8.1

Joseph Elikishvili

December 12, 2017

## 8.1

We will load data from 7.2 as per instructions.

library(mlbench)  
set.seed(202)  
simulated <- mlbench.friedman1(200, sd = 1)  
simulated <- cbind(simulated$x, simulated$y)  
simulated <- as.data.frame(simulated)  
colnames(simulated)[ncol(simulated)] <- "y"

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

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

## Overall  
## V1 4.39203636  
## V2 8.54848388  
## V3 1.36216360  
## V4 10.09353398  
## V5 2.35489781  
## V6 -0.02689475  
## V7 0.12457474  
## V8 0.09277256  
## V9 0.16901670  
## V10 0.27216301

We can see that V1-V5 got much higher weights then V6-V10. By rough estimate we see a cumulative absolute weight of 25 for V1-V5 and an absolute cumulative weight of about 0.6 for V6-V10, this shows a significant relative difference Did the random forest model significantly use the uninformative predictors (V6 - V10)?

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

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

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?

Lets go ahead and add the predictor to the data and re run the model

simulated$V11 = duplicate1  
model2 <- randomForest(y ~ ., data = simulated,  
importance = TRUE,  
ntree = 1000)  
rfImp2 <- varImp(model2, scale = FALSE)  
rfImp2

## Overall  
## V1 2.98364173  
## V2 8.54289551  
## V3 1.14067755  
## V4 9.81482100  
## V5 1.96513668  
## V6 0.06254938  
## V7 0.04282992  
## V8 -0.06792789  
## V9 0.18501386  
## V10 0.28748225  
## V11 2.89615317

We can see that V1 score has decreased, as the highly corelated predictor has been added. So we can conclude that the importance of the variable will decrase if there are highly corelated predictors present.

### 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?

Lets run the model and review results

library(party)  
rf1 = cforest(y ~., data =simulated[, 1:11], controls = cforest\_control(ntree=1000))  
rf2 = cforest(y ~., data =simulated, controls = cforest\_control(ntree=1000))  
imp1= varImp(rf1)   
imp2= varImp(rf2)  
imp1

## Overall  
## V1 3.82003646  
## V2 8.87387156  
## V3 0.27458632  
## V4 12.15246108  
## V5 2.35611066  
## V6 -0.03701611  
## V7 -0.04672204  
## V8 -0.03247095  
## V9 0.07055175  
## V10 0.21559424

imp2

## Overall  
## V1 2.22847968  
## V2 8.58912042  
## V3 0.25889259  
## V4 11.68161820  
## V5 2.14115049  
## V6 0.02640094  
## V7 0.03121931  
## V8 -0.01765752  
## V9 0.05869952  
## V10 0.21726427  
## V11 2.64645437

We can see a smiliar behaivior, V1-V5 have the higherst importance while V6-V10 have the lower importance.

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

Lets try to generate results with Bagged Trees

library(ipred)  
bt1 = bagging(y ~., data = simulated[, 1:11], nbag =50)  
bt2 = bagging(y ~., data = simulated, nbag =50)  
imp\_bt1= varImp(bt1)   
imp\_bt2= varImp(bt2)  
imp\_bt1

## Overall  
## V1 1.7963534  
## V10 0.8297850  
## V2 1.7139834  
## V3 1.4817921  
## V4 2.2994632  
## V5 1.8843893  
## V6 0.7113112  
## V7 0.8754050  
## V8 0.7182479  
## V9 0.7914967

imp\_bt2

## Overall  
## V1 1.8856164  
## V10 0.8520204  
## V11 1.6965504  
## V2 1.8348689  
## V3 1.0828712  
## V4 2.1927115  
## V5 1.8026756  
## V6 0.5738606  
## V7 0.7198832  
## V8 0.8377508  
## V9 0.8205739

Bagged Trees are changing some of the results, we can see that V10 has a higher then normal importance and we see a more distributed importance then before, V6-V10 are no long just a franction of the V1-V5 importance levels.

Next we will try Cubist model

library(Cubist)  
cb1 = cubist(simulated[, 1:10], simulated$y, committees =100)  
cb2 = cubist(simulated[, names(simulated) !="y"], y=simulated$y, committees =100 )  
imp\_cb1 = varImp(cb1)  
imp\_cb2 = varImp(cb2)  
imp\_cb1

## Overall  
## V3 50.0  
## V1 53.0  
## V2 65.0  
## V4 52.5  
## V5 28.5  
## V7 11.5  
## V9 5.5  
## V10 2.0  
## V8 0.5  
## V6 0.0

imp\_cb2

## Overall  
## V3 51.5  
## V1 50.0  
## V2 64.0  
## V4 56.5  
## V5 30.0  
## V11 21.5  
## V8 1.0  
## V9 4.0  
## V7 6.0  
## V6 3.5  
## V10 2.5

Cubist model is similar to the first models V1-V5 have a much higher importance then V6-V10, although we see that V7 has moved up relative to V6-V10.

#### 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:

#### (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 model on the right focuses its importance on just first few predictors becuase of the fact that as the learning rate increases, it becomes more greedy and will use fewer predictors. Also another reason is due to bagging fraction, the higher is the fraction, less predictors will be identified as important.

#### (b) Which model do you think would be more predictive of other samples?

The left model is using more predictors, while the model on teh right relies mostly on top 3 predictors, so assumption can be made the the model on the left will be more accurate.

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

Increasing Interaction depth will most likely force the model to spread the impportance across predictors, so the model on the right will probably benefit more from it, but as a result, more weight wil be icnreased to predictors lagging importance.

#### 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:

We will import imputation and data splitting code from 6.3 below.

library(AppliedPredictiveModeling)  
library(caret)  
  
data(ChemicalManufacturingProcess)  
df = ChemicalManufacturingProcess  
#summary(df)  
  
library(missForest)  
df\_imp1 = missForest(df)

## missForest iteration 1 in progress...done!  
## missForest iteration 2 in progress...done!  
## missForest iteration 3 in progress...done!  
## missForest iteration 4 in progress...done!

df\_imp = df\_imp1$ximp  
  
data = df\_imp[,2:58]  
target = df\_imp[,1]  
  
training = createDataPartition( target, p=0.75 )  
predictor\_training = data[training$Resample1,]  
target\_training = target[training$Resample]  
  
predictor\_testing = data[-training$Resample1,]  
target\_testing = target[-training$Resample1]  
ctrl <- trainControl(method = "cv", number = 10)

We will use several models and will evaluate the results.

#### Regression tree

First we will use Single Tree model and evaluate the results.

#Tune  
rt\_grid <- expand.grid(maxdepth= seq(1,10,by=1))  
rt\_Tune <- train(x = predictor\_training, y = target\_training, method = "rpart2", metric = "Rsquared", tuneGrid = rt\_grid, trControl = ctrl)  
  
#Predict  
rt\_pred = predict(rt\_Tune, predictor\_testing)  
postResample(pred = rt\_pred, obs = target\_testing)

## RMSE Rsquared MAE   
## 1.4295136 0.4200005 1.0512342

Next we will use Random forest model to evaluate

#Tune  
rf\_grid <- expand.grid(mtry=seq(2,38,by=3))  
rf\_Tune <- train(x = predictor\_training, y = target\_training, method = "rf", tuneGrid = rf\_grid, metric = "Rsquared", importance = TRUE, trControl = ctrl)  
  
#Predict  
rf\_pred = predict(rt\_Tune, predictor\_testing)  
postResample(pred = rt\_pred, obs = target\_testing)

## RMSE Rsquared MAE   
## 1.4295136 0.4200005 1.0512342

Finally lets take a look at Cubist model

#Tune  
cube\_grid <- expand.grid(committees = c(1, 5, 10, 20, 50), neighbors = c(0, 1, 3, 5))  
  
cube\_Tune <- train(x = predictor\_training, y = target\_training, method = "cubist", metric = "Rsquared", tuneGrid = cube\_grid, trControl = ctrl)  
  
#Predict  
cube\_pred = predict(cube\_Tune, predictor\_testing)  
postResample(pred = cube\_pred, obs = target\_testing)

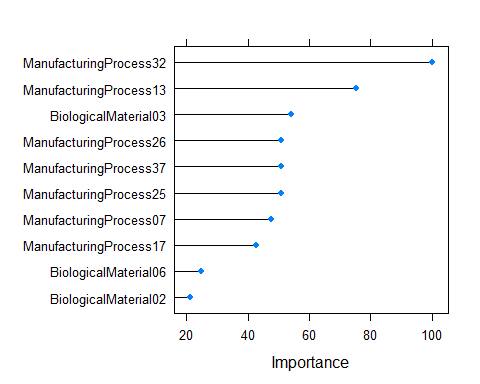
## RMSE Rsquared MAE   
## 0.8556065 0.7741709 0.6578451

It appear that Cubist model has the best RMSE score, it is an improvement over MARS scores and over PLS scores that we had.

#### (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?

Lets go ahead and review the predictor importance of cubist model

plot(varImp(cube\_Tune), top=10, scales = list(y = list(cex = 0.8)))

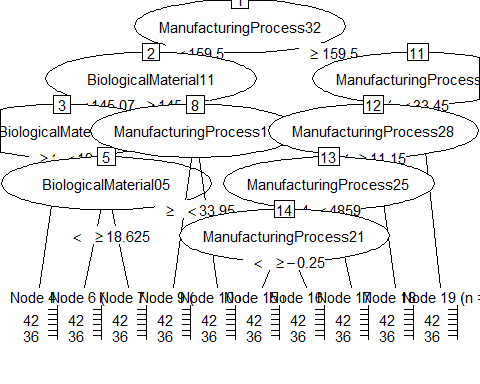


We can see that Manufacturing process32 made the top of the list which is not a surprise as we already established that it is the most importnant predictor, followed by Manufacturingprocess17. One thing to notice is that Cubist model heavily relys on top 2 predictors vs PLS, but we saw similar behaivior with MARS. One suprice is that Cubist is using Manufacturer process 17 and not 9 like MARS, but then agaon we saw similar usage with PLS and other models.

#### 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?

Next we will plot the OPtimal Sinlge tree

library(party)  
library(partykit)  
plot(as.party(rt\_Tune$finalModel),gp=gpar(fontsize=11))



As expected we can see the the top predictors are Manufacture related processes. Manufacturing process32 is at the top only few Biological processes affect target in any meaningful way.

#### Conlcusion.

At this point we have established that manufacturer processes have a much larger impacting role towards yild then biaological factors, Cubist model is the latest and strongest confirmation of that, so any further study should be focused on finding optimal values for manufacturing proceses.

### Market Basket Analysis

Lets go ahead and load the data and have a quick look

df = read.csv("c:/data/GroceryDataSet.csv", header = TRUE)  
summary(df)

## citrus.fruit semi.finished.bread margarine   
## sausage : 825 :2159 :3802   
## whole milk : 717 whole milk : 654 whole milk : 506   
## frankfurter : 580 other vegetables: 550 other vegetables: 415   
## tropical fruit : 482 root vegetables : 383 rolls/buns : 293   
## other vegetables: 460 rolls/buns : 378 yogurt : 289   
## citrus fruit : 452 tropical fruit : 355 soda : 229   
## (Other) :6318 (Other) :5355 (Other) :4300   
## ready.soups X X.1   
## :5101 :6105 :6960   
## whole milk : 315 rolls/buns : 176 soda : 150   
## other vegetables: 254 soda : 168 rolls/buns : 146   
## rolls/buns : 238 yogurt : 160 shopping bags: 107   
## soda : 211 whole milk : 149 bottled water: 95   
## yogurt : 202 shopping bags: 145 yogurt : 93   
## (Other) :3513 (Other) :2931 (Other) :2283   
## X.2 X.3 X.4   
## :7605 :8150 :8588   
## soda : 120 shopping bags: 76 soda : 61   
## shopping bags: 107 bottled water: 68 shopping bags : 56   
## rolls/buns : 92 newspapers : 66 fruit/vegetable juice: 55   
## newspapers : 68 rolls/buns : 59 bottled water : 54   
## domestic eggs: 57 soda : 59 newspapers : 51   
## (Other) :1785 (Other) :1356 (Other) : 969   
## X.5 X.6   
## :8938 :9184   
## shopping bags : 49 shopping bags: 40   
## soda : 39 newspapers : 36   
## fruit/vegetable juice: 34 pastry : 27   
## newspapers : 33 bottled water: 25   
## bottled water : 26 napkins : 23   
## (Other) : 715 (Other) : 499   
## X.7 X.8   
## :9366 :9483   
## soda : 30 soda : 24   
## shopping bags : 19 shopping bags : 18   
## chocolate : 17 fruit/vegetable juice: 16   
## fruit/vegetable juice: 17 napkins : 14   
## napkins : 17 newspapers : 14   
## (Other) : 368 (Other) : 265   
## X.9 X.10   
## :9561 :9638   
## shopping bags : 18 shopping bags : 16   
## fruit/vegetable juice: 17 napkins : 13   
## newspapers : 14 fruit/vegetable juice: 11   
## soda : 14 hygiene articles : 11   
## napkins : 11 candy : 9   
## (Other) : 199 (Other) : 136   
## X.11 X.12   
## :9693 :9739   
## shopping bags : 11 napkins : 8   
## napkins : 9 chocolate : 5   
## chocolate : 8 newspapers : 5   
## hygiene articles : 6 candy : 4   
## long life bakery product: 6 fruit/vegetable juice: 4   
## (Other) : 101 (Other) : 69   
## X.13 X.14 X.15   
## :9768 :9782 :9796   
## candy : 5 detergent : 4 bottled beer : 3   
## chocolate : 5 fruit/vegetable juice: 4 napkins : 3   
## napkins : 5 shopping bags : 4 pot plants : 3   
## newspapers : 3 chocolate : 3 candy : 2   
## bottled water: 2 bottled water : 2 hygiene articles: 2   
## (Other) : 46 (Other) : 35 (Other) : 25   
## X.16 X.17   
## :9805 :9816   
## napkins : 4 napkins : 2   
## fruit/vegetable juice : 2 baking powder : 1   
## house keeping products: 2 bottled beer : 1   
## hygiene articles : 2 cleaner : 1   
## candles : 1 cling film/bags: 1   
## (Other) : 18 (Other) : 12   
## X.18 X.19   
## :9820 :9826   
## waffles : 2 bottled beer : 2   
## chocolate marshmallow: 1 bottled water: 1   
## cling film/bags : 1 cake bar : 1   
## dental care : 1 coffee : 1   
## dog food : 1 flour : 1   
## (Other) : 8 (Other) : 2   
## X.20 X.21   
## :9827 :9827   
## chocolate : 2 chocolate : 1   
## fruit/vegetable juice : 1 female sanitary products: 1   
## liquor (appetizer) : 1 long life bakery product: 1   
## long life bakery product: 1 margarine : 1   
## pasta : 1 rum : 1   
## white wine : 1 (Other) : 2   
## X.22 X.23   
## :9828 :9829   
## abrasive cleaner : 1 chocolate : 1   
## chocolate : 1 hygiene articles: 1   
## hygiene articles : 2 napkins : 2   
## long life bakery product: 1 sugar : 1   
## specialty fat : 1   
##   
## X.24 X.25 X.26   
## :9830 :9833 :9833   
## cooking chocolate : 1 skin care: 1 hygiene articles: 1   
## house keeping products: 2   
## soups : 1   
##   
##   
##   
## X.27   
## :9833   
## candles: 1   
##   
##   
##   
##   
##

Lets see how many variables we are dealing with

dim(df)

## [1] 9834 32

We will next view top 30 items

#library(arules)  
#library(arulesViz)  
#library(datasets)  
#rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))  
#options(digits=2)  
#inspect(rules[1:5])