R Script

```r
## Dependencies
# SOURCE DEFAULT SETTINGS
### UNIVERSAL DATA SOURCING & DEFAULT SETTINGS FOR PROJECT
library(knitr)
library(kableExtra)
library(default)
library(ggplot2)
library(easypackages)
library(formatR)

# Load .rd file
setwd('~/GitHub/CUNY_DATA_624/Homework-Two')
load(file = "hw2.rds")

# SOURCE DEFAULT SETTINGS
source('~/GitHub/CUNY_DATA_624/Homework-Two/defaults.R')
# SOURCE HW ANSWERS
#source('~/GitHub/CUNY_DATA_624/Homework-Two/answers_final.R')
#load(file = "hw2nnet_mod2.rds")
load(file = "hw2.rds")

# SOURCE HW ANSWERS

# DEPENDENCIES
# Predicitve Modeling
libraries('AppliedPredictiveModeling',
          'mice',
          'caret',
          'tidyverse',
          'impute','pls','caTools','mlbench',
          'randomForest','party','gbm','Cubist','rpart')
# Formatting Libraries
libraries('default', 'knitr',
          'kableExtra','gridExtra','sqldf')
# Plotting Libraries
libraries('ggplot2', 'grid',
          'ggfortify','rpart.plot')


# Data Wrangling
library(AppliedPredictiveModeling); library(mice);
library(caret); library(tidyverse); library(pls);
library(caTools); library(mlbench); library(stringr);
# Formatting
library(default); library(knitr); library(kableExtra);
# Plotting
library(ggplot2); library(grid); library(ggfortify)

# THEME COLORS
dark_gold <- "#745010"
medium_gold <- "#cbbda5"
light_gold <- "#dcd3c3"
```

```r
# SET SEED

set.seed(58677)

# ASSIGNMENT 1
# KJ 6.3
data("ChemicalManufacturingProcess")

# (6.3a)
Plt_CMP.Yield <-ggplot(ChemicalManufacturingProcess, aes(x = Yield))+
  geom_histogram(color=light_gold, fill = light_gold)+
  scale_x_continuous(labels = scales::number_format(accuracy = .1))+
  labs(title="Distribution of Yield") +
  theme_bw()+theme(plot.title = element_text(color="#745010",
                                             size=10, face="bold"),
                   axis.title.y = element_blank())

# (6.3b)

# Total NA Values
#na_table<- table(is.na(ChemicalManufacturingProcess))
CMP_na <- ChemicalManufacturingProcess %>% select(-Yield) %>% summarise_all(funs(sum(is.na(.)))) %>% t(
  arrange(desc(V1)) %>% rename(n=V1)
CMP_na_left <-CMP_na %>% slice(1:14);
CMP_na_right <- CMP_na %>% slice(15:28);
CMP.total_na <- cbind(CMP_na_left, ` `=" ", CMP_na_right)

# use mice w/ default settings to impute missing data
miceImp <- mice(ChemicalManufacturingProcess, printFlag = FALSE)

# add imputed data to original data set
CMP_DF <-mice::complete(miceImp)

# (6.3c)

#code
set.seed(58677)   #  set seed to ensure you always have same random numbers generated

sample = sample.split(CMP_DF, SplitRatio = 0.80)
# splits the data in the ratio mentioned in SplitRatio.
# After splitting marks these rows as logical TRUE and the the remaining are marked as logical FALSE

chem_train =subset(CMP_DF,sample ==TRUE) # creates a training dataset named train1
# with rows which are marked as TRUE

chem_test=subset(CMP_DF, sample==FALSE)

#code
pls_model <- plsr(Yield~., data=chem_train,
                  method = 'kernelpls',
                  scale = TRUE,
                  center = TRUE)
```

```r
pls_model2 <- plsr(Yield~., data=chem_train,
                   method = 'kernelpls',
                   scale = TRUE,
                   center = TRUE,
                   ncomp =41)


#  Train Metrics
train_eval=data.frame('obs' = chem_train$Yield, 'pred' =pls_model$fitted.values)
colnames(train_eval) <- c('obs', 'pred')

# (6.3d)
#code
# #Test Predictions & Metrics
pls2_pred<- predict(pls_model2, chem_test, ncomp=41)

pls2test_eval=data.frame('obs' = chem_test$Yield, 'pred' =pls2_pred)

colnames(pls2test_eval) <- c('obs', 'pred')



caret::defaultSummary(pls2test_eval)%>%
  kable(caption="PLS Performance Metrics on Test Subset") %>%
  kable_styling()# %>% row_spec()

eval_plot <- ggplot(pls2test_eval, aes(obs, pred)) +
  labs(title="Observed vs. Predicted Results for Test Data",
       subtitle="Partial Least Squares Model")+
  geom_point()+
  coord_flip()

CMP.sample = sample.split(CMP_DF, SplitRatio = 0.80)
# splits the data in the ratio mentioned in SplitRatio.
# After splitting marks these rows as logical TRUE and the the remaining
# are marked as logical FALSE
CMP.train = subset(CMP_DF, CMP.sample ==TRUE)
# creates a training dataset named train1 with rows which are marked as TRUE
CMP.test = subset(CMP_DF, CMP.sample==FALSE)

# Train model
CMP.pls.fit <- train(Yield~., data=CMP.train, method = 'pls',
                     preProcess=c('zv', 'nzv', 'center', 'scale'),
                     trControl = trainControl(method = "cv", number = 5,
                                              savePredictions = T),
                     tuneLength=10)
CMP.pls.fit.obs_vs_pred <- cbind(Observed = CMP.pls.fit$finalModel$model$.outcome,
                                 Predicted = CMP.pls.fit$finalModel$fitted.values) %>%
  as.data.frame()
Plt_CMP.fit.obs_vs_pred <- ggplot(CMP.pls.fit.obs_vs_pred, aes(Observed, Predicted)) +
  geom_point(color=medium_gold) +
  geom_smooth(method="lm", color=dark_gold, fill=light_gold)+
  scale_x_continuous(labels = scales::number_format(accuracy = 1))+
  scale_y_continuous(labels = scales::number_format(accuracy = 1))+
  labs(title="Train Set: Observed vs. Predicted Values")+
  theme_bw()+
  theme()

#  Train Metrics
CMP.pls.train.perf <- CMP.pls.fit$results %>%
  as.data.frame() %>%
  filter(RMSE==min(RMSE)) %>%
  select(RMSE, Rsquared, MAE)

Plt_CMP.RMSE <- ggplot(CMP.pls.fit) + geom_line(color=dark_gold) + geom_point(color=medium_gold)
+theme_bw()+theme()
+labs(title="PLS Cross-Validation", y="RMSE", x="Components")
+scale_x_continuous(labels = scales::number_format(accuracy = 1))

# (6.3d)
## Test Predictions & Metrics
CMP.pls.pred <- predict(CMP.pls.fit, CMP.test)
CMP.pls.test.perf <- postResample(pred = CMP.pls.pred, obs = CMP.test$Yield)
%>% t() %>% as.data.frame()
CMP.pls.test.obs_vs_pred <- cbind(Predicted = CMP.pls.pred, Observed = CMP.test$Yield)
%>% as.data.frame()
Plt_CMP.test.obs_vs_pred <- ggplot(CMP.pls.test.obs_vs_pred, aes(Observed, Predicted)) +
  geom_point(color=medium_gold) +
  geom_smooth(method="lm", color=dark_gold, fill=light_gold)+
  labs(title="Test Set: Observed vs. Predicted Values")+
  scale_x_continuous(labels = scales::number_format(accuracy = 1))+
  scale_y_continuous(labels = scales::number_format(accuracy = 1))+
  theme_bw()+
  theme(plot.title = element_text(color="#745010", size=10, face="bold"))

# (6.3e)
CMP.pls.imp <- caret::varImp(CMP.pls.fit, scale=T)

CMP.pls.imp.df <- CMP.pls.imp$importance %>% as.data.frame() %>%
  rownames_to_column("Variable")%>%
  arrange(desc(Overall))

Plt_CMP.pls.imp <- CMP.pls.imp.df %>% top_n(15, Overall) %>%
  ggplot(aes(x=reorder(Variable, Overall), y=Overall)) +
  geom_point(colour = medium_gold) +
  geom_segment(aes(x=Variable,xend=Variable,y=0,yend=Overall),colour = dark_gold) +
  labs(title="Variable Importance", subtitle="Top 15 Predictors", x="Variable", y="Scaled Importance")+
  scale_y_continuous(labels = function(x) paste0(x, "%"))+
  coord_flip()+
  theme_bw()+
  theme(axis.title.y = element_blank())

# (6.3f)
# Scatter Plot Comparison
CMP.varImp.top5 <- CMP.pls.imp.df %>%
  top_n(5, Overall)
```

```r
CMP_DF.gather <- CMP_DF %>%
  gather(Variable, Value, -Yield)
Plt_CMP.Scatter <- CMP_DF.gather %>%
  filter(Variable %in% CMP.varImp.top5$Variable) %>%
  ggplot(aes(x=Value, y=Yield)) +
  geom_point(color=medium_gold)+
  geom_smooth(method = "lm", color=dark_gold, fill=light_gold)+
  labs(title="Scatter Plots of Top 5 Predictors Against Yield")+
  facet_wrap(~Variable, scales = "free_x", nrow = 1)+
  theme_bw()+
  theme()

# Correlation
CMP_DF.subset <- CMP_DF[(names(CMP_DF) %in% c(CMP.varImp.top5$Variable, "Yield"))]
CMP_DF.corr <-as.data.frame(as.matrix(cor(CMP_DF.subset)))
CMP_DF.corr.tbl <- CMP_DF.corr %>%
  select(Yield) %>%
  rownames_to_column('Variable') %>%
  filter(Variable!="Yield") %>%
  arrange(desc(Yield))

# ASSIGNMENT 2
# KJ 7.2; KJ 7.5
# The package `mlbench` contains a function called `mlbench.friedman1`
# that simulates these data:
set.seed(200)
trainingData <- mlbench.friedman1(200, sd = 1)
## We convert the 'x' data from a matrix to a data frame
## One reason is that this will give the columns names.
trainingData$x <- data.frame(trainingData$x)
## Look at the data using
#featurePlot(trainingData$x, trainingData$y)
## or other methods.
## This creates a list with a vector 'y' and a matrix
## of predictors 'x'. Also simulate a large test set to
## estimate the true error rate with good precision:
testData <- mlbench.friedman1(5000, sd = 1)
testData$x <- data.frame(testData$x)


#over ride set seed from sample data simulation
set.seed(58677)

#KNN provided by literature
knnModel <- train(x = trainingData$x,
                  y = trainingData$y,
                  method = "knn",
                  preProc = c("center", "scale"),
                  tuneLength = 10)
knnModel
knnPred <- predict(knnModel, newdata = testData$x)
## The function 'postResample' can be used to get the test set performance values
postResample(pred = knnPred, obs = testData$y)
```

```r
# instructions from text
set.seed(200)
trainingData <- mlbench.friedman1(200, sd = 1)
trainingData$x <- data.frame(trainingData$x)
testData <- mlbench.friedman1(5000, sd = 1)
testData$x <- data.frame(testData$x)
#featurePlot(trainingData$x, trainingData$y)

# created ggplot instead of featurePlot()

Sim.featurePlot <- trainingData %>%
  as.data.frame() %>%
  gather(x, value, -y) %>%
  mutate(x = str_replace(x, "x.","")) %>%
  arrange(desc(x)) %>%
  mutate(x = as.factor(x))
Sim.featurePlot$x <- factor(Sim.featurePlot$x,
                            levels = c("X1","X2", "X3", "X4",
                                       "X5", "X6", "X7", "X8", "X9","X10"))
Plt_Sim.featurePlot <- ggplot(Sim.featurePlot, aes(value, y)) +
  geom_point(color=dark_gold, alpha=.5)+
  facet_wrap(~ x, nrow=2, scales = "fixed")+
  theme_bw()+theme()+
  labs(title="XY Scatter Plots of Simulated Data")
# revert seed back to our set group number:
set.seed(58677)

# (7.2a)
#mars
marsGrid <- expand.grid(degree =1:2, nprune=seq(2,14,by=2))
hw2mars_mod <- train(x = trainingData$x,
                     y = trainingData$y,
                     method='earth',
                     tuneGrid = marsGrid,
                     trControl = trainControl(method = "cv"))

hw2mars_modplot<- ggplot(hw2mars_mod)+theme_bw()+
  theme()+
  labs(title="MARS Cross-Validated RMSE Profile")

#svm
hw2svm_mod <- train(x = trainingData$x,
                    y = trainingData$y,
                    method='svmRadial',
                    tuneLength = 14,
                    trControl = trainControl(method = "cv"))
#hw2svm_mod$finalModel
hw2svm_modplot<- ggplot(hw2svm_mod)+theme_bw()+
  theme()+
  labs(title="SVM Cross-Validated RMSE Profile")


##nnet (Taken from Andy)
```

```r
# hyperparameter tuning for nnet
nnetGrid <- expand.grid(.size = c(1:10), .decay = c(0, 0.01, .1))

hw2nnet_mod <- train(trainingData$x, trainingData$y,method = "nnet", tuneGrid = nnetGrid,trControl = tra
                     ## Automatically standardize data prior to modeling and prediction
                     preProc = c("center", "scale"),
                     linout = TRUE,
                     trace = FALSE,
                     MaxNWts = 10 * (ncol(trainingData$x) + 1)  + 10 +  1,
                     maxit = 500)

hw2nnet_modplot<- ggplot(hw2nnet_mod)+theme_bw()+
  theme()+labs(title="NNET Cross-Validated RMSE Profile")


# (7.2b)
#knn pred given to us
hw2marsPred <- predict(hw2mars_mod, newdata = testData$x)
hw2svmPred <- predict(hw2svm_mod, newdata = testData$x)
hw2nnetPred <- predict(hw2nnet_mod, newdata = testData$x)

knn_performance <- postResample(pred = knnPred, obs = testData$y)
hw2marsPerf <- postResample(pred = hw2marsPred, obs = testData$y)
hw2svmPerf <- postResample(pred = hw2svmPred, obs = testData$y)
hw2nnetPerf <- postResample(pred = hw2nnetPred, obs = testData$y)

hw2.2.bperformance_table <- rbind("knnTrain"=c("RMSE"=max(knnModel$results$RMSE),
                                               "RSquared"=max(knnModel$results$RMSE),
                                               "MAE"=max(knnModel$results$RMSE)),
                                  "knnTest"=knn_performance,
                                  "MARSTrain"=c("RMSE"=max(hw2mars_mod$results$RMSE),
                                                "RSquared"=max(hw2mars_mod$results$Rsquared),
                                                "MAE"=max(hw2mars_mod$results$MAE)),
                                  "MARSTest"=hw2marsPerf,
                                  "SVMTrain"=c(max(hw2svm_mod$results$RMSE),
                                               max(hw2svm_mod$results$Rsquared),
                                               max(hw2svm_mod$results$MAE)),
                                  "SVMTest"=hw2svmPerf,
                                  "NNETTrain"=c(max(hw2nnet_mod$results$RMSE),
                                                max(hw2nnet_mod$results$Rsquared),
                                                max(hw2nnet_mod$results$MAE)),
                                  "NNETTest"=hw2nnetPerf) %>% kable(caption="Model Performance", digits=

#hw2marsImp <- varImp(hw2mars_mod)
#hw2marsImptbl <- hw2marsImp$importance %>%
##kable(caption="MARS Model - Variable Importance", digits=2) %>% kable_styling()

hw2marsImp <- caret::varImp(hw2mars_mod)
#hw2marsImp<-hw2marsImp%>%
#    mutate(Variable = row.names(hw2marsImp))%>%
#    remove_rownames()%>%
#    select(Variable, Overall)%>%
#    arrange(desc(Overall))
```

```r
hw2marsImp<-hw2marsImp$importance %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  arrange(Overall) %>%
  mutate(rowname = forcats::fct_inorder(rowname ))

hwmarsimp_plot <- ggplot(head(hw2marsImp, 15), aes(x=reorder(rowname, Overall), y=Overall)) +
  geom_point(colour = 'violetred4') +
  geom_segment(aes(x=rowname,xend=rowname,y=0,yend=Overall),colour = 'violetred4') +
  labs(title="Variable Importance",
       subtitle="MARS for Simulated Data Set", x="Variable", y="Importance")+
  coord_flip()+
  theme_bw()+
  theme()

# (7.5a)
##knn
hw2knn_mod2 <- train(Yield~.,
                     data=chem_train,
                     method = "knn",
                     preProc = c("center", "scale"),
                     tuneLength = 10)

#nnet
nnetGrid_75 <- expand.grid(.decay = c(0, 0.01, .1),
                           .size = c(1:10),
                           .bag = FALSE)

hw2nnet_mod2 <- train(Yield~.,
                      data=chem_train,
                      method = "avNNet",
                      tuneGrid = nnetGrid_75,
                      preProc = c("center", "scale"),
                      linout = TRUE,
                      trace = FALSE,
                      MaxNWts = 10 * (ncol(chem_train) + 1) + 5 + 1,
                      maxit = 500)

#MARS
# Define the candidate models to test
marsGrid_75 <- expand.grid(.degree = 1:2, .nprune = 2:38)

hw2mars_mod2 <- train(Yield~.,
                      data=chem_train,
                      method = "earth",
                      tuneGrid = marsGrid_75,
                      trControl = trainControl(method = "cv"))

#SVM
hw2svm_mod2 <- train(Yield~.,
                     data=chem_train,
                     method = "svmRadial",
                     preProc = c("center", "scale"),
                     tuneLength = 14,
                     trControl = trainControl(method = "cv"))


#model performances
#knn pred given to us
hw2knnPred2 <- predict(hw2knn_mod2, newdata = chem_test)
hw2nnetPred2 <- predict(hw2nnet_mod2, newdata = chem_test)
hw2marsPred2 <- predict(hw2mars_mod2, newdata = chem_test)
hw2svmPred2 <- predict(hw2svm_mod2, newdata = chem_test)

hw2knnPerf2 <- postResample(pred = hw2knnPred2, obs = chem_test$Yield)
hw2marsPerf2 <- postResample(pred = hw2marsPred2, obs = chem_test$Yield)
hw2svmPerf2 <- postResample(pred = hw2svmPred2, obs = chem_test$Yield)
hw2nnetPerf2 <- postResample(pred = hw2nnetPred2, obs = chem_test$Yield)

hw2.2.cperformance_table <- rbind("knnTrain"=c("RMSE"=max(hw2knn_mod2$results$RMSE),
                                               "RSquared"=max(hw2knn_mod2$results$Rsquared),
                                               "MAE"=max(hw2knn_mod2$results$MAE)),
                                  "knnTest"=hw2knnPerf2,
                                  "MARSTrain"=c("RMSE"=max(hw2mars_mod2$results$RMSE),
                                                "RSquared"=max(hw2mars_mod2$results$Rsquared),
                                                "MAE"=max(hw2mars_mod2$results$MAE)),
                                  "MARSTest"=hw2marsPerf2,
                                  "SVMTrain"=c(max(hw2svm_mod2$results$RMSE),
                                               max(hw2svm_mod2$results$Rsquared),
                                               max(hw2svm_mod2$results$MAE)),
                                  "SVMTest"=hw2svmPerf2,
                                  "NNETTrain"=c(max(hw2nnet_mod2$results$RMSE),
                                                max(hw2nnet_mod2$results$Rsquared),
                                                max(hw2nnet_mod2$results$MAE)),
                                  "NNETTest"=hw2nnetPerf2) %>% kable(caption="Model Performance on Chem

# (7.5b)
#hw2svmImp2 <- varImp(hw2svm_mod2)
#hw2svmImptbl2 <- hw2svmImp2$importance %>% kable(caption="SVM Model - Variable Importance", digits=2)

hw2svmImp2 <- caret::varImp(hw2svm_mod2)

hw2svmImp2<-hw2svmImp2$importance %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  arrange(Overall) %>%
  mutate(rowname = forcats::fct_inorder(rowname ))

hwsvmimp_plot2 <- ggplot(head(hw2svmImp2, 15), aes(x=reorder(rowname, Overall), y=Overall)) +
  geom_point(colour = light_gold) +
  geom_segment(aes(x=rowname,xend=rowname,y=0,yend=Overall),colour = light_gold) +
  labs(title="Variable Importance",
       subtitle="SVM Model Importance for ChemicalManufacturing Data", x="Variable", y="Importance")+
  coord_flip()+
  theme_bw()+
  theme()
```

```
# (7.5c)
#alterate apprach (use plot importance to identify top few important features)
hw2imp <- CMP_DF %>%select(Yield,
                           ManufacturingProcess14,
                           ManufacturingProcess02,
                           ManufacturingProcess03,
                           ManufacturingProcess38,
                           ManufacturingProcess37 )

hw2cor_pre_df<- as.data.frame(as.matrix(cor(hw2imp)))

hw2cor_df<-tibble::rownames_to_column(hw2cor_pre_df, "VALUE")

hw2cor_df2<-sqldf("select VALUE, Yield from hw2cor_df order by Yield desc")%>%
  kable(caption="Correlation") %>%
  kable_styling()


# ASSIGNMENT 3
# KJ 8.1-8.3; KJ 8.7

# (8.1a)
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"

# revert seed back to our set group number:
set.seed(58677)

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

rfImp1 <- varImp(model1, scale = FALSE)

rfImp1.df <-tibble::rownames_to_column(as.data.frame(as.matrix(varImp(model1, scale = FALSE))), "VALUE"

#colnames(hw3_rfdt3) <- c("Value","Importance")

rfImp1plot<-ggplot(rfImp1.df, aes(x=reorder(VALUE, Overall), y=Overall)) +
  geom_point(color ='darkorange') +
  geom_segment(aes(x=VALUE,xend=VALUE,y=0,yend=Overall, color = 'darkorange')) +
  labs(title="Variable Importance", subtitle="Random Forest Model for Simulated Data", x="", y="Importa

# (8.1b)
simulated$duplicate1 <- simulated$V1 + rnorm(200) * .1

hw3model2 <- randomForest(y ~ ., data = simulated,
                          importance = TRUE,
                          ntree = 1000)
rfImp2 <- varImp(hw3model2, scale = FALSE)
```

```
rfImp2.df <-tibble::rownames_to_column(as.data.frame(as.matrix(varImp(hw3model2, scale = FALSE))), "VALU

rfImp2plot<-ggplot(rfImp2.df, aes(x=reorder(VALUE, Overall), y=Overall)) +
  geom_point(color ='darkorange') +
  geom_segment(aes(x=VALUE,xend=VALUE,y=0,yend=Overall, color = 'darkorange')) +
  labs(title="Variable Importance", subtitle="Random Forest Model (with duplicate 1) for Simulated Data



#hw3rf2_imp_table<- rfImp2%>% kable(caption="Random Forest Variable Importance on Simulated Dataset") %


# (8.1c)
#hw3_rfmodel3 <- cforest(y ~ ., data=simulated)
#rfdt3 <-as.data.frame(as.matrix(varimp(rf_mod3)))
#hw3_rfdt3 <-tibble::rownames_to_column(as.data.frame(as.matrix(varimp(hw3_rfmodel3))), "VALUE")    #as

#colnames(hw3_rfdt3) <- c("Value","Importance")

#hw3_rfdt3a <- hw3_rfdt3[order(-hw3_rfdt3$Importance),]

#hw3_rfdt3.tbl <- hw3_rfdt3a  %>% kable(caption="Unconditional CForest Model: Variable Importance") %>%

#hw3_rfdt3b <-  tibble::rownames_to_column(as.data.frame(as.matrix(varimp(hw3_rfmodel3,conditional=T)))

#colnames(hw3_rfdt3b) <- c("Value","Importance")

#hw3_rfdt3ba <- hw3_rfdt3b[order(-hw3_rfdt3b$Importance),]

#hw3_rfdt3b.tbl<-hw3_rfdt3ba  %>%  kable(caption="Conditional CForest Model: Variable Importance") %>%


# Now remove correlated predictor
simulated$duplicate1 <- NULL
bagCtrl <- cforest_control(mtry = ncol(simulated) - 1)
baggedTree <- party::cforest(y ~ ., data = simulated, controls = bagCtrl)
cfImp <- party::varimp(baggedTree, conditional = T)
#cfImp <- kable(sort(cfImp, decreasing = TRUE))
cfImp1 <- party::varimp(baggedTree, conditional = F)
#cfImp1 <- kable(sort(cfImp1, decreasing = TRUE))
# Keep correlated predictor
simulated$duplicate1 <- simulated$V1 + rnorm(200) * .1
bagCtrl <- cforest_control(mtry = ncol(simulated) - 1)
baggedTree <- party::cforest(y ~ ., data = simulated, controls = bagCtrl)
cfImp2 <- party::varimp(baggedTree, conditional = T)
#cfImp2 <- kable(sort(cfImp2, decreasing = TRUE))
cfImp22 <- party::varimp(baggedTree, conditional = F)
#cfImp22 <- kable(sort(cfImp22, decreasing = TRUE))
simulated$duplicate1 <- NULL

a <- data.frame(features = rownames(rfImp1), RF = rfImp1[,1])
b <- data.frame(features = rownames(rfImp2), RF.cor = rfImp2[,1])
c <- data.frame(features = names(cfImp), CF.cond = cfImp)
```

```r
d <- data.frame(features = names(cfImp1), CF = cfImp1)
e <- data.frame(features = names(cfImp2), CF.cor.cond = cfImp2)
f <- data.frame(features = names(cfImp22), CF.cor = cfImp22)
aa <- merge(a,d, all=T)
bb <- merge(b,f,all=T)
cc <- merge(c,e,all=T)
dd <-merge(aa,bb,all=T)
hw3final_df <- merge(dd,cc,all=T)
hw3final_df <- rbind(hw3final_df[-3,], hw3final_df[3,])
rownames(hw3final_df) <- c(1:11)

hw3final_dfb<-hw3final_df%>%  kable(caption="Conditional vs Unconditional CForest Model: Variable Impor


# (8.1d)

#GBM
gbmModel_nodup <- gbm(y ~ ., data = simulated, distribution = "gaussian", n.trees=1000)

#gbmModel_nodupb <-caret::varImp(gbmModel_nodup)    #as.data.frame(as.matrix(varimp(hw3_rfmodel3)))


simulated$duplicate1 <- simulated$V1 + rnorm(200) * .1
gbmModel_wdup <- gbm(y ~ ., data = simulated, distribution = "gaussian", n.trees=1000)

#gbmModel_wdupb <-varImp(gbmModel_wdup)   #as.data.frame(as.matrix(varimp(hw3_rfmodel3)))


simulated$duplicate1 <- NULL

#Cubist
cubistMod_nodup <- cubist(simulated[-11], simulated$y, committees = 100)

cubistMod_nodupb <-varImp(cubistMod_nodup)

cubistMod_nodupb.df <-
  tibble::rownames_to_column(as.data.frame(as.matrix(varImp(cubistMod_nodup))), "VALUE") #as.data.frame

cubistMod_nodupbplot<-ggplot(cubistMod_nodupb.df,
                             aes(x=reorder(VALUE, Overall),y=Overall)) +
  geom_point(color ='darkorange') +
  geom_segment(aes(x=VALUE,
                   xend=VALUE,
                   y=0,
                   yend=Overall,
                   color = 'darkorange')) +
  labs(title="Variable Importance",
       subtitle="Cubist Model without Duplicate",
       x="",
       y="Importance")+coord_flip()+theme_bw()+theme(legend.position = "none")
```

```r
simulated$duplicate1 <- simulated$V1 + rnorm(200) * .1
cubistMod_wdup <- cubist(simulated[-11], simulated$y, committees = 100)

cubistMod_wdupb <-varImp(cubistMod_wdup)

cubistMod_wdupb.df <-tibble::rownames_to_column(as.data.frame(as.matrix(varImp(cubistMod_wdup))), "VALU

cubistMod_wdupbplot<-ggplot(cubistMod_wdupb.df,
                            aes(x=reorder(VALUE, Overall),
                                y=Overall)) +
  geom_point(color ='darkorange') +
  geom_segment(aes(x=VALUE,xend=VALUE,y=0,yend=Overall, color = 'darkorange')) +
  labs(title="Variable Importance", subtitle="Cubist Model With Duplicate", x="", y="Importance")+ coor



simulated$duplicate1 <- NULL



# (8.2)

random_predictor <- data.frame(V1=sample(1:2, 100, replace=TRUE),
                               V2=sample(1:100, 100, replace=TRUE),
                               V3=sample(1:1000, 100, replace=TRUE),
                               V4=sample(1:5000, 100, replace=TRUE))
sim_df <- random_predictor %>% mutate(y=V1*V2*V3+rnorm(100))
sim_rf <- randomForest(y ~ ., data = sim_df, importance = TRUE, ntree = 1000)
sim_varImp <- varImp(sim_rf, scale=T)

sim_varImp.df <-tibble::rownames_to_column(as.data.frame(as.matrix(varImp(sim_rf))),
                                           "VALUE")    #as.data.frame(as.matrix(varimp(hw3_rfmodel3)))

sim_varImp.plot<-ggplot(cubistMod_wdupb.df, aes(x=reorder(VALUE, Overall), y=Overall)) +
  geom_point(color ='darkorange') +
  geom_segment(aes(x=VALUE,xend=VALUE,y=0,yend=Overall, color = 'darkorange')) +
  labs(title="Variable Importance",
       subtitle="Simulated Importance",
       x="",
       y="Importance")+ coord_flip()+theme_bw()+theme(legend.position = "none")


# (8.3a)

#No code needed for this problem

# (8.3b)

#No code needed for this problem

# (8.3c)

#No code needed for this problem
```

```r
# (8.7a)
#GBM
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(Yield~.,
                    data=chem_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 = chem_test)
gb_test_87 <- postResample(pred = gbmPred_87, obs = chem_test$Yield)

# Random Forest

rf_87_grid <- expand.grid(mtry= seq(100, 1000, by=50))

rfTune_87 <-  train(Yield~.,
                    data=chem_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 = chem_test)
rf_test_87 <- postResample(pred = rfPred_87, obs = chem_test$Yield)
rf_87<-randomForest(Yield~.,
                    data=chem_train,
                    importance = TRUE,
                    ntree = 900)
#  Cubist
set.seed(58677)
cb_grid_87 <- expand.grid(committees = c(25:45), neighbors = c(1, 3, 5 ))
cbTune_87 <- train(Yield~.,
                   data=chem_train,
                   method = "cubist",
                   metric="RMSE",
                   na.action = na.pass,
                   tuneGrid = cb_grid_87,
                   trControl = trainControl(method = 'cv'))
#plot(cbTune_87)
#min(cbTune_87$results$RMSE, na.rm = TRUE)
cbPred_87 <- predict(cbTune_87, newdata = chem_test)
cb_test_87 <- postResample(pred = cbPred_87, obs = chem_test$Yield)
```

```r
hw2.3.dperformance_table <- rbind("GBM"=c("RMSE"=max(gbmTune_87$results$RMSE),
                                          "RSquared"=max(gbmTune_87$results$Rsquared),
                                          "MAE"=max(gbmTune_87$results$MAE)),
                                  "GBMTest"=gb_test_87,

                                  "RFTrain"=c("RMSE"=max(rfTune_87$results$RMSE),
                                              "RSquared"=max(rfTune_87$results$Rsquared),
                                              "MAE"=max(rfTune_87$results$MAE)),
                                  "RFTest"=rf_test_87,

                                  "CubistTrain"=c(max(cbTune_87$results$RMSE),
                                                  max(cbTune_87$results$Rsquared),
                                                  max(cbTune_87$results$MAE)),
                                  "CubistTest"=cb_test_87) %>% kable(caption="Tree Model Performance on
  kable_styling() %>%
  row_spec() %>% row_spec(row=5:6, background ="#d9f2e6")



# (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()


# (8.7c)

rpartTune <- train(Yield~.,
                   data=chem_train,
                   method = "rpart2",
                   tuneLength = 10,
                   trControl = trainControl(method = "cv"))
#plot(rpartTune)
best_rpart <- rpart(Yield~., data =chem_train,
                    control = rpart.control(maxdepth = 4))
#decision_plot <- rpart.plot(best_rpart,
# type = 1,
# extra = 1)


## Libraries for .rmd file

# Predicitve Modeling
libraries('AppliedPredictiveModeling', 'mice','caret', 'tidyverse','impute','pls','caTools','mlbench','

# Formatting Libraries
libraries('default', 'knitr', 'kableExtra','gridExtra','sqldf','tibble')

# Plotting Libraries
libraries('ggplot2', 'grid', 'ggfortify','rpart.plot')

# Data Wrangling
library(AppliedPredictiveModeling); library(mice); library(caret); library(tidyverse); library(pls); li
```

```
# Formatting
library(default); library(knitr); library(kableExtra);

# Plotting
library(ggplot2); library(grid); library(ggfortify); library(gridExtra)


# Assignment 1
## kj-6.3a
data("ChemicalManufacturingProcess")

## kj-6.3a-plot
Plt_CMP.Yield

## kj-6.3b
CMP.total_na %>% kable(caption="Variables with Missing Values", booktabs=T) %>%  kable_styling(full_wid
  column_spec(3, bold = T, border_left = F, border_right = F, width = '0.1cm') %>%
  row_spec()

## kj-6.3c
CMP.pls.train.perf %>%
  kable(caption="PLS Performance Metrics on Training Subset", booktabs=T) %>%
  kable_styling() %>%
  row_spec()

## kj-6.3c2
grid.arrange(Plt_CMP.RMSE, Plt_CMP.fit.obs_vs_pred, ncol=2)

## kj-6.3d-1
CMP.pls.test.perf %>%
  kable(caption="PLS Performance Metrics on Test Subset", booktabs=T) %>%
  kable_styling() %>%
  row_spec()

## kj-6.3d-2

## kj-6.3e
Plt_CMP.pls.imp

## kj-6.3f-1
Plt_CMP.Scatter

## kj-6.3f-2
CMP_DF.corr.tbl %>%
  kable(caption="Variable Correlation with Yield", booktabs=T) %>%
  kable_styling(full_width = F) %>%
  row_spec()


# Assignment 2
## kj-7.2-ex1
set.seed(200)
trainingData <- mlbench.friedman1(200, sd = 1)
```

```
## We convert the 'x' data from a matrix to a data frame
## One reason is that this will give the  columns names.
trainingData$x <- data.frame(trainingData$x)
testData <- mlbench.friedman1(5000, sd = 1)
testData$x <- data.frame(testData$x)

## kj-7.2-ex3, echo=F
Plt_Sim.featurePlot

## kj-7.2-ex4, echo=T, eval=F
knnModel <- train(x = trainingData$x,
                  y = trainingData$y,
                  method = "knn",
                  preProc = c("center", "scale"),
                  tuneLength = 10)
knnModel
knnPred <- predict(knnModel, newdata = testData$x)
postResample(pred = knnPred, obs = testData$y)

###Model 1-MARS Regression:
## kj-7.2-1
hw2mars_modplot

###Model 2 SVM:
## kj-7.2-2
hw2svm_modplot

###Model 3 NNET:
## kj-7.2-3
hw2nnet_modplot

## kj-7.2-4
hw2.2.bperformance_table
#%>% kable(caption="Model Performance", digits=4)
#%>% kable_styling()
#%>% row_spec() %>% row_spec(row=3:4,background ="#d9f2e6")

## kj-7.2-4b
hwmarsimp_plot

## kj-7.5a
hw2.2.cperformance_table

## kj-7.5b
hwsvmimp_plot2

## kj-7.5c
hw2cor_df2


# Assignment 3
## kj-8.1
set.seed(200)
```

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

## kj-8.1a
hw3model1 <- randomForest(y ~ ., data = simulated,
                          importance = TRUE,
                          ntree = 1000)
rfImp1 <- varImp(model1, scale = FALSE)
rfImp1plot

## kj-8.1b-ex
simulated$duplicate1 <- simulated$V1 + rnorm(200) * .1

## kj-8.1b
#code
rfImp2plot

## kj-8.1c
#code
hw3final_dfb

## kj-8.1d
summary(gbmModel_nodup)

## kj-8.1da
summary(gbmModel_wdup)

## kj-8.1db
cubistMod_nodupbplot;cubistMod_wdupbplot

## kj-8.2
sim_varImp.plot

## kj-8.7a
hw2.3.dperformance_table

## kj-8.7b
grid.arrange(boost, random, cube, ncol=3)

##kj-8.7c
rpart.plot(best_rpart,
           type = 1,
           extra = 1)
```