DATA624 Homework 8

Bin Lin

2018-5-9

8.1. Recreate the simulated data from Exercise 7.2:

```
library(mlbench)
set.seed(200)
simulated <- mlbench.friedman1(200, sd = 1)
simulated <- cbind(simulated$x, simulated$y)
simulated <- as.data.frame(simulated)
colnames(simulated)[ncol(simulated)] <- "y"
```

a. Fit a random forest model to all of the predictors, then estimate the variable importance scores:

```
#install.packages("randomForest")
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(caret)
```

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

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

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
set.seed(100)
model1 <- randomForest(y ~ ., data = simulated, importance = TRUE, ntree = 1000)
rfImp1 <- varImp(model1, scale = FALSE)
```

Did the random forest model significantly use the uninformative predictors (V6 - V10)?

The random forest model did not significantly use the uninformative predictors(V6-V10) due to their importance level are much smaller than the informative predictors(V1-V5).

```
rfImp1
```

```
##         Overall
## V1   8.63576145
## V2   6.36633951
## V3   0.73342992
## V4   7.72426144
## V5   2.28781522
## V6   0.04503439
## V7   0.01582424
## V8  -0.10274799
## V9  -0.07855803
## V10 -0.05070734
```

b. Now add an additional predictor that is highly correlated with one of the informative predictors. For example:

```
simulated2 <- simulated
simulated2$V11 <- simulated$V1 + rnorm(200) * .1
cor(simulated2$V11, simulated2$V1)
```

```
## [1] 0.9414719
```

Fit another random forest model to these data. Did the importance score for V1 change? What happens when you add another predictor that is also highly correlated with V1?

```
set.seed(100)
model2 <- randomForest(y ~ ., data = simulated2, importance = TRUE, ntree = 1000)
rfImp2 <- varImp(model2, scale = FALSE)
rfImp2
```

```
##         Overall
## V1   5.73035635
## V2   6.33253246
## V3   0.64453732
## V4   6.75632730
## V5   2.08424598
## V6   0.14382046
## V7   0.01590056
## V8   0.01734896
## V9  -0.08433143
## V10 -0.05625675
## V11  4.19983475
```

The importance score for V1 change from 8.636 to 5.727. After I added 11th predictor that is also highly correlated with V1, all the informative predictors have decreasing importance levels. For unimformative predictors the changes are not obvious. In addition to that, the 11th predictor also gain high importance level just like V1.

```
rfImp1 <- rbind(rfImp1, NA)
cbind(rfImp1, rfImp2)
```

```
##         Overall     Overall
## V1   8.63576145  5.73035635
## V2   6.36633951  6.33253246
## V3   0.73342992  0.64453732
## V4   7.72426144  6.75632730
## V5   2.28781522  2.08424598
## V6   0.04503439  0.14382046
## V7   0.01582424  0.01590056
## V8  -0.10274799  0.01734896
## V9  -0.07855803 -0.08433143
## V10 -0.05070734 -0.05625675
## 11           NA  4.19983475
```

c. Use the cforest function in the party package to fit a random forest model using conditional inference trees. The party package function varimp can calculate predictor importance. The conditional argument of that function toggles between the traditional importance measure and the modified version described in Strobl et al. (2007). Do these importances show the same pattern as the traditional random forest model?

```
#install.packages("party")
library(party)
```

```
## Loading required package: grid
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
## 
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
## 
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
set.seed(100)

ctrl1 <- cforest_control(mtry = ncol(simulated) - 1)
tree1 <- cforest(y ~ ., data = simulated, controls = ctrl1)

cfimp1 <- varImp(tree1)

ctrl2 <- cforest_control(mtry = ncol(simulated2) - 1)
tree2 <- cforest(y ~ ., data = simulated2, controls = ctrl2)

cfimp2 <- varImp(tree2)
```

The importances show the same pattern as the traditional random forest model. The importance score for V1 change from 9.844 to 3.608. After I added 11th predictor that is also highly correlated with V1, all the informative predictors have decreasing importance levels. For unimformative predictors the changes are not obvious. In addition to that, the 11th predictor also gain high importance level.

```
cfimp1 <- rbind(cfimp1, NA)
cbind(cfimp1, cfimp2)
```

```
##          Overall      Overall
## V1   9.910468369  7.548067198
## V2   8.073769074  7.825289642
## V3   0.008692455 -0.012950356
## V4  10.183786204 10.462213365
## V5   2.353049419  2.327029283
## V6   0.023534821  0.003602874
## V7   0.103887618  0.035647509
## V8  -0.011166128 -0.018521013
## V9  -0.027114637 -0.025849459
## V10  0.028900824 -0.012828186
## 11            NA  2.305285264
```

d. Repeat this process with different tree models, such as boosted trees and Cubist. Does the same pattern occur?

Bagged Trees

```
library(ipred)
set.seed(100)

baggedTree1 <- bagging(y ~ ., data = simulated)
btimp1 <- varImp(baggedTree1)

baggedTree2 <- bagging(y ~ ., data = simulated2)
btimp2 <- varImp(baggedTree2)
```

```
btimp1 <- rbind(btimp1, NA)
cbind(btimp1, btimp2)
```

```
##       Overall   Overall
## V1  1.6784469 1.7033981
## V10 0.8333179 0.7064733
## V2  1.9844894 1.5559749
## V3  1.2248918 2.4104690
## V4  2.7076782 1.1620767
## V5  2.3904283 2.6479644
## V6  1.0202490 2.4559044
## V7  0.9756717 0.8106956
## V8  0.4303034 1.0994138
## V9  0.6979716 0.6422689
## 11         NA 0.6518883
```

Boosted Trees

```
#install.packages("gbm")
library(gbm)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

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

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

```
set.seed(100)

gbmModel1 <- gbm(y ~ ., data = simulated, distribution = "gaussian", n.trees = 100)
gbmimp1 <- varImp(gbmModel1, numTrees = 100)

gbmModel2 <- gbm(y ~ ., data = simulated2, distribution = "gaussian", n.trees = 100)
gbmimp2 <- varImp(gbmModel2, numTrees = 100)
```

```
gbmimp1 <- rbind(gbmimp1, NA)
cbind(gbmimp1, gbmimp2)
```

```
##       Overall    Overall
## V1  42254.18 25860.3476
## V2  16505.95 19453.3830
## V3      0.00     0.0000
## V4  14225.63 13699.5838
## V5      0.00   811.2224
## V6      0.00     0.0000
## V7      0.00     0.0000
## V8      0.00     0.0000
## V9      0.00     0.0000
## V10     0.00     0.0000
## 11        NA 13398.7897
```

Cubist

```
#install.packages("Cubist")
library(Cubist)

set.seed(100)
cubist1 <- cubist(simulated[,1:10], simulated$y, committees = 100)
cubimp1 <- varImp(cubist1)

cubist2 <- cubist(simulated2[,c(1:10,12)], simulated$y, committees = 100)
cubimp2 <- varImp(cubist2)
```

```
cubimp1 <- rbind(cubimp1, NA)
cbind(cubimp1, cubimp2)
```

```
##     Overall Overall
## V1     71.5    70.5
## V3     47.0    44.5
## V2     58.5    56.5
## V4     48.0    48.0
## V5     33.0    33.0
## V6     13.0     2.5
## V7      0.0     1.5
## V8      0.0     0.0
## V9      0.0     0.0
## V10     0.0     0.0
## 11       NA     0.0
```

8.2. Use a simulation to show tree bias with different granularities.

Approaches: Trees suffer from selection bias: predictors with a higher number of distinct values are favored over more granular predictors. According to what Loh and Shih said: “The danger occurs when a data set consists of a mix of informative and noise variables, and the noise variables have many more splits than the informative variables. Then there is a high probability that the noise variables will be chosen to split the top nodes of the tree. Pruning will produce either a tree with misleading structure or no tree at all.” In addition, the more missing values, the more biased the selection of predictors.

Interpretaion: The informative predictor in the following code has lower variance than than noise predictor. In addition, informative predictor also has higher correlation with the response predictor than the noise predictor. However, the noise predictor has higher importance than informative predictor.

```
library(rpart)
set.seed(100)

informative <- rep(1:5, 20)
noise <- rnorm(100, mean = 0, sd = 10)
y <- informative + rnorm(100, mean = 0, sd = 5)
df1 <- data.frame(informative, noise, y)

df2 = data.frame(SD = c(sd(informative), sd(noise)), Correlation = c(cor(informative, y), cor(no
ise, y)))
row.names(df2) <- c("Informative Predictors", "Noise Predictors")
df2
```

```
##                               SD Correlation
## Informative Predictors  1.421338   0.4851920
## Noise Predictors       10.207104  -0.1512738
```

```
rpartTree <- rpart(y ~ ., data = df1)
varImp(rpartTree)
```

```
##               Overall
## informative 0.5559885
## noise       0.7086826
```

8.3. In stochastic gradient boosting the bagging fraction and learning rate will govern the construction of the trees as they are guided by the gradient. Although the optimal values of these parameters should be obtained through the tuning process, it is helpful to understand how the magnitudes of these parameters affect magnitudes of variable importance. Figure 8.24 provides the variable importance plots for boosting using two extreme values for the bagging fraction (0.1 and 0.9) and the learning rate (0.1 and 0.9) for the solubility data. The left-hand plot has both parameters set to 0.1, and the right-hand plot has both set to 0.9:

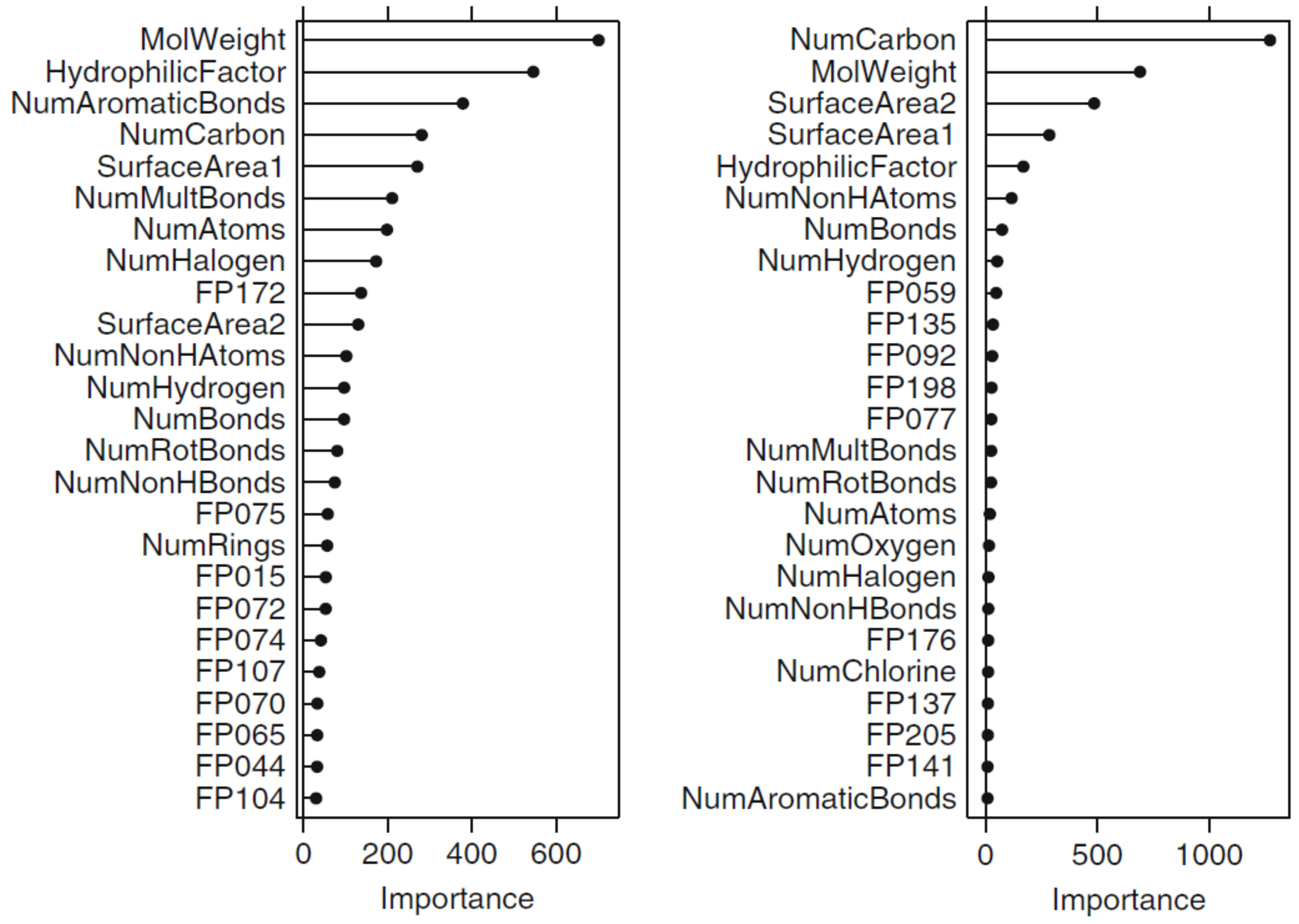


Figure 1

a. Why does the model on the right focus its importance on just the first few of predictors, whereas the model on the left spreads importance across more predictors?

According to “gbm” package documentation, shrinkage is a “parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction.” Therefore, the higher the shrinkage value, higher the learning rate, fewer predictors that it focuses on.

bag.fraction is the “fraction of the training set observations randomly selected to propose the next tree in the expansion. This introduces randomnesses into the model fit. If bag.fraction<1 then running the same model twice will result in similar but different fits. gbm uses the R random number generator so set.seed can ensure that the model can be reconstructed. Preferably, the user can save the returned gbm.object using save.” Higher bag.fraction values less randomnesses that the model creates, therefore, fewer predictos that it included in its model.

b. Which model do you think would be more predictive of other samples?

The model on the left would be more predictive of other samples, because it focuses on more predictors than the other model. The model that focuses on less predictors is greedier to run the model faster, however, it ignores the importance of certain predictors.

c. How would increasing interaction depth affect the slope of predictor importance for either model in Fig. 8.24?

interaction.depth is the maximum depth of variable interactions. 1 implies an additive model, 2 implies a model with up to 2-way interactions, etc. The higher interaction.depth value, the more spread out the importance among predictors. Therefore, the smaller slope of predictor importance figure.

8.7. Refer to Exercises 6.3 and 7.5 which describe a chemical manufacturing process. Use the same data imputation, data splitting, and pre-processing steps as before and train several tree-based models:

a. Which tree-based regression model gives the optimal resampling and test set performance?

```
library(mice)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(forecast)
library(e1071)
library(caret)
#install.packages("RWeka")
library(RWeka)
library(AppliedPredictiveModeling)
library(pls)
```

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

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

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

```
#data(package="AppliedPredictiveModeling")

data(ChemicalManufacturingProcess)

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

result <- C_M_P %>%
  select(everything()) %>%
  summarize_all(funs(sum(is.na(.))))

data.frame(sort(result, decreasing = TRUE))
```

```
##   Yield BiologicalMaterial01 BiologicalMaterial02 BiologicalMaterial03
## 1     0                    0                    0                    0
##   BiologicalMaterial04 BiologicalMaterial05 BiologicalMaterial06
## 1                    0                    0                    0
##   BiologicalMaterial07 BiologicalMaterial08 BiologicalMaterial09
## 1                    0                    0                    0
##   BiologicalMaterial10 BiologicalMaterial11 BiologicalMaterial12
## 1                    0                    0                    0
##   ManufacturingProcess01 ManufacturingProcess02 ManufacturingProcess03
## 1                      0                      0                      0
##   ManufacturingProcess04 ManufacturingProcess05 ManufacturingProcess06
## 1                      0                      0                      0
##   ManufacturingProcess07 ManufacturingProcess08 ManufacturingProcess09
## 1                      0                      0                      0
##   ManufacturingProcess10 ManufacturingProcess11 ManufacturingProcess12
## 1                      0                      0                      0
##   ManufacturingProcess13 ManufacturingProcess14 ManufacturingProcess15
## 1                      0                      0                      0
##   ManufacturingProcess16 ManufacturingProcess17 ManufacturingProcess18
## 1                      0                      0                      0
##   ManufacturingProcess19 ManufacturingProcess20 ManufacturingProcess21
## 1                      0                      0                      0
##   ManufacturingProcess22 ManufacturingProcess23 ManufacturingProcess24
## 1                      0                      0                      0
##   ManufacturingProcess25 ManufacturingProcess26 ManufacturingProcess27
## 1                      0                      0                      0
##   ManufacturingProcess28 ManufacturingProcess29 ManufacturingProcess30
## 1                      0                      0                      0
##   ManufacturingProcess31 ManufacturingProcess32 ManufacturingProcess33
## 1                      0                      0                      0
##   ManufacturingProcess34 ManufacturingProcess35 ManufacturingProcess36
## 1                      0                      0                      0
##   ManufacturingProcess37 ManufacturingProcess38 ManufacturingProcess39
## 1                      0                      0                      0
##   ManufacturingProcess40 ManufacturingProcess41 ManufacturingProcess42
## 1                      0                      0                      0
##   ManufacturingProcess43 ManufacturingProcess44 ManufacturingProcess45
## 1                      0                      0                      0
```

```
CMP_trans <- preProcess(C_M_P, method = c("nzv", "BoxCox", "center", "scale"))
transformed <- predict(CMP_trans, C_M_P)
```

```
set.seed(88)
trainingRows <- createDataPartition(transformed$Yield, p = .80, list= FALSE)

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

yield_test <- transformed[-trainingRows, 1]
predictor_test <- transformed[-trainingRows, -1]
```

```
ctrl <- trainControl(method = "cv", number = 10)

rpartTune  <- train(x = predictor_train, y = yield_train, method = "rpart2", tuneLength = 10,trC
ontrol = ctrl)
m5Tune <- train(x = predictor_train, y = yield_train, method = "M5", trControl = ctrl, control =
 Weka_control(M = 10))
btTune  <- train(x = predictor_train, y = yield_train, method = "treebag", trControl = ctrl)
rfTune  <- train(x = predictor_train, y = yield_train, method = "rf", trControl = ctrl, importan
ce=T)
cubistTune <- train(x = predictor_train, y = yield_train,method = "cubist")

#gbmGrid <- expand.grid(interaction.depth = seq(1, 7, by = 2), n.trees = seq(100, 1000, by = 5
0), shrinkage = c(0.01, 0.1), n.minobsinnode = 10)
#gbmTune2 <- train(x = predictor_train, y = yield_train, method = "gbm", tuneGrid = gbmGrid, ver
bose = FALSE)
```

```
rpart_RMSE <- min(rpartTune$results$RMSE)
m5_RMSE <- min(m5Tune$results$RMSE)
bt_RMSE <- min(btTune$results$RMSE)
rf_RMSE <- min(rfTune$results$RMSE)
cubist_RMSE <- min(cubistTune$results$RMSE)

result <- data.frame(Model = c("Single Trees", "Model Trees", "Bagged Trees", "Random Forest",
"Cubist"), RMSE = c(rpart_RMSE, m5_RMSE, bt_RMSE, rf_RMSE, cubist_RMSE))

result
```

```
##           Model      RMSE
## 1  Single Trees 0.7463874
## 2   Model Trees 0.6259828
## 3  Bagged Trees 0.6436255
## 4 Random Forest 0.5992231
## 5        Cubist 0.5798986
```

Interpretation: Random Forest model gives the optimal resampling and test set performance since it generate lowest RMSE value.

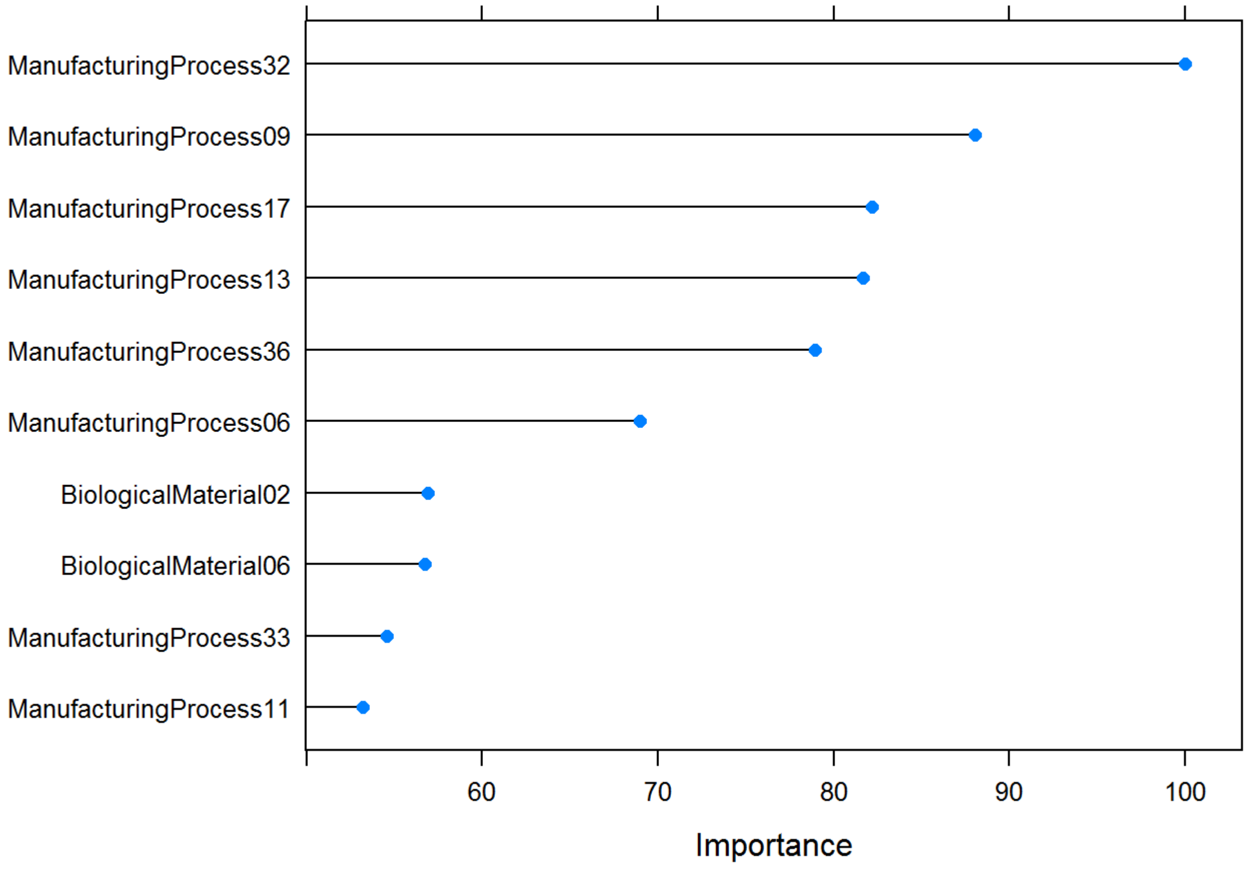
b. Which predictors are most important in the optimal tree-based regression model? Do either the biological or process variables dominate the list? How do the top 10 important predictors compare to the top 10 predictors from the optimal linear and nonlinear models?

Interpretation: ManufacturingProcess32 is the most important in the optimal tree-based regression model. Manufacturing processes is apparently dominate the list, because they make up 12 out of the top 20 most important predictors. In addition, ManufacturingProcess32 alone include around 60% overall in terms of weight. FOr linear models, top 5 out of top 10 are all from manufacturing process, on the other hand, nonlinear model gives a little more weight to biological material variables. For tree-based regression model, even more weight is given to biological material variables. However, the slop becomes steeper since big portion of the importance is given to only one variable ManufacturingProcess32.

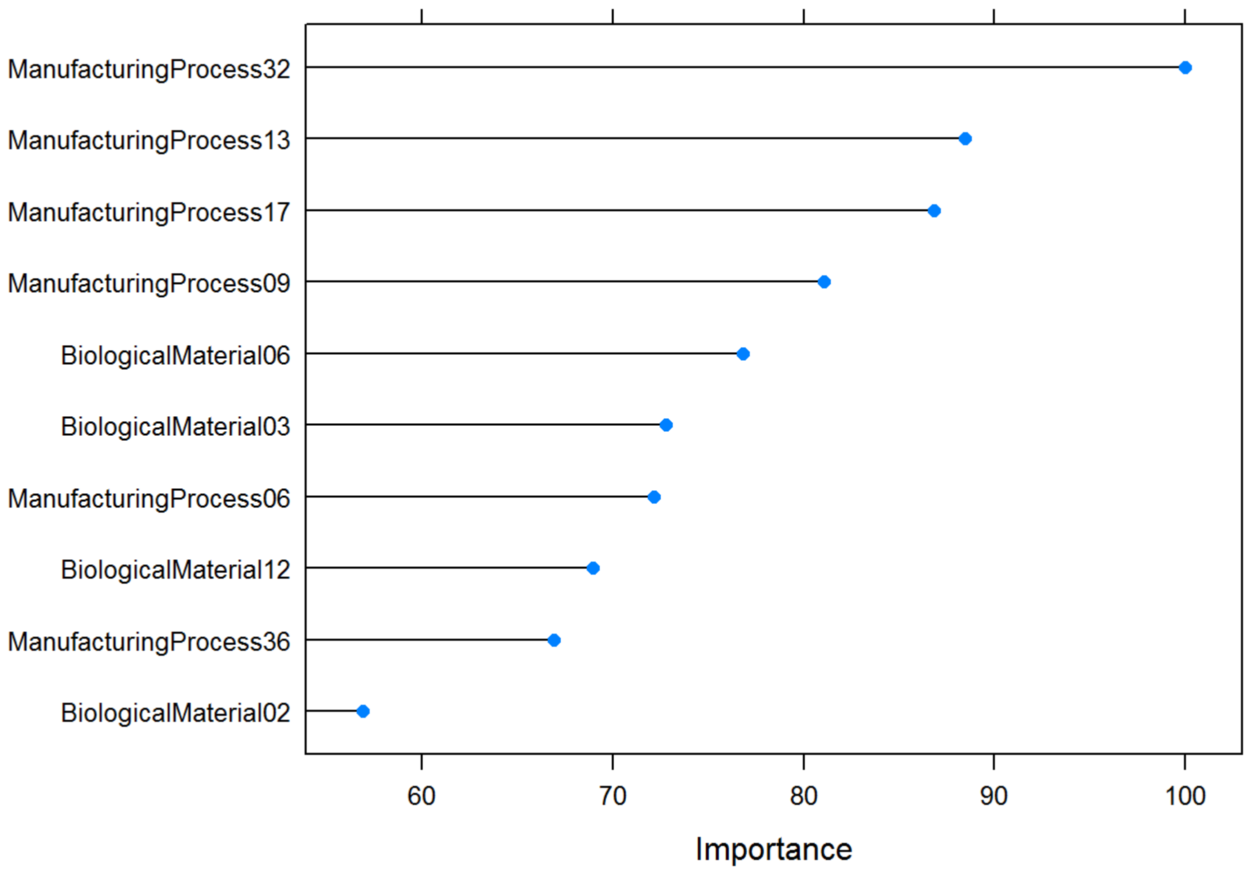
```
rpartimp3 <- varImp(rpartTune)
m5imp3 <- varImp(m5Tune)
btimp3 <- varImp(btTune)
rfimp3 <- varImp(rfTune)
cunistimp3 <- varImp(cubistTune)

rfimp3
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 56)
##
##                         Overall
## ManufacturingProcess32  100.00
## BiologicalMaterial12     37.09
## ManufacturingProcess17   36.89
## BiologicalMaterial03     36.39
## ManufacturingProcess31   33.75
## ManufacturingProcess13   31.78
## BiologicalMaterial06     31.75
## BiologicalMaterial05     29.73
## BiologicalMaterial11     28.40
## ManufacturingProcess06   28.17
## ManufacturingProcess09   27.58
## ManufacturingProcess11   26.75
## ManufacturingProcess36   24.03
## BiologicalMaterial01     23.19
## ManufacturingProcess18   22.59
## BiologicalMaterial08     22.27
## BiologicalMaterial02     21.95
## ManufacturingProcess25   21.68
## ManufacturingProcess39   21.20
## ManufacturingProcess01   20.81
```

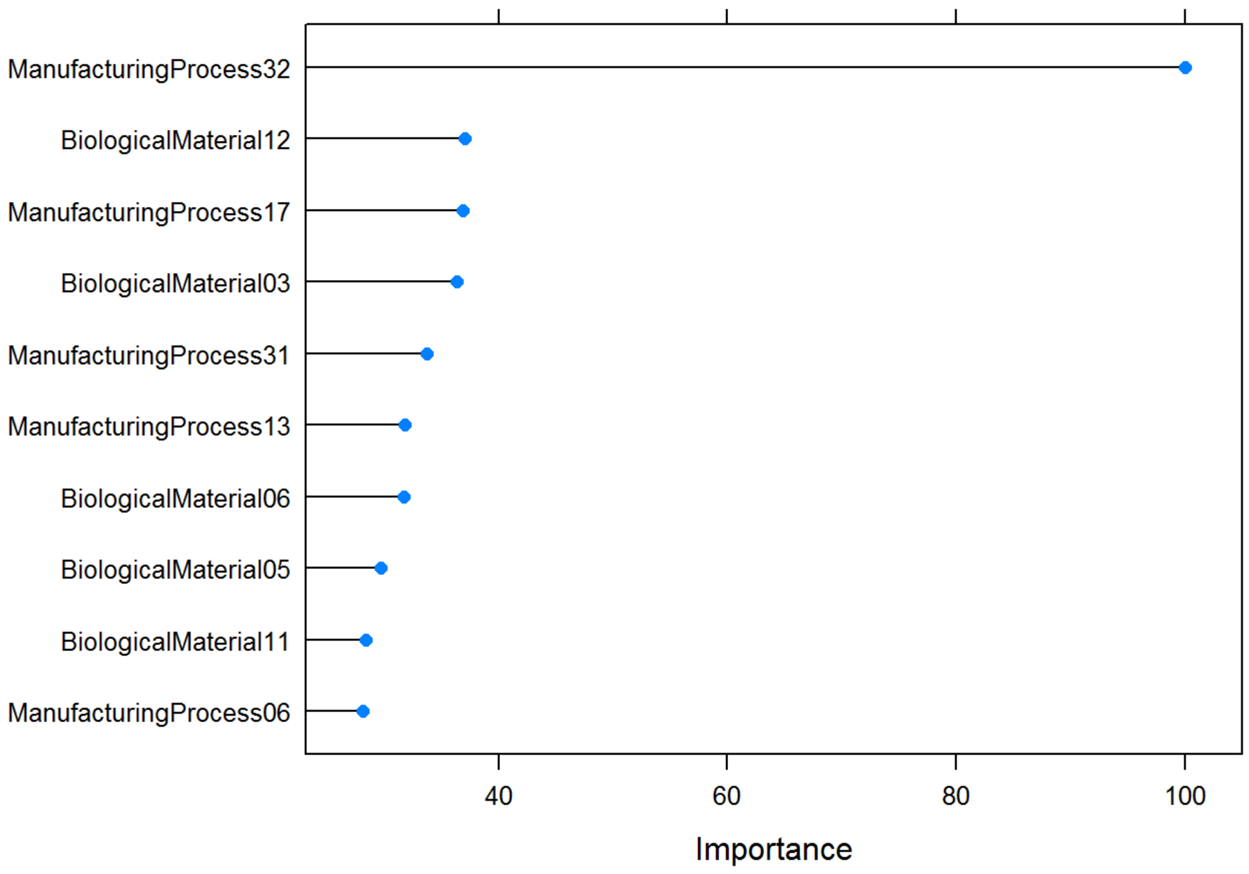
```
set.seed(88)
plsTune <- train(predictor_train, yield_train, method = "pls", tuneLength = 20, trControl = ctr
l)
svmRTune <- train(predictor_train, yield_train, method = "svmRadial", tuneLength = 20, trControl
 = ctrl)


par(mfrow = c(1, 3))
plot(varImp(plsTune), top = 10)
```



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



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



c. Plot the optimal single tree with the distribution of yield in the terminal nodes. Does this view of the data provide additional knowledge about the biological or process predictors and their relationship with yield?

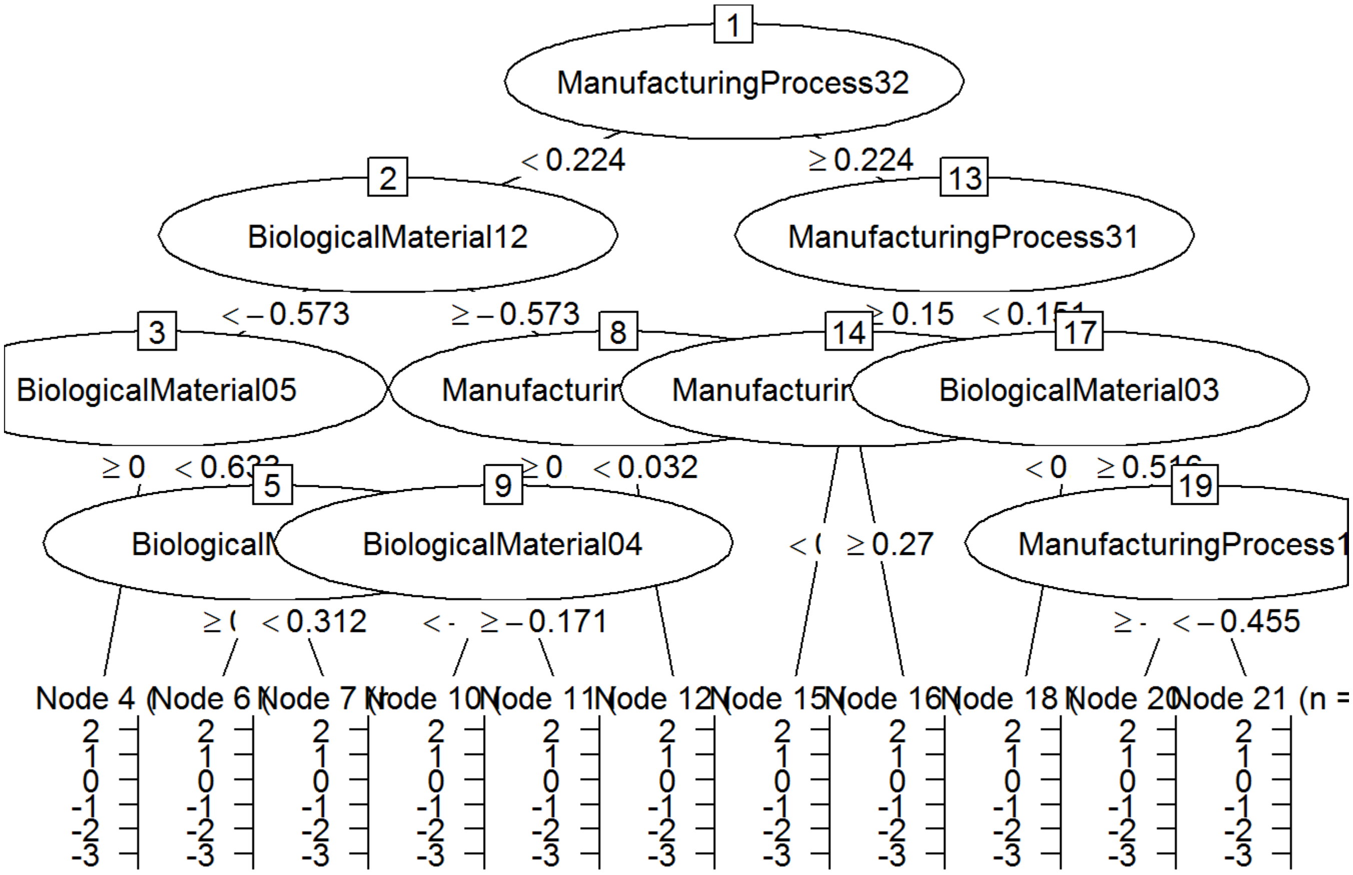
```
#install.packages("partykit")
library(partykit)
```

```
## Loading required package: libcoin
```

```
##
## Attaching package: 'partykit'
```

```
## The following objects are masked from 'package:party':
##
##     cforest, ctree, ctree_control, edge_simple, mob, mob_control,
##     node_barplot, node_bivplot, node_boxplot, node_inner,
##     node_surv, node_terminal, varimp
```

```
rpartTree <- as.party(rpartTune$finalModel)
plot(rpartTree)
```


1
ManufacturingProcess32
< 0.224
≥ 0.224
2
BiologicalMaterial12
13
ManufacturingProcess31
< −0.573
≥ −0.573
≥ 0.15
< 0.151
3
BiologicalMaterial05
8
14
17
BiologicalMaterial03
≥ 0
< 0.032
< 0
≥ 0.27
5
9
BiologicalMaterial04
19
< 0.312
≥ −0.171
< −0.455
≥ 0.27
Node 4
Node 6
Node 7
Node 10
Node 11
Node 12
Node 15
Node 16
Node 18
Node 20
Node 21 (n =