Team 2 - Homework Two

Assignment 3: KJ 8.1-8.3; KJ 8.7

Bethany

November 9, 2019

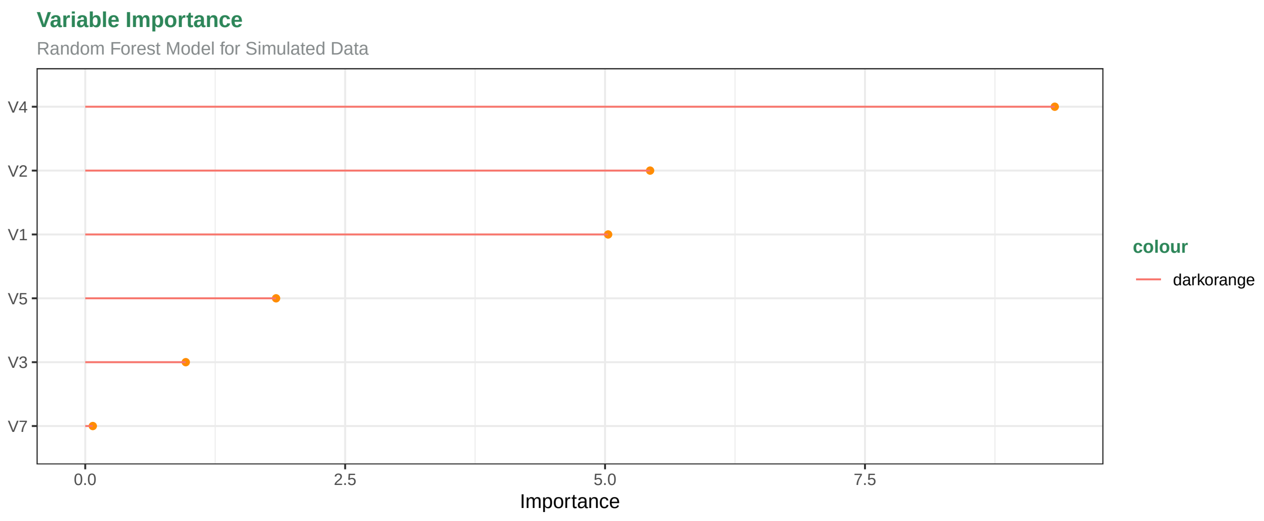
Dependencies

```
# Forecast libraries
libraries("mlbench", "AppliedPredictiveModeling")

# Regression libraries
libraries("randomForest", "caret")

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

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

(1) Kuhn & Johnson 8.1

Recreate the simulated data from Exercise 7.2:

(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)?

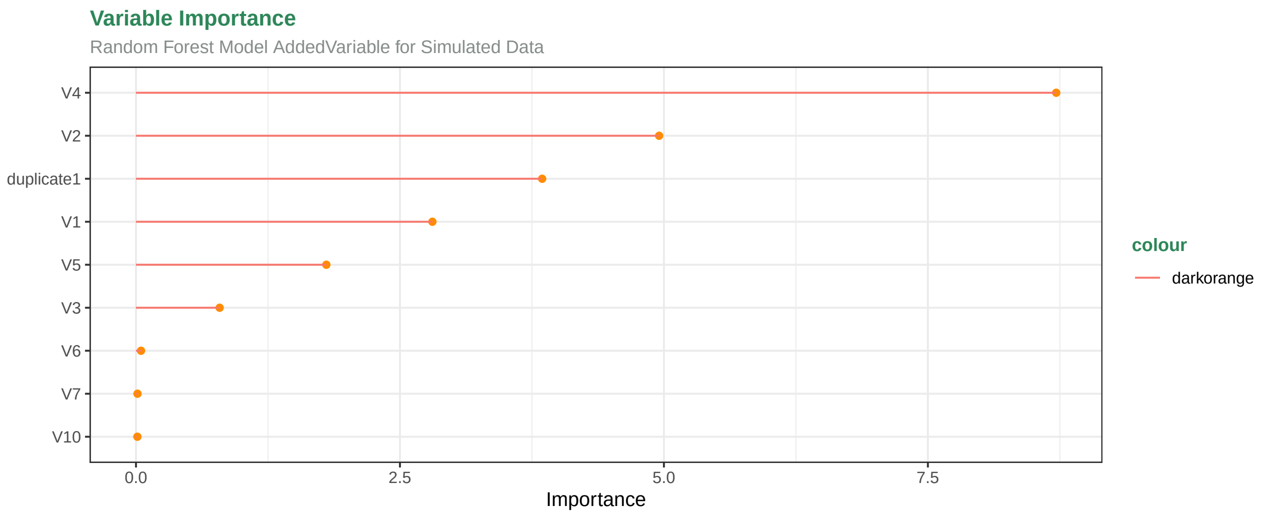


Based on the first random forest model, the first five variables in the data `v1 - v5` are considered the most important.

(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:

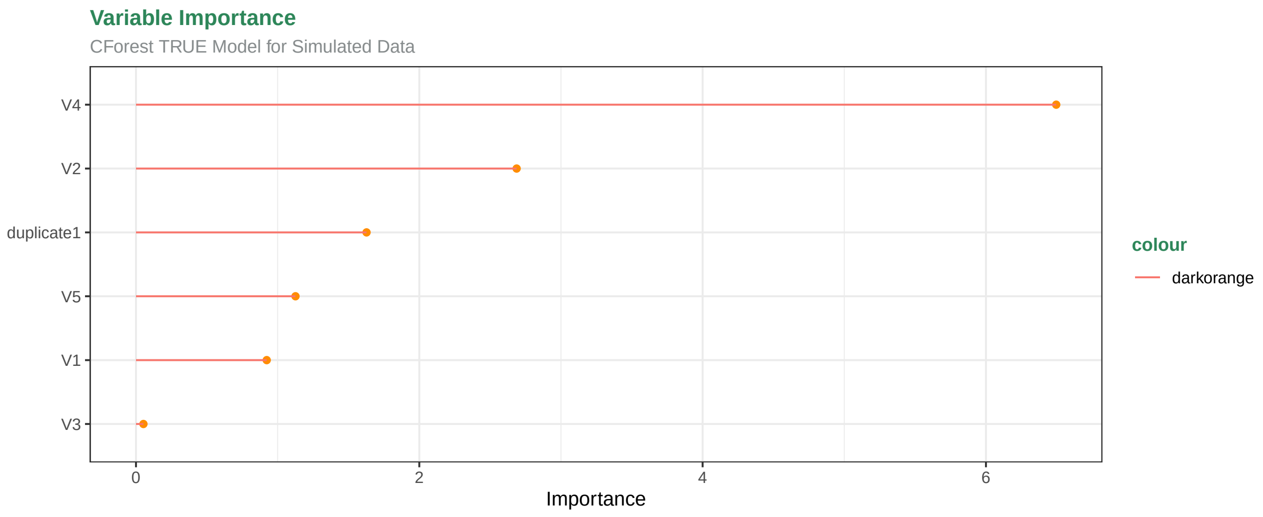
Correlated Variable Relationship to `Variable 1` :



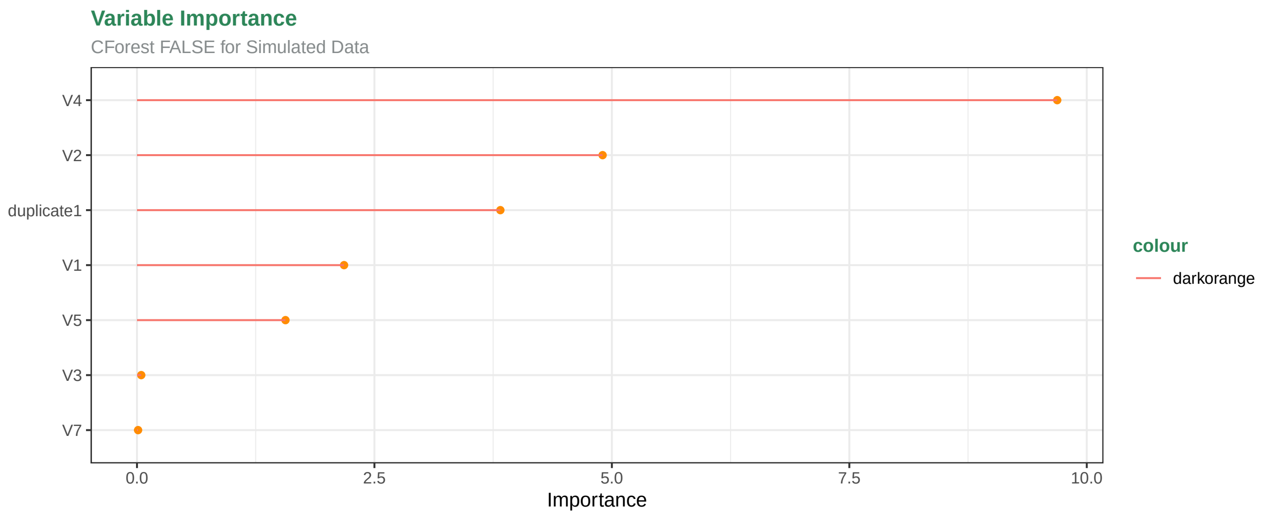
After adding the highly correlated variable, the first five variables remaint he most important, however, the correlated variable becomes one of the top three predictors too. In fact it is more important that the `v1` variable upon which it is based.

(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?

C Forest True

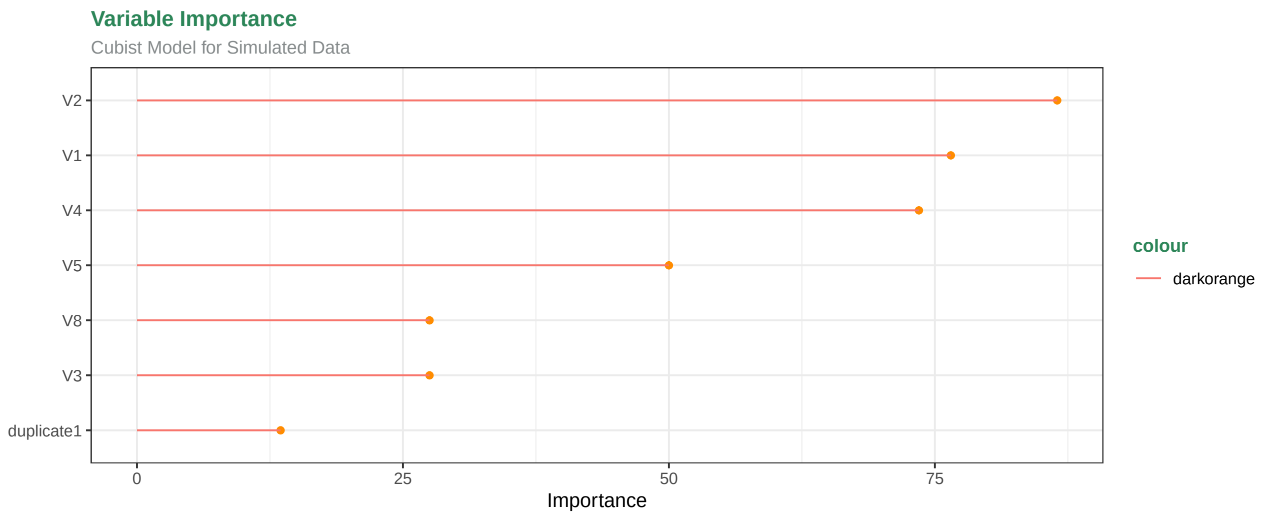


C Forest False

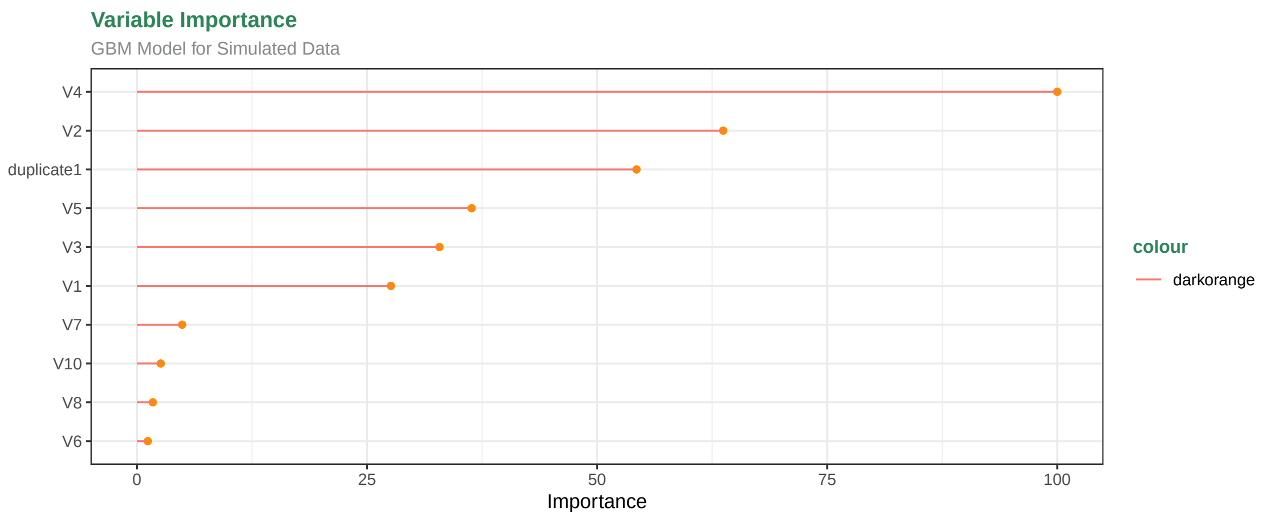


(d). Repeat this process with different tree models, such as boosted trees and Cubist. Does the same pattern occur?

Cubist Tree Model



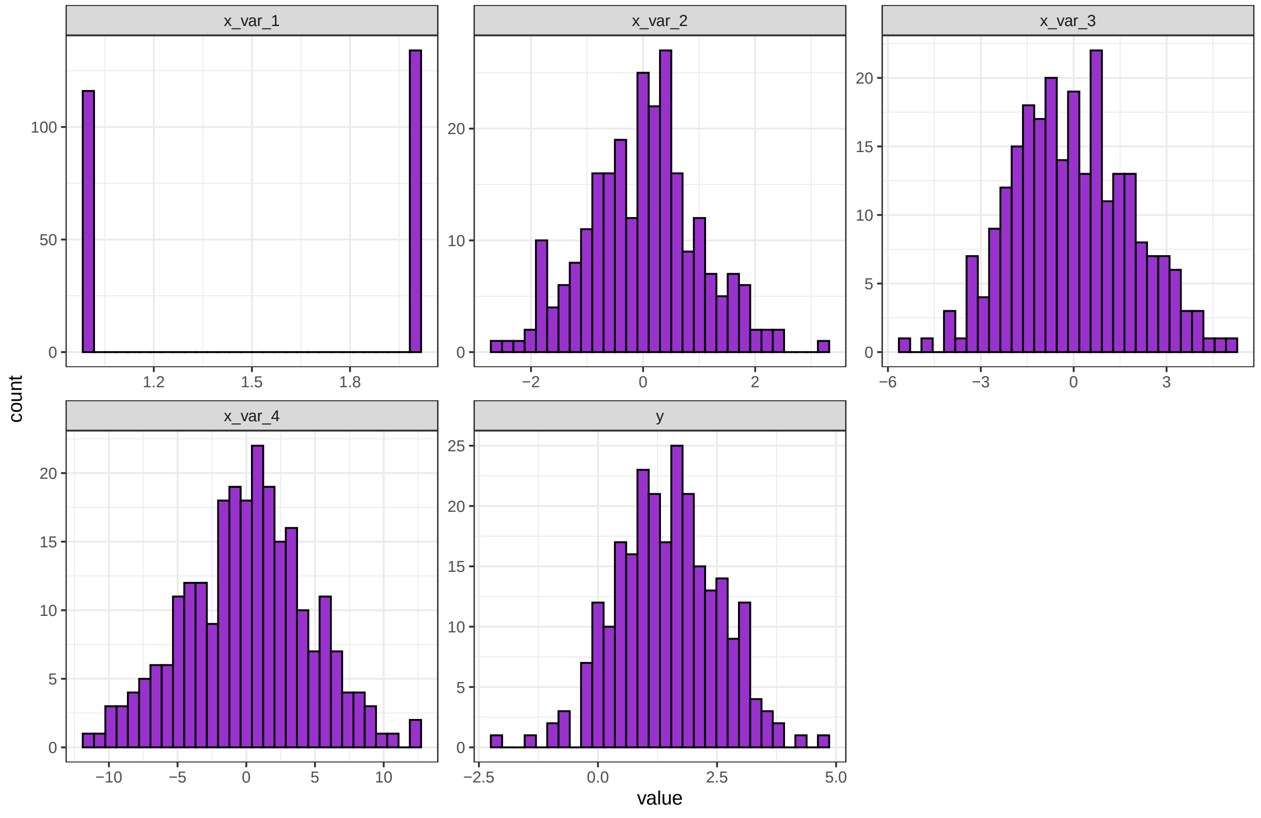
Boosted Trees Model



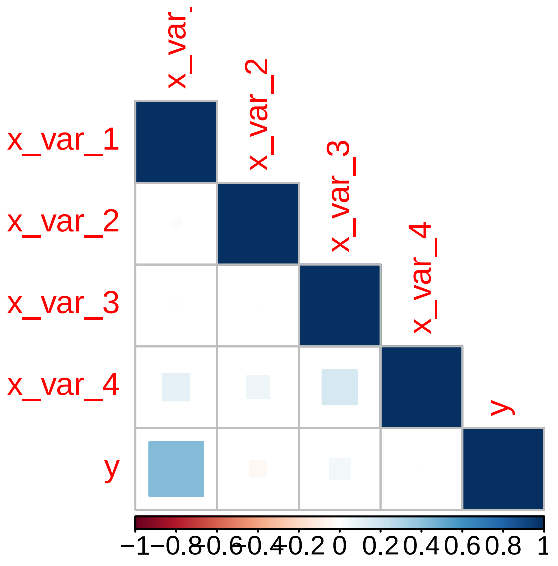
(2) Kuhn & Johnson 8.2

Use a simulation to show tree bias with different granularities.

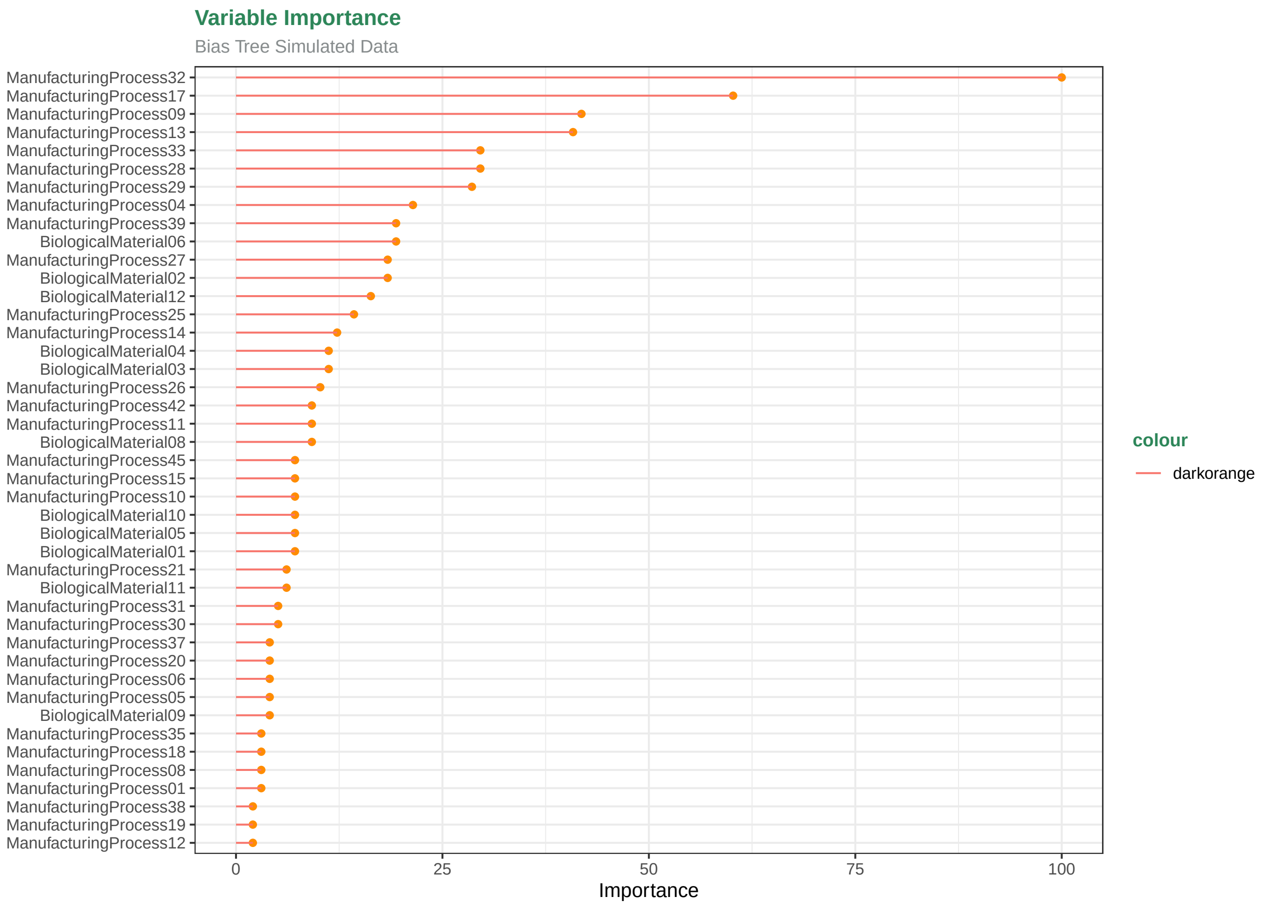
We created four variables - a binary numerical variable with the numbers 1 and 2 - a standard normal variable - a normal variable with a standard deviation of 2 - a normal variable with a standard deviation of 4 - a y variable that is the binary variable, with a random normal added to it



Based on this data and the correlation plot, the most related variable is the binary variable.



However, when you look look at the most import variables in the decision making process, `variable 1`, the least important of all the variables. When you look at the variable distributions, `variable 4` has the widest range of values providing more opportunity to split in more places than `variable 1`. Accordingly you might expect `variable 3` to be then next most important, however, the splits made prior influence the decisions further down in the tree such that variable two becomes the second most important variable.



(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?**

Because the model on the right has hig learning rate and high bagging fractions, it is doing two things, using 90% of the variables in each tree and using 90% of the error for a given model. Imagine there were ten variables in this model, if you used a bagging fraction of .9, the every tree would have 9 of the ten trees in it, only one variable would be different from the set of 9 trees in the first model each time. The means that most of the possbile break-points in the trees would be the same from tree to tree, only offering new opportunities for initial splits when the most dominating variabl is removed. Because of this, you would very likely find the model making the same first few decisions each time. And with a learning rate of .9, 90% of the error is added in from each tree,this means that in addition to consistently choosing from one or a few initial splits, you are also maximizing their contributions to the models.

Because of these two factors, the trees contributing to the Stockastic Boosted Tree in this example will make very few decisions (maening they will make the same decisions repeately). That lack of variation in initial choices means that the number of paths the learning can take is limited, and with a high learning rate, a core set of variables is selected early on from the trees built very similarly. In essence, the model never has the opportunity to evaluate other possible variables because the greediness of the model makes the same first few choices every time(

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

The .9, .9 model would likely be overfit to the training data, because the variation in values within those most important variables may not be reflective of the general population. So, this model will be tuned to choose from sample members following the samples distribution of values in those most important features at the expense of recognizing potiential splits in other variables which might be more common in the poolation that the sample.

With a learning rate of .1 and a bagging fraction of .1, the left model is more likely to build truly weak predictors, from smallers sets of variables, consider more distinct breaking points, and therefore extend better to wild data not fully described by the first few variables in the importance summarise_layers

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

Increasing the tree depth would affect both models, but differently. The model with bagging fraction and learning rate equal to .1, increasing the number of nodes in the tree would likely increase the importance of the lower variables, creating a less polynomial slope. this is because making more decisions, means giving weight the variables and values where those decisions are made. This would give importance to those variables. For the model with bagging fraction and learning rate equal to .9, increasing the depth would likely not change the slope of the lower variables much at all just ad new variable or two to the top. The difference is that in this model, with the high fraction and learning rate, we again will still be making most of the same decisions, from tree to tree, so the only increases in importance come from the added nodes, which will be downstream, and they too are highly likely to be the same from tree to tree, such that you will add a importance low on the scale to those variables upon which the new nodes break (or upper variables with a second or third break will grow in importance). So you might see a reshuffling of upper nodes and the slight increase of a one or two less important variables. However, the overall slope, of quickly going to zero will be contrained by the high bagging fraction and learning rate.

(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?

The boosted model has the lowest training error, and the Cubist the highest, but the cubist model seems to have the lowest test $RMSE$

Training Model

Random Forest Train

RMSE	Rsquared	MAE
1.283481	0.5612445	0.9835251

Test

RMSE	Rsquared	MAE
0.9089474	0.728537	0.7204187

Boosted Model

Train

RMSE	Rsquared	MAE
1.254821	0.5470138	0.9589064

Test

RMSE	Rsquared	MAE
0.8192842	0.7689348	0.6330436

Cubist

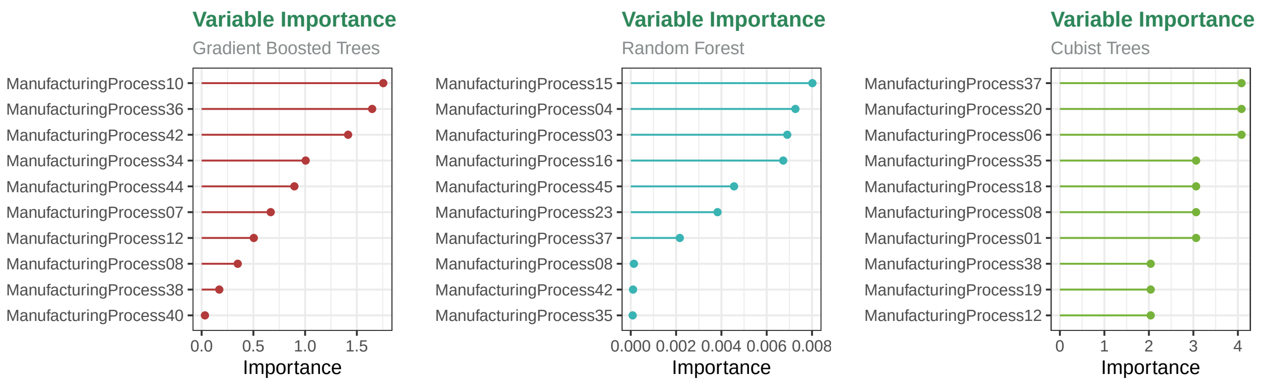
Train

RMSE	Rsquared	MAE
37	3	0.9516027

Test

RMSE	Rsquared	MAE
0.7574218	0.8115943	0.5379256

(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?

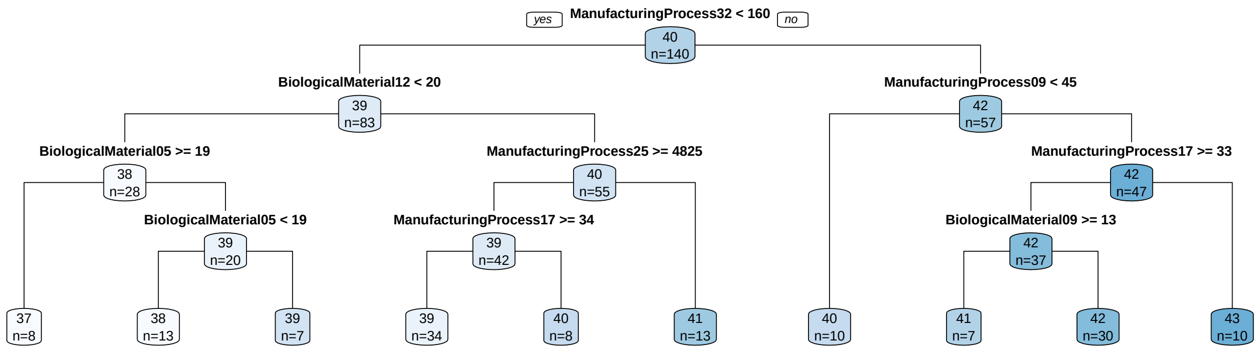


In all the models the Manufacturing processes dominate the list, with slight differences on where in the list and how much influence each has

Comparing the top 10 variables in each model reveals some strong differences. the Boosted tree follows a rather linear drop-off of importance through a list of exclusively process-based variables. The random forest falls off slower in the first five variables but becomes rapidly linear. All but the last variable in this list are also process based, the last is a biological material variable. The cubist tree, however takes on a very different depreciation, as the values are discrete, with the first variable (biological) at 4, and the next six process variables all at exactly thre and the last three variables at 2. Other than the first and last variables all of the cubists top 10 are manufacturing process variable.names

What is as interesting as the importance values and the transitions between the variables is the fact that from model to model, the only variable that was in the top 10 for all three was ManufacturingProcess38, and it was near the bottom of all three lists.

(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?


yes
ManufacturingProcess32 < 160
no
40
n=140
BiologicalMaterial12 < 20
39
n=83
ManufacturingProcess09 < 45
42
n=57
BiologicalMaterial05 >= 19
38
n=28
ManufacturingProcess25 >= 4825
40
n=55
ManufacturingProcess17 >= 33
42
n=47
BiologicalMaterial05 < 19
39
n=20
ManufacturingProcess17 >= 34
39
n=42
BiologicalMaterial09 >= 13
42
n=37
37
n=8
38
n=13
39
n=7
39
n=34
40
n=8
41
n=13
40
n=10
41
n=7
42
n=30
43
n=10

Based on this regression tree, the differences between process and matrial is that process variables differentiate more observations at each break, which is why the break first, they increase the purity most quickly. However, the final decisions seem to be based rather wholly on biological material variables, such that only looking at Process or prunning too soon, might lead to overfitting.

R Code

```
# insert code here
require(caret)
require(impute)
require(mlbench)
require(randomForest)
require(caret)
require(partykit)
require(party)
require(Cubist)
require(gbm)
require(corrplot)
require(rpart)
require(AppliedPredictiveModeling)
require(caTools)
require(rpart.plot)
require(gridExtra)
require(tidyverse)


set.seed(58677)
# (8.1)

simulated <- mlbench.friedman1(200, sd = 1)
simulated <- cbind(simulated$x, simulated$y)
simulated <- as.data.frame(simulated)
colnames(simulated)[ncol(simulated)] <- "y"
x <- simulated %>% select(-y)
y = simulated %>% select(y)

# (8.1a)


model1_81 <- randomForest(y ~ ., data = simulated, importance = TRUE,
    ntree = 1000)
```

```r
rfImp1_81 <- varImp(model1_81, scale = FALSE)
rfImp1_81 <- as.data.frame(rfImp1_81) %>% rownames_to_column("Variable") %>%
    filter(Overall > 0)
plot_81a <- ggplot(rfImp1_81, aes(x = reorder(Variable, Overall),
    y = Overall)) + geom_point(color = "darkorange") + geom_segment(aes(x = Variable,
    xend = Variable, y = 0, yend = Overall, color = "darkorange")) +
    labs(title = "Variable Importance", subtitle = "Random Forest Model for Simulated Data",
        x = "", y = "Importance") + coord_flip() + theme_bw() +
    theme()


# (8.1b)

simulated$duplicate1 <- simulated$V1 + rnorm(200) * 0.1
# cor(simulated$duplicate1, simulated$V1)

cors <- cor(simulated$duplicate1, simulated$V1)
model2_81 <- randomForest(y ~ ., data = simulated, importance = TRUE,
    ntree = 1000)
rfImp2_81 <- varImp(model2_81, scale = FALSE)
rfImp2_81 <- as.data.frame(rfImp2_81) %>% rownames_to_column("Variable") %>%
    filter(Overall > 0)
eight_1_rf <- ggplot(rfImp2_81, aes(x = reorder(Variable, Overall),
    y = Overall)) + geom_point(color = "darkorange") + geom_segment(aes(x = Variable,
    xend = Variable, y = 0, yend = Overall, color = "darkorange")) +
    labs(title = "Variable Importance", subtitle = "Random Forest Model AddedVariable for Simulated Data
        x = "", y = "Importance") + coord_flip() + theme_bw() +
    theme()


# (8.1c)





model3_81 <- cforest(y ~ ., data = simulated)
rfImp3_true_81 <- varImp(model3_81, conditional = TRUE)
rfImp3_true_81 <- as.data.frame(rfImp3_true_81) %>% rownames_to_column("Variable") %>%
    filter(Overall > 0)
eight_1_cforest_t <- ggplot(rfImp3_true_81, aes(x = reorder(Variable,
    Overall), y = Overall)) + geom_point(color = "darkorange") +
    geom_segment(aes(x = Variable, xend = Variable, y = 0, yend = Overall,
        color = "darkorange")) + labs(title = "Variable Importance",
    subtitle = "CForest TRUE Model for Simulated Data", x = "",
    y = "Importance") + coord_flip() + theme_bw() + theme()


rfImp3_false_81 <- varImp(model3_81, conditional = FALSE)
rfImp3_false_81 <- as.data.frame(rfImp3_false_81) %>% rownames_to_column("Variable") %>%
    filter(Overall > 0)
eight_1_cforest_f <- ggplot(rfImp3_false_81, aes(x = reorder(Variable,
    Overall), y = Overall)) + geom_point(color = "darkorange") +
    geom_segment(aes(x = Variable, xend = Variable, y = 0, yend = Overall,
        color = "darkorange")) + labs(title = "Variable Importance",
    subtitle = "CForest FALSE for Simulated Data", x = "", y = "Importance") +
    coord_flip() + theme_bw() + theme()

x <- simulated %>% select(-y)


# (8.1d)

model4_81 <- cubist(x = x, y = simulated$y)
rfImp4_81 <- varImp(model4_81, scale = FALSE)

rfImp4_81 <- as.data.frame(rfImp4_81) %>% rownames_to_column("Variable") %>%
    filter(Overall > 0)
eight_1_cube <- ggplot(rfImp4_81, aes(x = reorder(Variable, Overall),
    y = Overall)) + geom_point(color = "darkorange") + geom_segment(aes(x = Variable,
    xend = Variable, y = 0, yend = Overall, color = "darkorange")) +
    labs(title = "Variable Importance", subtitle = "Cubist Model for Simulated Data",
        x = "", y = "Importance") + coord_flip() + theme_bw() +
    theme()



# GBM
gbmGrid_81 <- expand.grid(interaction.depth = seq(1, 7, by = 2),
    n.trees = seq(100, 1000, by = 50), shrinkage = c(0.01, 0.1),
    n.minobsinnode = 10)

gbmTune_81 <- train(x, simulated$y, tuneGrid = gbmGrid_81, method = "gbm",
    verbose = FALSE)

gbmImp_81 <- varImp(gbmTune_81)
gbmImp_81 <- as.data.frame(gbmImp_81$importance) %>% rownames_to_column("Variable") %>%
    filter(Overall > 0)
eight_1_boost <- ggplot(gbmImp_81, aes(x = reorder(Variable,
    Overall), y = Overall)) + geom_point(color = "darkorange") +
    geom_segment(aes(x = Variable, xend = Variable, y = 0, yend = Overall,
        color = "darkorange")) + labs(title = "Variable Importance",
    subtitle = "GBM Model for Simulated Data", x = "", y = "Importance") +
    coord_flip() + theme_bw() + theme()



# (8.2)

set.seed(58677)
x_var_1 <- sample(x = c(1, 2), size = 250, replace = TRUE)
x_var_2 <- rnorm(n = 250, mean = 0, sd = 1)
x_var_3 <- rnorm(n = 250, mean = 0, sd = 2)
x_var_4 <- rnorm(n = 250, mean = 0, sd = 4)
y = x_var_1 + rnorm(n = 250, mean = 0, sd = 1)
```

```r
sim_data <- data.frame(cbind(x_var_1, x_var_2, x_var_3, x_var_4,
    y))
simCor <- cor(sim_data)
# corrplot(simCor, method = 'square', type = 'lower')
# psych::describe(sim_data)
sim_data %>% keep(is.numeric) %>% gather() %>% ggplot(aes(value)) +
    facet_wrap(~key, scales = "free") + geom_histogram(fill = "darkorchid",
    color = "black") + theme_bw()

eight_two_tree <- rpart(y ~ ., data = sim_data)
import <- varImp(eight_two_tree)
import <- as.data.frame(import) %>% rownames_to_column("Variable") %>%
    filter(Overall > 0)
eight_two_plot <- ggplot(import, aes(x = reorder(Variable, Overall),
    y = Overall)) + geom_point(color = "darkorange") + geom_segment(aes(x = Variable,
    xend = Variable, y = 0, yend = Overall, color = "darkorange")) +
    labs(title = "Variable Importance", subtitle = "Bias Tree Simulated Data",
        x = "", y = "Importance") + coord_flip() + theme_bw() +
    theme()


# (8.3a)


# #a.  Because the model on the right has high learning rate
# and high bagging fractions, it is doing two things, using
# 90% of the variables in each tree and using 90% of the
# error for a given model. Imagine there were ten variables
# in this model, if you used a bagging fraction of .9, the
# every tree would have 9 of the ten trees in it, only one
# variable would be different from the set of 9 trees in the
# first model each time. The means that most of the possbile
# break-points in the trees would be the same from tree to
# tree, only offering new opportunities for initial splits
# when the most dominating variabl is removed. Because of
# this, you would very likely find the model making the same
# first few decisions each time. And with a learning rate of
# .9, 90% of the error is added in from each tree,this means
# that in addition to consistently choosing from one or a few
# initial splits, you are also maximizing their contributions
# to the models.  Because of these two factors, the trees
# contributing to the Stockastic Boosted Tree in this example
# will make very few decisions (maening they will make the
# same decisions repeatedly). That lack of variation in
# initial choices means that the number of paths the learning
# can take is limited, and with a high learning rate, a core
# set of variables is selected early on from the trees built
# very similarly. In essence, the model never has the
# opportunity to evaluate other possible variables because
# the greediness of the model makes the same first few
# choices every time( (8.3b)

# The .9, .9 model would likely be overfit to the training
# data, because the variation in values within those most
# important variables may not be reflective of the general
# population. So, this model will be tuned to choose from
# sample members following the samples distribution of values
# in those most important features at the expense of
# recognizing potential splits in other variables which
# might be more common in the poolation that the sample.
# With a learning rate of .1 and a bagging fraction of .1,
# the left model is more likely to build truly weak
# predictors, from smallers sets of variables, consider more
# distinct breaking points, and therefore extend better to
# wild data not fully described by the first few variables in
# the importance summarise_layers (8.3c) Increasing the tree
# depth would affect both models, but differently.  The model
# with bagging fraction and learning rate equal to .1,
# increasing the number of nodes in the tree would likely
# increase the importance of the lower variables, creating a
# less polynomial slope. this is because making more
# decisions, means giving weight the variables and values
# where those decisions are made. This would give importance
# to those variables.  For the model with bagging fraction
# and learning rate equal to .9, increasing the depth would
# likely not change the slope of the lower variables much at
# all just ad new variable or two to the top. The difference
# is that in this model, with the high fraction and learning
# rate, we again will still be making most of the same
# decisions, from tree to tree, so the only increases in
# importance come from the added nodes, which will be
# downstream, and they too are highly likely to be the same
# from tree to tree, such that you will add a importance low
# on the scale to those variables upon which the new nodes
# break (or upper variables with a second or third break will
# grow in importance). So you might see a reshuffling of
# upper nodes and the slight increase of a one or two less
# important variables. However, the overall slope, of quickly
# going to zero will be contrained by the high bagging
# fraction and learning rate.  (8.7a)



data("ChemicalManufacturingProcess")
# Total NA Values
na_table <- table(is.na(ChemicalManufacturingProcess))
total_na <- sapply(ChemicalManufacturingProcess[2:57], function(x) sum(is.na(x)))
na_table <- sapply(ChemicalManufacturingProcess, function(x) table(is.na(x)))
total_na <- data.frame(sort(total_na, decreasing = TRUE))
total_na <- cbind(Variable = rownames(total_na), total_na)
rownames(total_na) <- 1:nrow(total_na)
colnames(total_na) <- c("Variable", "Count")
total_na <- cbind(total_na[1:28, ], total_na[29:56, ])
# chem_hist <- ggplot(ChemicalManufacturingProcess, aes(x =
# Yield))+ geom_histogram(colour ='black', fill =
# 'violetred4')
```

```r
imputed_data = data.frame(impute.knn(as.matrix(ChemicalManufacturingProcess),
    k = 10, rowmax = 0.3, colmax = 0.85, rng.seed = 1942)$data)


sample = sample.split(imputed_data, SplitRatio = 0.8)  # splits the data in the ratio mentioned in Spli
trainingData = subset(imputed_data, sample == TRUE)  # creates a training dataset named train1 with row
testData = subset(imputed_data, sample == FALSE)
x_train <- trainingData[, 2:58]
x_test <- testData[, 2:58]
y_train <- as.vector(trainingData$Yield)

# Boosted
set.seed(58677)
gbmGrid_87 <- expand.grid(interaction.depth = seq(1, 7, by = 2),
    shrinkage = c(0.01, 0.1), n.trees = seq(100, 1000, by = 50),
    n.minobsinnode = 10)

gbmTune_87 <- train(x_train, y_train, method = "gbm", verbose = FALSE,
    tuneGrid = gbmGrid_87)

# gbmTune_87 plot(gbmTune_87)
min(gbmTune_87$results$RMSE, na.rm = TRUE)

# 0.01 7 850 1.254821 0.5470138 0.9589064
gbmPred_87 <- predict(gbmTune_87, newdata = x_test)
gb_test_87 <- postResample(pred = gbmPred_87, obs = testData$Yield)

# Random Forest
set.seed(58677)
rf_87_grid <- expand.grid(mtry = seq(100, 1000, by = 50))
rfTune_87 <- train(x_train, y_train, method = "rf", ntree = 50,
    tuneGrid = rf_87_grid)





# plot(rfTune_87) min(rfTune_87$results$RMSE, na.rm = TRUE)

rfPred_87 <- predict(rfTune_87, newdata = x_test)
rf_test_87 <- postResample(pred = rfPred_87, obs = testData$Yield)
rf_87 <- randomForest(x_train, y_train, importance = TRUE, ntree = 900)

# Cubist


set.seed(58677)
cb_grid_87 <- expand.grid(committees = c(25:45), neighbors = c(1,
    3, 5))
cbTune_87 <- train(x_train, trainingData$Yield, method = "cubist",
    metric = "RMSE", na.action = na.pass, tuneGrid = cb_grid_87,
    trControl = trainControl(method = "cv"))
```

```r
# plot(cbTune_87)
min(cbTune_87$results$RMSE, na.rm = TRUE)

cbPred_87 <- predict(cbTune_87, newdata = x_test)
cb_test_87 <- postResample(pred = cbPred_87, obs = testData$Yield)

# (8.7b)



# Boosted Model
import <- varImp(gbmTune_87)
import <- as.data.frame(import$importance) %>% rownames_to_column("Variable") %>%
    filter(Overall > 0) %>% arrange(Overall)
boost <- ggplot(import[1:10, ], aes(x = reorder(Variable, Overall),
    y = Overall)) + geom_point(color = "#b33a3a") + geom_segment(aes(x = Variable,
    xend = Variable, y = 0, yend = Overall), color = "#b33a3a") +
    labs(title = "Variable Importance Chemical Manufacturing",
        subtitle = "Gradient Boosted Trees", x = "", y = "Importance") +
    coord_flip() + theme_bw() + theme()

# rf Model
import <- varImp(rf_87, scale = FALSE)
import <- as.data.frame(import) %>% rownames_to_column("Variable") %>%
    filter(Overall > 0) %>% arrange(Overall)
random <- ggplot(import[1:10, ], aes(x = reorder(Variable, Overall),
    y = Overall)) + geom_point(color = "#3ab3b3") + geom_segment(aes(x = Variable,
    xend = Variable, y = 0, yend = Overall), color = "#3ab3b3") +
    labs(title = "Variable Importance Chemical Manufacturing",
        subtitle = "Random Forest", x = "", y = "Importance") +
    coord_flip() + theme_bw() + theme()



# Cubist Model
import <- varImp(cbTune_87)
import <- as.data.frame(import$importance) %>% rownames_to_column("Variable") %>%
    filter(Overall > 0) %>% arrange(Overall)
cube <- ggplot(import[1:10, ], aes(x = reorder(Variable, Overall),
    y = Overall)) + geom_point(color = "#77b33a") + geom_segment(aes(x = Variable,
    xend = Variable, y = 0, yend = Overall), color = "#77b33a") +
    labs(title = "Variable Importance Chemical Manufacturing",
        subtitle = "Cubist Trees", x = "", y = "Importance") +
    coord_flip() + theme_bw() + theme()



# eight_seven <- grid.arrange(boost, random, cube, ncol=3)

# IN all the models the Manufacturing processes dominate the
# list, with slight differences on where in the list and how
# much influence each has
# Comparing the top 10 variables in each model reveals some
# strong differences. the Boosted tree follows a rather
# linear drop-off of importance through a list of exclusively
# process-based variables. The random forest falls off slower
# in the first five variables but becomes rapidly linear. All
# but the last variable in this list are also process based,
# the last is a biological material variable. The cubist
# tree, however takes on a very different depreciation, as
# the values are discrete, with the first variable
# (biological) at 4, and the next six process variables all
# at exactly thre and the last three variables at 2. Other
# than the first and last variables all of the cubists top 10
# are manufacturing process variable.names( WHat is as
# interesting as the importance values and the transitions
# between the variables is the fact that from model to model,
# the only variable that was in the top 10 for all three was
# ManufacturingProcess38, and it was near the bottom of all
# three lists.

# (8.7c)

set.seed(58677)
rpartTune <- train(x_train, y_train, method = "rpart2", tuneLength = 10,
    trControl = trainControl(method = "cv"))
# plot(rpartTune)

best_rpart <- rpart(Yield ~ ., data = trainingData, control = rpart.control(maxdepth = 4))
# decision_plot <- rpart.plot(best_rpart, type = 1, extra =
# 1) Based on this regression tree, the differences between
# process and matrial is that process variables differentiate
# more observations at each break, which is why the break
# first, they increase the purity most quickly. However, the
# final decisions seem to be based rather wholly on
# biological material variables, such that only looking at
# Process or prunning too soon, might lead to overfitting.
```