

# Team 2 - Homework Two

Assignment 3: KJ 8.1-8.3; KJ 8.7

*Juliann McEachern*

*11/8/19*

## Dependencies

```
# Forecast libraries
libraries("mlbench", "AppliedPredictiveModeling", "party", "randomForest",
         "caret")

# Formatting Libraries
libraries("default", "knitr", "kableExtra", "tidyverse")

# Plotting Libraries
libraries("ggplot2", "grid", "ggfortify")
```

## (1) Kuhn & Johnson 8.1

Recreate the simulated data from Exercise 7.2:

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

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

The random forest model predominately favored the predictor variables V1-V5, with the exception of V3. The uninformative predictors all had significantly lower overall importance.

Table 1: Variable Importance for Random Forest Model1

	V1	V4	V2	V5	V3	V6	V7	V10	V9	V8
Overall	8.73	7.62	6.42	2.02	0.76	0.17	-0.01	-0.07	-0.1	-0.17

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

```
simulated$duplicate1 <- simulated$V1 + rnorm(200) * 0.1
model2 <- randomForest(y ~ ., data = simulated, importance = T,
  ntree = 1000)
rfImp2 <- varImp(model2, scale = F)
```

The correlation of the simulated V1 and duplicate variable was 0.9338163. Adding a highly correlated value, such as this duplicate, decreased the overall variable importance of V1.

Table 2: Variable Importance for Random Forest Model2

	V4	V1	V2	duplicate1	V5	V3	V6	V7	V9	V10	V8
<b>Overall</b>	6.87	6.59	5.86	2.94	2.07	0.61	0.1	0.08	-0.07	-0.13	-0.17

(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?

Table 3: Variable Importance for Conditional Random Forest Model3

	V4	V1	V2	V5	V3	V9	V8	V10	V6	V7
<b>Overall</b>	6.66	5.5	5.19	1.21	0.03	0	0	-0.01	-0.01	-0.01

Table 4: Variable Importance for Conditional Random Forest Model3

	V4	V2	duplicate1	V1	V5	V3	V7	V6	V10	V9	V8
<b>Overall</b>	6.05	4.82	1.92	1.92	1.06	0.02	0.01	0	-0.01	-0.01	-0.02

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

Table 5: Variable Importance for Random Forest Model2

	V4	V1	V2	V5	V3	V7	V6	V8	V10	V9
<b>Overall</b>	100	85.68	74.23	38.2	27.74	4.95	1.92	0.37	0.22	0

Table 6: Variable Importance for Random Forest Model2

	V4	V2	duplicate1	V5	V1	V3	V6	V7	V8	V10	V9
<b>Overall</b>	100	72.03	52.34	39.8	38.86	30.24	2.07	0.53	0.44	0.22	0

## (2) Kuhn & Johnson 8.2

Use a simulation to show tree bias with different granularities.

### (3) Kuhn & Johnson 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?
- (b). Which model do you think would be more predictive of other samples?
- (c). How would increasing interaction depth affect the slope of predictor importance for either model in Fig.8.24?

### (4) Kuhn & Johnson 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?
- (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?
- (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 Code

```
# (8.1)
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"

# (8.1a)
model1 <- randomForest(y ~ ., data = simulated, importance = TRUE,
  ntree = 1000)
```

```

rfImp1 <- varImp(model1, scale = FALSE)

# (8.1b)
simulated$duplicate1 <- simulated$V1 + rnorm(200) * 0.1
cor(simulated$duplicate1, simulated$V1)
model2 <- randomForest(y ~ ., data = simulated, importance = T,
  ntree = 1000)
rfImp2 <- varImp(model2, scale = F)

# (8.1c) Rename variables for ease of reference
sim_original <- select(simulated, -duplicate1)
sim_duplicate <- simulated
model3 <- cforest(y ~ ., data = sim_original, controls = cforest_unbiased(ntree = 1000))
cfImp3 <- varimp(model3, conditional = TRUE)
model4 <- cforest(y ~ ., data = sim_duplicate, controls = cforest_unbiased(ntree = 1000))
cfImp4 <- varimp(model4, conditional = TRUE)

# (8.1d) Create tuning parameters
fitControl <- trainControl(method = "repeatedcv", number = 10,
  repeats = 10)
model5 <- train(y ~ ., data = sim_original, method = "gbm", trControl = fitControl,
  verbose = FALSE)
cfImp5 <- varimp(model5, conditional = TRUE)
model6 <- cforest(y ~ ., data = sim_duplicate, controls = cforest_unbiased(ntree = 1000))
cfImp6 <- varimp(model6, conditional = TRUE)

# (8.2)

# (8.3a)

# (8.3b)

# (8.3c)

# (8.7a)

# (8.7b)

# (8.7c)

```