DATA 624 Homework 7

Bin Lin

2018-5-7

7.2. Friedman (1991) introduced several benchmark data sets create by simulation. One of these simulations used the following nonlinear equation to create data:

$$y = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + N(0, \sigma^2)$$

where the x values are random variables uniformly distributed between [0, 1] (there are also 5 other non-informative variables also created in the simulation). The package mlbench contains a function called mlbench.friedman1 that simulates these data:

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

```
library(mlbench)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(earth)
```

```
## Loading required package: plotmo
```

```
## Loading required package: plotrix
```

```
## Loading required package: TeachingDemos
```

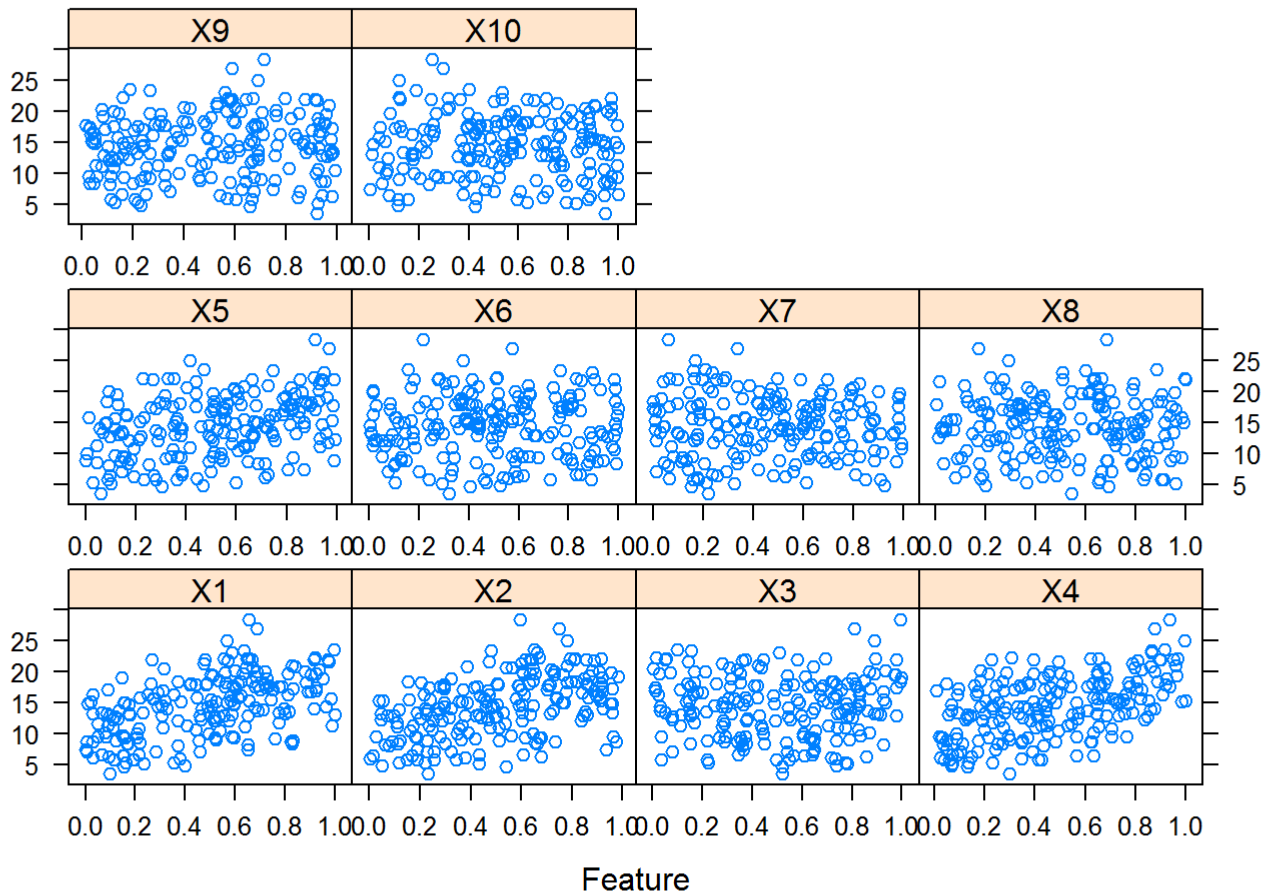
```
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:caret':
##
##     R2
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
#install.packages("earth")

set.seed(200)
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)
```

```
knnModel <- train(x = trainingData$x, y = trainingData$y, method = "knn", preProc = c("center",
"scale"), tuneLength = 10)
knnModel
```

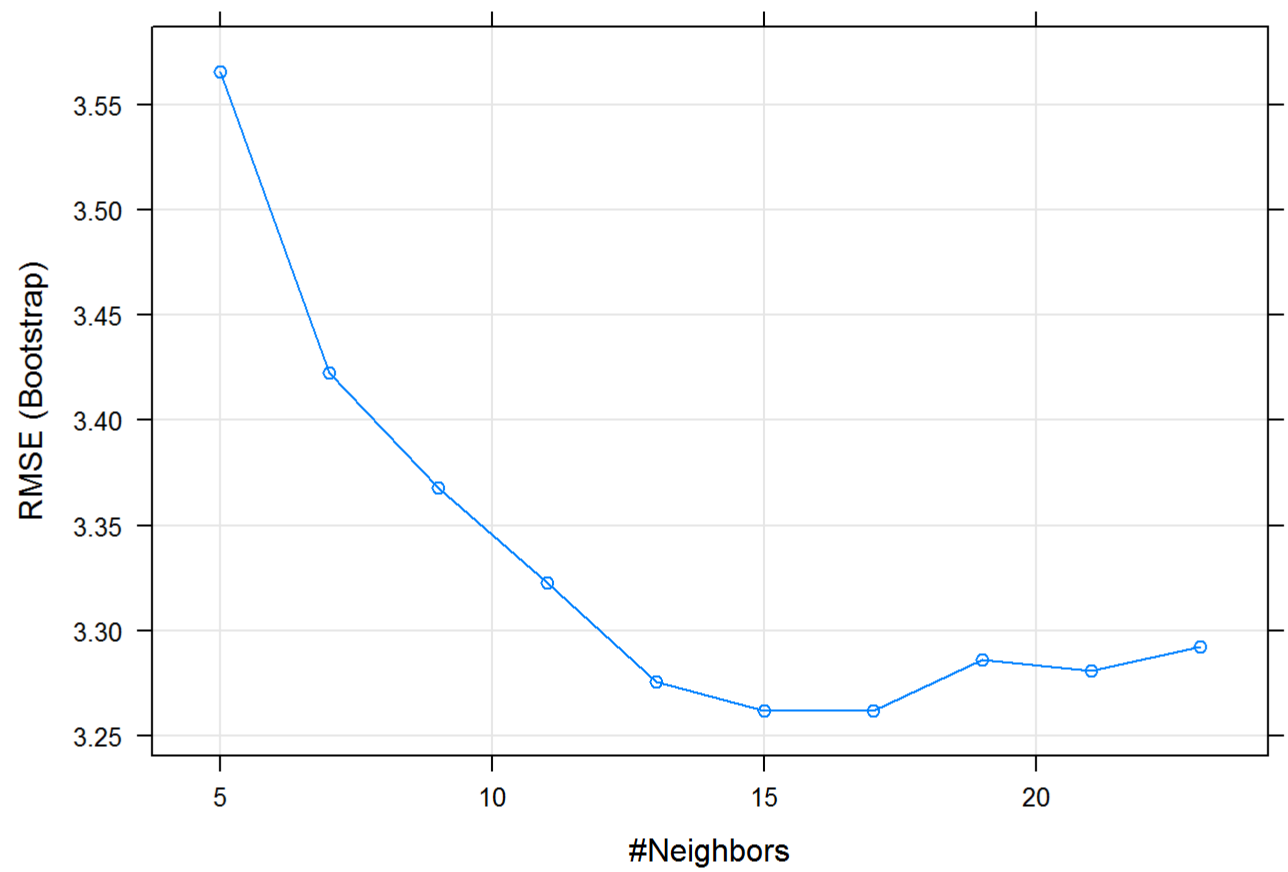
```
## k-Nearest Neighbors
##
## 200 samples
##  10 predictor
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 200, 200, 200, 200, 200, 200, ...
## Resampling results across tuning parameters:
##
##   k   RMSE      Rsquared   MAE
##    5  3.565620  0.4887976  2.886629
##    7  3.422420  0.5300524  2.752964
##    9  3.368072  0.5536927  2.715310
##   11  3.323010  0.5779056  2.669375
##   13  3.275835  0.6030846  2.628663
##   15  3.261864  0.6163510  2.621192
##   17  3.261973  0.6267032  2.616956
##   19  3.286299  0.6281075  2.640585
##   21  3.280950  0.6390386  2.643807
##   23  3.292397  0.6440392  2.656080
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 15.
```

```
knnPred <- predict(knnModel, newdata = testData$x)
postResample(pred = knnPred, obs = testData$y)
```

```
##      RMSE  Rsquared       MAE
## 3.1750657 0.6785946 2.5443169
```

According to the following graph, the optimal k-Nearest Neighbors model contains 15 neighbors with RMSE value at 3.175.

```
plot(knnModel)
```



The following code, the data was used to train for MARS model, after the preprocess procedures such as center, scale have already completed.

```
set.seed(888)
marsGrid <- expand.grid(degree = 1:2, nprune = 1:20)
marsTuned <- train(x = trainingData$x, y = trainingData$y, method = "earth", tuneGrid = marsGri
d, trControl = trainControl(method = "cv"), preProc = c("center", "scale"))
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

Interpretation: MARS model is apparently better than KNN model since its RMSE value is 1.323, which is much less than the one generated from KNN model. The final values used for the model were nprune = 16 and degree = 2.In addition, MARS selects the informative predictors (X1-X5) only. X6-X10 predictors have no importance at all.

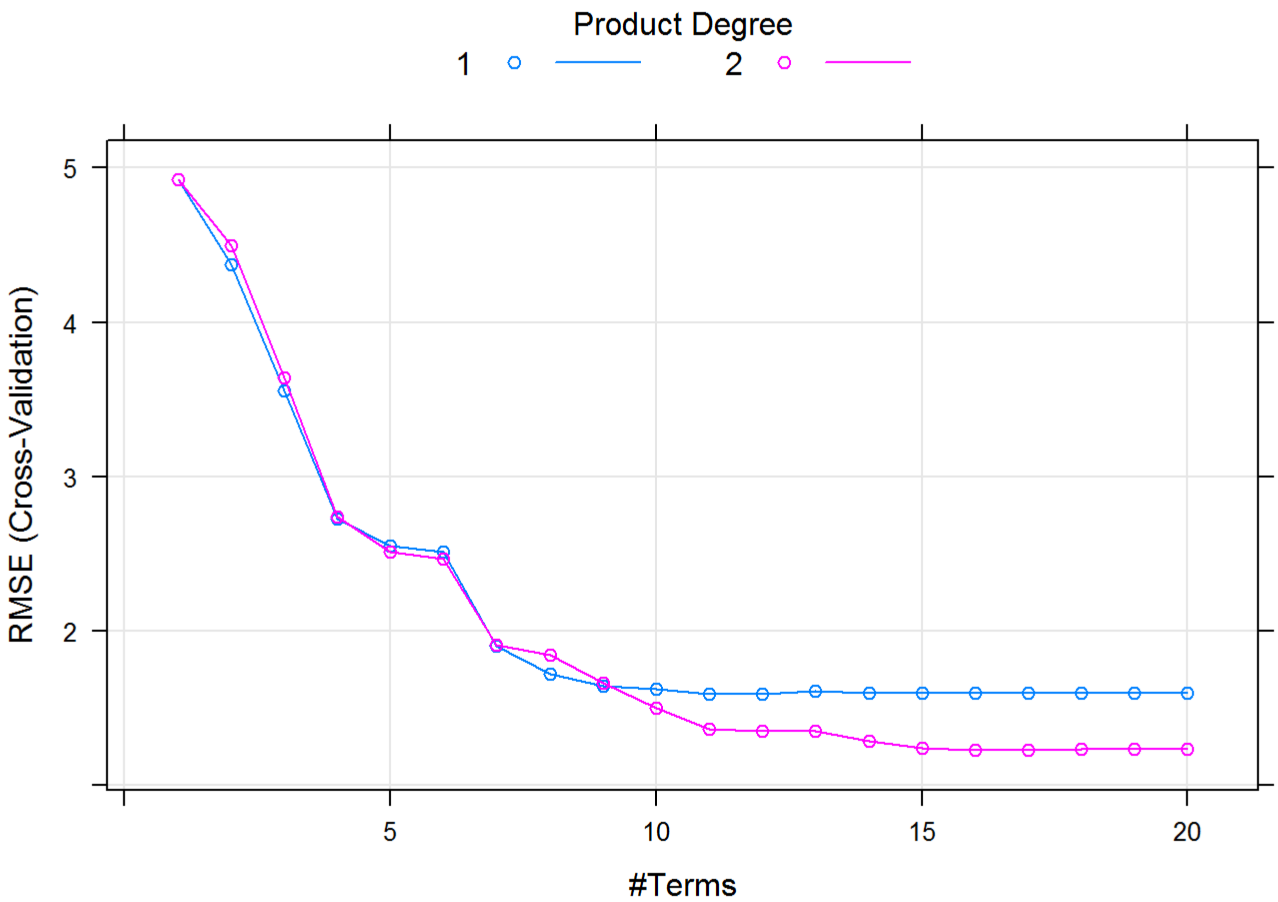
```
marsPred <- predict(marsTuned, newdata = testData$x)
postResample(pred = marsPred, obs = testData$y)
```

```
##      RMSE  Rsquared       MAE
## 1.2793868 0.9343367 1.0091132
```

```
varImp(marsTuned)
```

```
## earth variable importance
##
##      Overall
## X1   100.00
## X4    85.12
## X2    69.20
## X5    49.23
## X3    39.89
## X8     0.00
## X6     0.00
## X9     0.00
## X10    0.00
## X7     0.00
```

```
plot(marsTuned)
```




```
marsTuned
```

```
## Multivariate Adaptive Regression Spline
##
## 200 samples
##  10 predictor
##
## Pre-processing: centered (10), scaled (10)
## 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        1      4.921307        NaN  4.0422513
##   1        2      4.372600  0.2494859  3.6584051
##   1        3      3.555270  0.4985649  2.8726615
##   1        4      2.726352  0.7089775  2.1900423
##   1        5      2.547907  0.7479963  2.0594070
##   1        6      2.509322  0.7615072  1.9793715
##   1        7      1.901733  0.8634079  1.4904166
##   1        8      1.719886  0.8835157  1.3687368
##   1        9      1.644159  0.8921695  1.3043667
##   1       10      1.623350  0.8948379  1.2795465
##   1       11      1.589075  0.9002714  1.2555563
##   1       12      1.593492  0.8997416  1.2497490
##   1       13      1.610554  0.8972277  1.2600103
##   1       14      1.600163  0.8985099  1.2527223
##   1       15      1.600163  0.8985099  1.2527223
##   1       16      1.600163  0.8985099  1.2527223
##   1       17      1.600163  0.8985099  1.2527223
##   1       18      1.600163  0.8985099  1.2527223
##   1       19      1.600163  0.8985099  1.2527223
##   1       20      1.600163  0.8985099  1.2527223
##   2        1      4.921307        NaN  4.0422513
##   2        2      4.496073  0.2034636  3.7413677
##   2        3      3.638997  0.4724445  2.9952571
##   2        4      2.741191  0.7116752  2.2002016
##   2        5      2.513763  0.7579066  2.0381846
##   2        6      2.465818  0.7681812  1.9718287
##   2        7      1.911666  0.8594136  1.5023921
##   2        8      1.846490  0.8631160  1.3941037
##   2        9      1.660966  0.8904560  1.3065837
##   2       10      1.498962  0.9085211  1.1720222
##   2       11      1.366536  0.9224437  1.0776368
##   2       12      1.354300  0.9255822  1.0697916
##   2       13      1.349547  0.9253808  1.0753422
##   2       14      1.286753  0.9330843  1.0312404
##   2       15      1.238795  0.9377730  0.9894584
##   2       16      1.226555  0.9391833  0.9784266
##   2       17      1.230141  0.9391998  0.9815644
##   2       18      1.234600  0.9388660  0.9846447
##   2       19      1.234600  0.9388660  0.9846447
##   2       20      1.234600  0.9388660  0.9846447
##
```

```
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nprune = 16 and degree = 2.
```

7.5. Exercise 6.3 describes data for a chemical manufacturing process. Use the same data imputation, data splitting, and pre-processing steps as before and train several nonlinear regression models.

a. Which nonlinear regression model gives the optimal resampling and test set performance?

```
library(AppliedPredictiveModeling)
library(mice)
library(e1071)
#install.packages("RANN")
library(RANN)

data(ChemicalManufacturingProcess)
set.seed(888)

a <- mice(ChemicalManufacturingProcess, m = 1, method = "pmm", print = F)
C_M_P <- complete(a)

trainingRows <- createDataPartition(C_M_P$Yield, p = .80, list= FALSE)

yield_train <- C_M_P[trainingRows, 1]
predictor_train <- C_M_P[trainingRows, -1]

yield_test <- C_M_P[-trainingRows, 1]
predictor_test <- C_M_P[-trainingRows, -1]

CMP_trans <- preProcess(predictor_train, method = c("nzv", "BoxCox", "center", "scale", "knnImpu
te"))
ctrl <- trainControl(method = "cv", number = 10)
```

Neural Networks

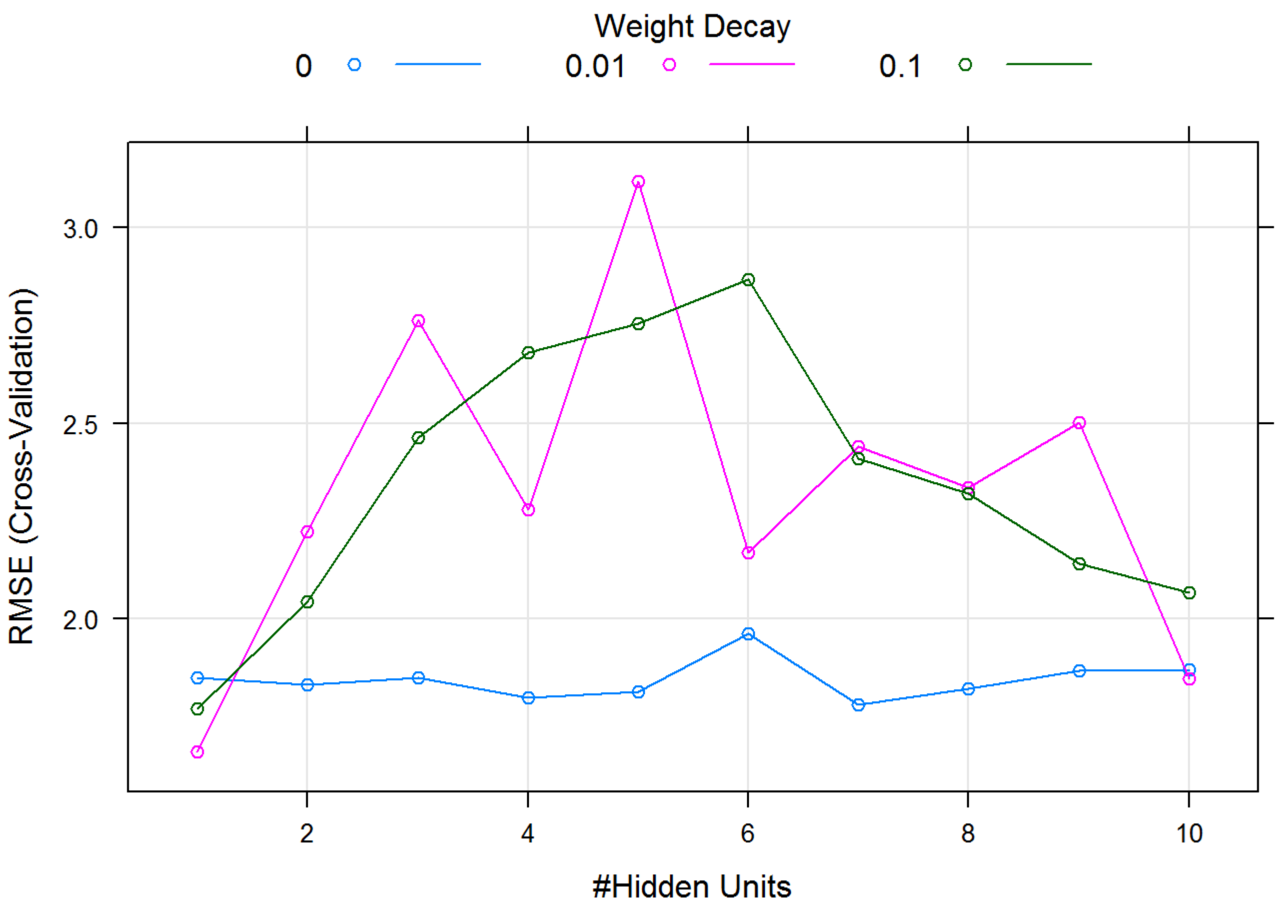
```
set.seed(888)

nnetGrid <- expand.grid(decay = c(0, 0.01, .1), size = c(1:10))

nnetTune <- train(x = predictor_train, y = yield_train, method = "nnet", tuneGrid = nnetGrid, tr
Control = ctrl, linout = TRUE, trace = FALSE, maxit = 500)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
nnetPred <- predict(nnetTune, newdata = predictor_test)
postResample(pred = nnetPred, obs = yield_test)
```

```
##      RMSE  Rsquared       MAE
## 1.6052444 0.3915792 1.3444695
```

```
plot(nnetTune)
```

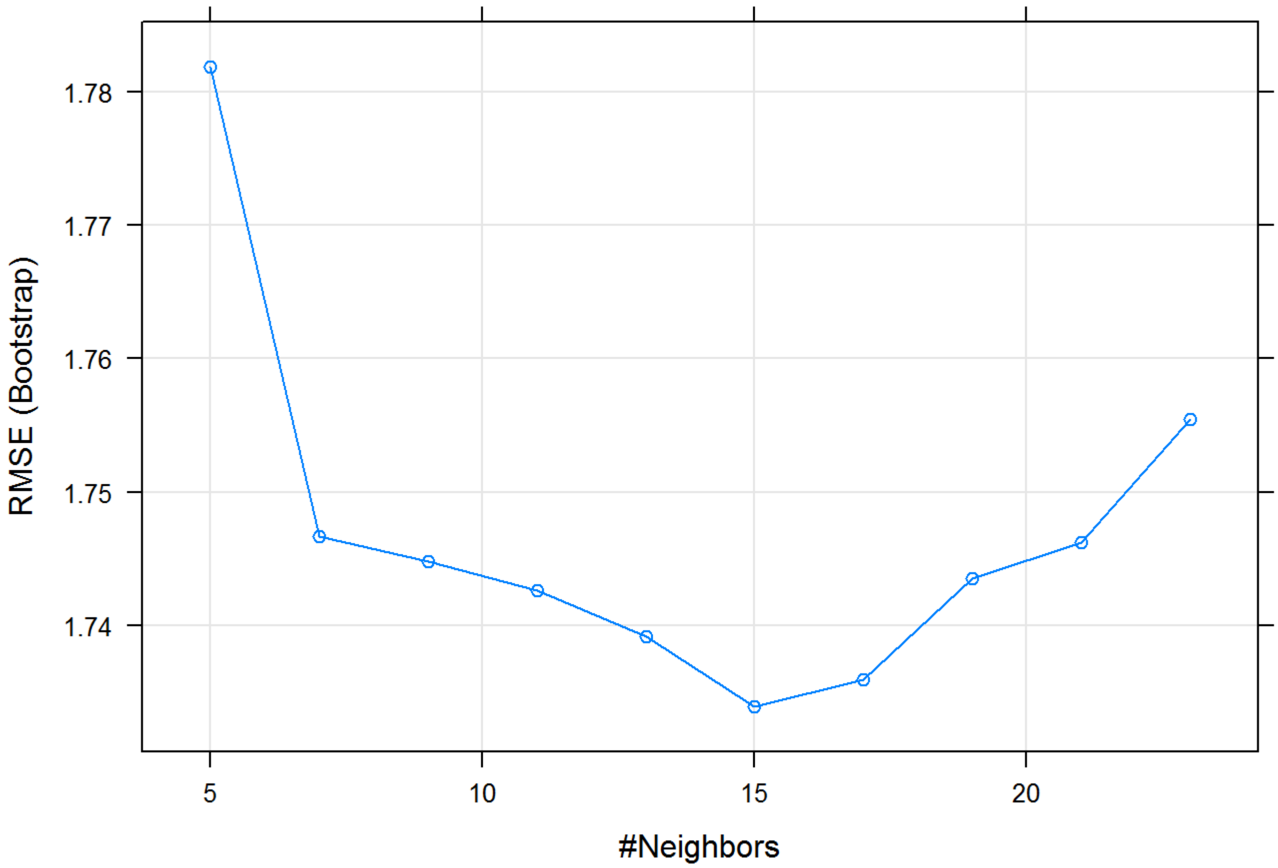


K-Nearest Neighbors

```
set.seed(888)
knnTune <- train(x = predictor_train, y = yield_train, method = "knn", tuneLength = 10)

knnPred <- predict(knnTune, newdata = predictor_test)
postResample(pred = knnPred, obs = yield_test)
```

```
##      RMSE  Rsquared       MAE
## 1.1436604 0.7049123 0.9106875
```

```
plot(knnTune)
```



Multivariate Adaptive Regression Splines

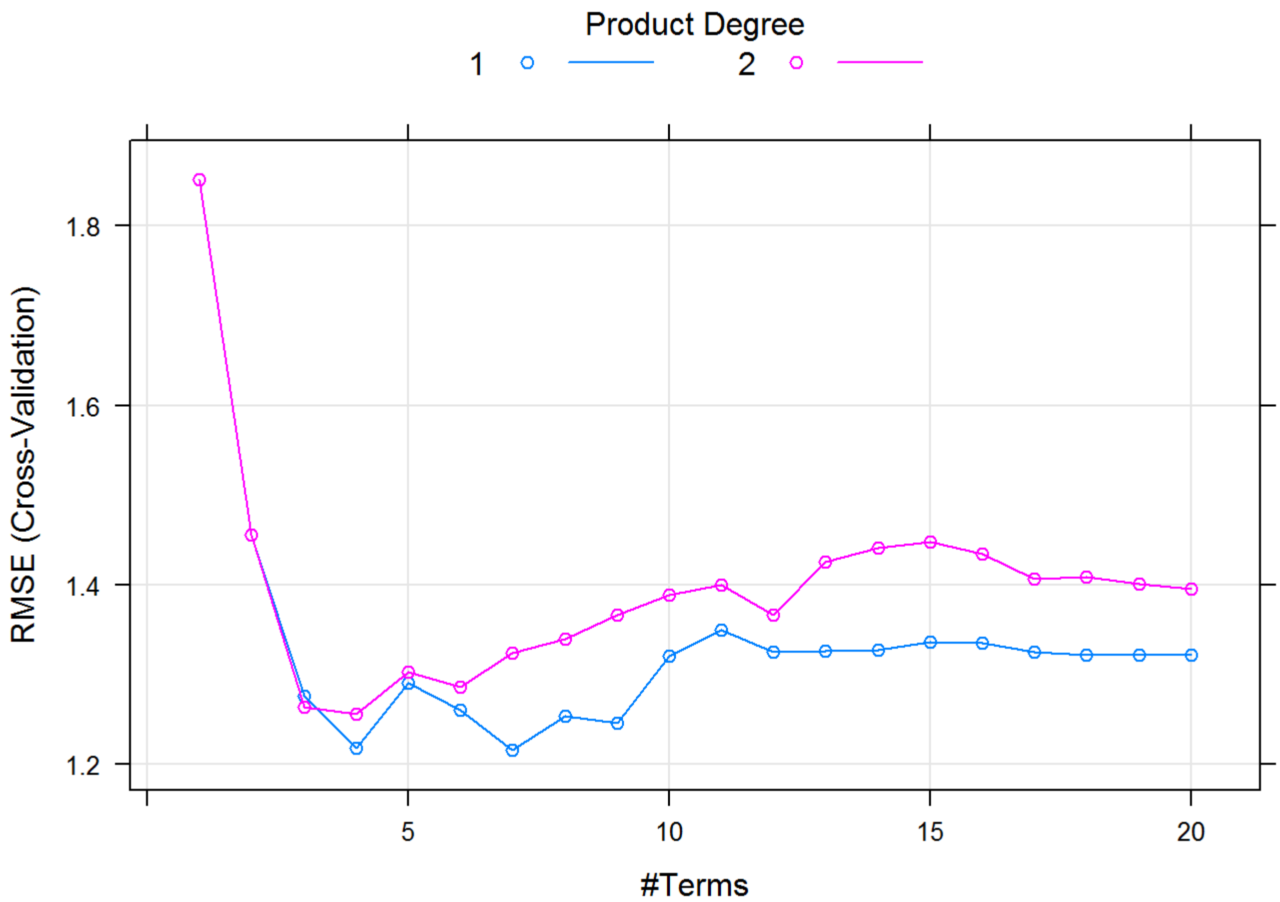
```
set.seed(888)

marsGrid <- expand.grid(degree = 1:2, nprune = 1:20)
marsTune <- train(x = predictor_train, y = yield_train, method = "earth", tuneGrid = marsGrid, t
rControl = ctrl)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
marsPred <- predict(marsTune, newdata = predictor_test)
postResample(pred = marsPred, obs = yield_test)
```

```
##      RMSE  Rsquared       MAE
## 1.0643609 0.6254264 0.9566682
```

```
plot(marsTune)
```



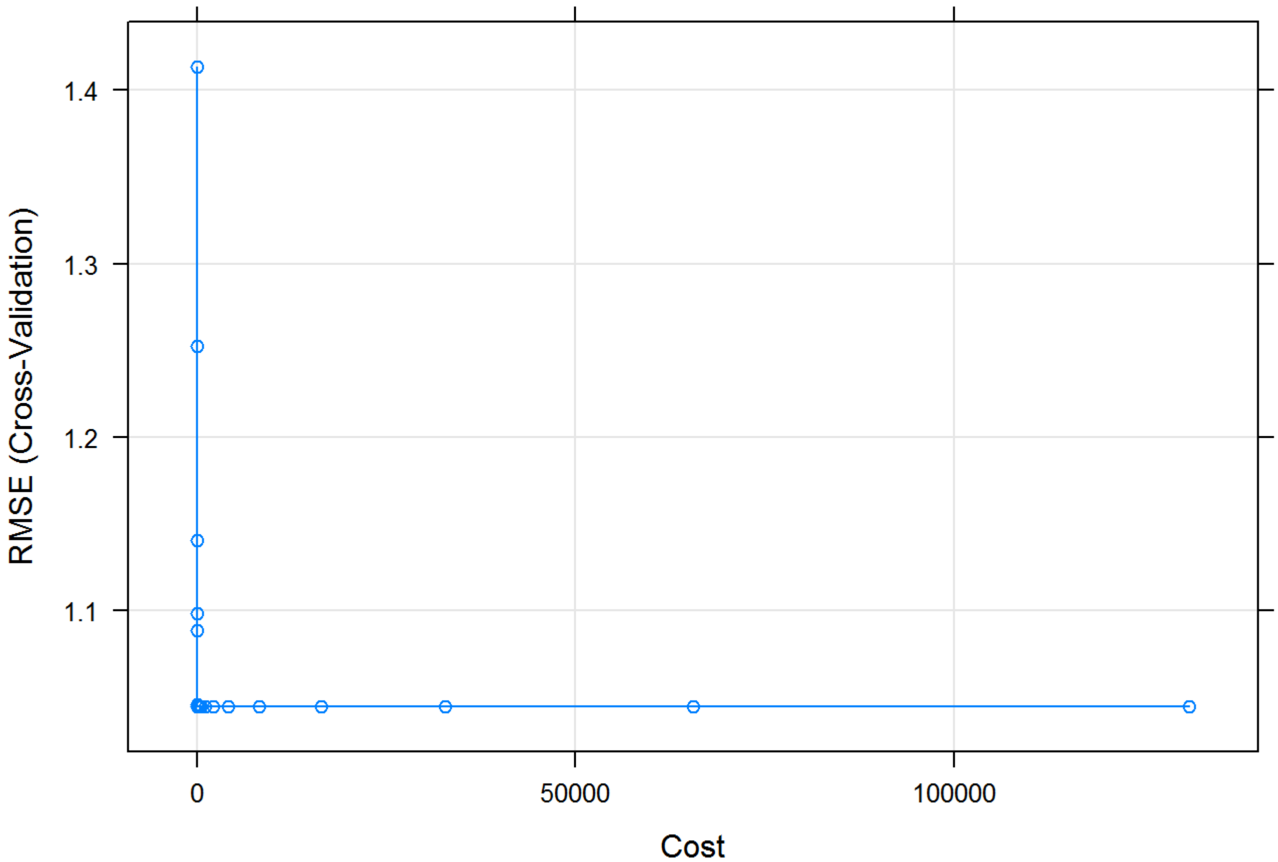
Support Vector Machines

```
set.seed(888)

svmRTune <- train(predictor_train, yield_train, method = "svmRadial", tuneLength = 20, trControl
 = ctrl)

#svmLTune <- train(predictor_train, yield_train, method = "svmLinear", tuneLength = 20, trContro
l = ctrl)

#svmPTune <- train(predictor_train, yield_train, method = "svmPoly", tuneLength = 20, trControl
 = ctrl)
```

```
svmRPred <- predict(svmRTune, newdata = predictor_test)
postResample(pred = svmRPred, obs = yield_test)
```

```
##      RMSE  Rsquared       MAE
## 1.0441130 0.6537003 0.8766606
```

```
#svmLPred <- predict(svmLTune, newdata = predictor_test)
#postResample(pred = svmLPred, obs = yield_test)

#svmPPred <- predict(svmPTune, newdata = predictor_test)
#postResample(pred = svmPPred, obs = yield_test)


plot(svmRTune)
```



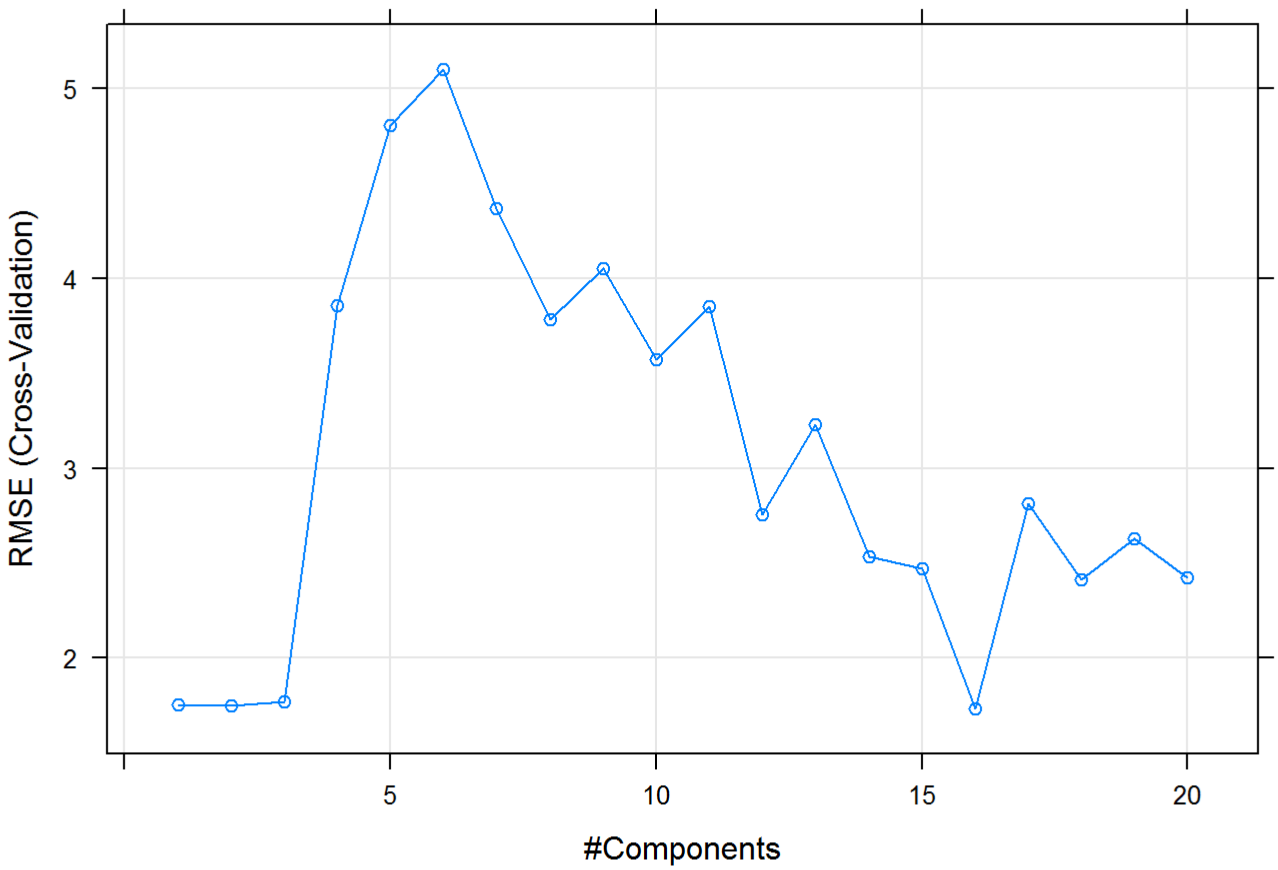

```
#plot(svmLTune)
#plot(svmPTune)
```

Partial Least Squares

```
set.seed(888)

plsTune <- train(predictor_train, yield_train, method = "pls", tuneLength = 20, trControl = ctr
l)

plsPred <- predict(plsTune, newdata = predictor_test)
postResample(pred = plsPred, obs = yield_test)
```

```
##      RMSE  Rsquared       MAE
## 2.7779022 0.2398376 1.3671123
```

```
plot(plsTune)
```




```
pls_RMSE <- min(plsTune$results$RMSE)
nnet_RMSE <- min(nnetTune$results$RMSE)
mars_RMSE <- min(marsTune$results$RMSE)
svmR_RMSE <- min(svmRTune$results$RMSE)
knn_RMSE <- min(knnTune$results$RMSE)


result <- data.frame(Model = c("pls", "nnet", "mars", "svmR", "knn"), RMSE = c(pls_RMSE, nnet_RM
SE, mars_RMSE, svmR_RMSE, knn_RMSE))

result
```

```
##   Model     RMSE
## 1   pls 1.732079
## 2  nnet 1.660641
## 3  mars 1.215567
## 4  svmR 1.044491
## 5   knn 1.733900
```

Based on the result that I obtained for each model, the best model that generate optimal resampling and test set performance is the Support Vector Machines model using the method “svmRadial”. Because this model has the lowest RMSE value which is 1.044.

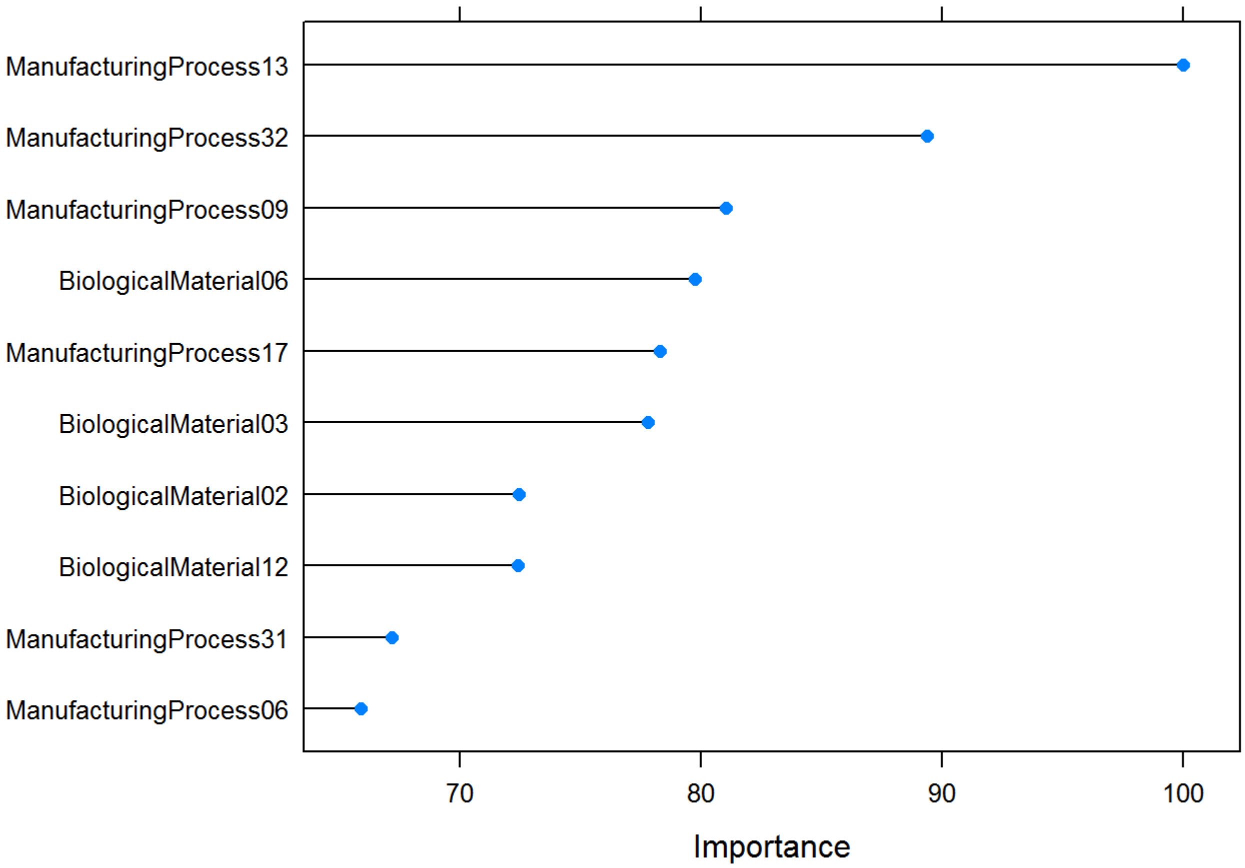
b. Which predictors are most important in the optimal nonlinear regression model? Do either the biological or process variables dominate the list? How do the top ten important predictors compare to the top ten predictors from the optimal linear model?

```
varImp(svmRTune)
```

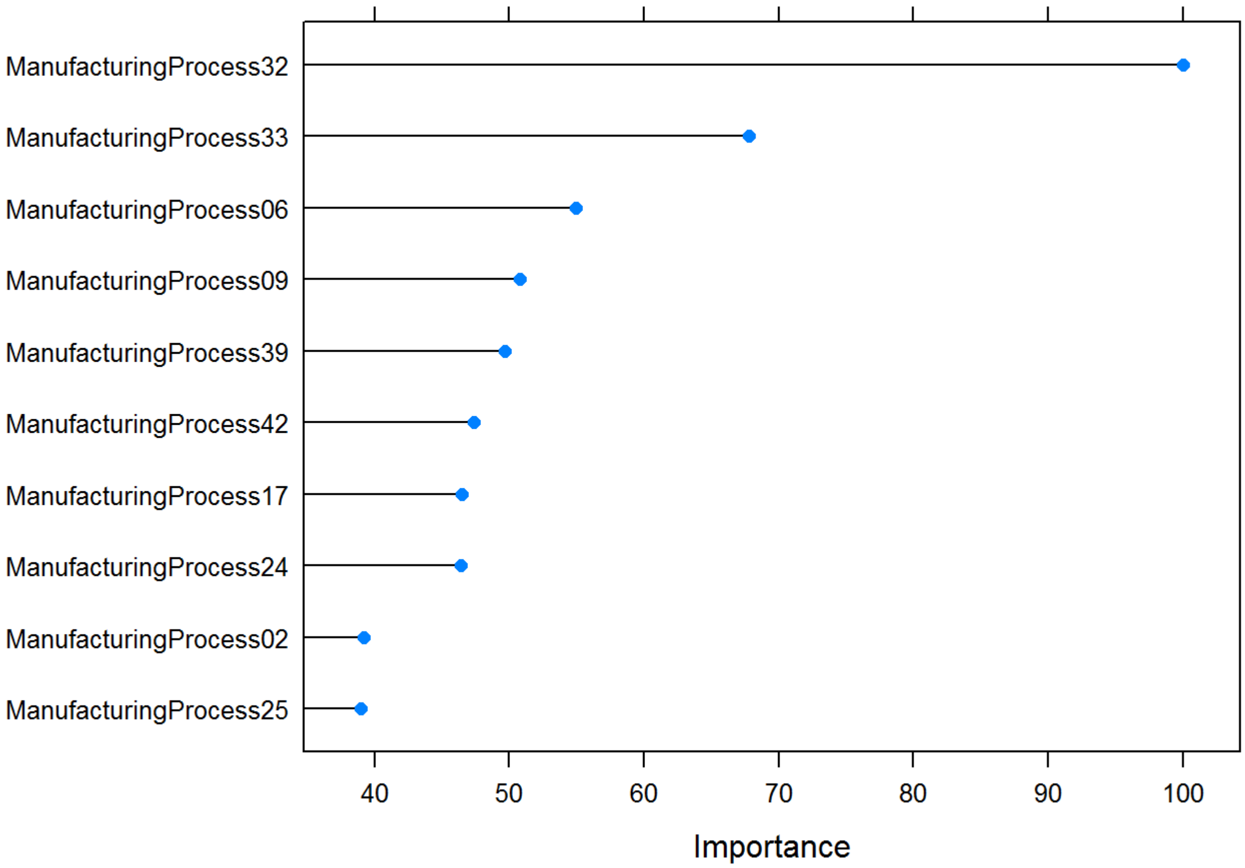
```
## loess r-squared variable importance
##
##   only 20 most important variables shown (out of 57)
##
##                        Overall
## ManufacturingProcess13  100.00
## ManufacturingProcess32   89.39
## ManufacturingProcess09   81.01
## BiologicalMaterial06     79.74
## ManufacturingProcess17   78.30
## BiologicalMaterial03     77.77
## BiologicalMaterial02     72.42
## BiologicalMaterial12     72.38
## ManufacturingProcess31   67.15
## ManufacturingProcess06   65.88
## BiologicalMaterial11     61.96
## ManufacturingProcess36   60.51
## ManufacturingProcess30   48.66
## BiologicalMaterial08     48.35
## BiologicalMaterial04     45.39
## ManufacturingProcess33   43.97
## ManufacturingProcess29   43.70
## BiologicalMaterial01     42.25
## ManufacturingProcess11   38.09
## BiologicalMaterial09     37.04
```

By comparing the importance figure between the optimal non-linear model and the optimal linea model, we are able know ManufacturingProcess32, 06. 09 all have significant weights on both model. PLS model is solely built upon manufacturing process predictors. However, SVM model gives more weight to Biological Material, when4 out of top 10 predictors are from this class. The mode suprising finding is that ManufacturingProcess13 is the most important predictor at SVM model, but it is not one of the top 10 predictors for PLS model.

```
par(mfrow = c(1, 2))
plot(varImp(svmRTune), top = 10)
```



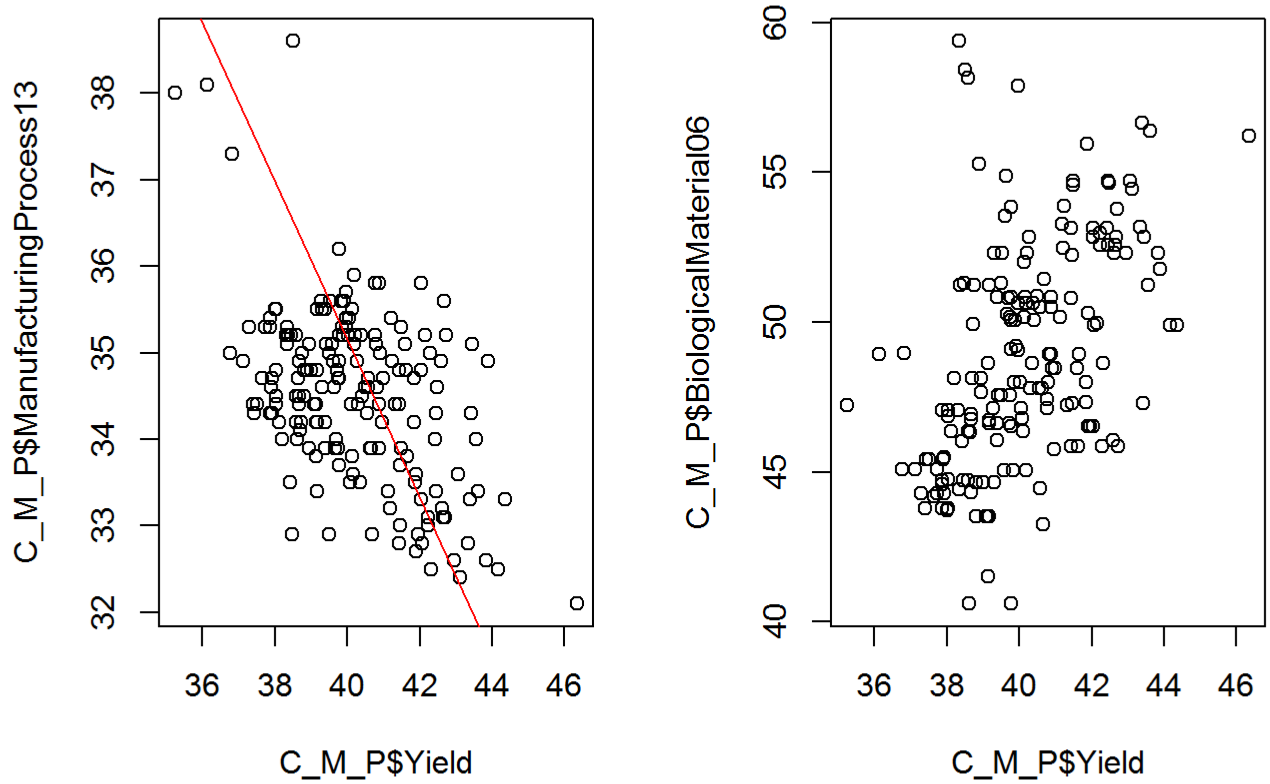

```
plot(varImp(plsTune), top = 10)
```



c. Explore the relationships between the top predictors and the response for the predictors that are unique to the optimal nonlinear regression model. Do these plots reveal intuition about the biological or process predictors and their relationship with yield?

Out of the top 5 predictors, SVM model has large weight on ManufacturingProcess13 and BiologicalMaterial06, which are unique to this model. Therefore, I am going to investigate on the relationship between these two predictors with their respective yield in particular. The following graph proves that the yield has negative relationship with ManufacturingProcess13 and positive relationship with BiologicalMaterial06

```
par(mfrow = c(1, 2))
plot(C_M_P$Yield, C_M_P$ManufacturingProcess13)
abline(lm(C_M_P$Yield ~ C_M_P$ManufacturingProcess13), col="red")

plot(C_M_P$Yield, C_M_P$BiologicalMaterial06)
abline(lm(C_M_P$Yield ~ C_M_P$BiologicalMaterial06), col="red")
```


C_M_P$ManufacturingProcess13
32
33
34
35
36
37
38
36
38
40
42
44
46
C_M_P$Yield
C_M_P$BiologicalMaterial06
40
45
50
55
60
36
38
40
42
44
46
C_M_P$Yield