10      1.290678  0.5461481  1.0252852
```

1	11	1.274922	0.5554307	1.0060641
1	12	1.341004	0.5195106	1.0623705
1	13	1.369019	0.5059586	1.0870393
1	14	1.381312	0.5020588	1.0876543
1	15	1.349563	0.5213324	1.0707806
1	16	1.346936	0.5226876	1.0680519
1	17	1.358106	0.5188035	1.0801521
1	18	1.358106	0.5188035	1.0801521
1	19	1.358106	0.5188035	1.0801521
1	20	1.358106	0.5188035	1.0801521
1	21	1.358106	0.5188035	1.0801521
1	22	1.358106	0.5188035	1.0801521
1	23	1.358106	0.5188035	1.0801521
1	24	1.358106	0.5188035	1.0801521
1	25	1.358106	0.5188035	1.0801521
1	26	1.358106	0.5188035	1.0801521
1	27	1.358106	0.5188035	1.0801521
1	28	1.358106	0.5188035	1.0801521
1	29	1.358106	0.5188035	1.0801521
1	30	1.358106	0.5188035	1.0801521
1	31	1.358106	0.5188035	1.0801521
1	32	1.358106	0.5188035	1.0801521
1	33	1.358106	0.5188035	1.0801521
1	34	1.358106	0.5188035	1.0801521
1	35	1.358106	0.5188035	1.0801521
1	36	1.358106	0.5188035	1.0801521
1	37	1.358106	0.5188035	1.0801521
1	38	1.358106	0.5188035	1.0801521
1	39	1.358106	0.5188035	1.0801521
1	40	1.358106	0.5188035	1.0801521
1	41	1.358106	0.5188035	1.0801521
1	42	1.358106	0.5188035	1.0801521
1	43	1.358106	0.5188035	1.0801521
1	44	1.358106	0.5188035	1.0801521
1	45	1.358106	0.5188035	1.0801521
1	46	1.358106	0.5188035	1.0801521
1	47	1.358106	0.5188035	1.0801521
1	48	1.358106	0.5188035	1.0801521
1	49	1.358106	0.5188035	1.0801521
1	50	1.358106	0.5188035	1.0801521
2	2	1.502409	0.3816048	1.1739426
2	3	1.367764	0.4805279	1.0917356
2	4	1.388190	0.5108278	1.0608128
2	5	1.304490	0.5634132	1.0192828
2	6	1.269676	0.5708952	0.9881550
2	7	1.372822	0.4961310	1.0700828
2	8	1.355434	0.5008839	1.0614555
2	9	1.356070	0.4971033	1.0653667
2	10	1.424864	0.4751246	1.1108653
2	11	1.403176	0.4911517	1.0879520
2	12	1.529606	0.4156696	1.1697298
2	13	1.581811	0.4142080	1.1710376
2	14	1.671390	0.3840484	1.2070565
2	15	1.653477	0.3933017	1.1966139
2	16	1.654954	0.3944623	1.2000589
2	17	1.595685	0.4240096	1.1585922
2	18	1.630129	0.4150621	1.1868511
2	19	1.646315	0.4098851	1.1975194
2	20	1.640001	0.4081172	1.2124948
2	21	1.648272	0.3985678	1.2098688
2	22	1.662832	0.3958711	1.2249462
2	23	1.659475	0.3998134	1.2248941
2	24	1.654106	0.4151083	1.2393631
2	25	1.650484	0.4217019	1.2320061

```
  2       26      1.667116  0.4217100  1.2428794
  2       27      1.667116  0.4217100  1.2428794
  2       28      1.667116  0.4217100  1.2428794
  2       29      1.667116  0.4217100  1.2428794
  2       30      1.667116  0.4217100  1.2428794
  2       31      1.667116  0.4217100  1.2428794
  2       32      1.667116  0.4217100  1.2428794
  2       33      1.667116  0.4217100  1.2428794
  2       34      1.667116  0.4217100  1.2428794
  2       35      1.667116  0.4217100  1.2428794
  2       36      1.667116  0.4217100  1.2428794
  2       37      1.667116  0.4217100  1.2428794
  2       38      1.667116  0.4217100  1.2428794
  2       39      1.667116  0.4217100  1.2428794
  2       40      1.667116  0.4217100  1.2428794
  2       41      1.667116  0.4217100  1.2428794
  2       42      1.667116  0.4217100  1.2428794
  2       43      1.667116  0.4217100  1.2428794
  2       44      1.667116  0.4217100  1.2428794
  2       45      1.667116  0.4217100  1.2428794
  2       46      1.667116  0.4217100  1.2428794
  2       47      1.667116  0.4217100  1.2428794
  2       48      1.667116  0.4217100  1.2428794
  2       49      1.667116  0.4217100  1.2428794
  2       50      1.667116  0.4217100  1.2428794

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nprune = 5 and degree = 1.

> summary(marsTuned2)
Call: earth(x=data.frame[144,57], y=c(38,42.03,41.4...), keepxy=TRUE, degree=
1, nprune=5)

                                        coefficients
(Intercept)                                38.260768
h(-1.01531-ManufacturingProcess09)         -1.102648
h(ManufacturingProcess09- -1.01531)         0.541082
h(-0.992785-ManufacturingProcess13)         1.660157
h(ManufacturingProcess32- -1.01272)         1.158230

Selected 5 of 21 terms, and 3 of 57 predictors
Termination condition: RSq changed by less than 0.001 at 21 terms
Importance: ManufacturingProcess32, ManufacturingProcess09, ManufacturingProc
ess13, BiologicalMaterial01-unused, ...
Number of terms at each degree of interaction: 1 4 (additive model)
GCV 1.39529    RSS 176.5914    GRSq 0.6048384    RSq 0.6478156
```

Plot of MARS:



RMSE and R square values:

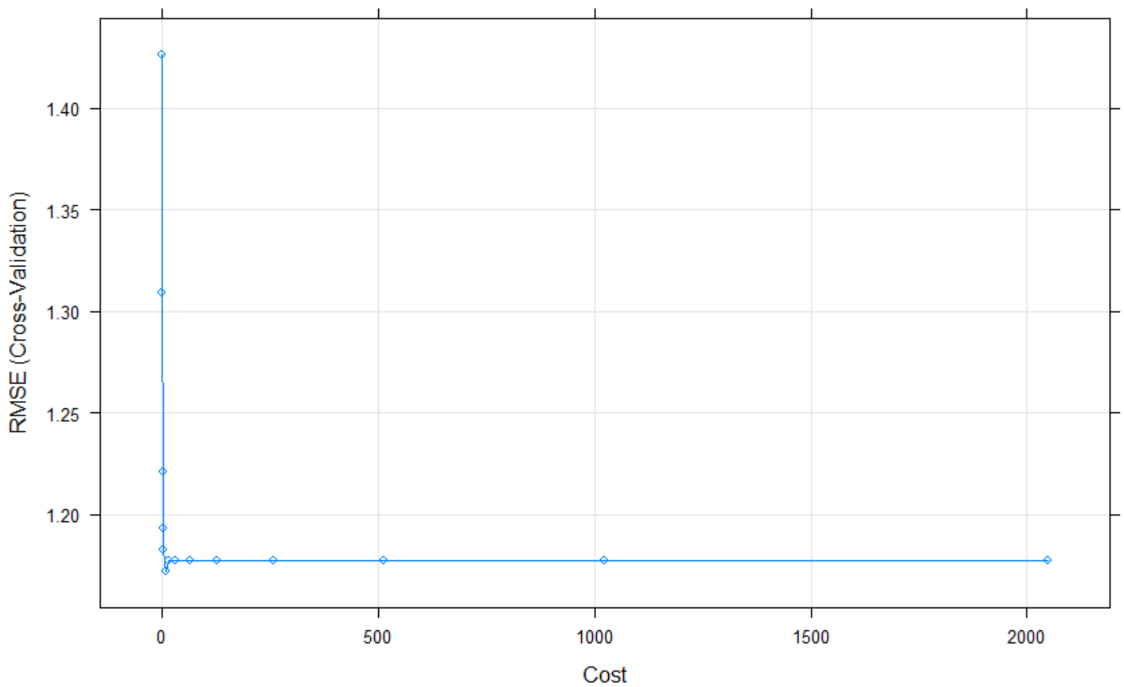
```
> RMSE
[1] 0.976105
> Rsquared
             y
[1,] 0.6829522
```

*******SVM Model*******

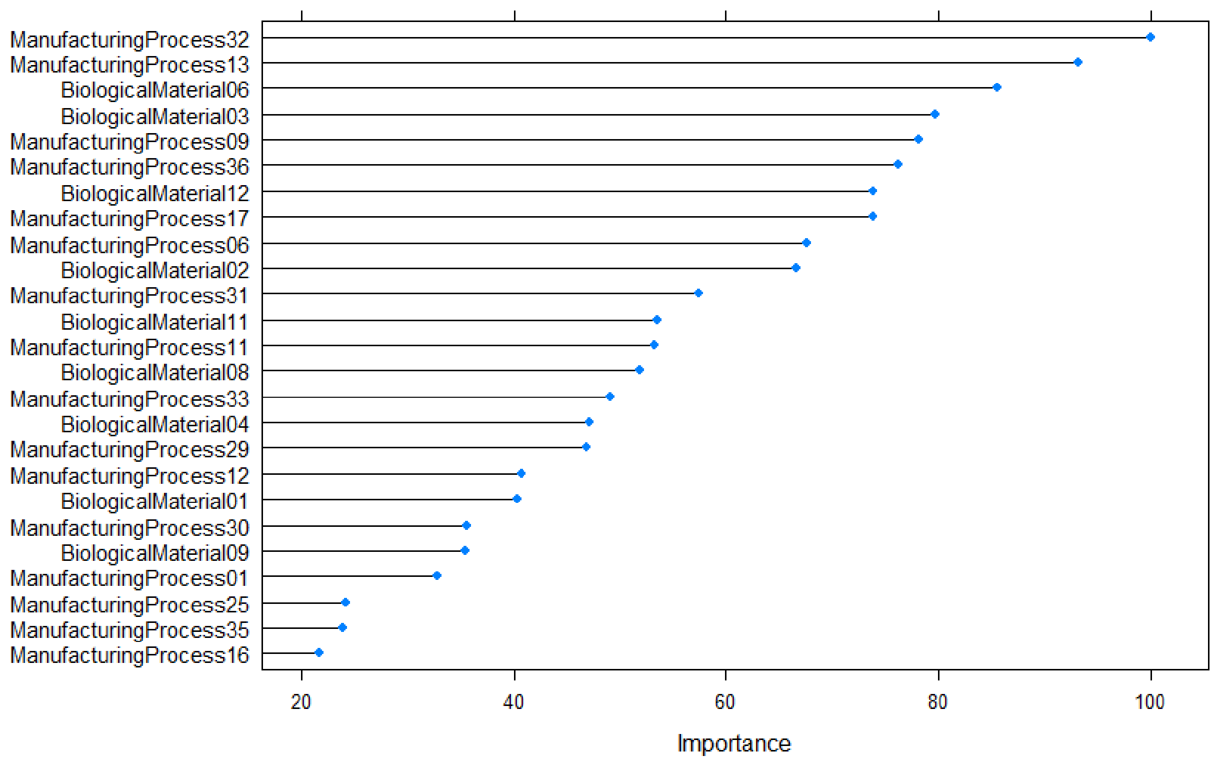
```
> svmRTuned
Support Vector Machines with Radial Basis Function Kernel

144 samples
 57 predictor

Pre-processing: centered (57), scaled (57)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 130, 129, 129, 130, 130, 130, ...
Resampling results across tuning parameters:

  C       RMSE      Rsquared   MAE
    0.25  1.426182  0.4970065  1.1598629
    0.50  1.309048  0.5496127  1.0666378
    1.00  1.221146  0.5934250  0.9951853
    2.00  1.193059  0.6075233  0.9669143
    4.00  1.182927  0.6120168  0.9609258
    8.00  1.172228  0.6192141  0.9549031
   16.00  1.177742  0.6161415  0.9607608
   32.00  1.177742  0.6161415  0.9607608
   64.00  1.177742  0.6161415  0.9607608
  128.00  1.177742  0.6161415  0.9607608
  256.00  1.177742  0.6161415  0.9607608
```

```
   512.00  1.177742  0.6161415  0.9607608
  1024.00  1.177742  0.6161415  0.9607608
  2048.00  1.177742  0.6161415  0.9607608

Tuning parameter 'sigma' was held constant at a value of 0.01457755
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.01457755 and C = 8.
```

Plot of SVM:



Plot of important predictors:



RMSE and R square values:

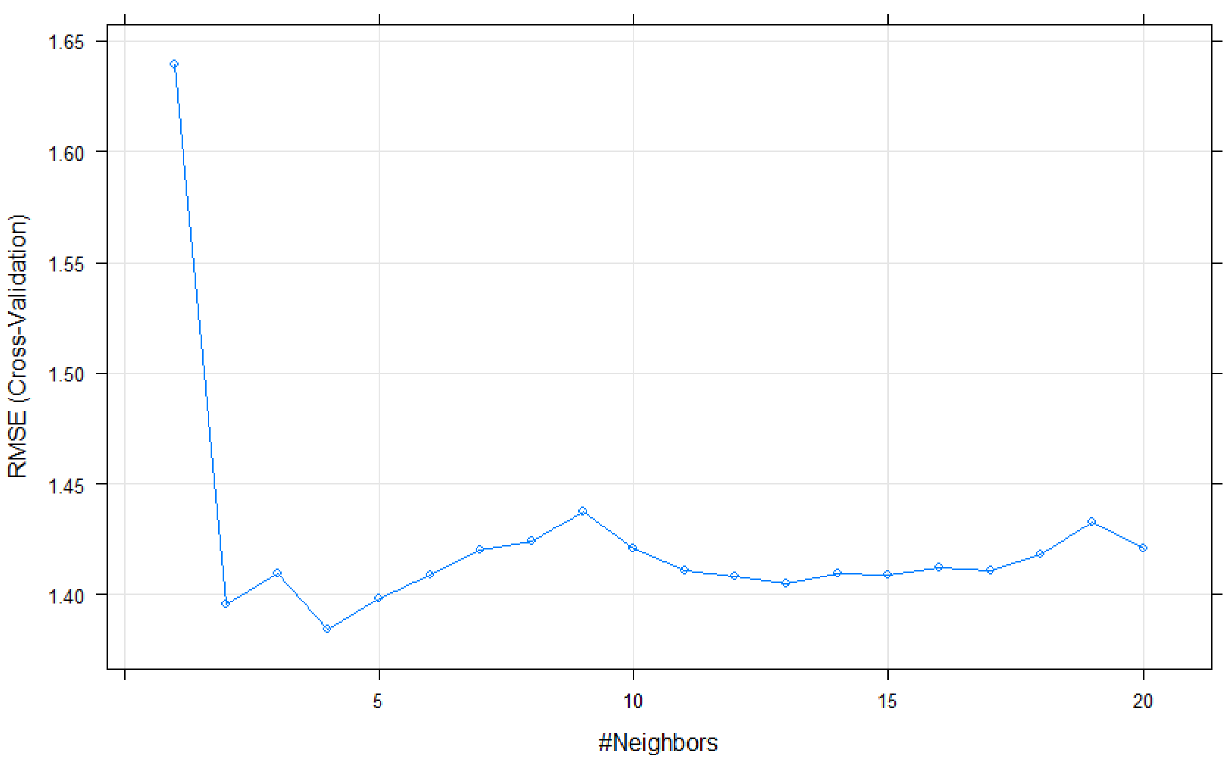
```
> RMSE
[1] 0.8771186
> Rsquared
[1] 0.7410714
```

*******KNN Model*******

```
> knnTune
k-Nearest Neighbors

144 samples
 56 predictor

Pre-processing: centered (56), scaled (56)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 129, 130, 130, 129, 130, 131, ...
Resampling results across tuning parameters:

  k   RMSE      Rsquared   MAE
  1   1.639293  0.3430869  1.262014
  2   1.395296  0.4752058  1.114661
  3   1.409294  0.4485807  1.097158
  4   1.384346  0.4858396  1.086574
  5   1.398517  0.4581576  1.106972
  6   1.409157  0.4454810  1.121231
  7   1.420295  0.4388996  1.151864
  8   1.423832  0.4398441  1.163044
```

```
  9  1.437183  0.4287864  1.172073
 10  1.421155  0.4378686  1.155454
 11  1.411128  0.4473777  1.152830
 12  1.408270  0.4500300  1.151142
 13  1.404617  0.4550758  1.146724
 14  1.409763  0.4514934  1.147584
 15  1.409020  0.4589753  1.139574
 16  1.411879  0.4619994  1.139984
 17  1.410879  0.4721659  1.137554
 18  1.418117  0.4677082  1.145305
 19  1.432810  0.4603145  1.154873
 20  1.421041  0.4750207  1.150003

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 4.
```

Plot of KNN:



RMSE and R square values:

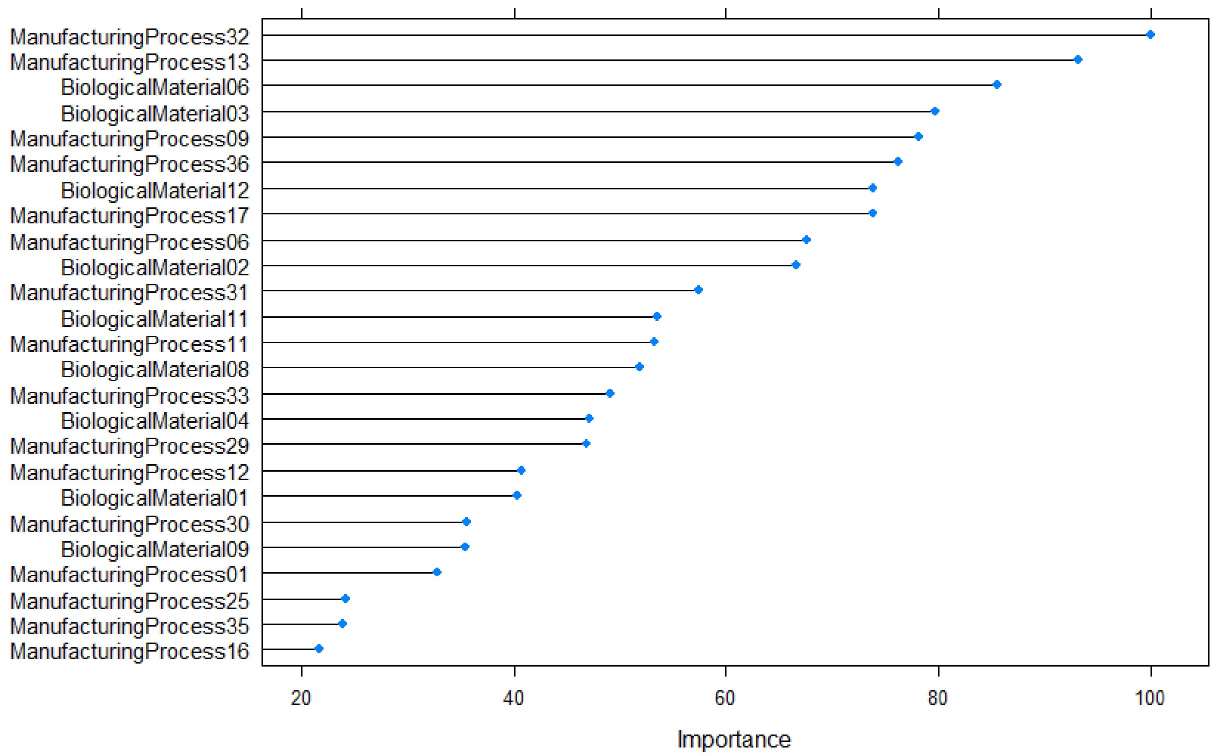
```
> RMSE
[1] 1.15631
> Rsquared
[1] 0.5810843
```

Question 7.5 (b)

Which predictors are most important in the optimal nonlinear regression model? Do either the biological or process variables dominate the list? How do the top ten important predictors compare to the top ten predictors from the optimal linear model?

Answer:

SVM being the most optimal nonlinear regression model has the below important predictors:



ManufacturingProcess variables dominate the list for SVM.

From previous homework

Model	RMSE
Lm	10.93
Lasso	1.82
Ridge	1.83
Enet	2.11

We see that Lasso is the best linear model on the Chemical Manufacturing Process data. Comparing the predictors of lasso and SVM we get:

- LASSO

ManufacturingProcess variables dominate the list for Lasso as well.

```
> varImp(lasso_model, lambda = cv$lambda.min)
                          Overall
BiologicalMaterial01   0.000000000
BiologicalMaterial02   0.000000000
BiologicalMaterial03   0.112708621
BiologicalMaterial04   0.000000000
BiologicalMaterial05   0.000000000
BiologicalMaterial06   0.000000000
BiologicalMaterial07   0.009509862
BiologicalMaterial08   0.000000000
BiologicalMaterial09   0.000000000
BiologicalMaterial10   0.000000000
BiologicalMaterial11   0.000000000
BiologicalMaterial12   0.000000000
ManufacturingProcess01 0.000000000
ManufacturingProcess02 0.000000000
ManufacturingProcess03 0.000000000
ManufacturingProcess04 0.162928695
ManufacturingProcess05 0.000000000
ManufacturingProcess06 0.085777342
ManufacturingProcess07 0.062973310
ManufacturingProcess08 0.000000000
ManufacturingProcess09 0.284614864
ManufacturingProcess10 0.000000000
ManufacturingProcess11 0.000000000
ManufacturingProcess12 0.000000000
ManufacturingProcess13 0.096286395
ManufacturingProcess14 0.000000000
ManufacturingProcess15 0.000000000
ManufacturingProcess16 0.000000000
ManufacturingProcess17 0.357619767
ManufacturingProcess18 0.000000000
ManufacturingProcess19 0.000000000
ManufacturingProcess20 0.000000000
ManufacturingProcess21 0.000000000
ManufacturingProcess22 0.000000000
ManufacturingProcess23 0.008925333
ManufacturingProcess24 0.093393325
ManufacturingProcess25 0.000000000
ManufacturingProcess26 0.000000000
ManufacturingProcess27 0.000000000
ManufacturingProcess28 0.000000000
ManufacturingProcess29 0.591763124
ManufacturingProcess30 0.137948363
ManufacturingProcess31 0.000000000
ManufacturingProcess32 0.747713305
ManufacturingProcess33 0.000000000
ManufacturingProcess34 0.076817805
ManufacturingProcess35 0.000000000
ManufacturingProcess36 0.179127984
ManufacturingProcess37 0.221060479
ManufacturingProcess38 0.000000000
ManufacturingProcess39 0.048027780
ManufacturingProcess40 0.000000000
ManufacturingProcess41 0.000000000
ManufacturingProcess42 0.000000000
ManufacturingProcess43 0.051966758
ManufacturingProcess44 0.000000000
ManufacturingProcess45 0.061329224
```

- SVM

```
> varImp(svmRTuned)$importance
                          Overall
BiologicalMaterial01    40.2779608
BiologicalMaterial02    66.5824394
BiologicalMaterial03    79.6906796
BiologicalMaterial04    47.1507324
BiologicalMaterial05    10.8623772
BiologicalMaterial06    85.5635314
BiologicalMaterial07     3.3351954
BiologicalMaterial08    51.8451978
BiologicalMaterial09    35.4803625
BiologicalMaterial10    21.5067030
BiologicalMaterial11    53.5969206
BiologicalMaterial12    73.8215317
ManufacturingProcess01  32.7592301
ManufacturingProcess02  10.8197488
ManufacturingProcess03   3.4911254
ManufacturingProcess04  21.0223053
ManufacturingProcess05   4.2172945
ManufacturingProcess06  67.5571363
ManufacturingProcess07   0.2771389
ManufacturingProcess08   0.1033685
ManufacturingProcess09  78.2091653
ManufacturingProcess10  14.5255441
ManufacturingProcess11  53.2271800
ManufacturingProcess12  40.7319774
ManufacturingProcess13  93.1932828
ManufacturingProcess14   8.4507257
ManufacturingProcess15  19.0578586
ManufacturingProcess16  21.6730438
ManufacturingProcess17  73.7853670
ManufacturingProcess18  17.2068235
ManufacturingProcess19   5.2462448
ManufacturingProcess20  20.4856909
ManufacturingProcess21  12.2021501
ManufacturingProcess22   0.6078945
ManufacturingProcess23   2.4651551
ManufacturingProcess24  15.8760370
ManufacturingProcess25  24.2331286
ManufacturingProcess26  13.7209560
ManufacturingProcess27  16.1986253
ManufacturingProcess28  17.3060225
ManufacturingProcess29  46.7965363
ManufacturingProcess30  35.5739492
ManufacturingProcess31  57.4009288
ManufacturingProcess32 100.0000000
ManufacturingProcess33  49.0972274
ManufacturingProcess34   7.6558269
ManufacturingProcess35  23.9400984
ManufacturingProcess36  76.2131059
ManufacturingProcess37  13.1438491
ManufacturingProcess38   1.4337566
ManufacturingProcess39   0.4978371
ManufacturingProcess40   1.5448843
ManufacturingProcess41   0.7734379
ManufacturingProcess42   0.0000000
ManufacturingProcess43   5.9478364
ManufacturingProcess44   1.7741678
ManufacturingProcess45   0.1311078
```

Looking at top 20 important predictors of SVM

```
> varImp(svmRTuned)
loess r-squared variable importance

  only 20 most important variables shown (out of 57)

                       Overall
ManufacturingProcess32  100.00
ManufacturingProcess13   93.19
BiologicalMaterial06     85.56
BiologicalMaterial03     79.69
ManufacturingProcess09   78.21
ManufacturingProcess36   76.21
BiologicalMaterial12     73.82
ManufacturingProcess17   73.79
ManufacturingProcess06   67.56
BiologicalMaterial02     66.58
ManufacturingProcess31   57.40
BiologicalMaterial11     53.60
ManufacturingProcess11   53.23
BiologicalMaterial08     51.85
ManufacturingProcess33   49.10
BiologicalMaterial04     47.15
ManufacturingProcess29   46.80
ManufacturingProcess12   40.73
BiologicalMaterial01     40.28
ManufacturingProcess30   35.57
```

**************************************Homework Code***

```
## We will use packeges: caret, earth, kernlab, and nnet
install.packages(c("caret", "earth", "kernlab", "nnet"))

set.seed(1)
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))
```

```
library(kernlab)
par(mfrow = c(1,3))

rbfSVM <- ksvm(x = x, y = y, data = sinData, kernel ="rbfdot", kpar = "automatic",
           C = 1, epsilon = 0.001)
modelPrediction <- predict(rbfSVM, newdata = dataGrid)
plot(x,y)
points(x = dataGrid$x, y = modelPrediction[,1], type = "l", col = "blue")

rbfSVM <- ksvm(x = x, y = y, data = sinData, kernel ="rbfdot", kpar = "automatic",
           C = 1, epsilon = 0.1)
modelPrediction <- predict(rbfSVM, newdata = dataGrid)
plot(x,y)
points(x = dataGrid$x, y = modelPrediction[,1], type = "l", col = "blue")


rbfSVM2 <- ksvm(x = x, y = y, data = sinData, kernel ="rbfdot", kpar = "automatic",
           C = 1, epsilon = 1)
modelPrediction2 <- predict(rbfSVM2, newdata = dataGrid)
plot(x,y)
points(x = dataGrid$x, y = modelPrediction2[,1], type = "l", col = "blue")

------------------
par(mfrow = c(1,3))
rbfSVM3 <- ksvm(x = x, y = y, data = sinData, kernel ="rbfdot", kpar = "automatic",
           C = 0.0001, epsilon = 0.1)
modelPrediction3 <- predict(rbfSVM3, newdata = dataGrid)
plot(x,y)
points(x = dataGrid$x, y = modelPrediction3[,1], type = "l", col = "blue")


rbfSVM4 <- ksvm(x = x, y = y, data = sinData, kernel ="rbfdot", kpar = "automatic",
            C = 1, epsilon = 0.1)
modelPrediction4 <- predict(rbfSVM4, newdata = dataGrid)
plot(x,y)
points(x = dataGrid$x, y = modelPrediction4[,1], type = "l", col = "blue")
```

```
rbfSVM5 <- ksvm(x = x, y = y, data = sinData, kernel ="rbfdot", kpar = "automatic",
              C = 10000000, epsilon = 0.1)
modelPrediction5 <- predict(rbfSVM5, newdata = dataGrid)
plot(x,y)
points(x = dataGrid$x, y = modelPrediction5[,1], type = "l", col = "blue")



#7.1(b)
par(mfrow = c(2,2))
rbfSVM6 <- ksvm(x = x, y = y, data = sinData, kernel ="rbfdot", kpar = list(sigma =
0.001), C = 1, epsilon = 0.1)
modelPrediction6 <- predict(rbfSVM6, newdata = dataGrid)
plot(x, y)
points(x = dataGrid$x, y = modelPrediction6[,1],type = "l", col = "blue")


rbfSVM7 <- ksvm(x = x, y = y, data = sinData, kernel ="rbfdot", kpar = list(sigma = 0.1),
C = 1, epsilon = 0.1)
modelPrediction7 <- predict(rbfSVM7, newdata = dataGrid)
plot(x, y)
points(x = dataGrid$x, y = modelPrediction7[,1],type = "l", col = "blue")

rbfSVM8 <- ksvm(x = x, y = y, data = sinData, kernel ="rbfdot", kpar = list(sigma = 10),
C = 1, epsilon = 0.1)
modelPrediction8 <- predict(rbfSVM8, newdata = dataGrid)
plot(x, y)
points(x = dataGrid$x, y = modelPrediction8[,1],type = "l", col = "blue")

rbfSVM9 <- ksvm(x = x, y = y, data = sinData, kernel ="rbfdot", kpar = list(sigma =
100), C = 1, epsilon = 0.1)
modelPrediction9 <- predict(rbfSVM9, newdata = dataGrid)
plot(x, y)
points(x = dataGrid$x, y = modelPrediction9[,1],type = "l", col = "blue")
```

```
#7.2 (a)
library(mlbench)
library(caret)
set.seed(1)
trainingData <- mlbench.friedman1(200, sd = 1)
trainingData$x <- data.frame(trainingData$x)
featurePlot(trainingData$x, trainingData$y)
testData <- mlbench.friedman1(5000, sd = 1)
testData$x <- data.frame(testData$x)

##knn model

knnModel <- train(x = trainingData$x,y = trainingData$y,method = "knn",preProc =
c("center", "scale"), tuneLength = 10)
knnModel
knnPred <- predict(knnModel, newdata = testData$x)
plot(knnModel)
postResample(pred = knnPred, obs = testData$y)


##MARS model
library(earth)
library(AppliedPredictiveModeling)
marsGrid <- expand.grid(.degree = 1:2, .nprune = 2:30)
set.seed(1)
marsTuned <- train(trainingData$x, trainingData$y,method = "earth", tuneGrid =
marsGrid, trControl = trainControl(method = "cv"))
marsTuned
summary(marsTuned)
marsPred <- predict(marsTuned, newdata = testData$x)
plotmo(marsTuned)
plot(marsTuned)
varImp(marsTuned)
postResample(pred = marsPred, obs = testData$y)
```

```
##7.5

library(RANN)

data("ChemicalManufacturingProcess")


predictors<-subset(ChemicalManufacturingProcess, select = -Yield)
yield<-subset(ChemicalManufacturingProcess, select="Yield")

P1<- preProcess(predictors,method=c("knnImpute"))
Predictors1 <- predict(P1,predictors)

P2<- preProcess(Predictors1, method = c("center","scale"))
Predictors2 <- predict(P2,Predictors1)


Split<-createDataPartition(yield$Yield, p=0.8, list = FALSE)
TrainP<- Predictors2[Split,]
TrainY<-yield[Split,]

TestP<- Predictors2[-Split,]
TestY<-yield[-Split,]



##Neural Network with resampling

nnetGrid <- expand.grid(.decay = c(0, 0.01, .1),
                        .size = c(1:10))
set.seed(1)
ctrl <- trainControl(method = "cv", number = 10)
nnetTune <- train(TrainP, TrainY,
                  method = "nnet",
                  tuneGrid = nnetGrid,
                  trControl = ctrl,
                  preProc = c("center", "scale"),
```

```
                 linout = TRUE,
                 trace = FALSE,
                 MaxNWts = 10 * (ncol(TrainP) + 1) + 10 + 1,
                 maxit = 500)

nnetTune
plot(nnetTune)

Pred=predict(nnetTune, TestP)
SSEnnet=mean((TestY-Pred)^2)
RMSE=sqrt(SSEnnet)
Rsquared=(cor(TestY, Pred))^2
RMSE
Rsquared


##MARS with resampling

library(earth)
library(AppliedPredictiveModeling)

marsGrid2 <- expand.grid(.degree = 1:2, .nprune = 2:50)
set.seed(1)

marsTuned2 <- train(TrainP, TrainY,
                  method = "earth",
                  tuneGrid = marsGrid2,
                  trControl = trainControl(method = "cv"))

marsTuned2
summary(marsTuned2)
predict(marsTuned2, newdata = TestP)
plot(marsTuned2)


varImp(marsTuned2)
```

```
Pred=predict(marsTuned2, TestP)
SSEmars=mean((TestY-Pred)^2)
RMSE=sqrt(SSEmars)
Rsquared=(cor(TestY, Pred))^2
RMSE
Rsquared


##SVM

install.packages("kernlab")
library(kernlab)
library(AppliedPredictiveModeling)

svmRTuned <- train(TrainP, TrainY,
              method = "svmRadial",
              preProc = c("center", "scale"),
              tuneLength = 14,
              trControl = trainControl(method = "cv"))



svmRTuned
plot(svmRTuned)

## The subobject named finalModel contains the model created by the ksvm
## function:

svmRTuned$finalModel

important=varImp(svmRTuned)
plot(important, top = 25, scales = list(y = list(cex = .95)))

Pred=predict(svmRTuned, TestP)
SSEsvm=mean((TestY-Pred)^2)
RMSE=sqrt(SSEsvm)
Rsquared=(cor(TestY, Pred))^2
```

```
RMSE
Rsquared



##knn

library(AppliedPredictiveModeling)
library(caret)
knnDescr <- TrainP[, -nearZeroVar(TrainP)]
set.seed(1)
knnTune <- train(knnDescr,
                TrainY,
                method = "knn",
                # Center and scaling will occur for new predictions too
                preProc = c("center", "scale"),
                tuneGrid = data.frame(.k = 1:20),
                trControl = trainControl(method = "cv"))
knnTune
plot(knnTune)

Pred=predict(knnTune, TestP)
SSEknn=mean((TestY-Pred)^2)
RMSE=sqrt(SSEknn)
Rsquared=(cor(TestY, Pred))^2
RMSE
Rsquared

#--------------------------------------------------------

#lasso
set.seed(1)
cv <- cv.glmnet(as.matrix(TrainP), TrainY, alpha = 1,
               nfolds=3, standardize = TRUE)
plot(cv)
cv$lambda.min
```

```
lasso_model <- glmnet(as.matrix(TrainP), TrainY, alpha = 1, lambda = cv$lambda.min,
standardize = TRUE)
#Predicting on Test data
y_hat_cv <- predict(lasso_model, as.matrix(TestP))

postResample(y_hat_cv, TestY)
#<<<<<<<<<<<<<<<<<<<<<<<<<lasso>>>>>>>>>>>>>>>>>>>>>>>>

library(caret)
varImp(lasso_model, lambda = cv$lambda.min)
```