DATA 624: PREDICTIVE ANALYTICS HW 9

Gabriel Campos

Last edited April 14, 2024

Library

```r
library(AppliedPredictiveModeling)
library(caret)
library(corrplot)
library(Cubist)
library(dplyr)
library(earth)
library(forecast)
library(fpp3)
library(gbm)
library(ggplot2)
library(MASS)
library(mice)
library(mlbench)
library(party)
library(randomForest)
library(RANN)
library(rpart)
library(rpart.plot)
library(tidyr)
```

Description

Do problems 8.1, 8.2, 8.3, and 8.7 in Kuhn and Johnson. Please submit the Rpubs link along with the .rmd file.

8.1.

Recreate the simulated data from Exercise 7.2:

```r
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:

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

```
rownames(rfImp1) <- paste0("V", 1:10)
rfImp1_sorted <- rfImp1[order(-rfImp1$Overall), , drop = FALSE] # 'drop = FALSE' ensures the result is
print(rfImp1_sorted)
```

```
##          Overall
## V1   8.732235404
## V4   7.615118809
## V2   6.415369387
## V5   2.023524577
## V3   0.763591825
## V6   0.165111172
## V7  -0.005961659
## V10 -0.074944788
## V9  -0.095292651
## V8  -0.166362581
```

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

No. The values of `V6-V10` are very low, even most being a negative value indicating a low importance score.

(b)

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

```
simulated$duplicate1 <- simulated$V1 + rnorm(200) * .1
cor(simulated$duplicate1, simulated$V1)
```

```
## [1] 0.9460206
```

```
simulated2<-simulated
simulated2$duplicate1 <- simulated2$V1 + rnorm(200) * .1
cor(simulated2$duplicate1, simulated2$V1)
```

```
## [1] 0.9447637
```

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`?

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

```
rownames(rfImp2) <- c(paste0("V", 1:10), "duplicate1")
rfImp2_sorted <- rfImp2[order(-rfImp2$Overall), , drop = FALSE] # 'drop = FALSE' ensures the result is
print(rfImp2_sorted)
```

```
##                  Overall
## V4            6.92191640
## V1            6.51537580
## V2            6.28207942
## duplicate1    3.45558605
## V5            2.05234722
## V3            0.55644584
## V6            0.11516713
## V10          -0.03254994
## V7           -0.06291275
## V9           -0.09013857
## V8           -0.14139209
```

The highly correlated rearranged the importance score for all predictors. Predictor V1 was demoted from most important to second most important. The highly correlated duplicate lands in fourth place.

(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?

```
model3 <- cforest(y ~., data = simulated)

order(-varimp(model3, conditional = FALSE)) #default conditional: FALSE
```

```
##  [1]  4  2 11  1  5  7  9 10  6  3  8
```

```
order(-varimp(model3, conditional = TRUE))
```

```
##  [1]  4  2 11  1  5  3  9  6  7 10  8
```

Using the party package on varimp

cforest conditional=false is 4 2 11 1 5 3 7 9 6 8 10 cforest conditional=true is 4 2 1 11 5 6 3 7 8 9 10 while randomforest is 1 4 2 5 3 6 7 10 9 8

which differ in pattern, but do support that v6-v10 have relatively low importance levels.

(d)

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

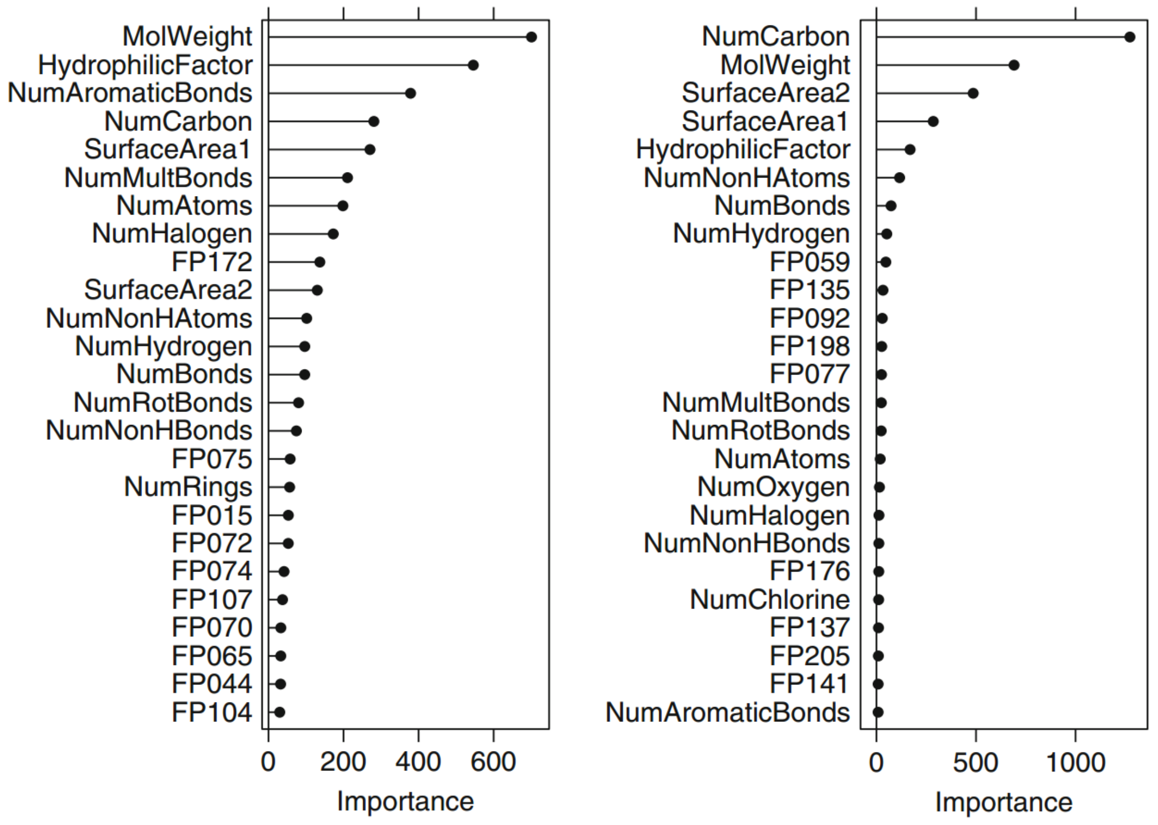


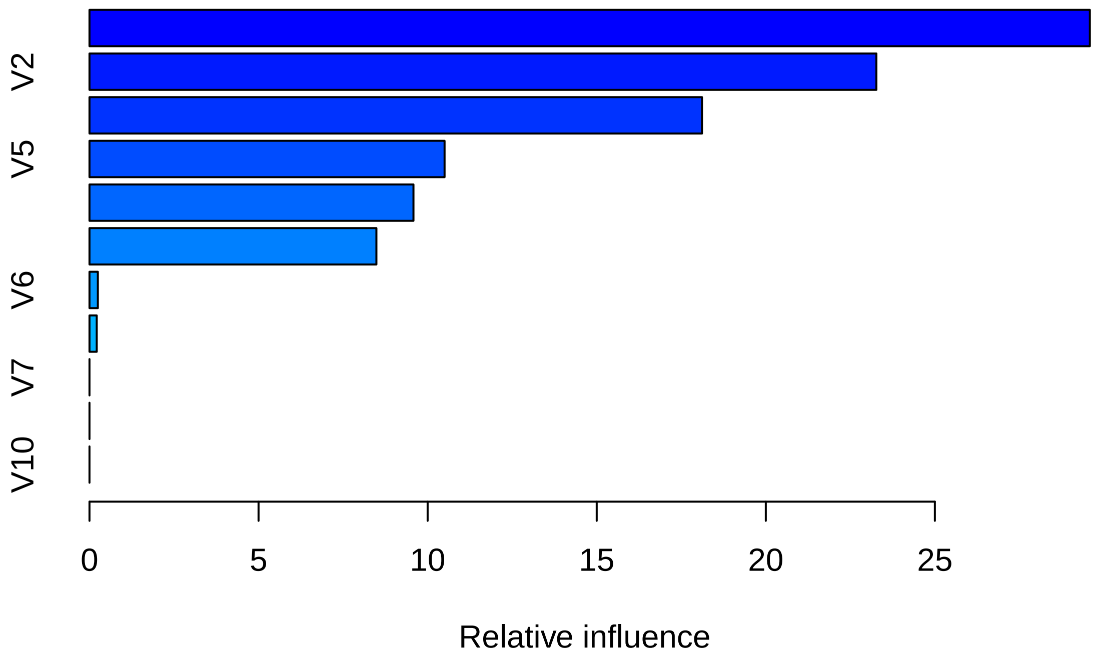
Fig. 8.24: A comparison of variable importance magnitudes for differing values of the bagging fraction and shrinkage parameters. Both tuning parameters are set to 0.1 in the *left* figure. Both are set to 0.9 in the *right* figure

Boosted

```
set.seed(624)

model_gbm<- gbm(y ~., data = simulated, distribution = "gaussian")

summary.gbm(model_gbm)
```



```
##                   var    rel.inf
## V4                 V4 29.5838443
## V2                 V2 23.2710334
## V1                 V1 18.1143091
## V5                 V5 10.5014583
## duplicate1 duplicate1  9.5810923
## V3                 V3  8.4854137
## V6                 V6  0.2478833
## V9                 V9  0.2149655
## V7                 V7  0.0000000
## V8                 V8  0.0000000
## V10               V10  0.0000000
```

Cubist

```
simulated_x <- subset(simulated, select = -c(y))

cubist_model <- cubist(x = simulated_x, y = simulated$y, committees = 100)
```

```
cube_model_v2<-varImp(cubist_model)
rownames(cube_model_v2) <- c(paste0("V", 1:10), "duplicate1")
cubist_sorted <- cube_model_v2[order(-cube_model_v2$Overall), , drop = FALSE]
print(cubist_sorted)
```

```
##              Overall
## V3              59.5
## V2              52.5
## V5              46.0
## V1              43.5
## V4              27.5
## V7              27.0
## V8              10.0
## V6               4.0
## V9               1.0
## V10              0.0
## duplicate1       0.0
```

Cubist model has V3 as the most important predictor, and the Bagged Trees has V4 as the most important predictor. Predictors V6 – V10 remain uninformative.

8.2.

Use a simulation to show tree bias with different granularities.

```r
set.seed(624)
#samples for predictors
low <- sample(0:50, 500, replace = T)
medium <- sample(0:500, 500, replace = T)
high <- sample(0:5000, 500, replace = T)

#response
y <- low + medium + high + rnorm(250)

#check variance of predictors
var(low)
```

```
## [1] 221.9536
```

```r
var(medium)
```

```
## [1] 21752.82
```

```r
var(high)
```

```
## [1] 2016913
```

```r
df_sim <- data.frame(low, medium, high, y)

diff_gran_model <- randomForest(y ~., data = df_sim, importance = TRUE, ntree = 1000)

varImp(diff_gran_model, scale=FALSE)
```

```
##             Overall
## low       -19996.77
## medium     15806.52
## high     3605673.96
```

Sample data made using `sample` has `low` variable with the highest granularity. This is confirmed by it's variance of 0.71. Variables `medium` and `high` have medium and high variance, when compared to `low`. Higher variance indicates lower granularity.

The tree model confirms tree bias where highest variance variables (lowest granularity) get ranked with highest importance.

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:

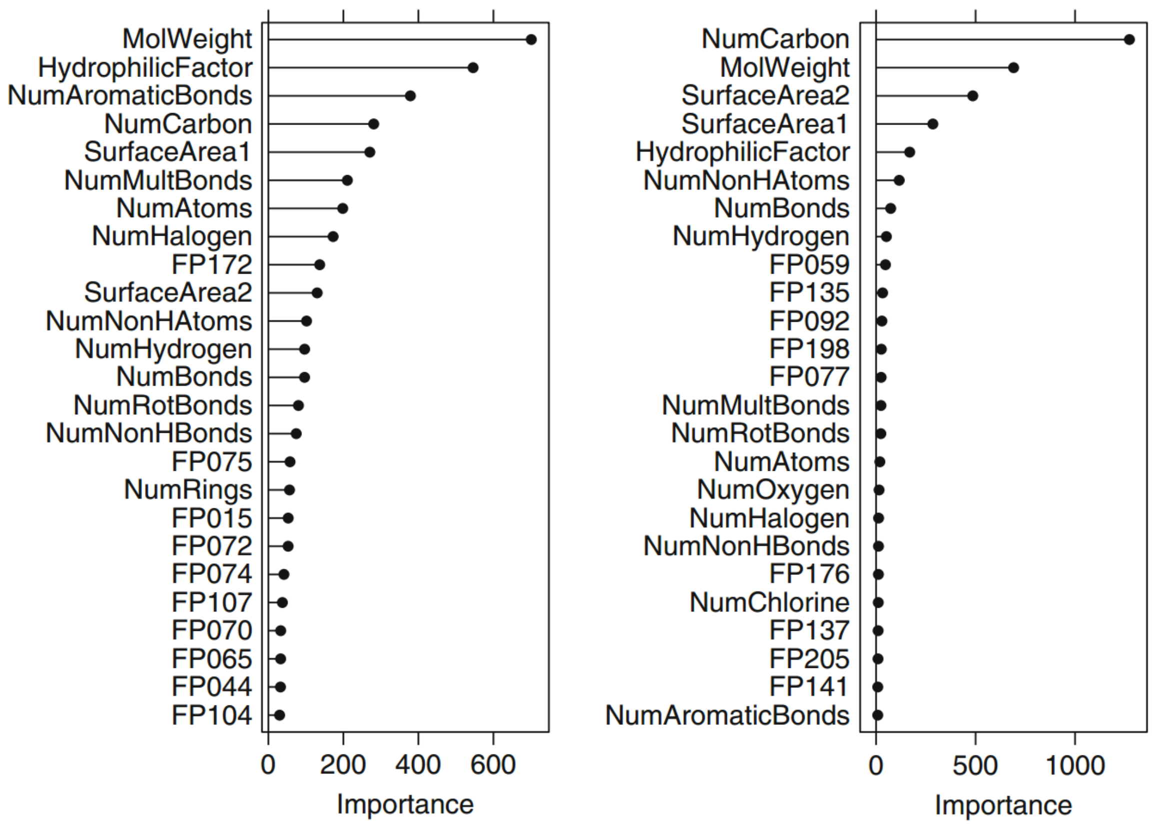


Fig. 8.24: A comparison of variable importance magnitudes for differing values of the bagging fraction and shrinkage parameters. Both tuning parameters are set to 0.1 in the *left* figure. Both are set to 0.9 in the *right* figure

package gbm

(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?

Left Model

- The left model with `shrinkage` = 0.1 & `bag.fraction` = 0.1 focuses on importance for just the first few predictors because the training set observations fraction is very small(0.1 or 10%).
- Small portion data used to create the trees, means there's a focus to it's importance only on a small amount of predictors.
- `shrinkage` parameter or learning rate is set to the highest recommended value of what "usually works" according to documentation for the `gbm` package. The documentation states "smaller learning rate typically require(s) more trees" meaning that smaller values lead to more trees and higher values lead to less trees.

Right Model

- The right model with `shrinkage` = 0.9, `bag.fraction` = 0.9 has a greater spread importance across the predictors because it uses .9 or 90% of the training set to create its trees.
- Larger `shrinkage` parameter suggests that less trees are created in this model.

(b)

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

The left model with `shrinkage` = 0.1, `bag.fraction` = 0.1 would be more predictive of other samples between the two. Considering the amount of trees it should have in comparison to the model on the right with `shrinkage` = 0.9, `bag.fraction` = 0.9.

(c)

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

According to pub_journals link Sec 4.2, pg. 15

- The relationship between shrinkage (learning rate) and interaction depth (tree complexity) in decision tree models, specifically in boosting.
- Shrinkage is inversely related to interaction depth, meaning as tree complexity increases, the learning rate should decrease to prevent overfitting and maintain model stability.
- This slower learning process necessitates using more trees.
- Larger datasets better support more complex trees, as they can handle the detailed structures these trees create without overfitting quickly. *Interaction depth/tree size or tree complexity, affects the maximum size for each tree.
- As mentioned there is a inverse relationship with shrinkage to interaction depth.
- The left (`shrinkage` = 0.1, `bag.fraction` = 0.1) is likely to benefit from increasing interaction depth.
- But the right might not since the shrinkage parameter is already so high.

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:

```
#load data
data(ChemicalManufacturingProcess)

set.seed(624)
imputed_data <- preProcess(ChemicalManufacturingProcess, method = c("knnImpute","nzv", "corr"))
df_full <- predict(imputed_data, ChemicalManufacturingProcess)

chem_index <- createDataPartition(df_full$Yield , p=.8, list=F)

chem_train <-  df_full[chem_index,]
chem_test <- df_full[-chem_index,]

train_pred <- chem_train[-c(1)]
test_pred<-  chem_test[-c(1)]
```

(a)

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

Boosted Trees

```
set.seed(624)
gbm_model <- gbm.fit(train_pred, chem_train$Yield, distribution = "gaussian")
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.9239             nan     0.0010    0.0004
##      2        0.9231             nan     0.0010    0.0006
##      3        0.9224             nan     0.0010    0.0007
##      4        0.9217             nan     0.0010    0.0006
##      5        0.9211             nan     0.0010    0.0006
##      6        0.9203             nan     0.0010    0.0006
##      7        0.9196             nan     0.0010    0.0007
##      8        0.9190             nan     0.0010    0.0006
##      9        0.9183             nan     0.0010    0.0007
##     10        0.9175             nan     0.0010    0.0005
##     20        0.9111             nan     0.0010    0.0007
##     40        0.8980             nan     0.0010    0.0006
##     60        0.8865             nan     0.0010    0.0004
##     80        0.8743             nan     0.0010    0.0005
##    100        0.8628             nan     0.0010    0.0002
```

```
gbm_model
```

```
## A gradient boosted model with gaussian loss function.
## 100 iterations were performed.
## There were 46 predictors of which 7 had non-zero influence.
```

```
gbm_pred <- predict(gbm_model, newdata = test_pred)
postResample(pred = gbm_pred, obs = chem_test$Yield)
```

```
##      RMSE  Rsquared       MAE
## 1.1097582 0.5627084 0.8634997
```

Cubist

```
set.seed(624)
cube_model <- cubist(train_pred, chem_train$Yield)
cube_model
```

```
##
## Call:
## cubist.default(x = train_pred, y = chem_train$Yield)
##
## Number of samples: 144
## Number of predictors: 46
##
## Number of committees: 1
## Number of rules: 2
```

```
cube_pred <- predict(cube_model, newdata = test_pred)
postResample(pred = cube_pred, obs = chem_test$Yield)
```

```
##      RMSE  Rsquared       MAE
## 0.6724398 0.6903309 0.5159054
```

Random Forest

```
set.seed(624)
#fit the model
rf_model <- randomForest(train_pred, chem_train$Yield, importance = TRUE, ntrees = 1000)
rf_model
```

```
##
## Call:
##  randomForest(x = train_pred, y = chem_train$Yield, importance = TRUE,      ntrees = 1000)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 15
##
##           Mean of squared residuals: 0.3711346
##                     % Var explained: 59.85
```

```
rf_pred <- predict(rf_model, newdata = test_pred)
postResample(pred = rf_pred, obs = chem_test$Yield)
```

```
##      RMSE  Rsquared       MAE
## 0.6715320 0.7323224 0.4828902
```

Random Forest has the optimal metrics because of its RMSE of 0.6715320 (lowest) and Rsquared of 0.7323224

(b)

Top 10:
ManufacturingProcess32, BiologicalMaterial06, BiologicalMaterial03, ManufacturingProcess13, ManufacturingProcess36, BiologicalMaterial11, ManufacturingProcess09, BiologicalMaterial08, ManufacturingProcess28, ManufacturingProcess11

```
rf_model_v2<-varImp(rf_model)
```

```
rf_model_sorted <- rf_model_v2[order(-rf_model_v2$Overall), , drop = FALSE] # 'drop = FALSE' ensures th
head(rf_model_sorted,10)
```

```
##                          Overall
## ManufacturingProcess32 21.996175
## BiologicalMaterial06   12.943335
## BiologicalMaterial03   11.767253
## ManufacturingProcess36  9.989997
## BiologicalMaterial11    9.396847
## ManufacturingProcess13  9.195379
## BiologicalMaterial08    8.546350
## ManufacturingProcess09  7.580592
## BiologicalMaterial09    7.019902
## ManufacturingProcess28  6.992895
```

i

Which predictors are most important in the optimal tree-based regression model? Do either the biological or process variables dominate the list?

`ManufacturingProcess32` is the most important.Neither of the predictors dominating the list with 5 `ManufacturingProcess` and 5 `BiologicalProcess` in the top ten.

ii

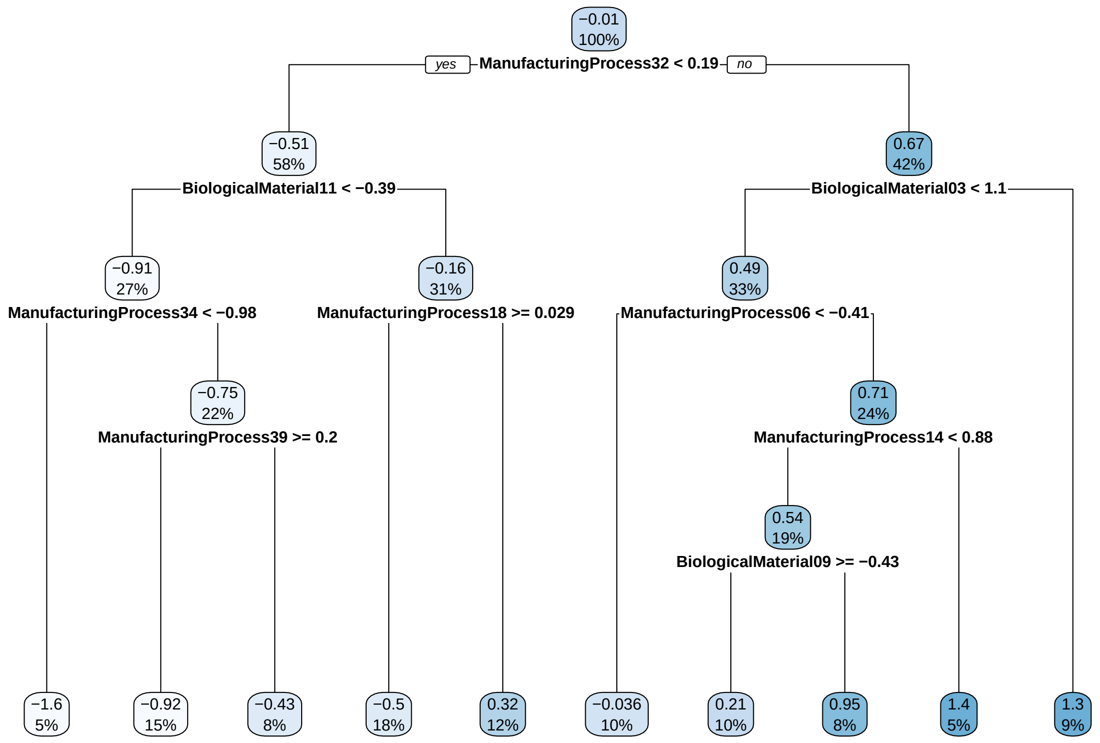
How do the top 10 important predictors compare to the top 10 predictors from the optimal linear and nonlinear models?

This combination is basically consistent with the importance variables of the non-linear model, not so much the linear model.

(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?

```r
rpart_tree <- rpart(Yield ~., data = chem_train)
rpart.plot(rpart_tree)
```


−0.01
100%
yes
ManufacturingProcess32 < 0.19
no
−0.51
58%
BiologicalMaterial11 < −0.39
0.67
42%
BiologicalMaterial03 < 1.1
−0.91
27%
ManufacturingProcess34 < −0.98
−0.16
31%
ManufacturingProcess18 >= 0.029
0.49
33%
ManufacturingProcess06 < −0.41
−0.75
22%
ManufacturingProcess39 >= 0.2
0.71
24%
ManufacturingProcess14 < 0.88
0.54
19%
BiologicalMaterial09 >= −0.43
−1.6
5%
−0.92
15%
−0.43
8%
−0.5
18%
0.32
12%
−0.036
10%
0.21
10%
0.95
8%
1.4
5%
1.3
9%