Jake Kruse

Geog 560

Lab 4

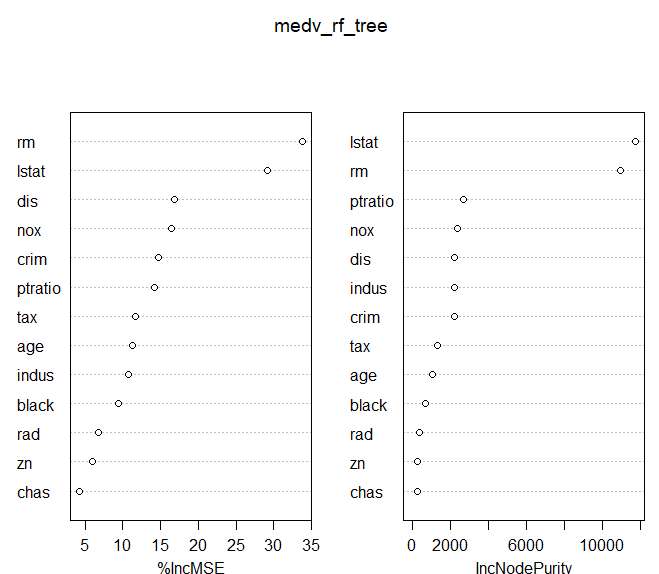
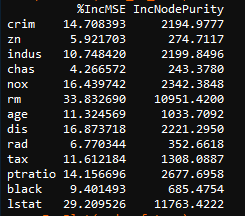
**(1) (2 pts) Apply the Random Forest (RF) Regression method with crossvalidation approach for all predictors in Boston Data to predict the “medv”**

**median value of owner-occupied homes in $1000s. From the importance()**

**table, which factors do you think are most important regarding the mean**

**decrease of accuracy in predictions?**

460 was the number for ntrees that produced the most accurate model, and the importance table below was generated from that model. The number of rooms per dwelling (rm variable) was the important variable in housing prices, and the second most important variable was dis (distance to employment centers).

Using 10-fold Cross Validation, 5 variables were able to best reduce cv-error while also reducing unnecessary variables. As such: nox, rm, dis, ptratio/crim, and lstat should be the variables used in the model (based on %IncMSE); while crim and ptratio had similar $IncMSE, the increase in node purity was greater for ptratio, so either or both could be included.



**(2) (3 pts) Based on the mean sum of square errors, compare the RF regression**

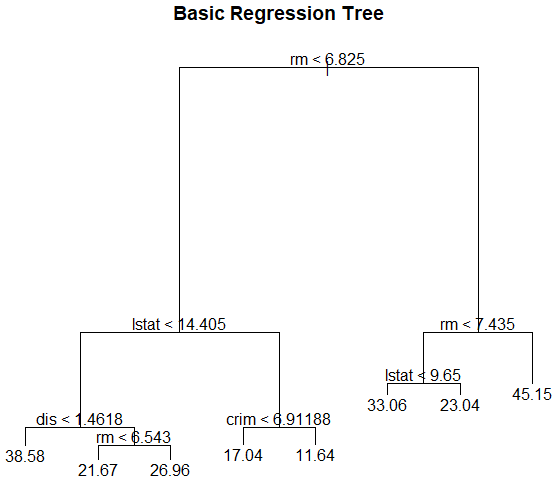
**model performance with (a) the basic Regression Tree using recursive binary**

**splitting; and (b) the Pruning Tree with the optimal tree node complexity.**

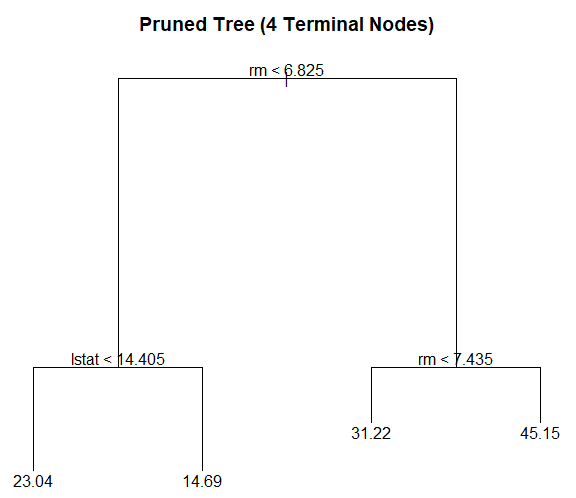
**Which one is the best?**

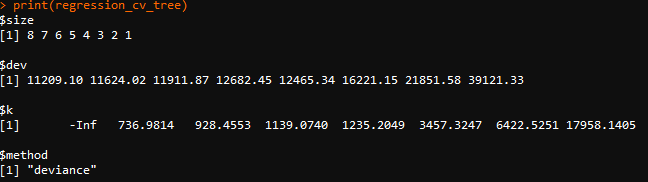
The MSE for the best RF predication above (with 460 trees) was 7.29.

The MSE for the basic Regression Tree (optimal tree node complexity) predication was 15.66.



The MSE for the Pruned Tree (recursive binary) predication was 35.00. This used an optimal terminal node size of 4, as 4 nodes achieved the majority of the reduction in deviance. Having 8 terminal nodes (no pruning) resulted in a lower MSE, but is not as simple of model.



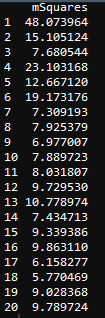
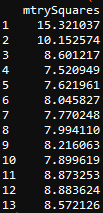


**(3) (3 pts) When changing the number of predictors (“mtry”) in each subtree or**

**changing the number of trees (“ntree”) in a forest, what are the trends for**

**the changes of the mean sum of square errors in predictions?**

Increasing the number of trees (ntree, left) generally reduced MSE until 610 trees, after which some tree combinations reduced MSE and some increased MSE. Increasing the number of predictors (mtry, right), in a forest with 460 trees, reduced MSE the most with 4 predictors, and afterwards increased MSE.

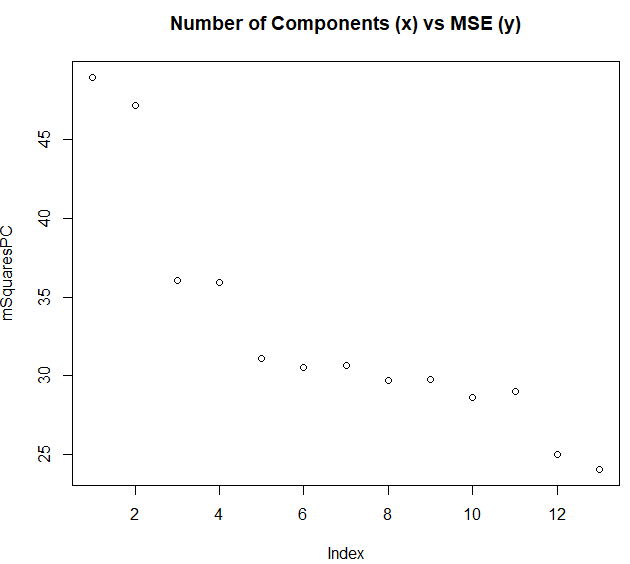
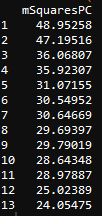
 

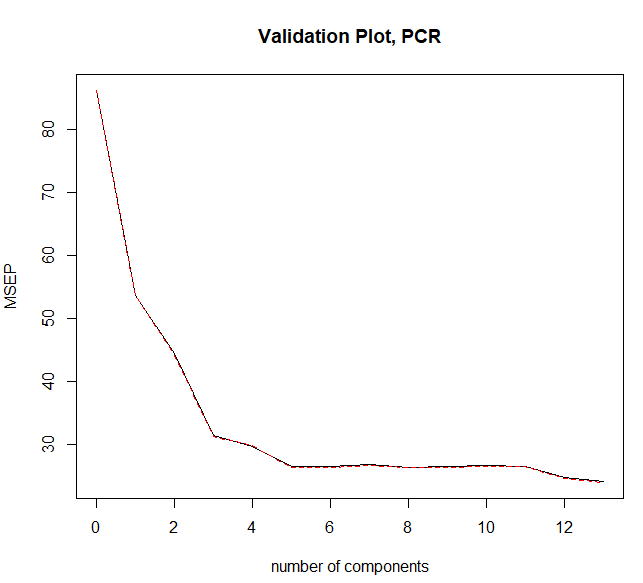
**(4) (2 pts) Try to include different number (parameter name: “ncomp”) of**

**principle components in regression pcr(), what is the trend for the changes**

**of the mean sum of square errors in predictions? Why?**

MSE continually decreases as the number of principal component in regression increases, however 5 principal components do the majority of reduction of the MSE (in the black box on the left, mSquaresPC is the MSE, and the numbers correspond to the ncomp values used). The continual reduction of MSE means that all the variables help explain the data (rather than being unrelated or irrelevant).





**#1**

**#RF w CV to predict medv**

**https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/**

**#rf.crossValidation**

**https://rdrr.io/cran/rfUtilities/man/rf.crossValidation.html**

**library("MASS")**

**data("Boston")**

**# Random Forest for regression, not classifcation**

**## add your code**

**#Install via Packages tab**

**if (!require("tree")) install.packages("tree")**

**library(tree)**

**##load randomForest via packages tabs**

**if (!require("randomForest")) install.packages("randomForest")**

**library("randomForest")**

**library(caret)**

**if (!require("caret")) install.packages("caret")**

**library(ggplot2)**

**if (!require("gdals")) install.packages("gdals")**

**library("gdals")**

**if (!require("rgdal")) install.packages("rgdal")**

**library(rgdal)**

**#training and test data**

**#get dimensions of data**

**dsobs <- dim(Boston)[1]**

**dsfeats <- dim(Boston)[2]**

**head(Boston)**

**#Seperate Training and Testing datasets**

**sample\_index <- sample(dsobs, dsobs\*0.9) # randomly generate the row index for 90% training dataset (the size ccould vary)**

**training\_data<- Boston[sample\_index,] # create the training dataset that contains 90% of the whole data**

**testing\_data <- Boston[-sample\_index,] # get the testing dataset that contains 10% of the whole data**

**# Growing an RF model for predicting the median value of owner-occupied homes**

**mSquares <- NULL**

**for (i in seq(1,20)){**

**ntree = 50\*i + 10**

**medv\_rf\_tree <- randomForest(medv ~ .-medv, data=training\_data, mtry=4, importance =TRUE, ntree=i)**

**regression\_tree\_pred = predict(medv\_rf\_tree,testing\_data)**

**mSquares[i] <- mean((regression\_tree\_pred-testing\_data$medv)^2)**

**}**

**data.frame(mSquares)**

**#Number of trees i=9 (660 trees total) produced the best model (mSquares=lowest at i=7)**

**medv\_rf\_tree <- randomForest(medv ~ .-medv, data=training\_data, mtry=4, importance =TRUE, ntree=460)**

**regression\_tree\_pred = predict(medv\_rf\_tree,testing\_data)**

**summary(medv\_rf\_tree)**

**train2<- data.frame(training\_data)**

**train2$medv <- NULL**

**head(train2)**

**rf\_cv <- rfcv(train2,training\_data$medv, cv.fold = 10,scale="fixed", step=-1)**

**rf\_cv**

**plot(rf\_cv)**

**importance(medv\_rf\_tree)**

**varImpPlot(medv\_rf\_tree)**

**##2**

**regression\_tree =tree(medv ~ .-medv , training\_data, split='deviance') # train a tree model**

**summary(regression\_tree)**

**plot(regression\_tree) #plot the decision tree**

**text(regression\_tree, pretty =0) #add the label of tree nodes**

**title("Basic Regression Tree")**

**# make the prediction on testing data**

**regression\_tree\_pred = predict(regression\_tree,testing\_data)**

**plot(regression\_tree\_pred, testing\_data$medv) #scatter plot of the prediction values and the testing values**

**title("Scatter Plot of the Prediction and Testing Values\n Basic Regression Tree")**

**# calculate the mean sum of square errors**

**mean((regression\_tree\_pred-testing\_data$medv)^2)**

**#K-folds Cross-validation for Choosing Tree Complexity**

**regression\_cv\_tree =cv.tree(regression\_tree,FUN=prune.tree)**

**print(regression\_cv\_tree)**

**plot(regression\_cv\_tree$size, regression\_cv\_tree$dev, type='b')**

**title("Tree Size vs Error")**

**#Pruning tree with the optimal terminal node size for the tree that minimizes the deviance**

**regression\_prune\_tree =prune.tree(regression\_tree, best=4) #change the 'best' node value if needed**

**plot(regression\_prune\_tree)**

**text(regression\_prune\_tree ,pretty =0)**

**title("Pruned Tree (4 Terminal Nodes)")**

**# make the prediction and calculate the mean sum of square errors**

**regression\_tree\_prune\_pred = predict(regression\_prune\_tree,testing\_data)**

**plot(regression\_tree\_prune\_pred, testing\_data$medv) #scatter plot of the prediction values and the testing values**

**title("Scatter Plot of the Prediction and Testing Values\n Basic Regression Tree")**

**# calculate the mean sum of square errors**

**mean((regression\_tree\_prune\_pred-testing\_data$medv)^2)**

**###3**

**mtrySquares <- NULL**

**for (i in seq(1,13)){**

**mtry=i**

**medv\_rf\_tree <- randomForest(medv ~ .-medv, data=training\_data, mtry=i, importance =TRUE, ntree=460)**

**regression\_tree\_pred = predict(medv\_rf\_tree,testing\_data)**

**mtrySquares[i] <- mean((regression\_tree\_pred-testing\_data$medv)^2)**

**}**

**data.frame(mtrySquares)**

**#4**

**# Principal Components Regression is similar to that for lm(), with a few additional options.**

**install.packages("pls")**

**library (pls)**

**help(pcr)**

**# train a PCRegression model**

**pcr.fit <- pcr(medv ~ ., data=training\_data ,scale=TRUE, validation ="CV")**

**summary(pcr.fit)**

**#We now perform PCR on the training data and evaluate its test set performance.**

**pcr.pred=predict (pcr.fit, testing\_data, ncomp = 5)**

**#mean sum of squared error**

**mean((pcr.pred - testing\_data$medv)^2)**

**mSquaresPC <- NULL**

**for (i in seq(1,13)){**

**ncomp = i**

**pcr.fit <- pcr(medv ~ ., data=training\_data ,scale=TRUE, validation ="CV")**

**pcr\_pred = predict(pcr.fit,testing\_data, ncomp=i)**

**mSquaresPC[i] <- mean((pcr\_pred-testing\_data$medv)^2)**

**}**

**data.frame(mSquaresPC)**

**plot(mSquaresPC)**

**title("Number of Components (x) vs MSE (y)")**

**validationplot(pcr.fit, val.type="MSEP", main="Validation Plot, PCR")**

**##^MSE of the different mdoels**