DATA624: Homework 9

Donald Butler

2023-11-19

Contents

Homework 9

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(AppliedPredictiveModeling)
library(mlbench)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
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
##
## Attaching package: 'strucchange'
##
## The following object is masked from 'package:stringr':
##
##     boundary
##
##
## Attaching package: 'party'
##
## The following object is masked from 'package:dplyr':
##
##     where
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

```r
library(Cubist)
library(rpart)
library(rpart.plot)
library(ipred)
```

Exercise 8.1

Recreate the simulated data from Exercise 7.2.

```
set.seed(31415)
simulated <- mlbench.friedman1(200, sd = 1)
simulated <- cbind(simulated$x, simulated$y) |> as.data.frame()
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))
```

```
##           Overall
## V1   5.4536774423
## V2   9.0924929892
## V3   1.3248082787
## V4  10.0796229224
## V5   1.3879158867
## V6  -0.1140931718
## V7   0.0733821187
## V8   0.0003603532
## V9  -0.0781190844
## V10 -0.1522243846
```

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

The importance of predictors `V6` - `V10` are all near zero which indicates that have little impact on the model.

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

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

```
## [1] 0.9433099
```

Fit another random forest model to these data. Did the importance score for `V1` change?

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

```
##                Overall
## V1          3.92252934
## V2          8.98010706
## V3          1.35006038
## V4          9.35757388
## V5          1.38046302
## V6          0.03609073
## V7          0.07539182
## V8          0.01830255
## V9         -0.01788776
## V10        -0.07082455
## duplicate1  2.38006415
```

The importance of `V1` decreased in the new model, but by less than `duplicate1`.

What happens when you add another predictor that is also highly correlated with `V1`?

```
simulated$duplicate2 <- simulated$V1 + rnorm(200) * .1
cor(simulated$duplicate2, simulated$V1)
```

```
## [1] 0.9494923
```

```
model3 <- randomForest(y ~ ., data = simulated, importance = TRUE, ntree = 1000)
(rfImp3 <- varImp(model3, scale = FALSE))
```

```
##                 Overall
## V1           3.32224021
## V2           9.58385743
## V3           1.13905238
## V4           9.79602219
## V5           1.45263144
## V6          -0.04918987
## V7           0.01889473
## V8          -0.09244135
## V9          -0.06300478
## V10         -0.06589530
## duplicate1   1.71718908
## duplicate2   1.73013505
```

Introducing another highly correlated variable has further reduced the importance of `V1`.

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?

```
model4 <- cforest(y ~ ., data = simulated[, c(1:11)])
(rfImp4 <- party::varimp(model4, conditional = TRUE))
```

```
##           V1           V2           V3           V4           V5           V6 
##  3.984513934  7.579834713  0.247489900  8.391174310  1.036284655  0.001109265 
##           V7           V8           V9          V10 
##  0.002636829 -0.003564613 -0.034962904 -0.018025626
```

The predictors are ranked in the same order as the initial model.

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

Boosted Trees

```r
model5 <- gbm(y ~ ., data = simulated[,c(1:11)], distribution = 'gaussian')
(rfImp5 <- varImp(model5, numTrees = 50))
```

```
##       Overall
## V1  2199.9364
## V2  4279.9036
## V3   653.9787
## V4  4954.3246
## V5   742.7120
## V6     0.0000
## V7     0.0000
## V8     0.0000
## V9     0.0000
## V10    0.0000
```

Cubist

```r
model6 <- train(y ~ ., data = simulated[,c(1:11)], method = 'cubist')
rfImp6 <- varImp(model6, scale = FALSE)
rfImp6$importance
```

```
##     Overall
## V2     65.5
## V3     49.5
## V1     58.5
## V4     50.0
## V5     37.0
## V7     11.0
## V8      2.0
## V9      1.5
## V10     1.0
## V6      1.0
```

Both the random forest and boosted trees models rank the predictors in the same order, but the cubist model significantly different.

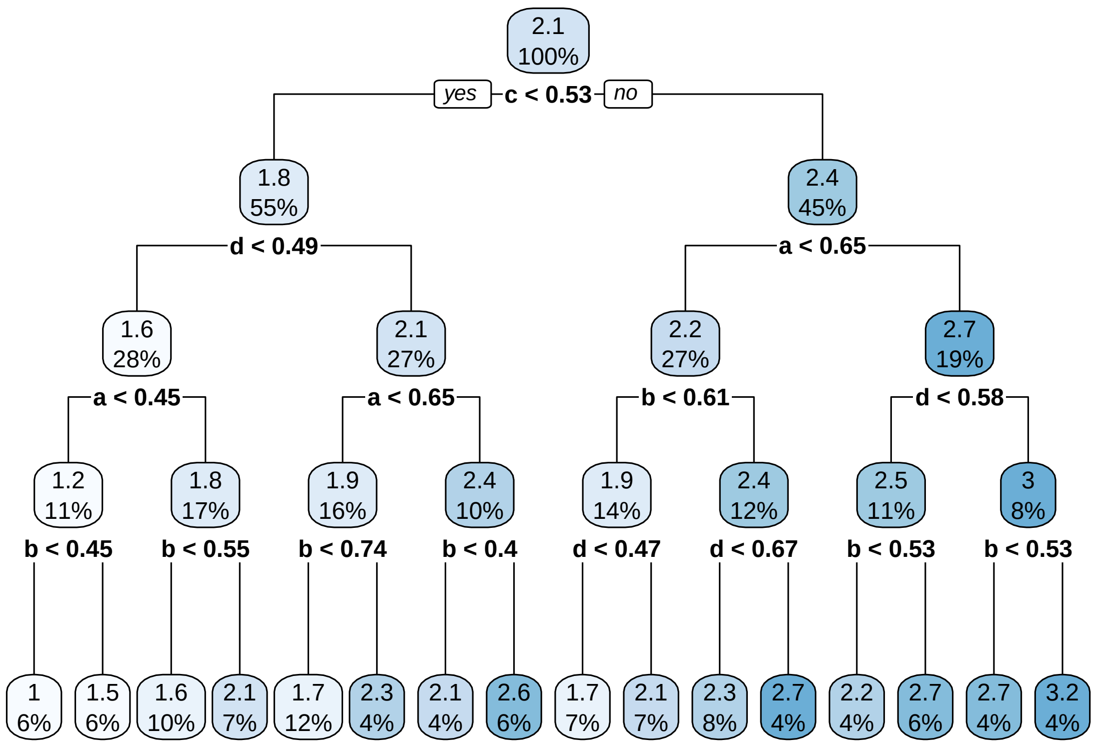
Exercise 8.2

Use a simulation to show tree bias with different granularities.

```r
a <- sample(1:10 / 10, 1000, replace = TRUE)
b <- sample(1:100 / 100, 1000, replace = TRUE)
c <- sample(1:1000 / 1000, 1000, replace = TRUE)
d <- sample(1:10000 / 10000, 1000, replace = TRUE)
y <- a + b + c + d
simulation = data.frame(a, b, c, d, y)
```

Created four random variables with increasing granularity. Predictor `a` has the lowest and `d` has the highest.

```r
model <- rpart(y ~ ., data = simulation)
rpart.plot(model)
```



```r
varImp(model)
```

```
##    Overall
## a 2.403314
## b 4.792188
## c 2.230621
## d 2.972393
```

With the four random samples a-d, should all have about the same importance.

Exercise 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:

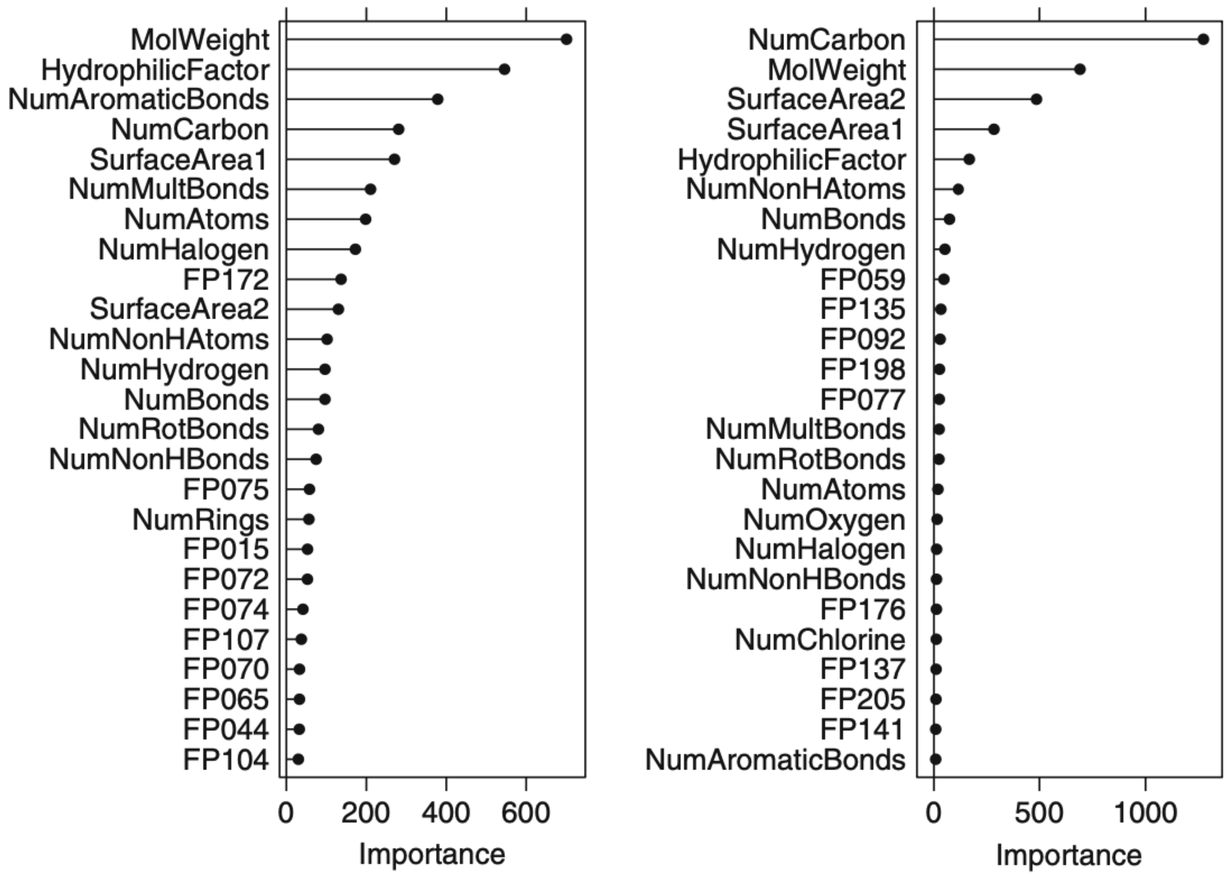


Fig. 8.24: A comparison of variable importance magnitudes for differing values of the bagging fraction and shrinkage parameters. Both tuning parameters are set to 0.1 in the *left* figure. Both are set to 0.9 in the *right* figure

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?

The bagging fraction is a percentage of data used to generate the tree. Shrinkage is the learning rate and is applied to each tree in the expansion. A higher learning rate leads to fewer iterations when generating the 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 of the lower learning rate, it doesn't focus on just a few predictors.

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

The interaction depth is the number of divisions to perform on the tree. With more splits, the importance of predictors increases, which allows lower importance predictors to contribute more.

Exercise 8.7

Refer to Exercise 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:

From Homework 7

```r
data(ChemicalManufacturingProcess)
imputed <- predict(preProcess(ChemicalManufacturingProcess, method = 'bagImpute'), ChemicalManufacturin

X <- imputed |>
  select(-Yield)
y <- imputed$Yield

X <- X[,-nearZeroVar(X)]

train <- createDataPartition(y, p = .8, list = FALSE)
X_train <- X[train,]
X_test <- X[-train,]
y_train <- y[train]
y_test <- y[-train]
```

Single Trees

```r
stModel <- train(X_train, y_train, method = 'rpart',
                 tuneLength = 10, trControl = trainControl(method = 'cv'))
stPred <- predict(stModel, newdata = X_test)
stResult <- data.frame(as.list(postResample(pred = stPred, obs = y_test))) |>
  mutate(model = 'Single Trees') |>
  relocate(model, RMSE, Rsquared, MAE)
stResult
```

```
##          model     RMSE  Rsquared      MAE
## 1 Single Trees 1.240939 0.5112617 1.031375
```

Bagged Trees

```r
btModel <- ipredbagg(y_train, X_train)
btPred <- predict(btModel, newdata = X_test)
btResult <- data.frame(as.list(postResample(pred = btPred, obs = y_test))) |>
  mutate(model = 'Bagged Trees') |>
  relocate(model, RMSE, Rsquared, MAE)
btResult
```

```
##          model      RMSE Rsquared      MAE
## 1 Bagged Trees 0.9556692  0.66381 0.708786
```

Random Forest

```
rfModel <- randomForest(X_train, y_train)
rfPred <- predict(rfModel, newdata = X_test)
rfResult <- data.frame(as.list(postResample(pred = rfPred, obs = y_test))) |>
  mutate(model = 'Random Forest') |>
  relocate(model, RMSE, Rsquared, MAE)
rfResult
```

```
##           model      RMSE  Rsquared       MAE
## 1 Random Forest 0.8703731 0.7162801 0.6942521
```

Boosted Trees

```
gbmModel <- gbm.fit(X_train, y_train, distribution = 'gaussian', verbose = FALSE, n.trees = 100)
gbmPred <- predict(gbmModel, newdata = X_test)
```

```
## Using 100 trees...
```

```
gbmResult <- data.frame(as.list(postResample(pred = gbmPred, obs = y_test))) |>
  mutate(model = 'Boosted Trees') |>
  relocate(model, RMSE, Rsquared, MAE)
gbmResult
```

```
##           model     RMSE  Rsquared      MAE
## 1 Boosted Trees 1.559316 0.5158594 1.304727
```

Cubist

```
cModel <- cubist(X_train, y_train)
cPred <- predict(cModel, newdata = X_test)
cResult <- data.frame(as.list(postResample(pred = cPred, obs = y_test))) |>
  mutate(model = 'Cubist') |>
  relocate(model, RMSE, Rsquared, MAE)
cResult
```

```
##    model      RMSE  Rsquared       MAE
## 1 Cubist 0.9907287 0.6967787 0.7720304
```

Summary

```
stResult |>
  union(btResult) |>
  union(rfResult) |>
  union(gbmResult) |>
  union(cResult) |>
  arrange(desc(Rsquared))
```

```
##            model      RMSE  Rsquared       MAE
## 1 Random Forest 0.8703731 0.7162801 0.6942521
## 2        Cubist 0.9907287 0.6967787 0.7720304
## 3  Bagged Trees 0.9556692 0.6638100 0.7087860
## 4 Boosted Trees 1.5593162 0.5158594 1.3047268
## 5  Single Trees 1.2409388 0.5112617 1.0313754
```

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

The random forest model produced the highest R^2 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?

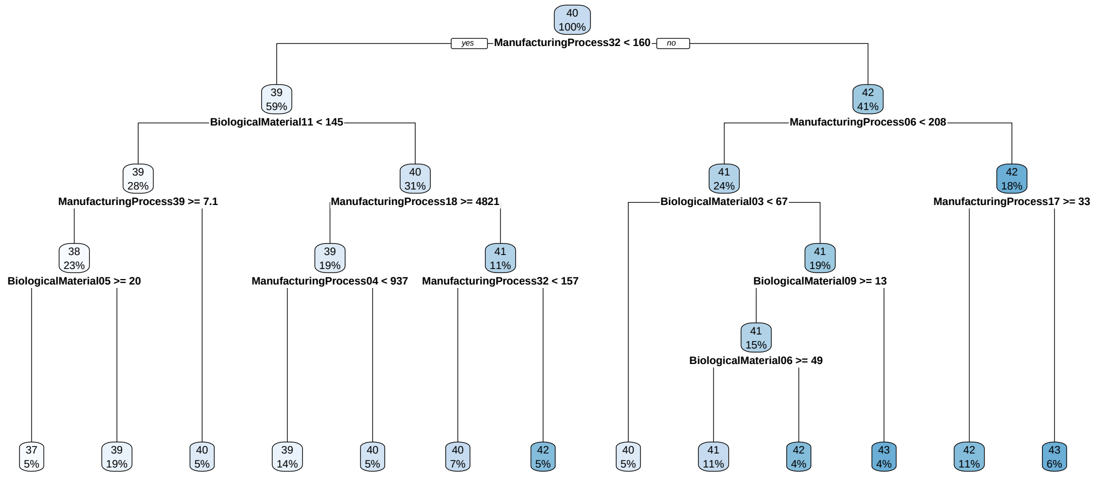
```r
top10 <- varImp(rfModel) |>
  arrange(desc(Overall)) |>
  head(10)
top10
```

```
##                          Overall
## ManufacturingProcess32 102.55700
## ManufacturingProcess13  34.63004
## BiologicalMaterial12    31.04151
## ManufacturingProcess31  24.25126
## BiologicalMaterial03    22.51428
## ManufacturingProcess17  21.53646
## BiologicalMaterial06    17.98136
## BiologicalMaterial11    16.42813
## ManufacturingProcess36  16.29007
## ManufacturingProcess09  16.17741
```

The tree model has most of the same predictors in the top 10 as the linear and nonlinear models produced in previous assignments.

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
rpartModel <- rpart(Yield ~ ., data = imputed)
rpart.plot(rpartModel)
```


40
100%
yes
ManufacturingProcess32 < 160
no
39
59%
BiologicalMaterial11 < 145
42
41%
ManufacturingProcess06 < 208
39
28%
ManufacturingProcess39 >= 7.1
40
31%
ManufacturingProcess18 >= 4821
41
24%
BiologicalMaterial03 < 67
42
18%
ManufacturingProcess17 >= 33
38
23%
BiologicalMaterial05 >= 20
39
19%
ManufacturingProcess04 < 937
41
11%
ManufacturingProcess32 < 157
41
19%
BiologicalMaterial09 >= 13
41
15%
BiologicalMaterial06 >= 49
37
5%
39
19%
40
5%
39
14%
40
5%
40
7%
42
5%
40
5%
41
11%
42
4%
43
4%
42
11%
43
6%

This graphical view gives you an indication of the combinations of predictors that affect the yield.