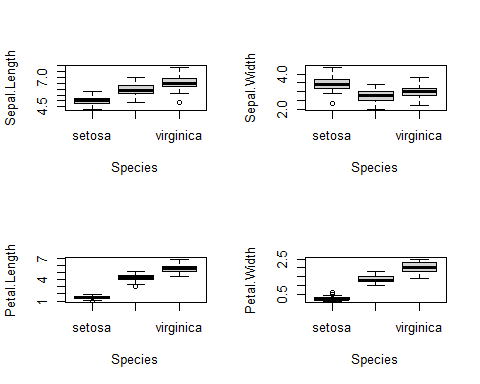
STAT318Assignment3

## Question 1

There are 150 rows and 5 columns/variables. The variables are Sepal.length, Sepal.Width, Petal.Length, Petal.Width and Species.

# b)  
# Group by species  
par(mfrow=c(2,2))  
boxplot(Sepal.Length~Species, data = iris)  
boxplot(Sepal.Width~Species, data = iris)  
boxplot(Petal.Length~Species, data = iris)  
boxplot(Petal.Width~Species, data = iris)

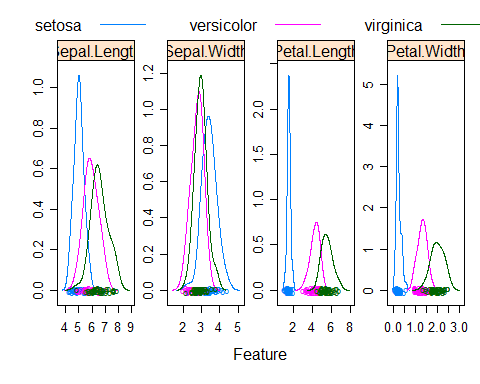


# c)  
summary(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width   
## Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100   
## 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300   
## Median :5.800 Median :3.000 Median :4.350 Median :1.300   
## Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199   
## 3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800   
## Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500   
## Species   
## setosa :50   
## versicolor:50   
## virginica :50   
##   
##   
##

The data is not unbalanced as there are 3 classes in species (setosa, versicolor, virginica) with all equal ratios 50:50:50.

#d)  
  
x <- iris[, 1:4]  
y <- iris$Species  
caret::featurePlot(x,   
 y,   
 plot="density",   
 scales = list(x = list(relation="free"),   
 y = list(relation="free")),   
 adjust = 1.5,   
 pch = "o",   
 layout = c(4, 1),   
 auto.key = list(columns = 3))

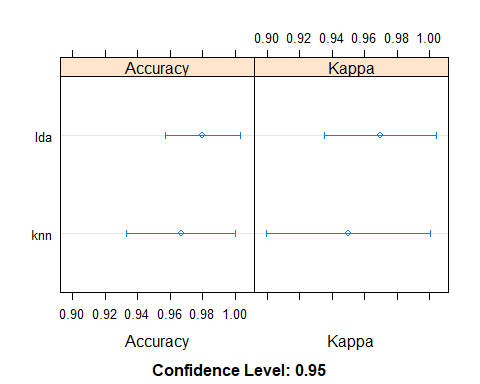


From the density plot for petal length and petal width, we can see that the density of setosa flower does not overlap with the densities of the versicolor and virginica iris flowers.Because there is no overlap these might be better predictors for flower species. In the Sepal.Length and Sepal.Width plot all three densities (setosa, versicolor and virginica) have overlap which may mean they are not as significant for predictions. There is a a lot more overlap between all three species for the sepal length and sepal width features, so these variables are not as significant in predictions.

#e)  
nfolds <- 10  
traincontrol <- trainControl(method = "cv",  
 number = nfolds)  
LDA <- train(Species ~ .,   
 data = iris,   
 method = "lda",   
 trControl = traincontrol)  
  
KNN <- train(Species ~ .,   
 data = iris,   
 method = "knn",   
 tuneGrid = expand.grid(k = c(1,2,3,4,5,6,7,8,9,10)),  
 trControl = traincontrol,   
 metric="Accuracy")  
  
results <- resamples(list(lda=LDA, knn=KNN))  
summary(results)

##   
## Call:  
## summary.resamples(object = results)  
##   
## Models: lda, knn   
## Number of resamples: 10   
##   
## Accuracy   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lda 0.9333333 0.9500000 1 0.9800000 1 1 0  
## knn 0.8666667 0.9333333 1 0.9666667 1 1 0  
##   
## Kappa   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lda 0.9 0.925 1 0.97 1 1 0  
## knn 0.8 0.900 1 0.95 1 1 0

dotplot(results)



From the plot we can see that LDA is more accurate than knn. We can also see that knn has a larger 95% confidence interval compared to lda. Because of these two facts the lda should be used for better accuracy.

## Question 2

#a  
library(ISLR2)  
summary(Credit)

## Income Limit Rating Cards   
## Min. : 10.35 Min. : 855 Min. : 93.0 Min. :1.000   
## 1st Qu.: 21.01 1st Qu.: 3088 1st Qu.:247.2 1st Qu.:2.000   
## Median : 33.12 Median : 4622 Median :344.0 Median :3.000   
## Mean : 45.22 Mean : 4736 Mean :354.9 Mean :2.958   
## 3rd Qu.: 57.47 3rd Qu.: 5873 3rd Qu.:437.2 3rd Qu.:4.000   
## Max. :186.63 Max. :13913 Max. :982.0 Max. :9.000   
## Age Education Own Student Married Region   
## Min. :23.00 Min. : 5.00 No :193 No :360 No :155 East : 99   
## 1st Qu.:41.75 1st Qu.:11.00 Yes:207 Yes: 40 Yes:245 South:199   
## Median :56.00 Median :14.00 West :102   
## Mean :55.67 Mean :13.45   
## 3rd Qu.:70.00 3rd Qu.:16.00   
## Max. :98.00 Max. :20.00   
## Balance   
## Min. : 0.00   
## 1st Qu.: 68.75   
## Median : 459.50   
## Mean : 520.01   
## 3rd Qu.: 863.00   
## Max. :1999.00

dim(Credit)

## [1] 400 11

There are 400 rows and 11 columns/variables. The variables are Income, Limit, Rating, Cards, Age, Education, Own, Student, Married, Region, Balance.

#b  
creditBalanceMean <- mean(Credit$Balance)  
creditBalanceMean

## [1] 520.015

Estimate for the population mean of the credit card balance is $520.015.

#c  
n\_observation <- nrow(Credit)  
SE <- sd(Credit$Balance) / sqrt(n\_observation)  
SE

## [1] 22.98794

Estimate of the standard error is $22.98794.

#d  
library(boot)

##   
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':  
##   
## melanoma

meanfunc = function(x, index) {  
 return(mean(x$Balance[index]))  
}  
  
bootSE <- boot(Credit, meanfunc, R=1000)  
bootSE

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Credit, statistic = meanfunc, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 520.015 0.3326725 22.86118

bootSEval <- 22.23329

The standard error using bootstrap is $22.23329. Our standard error in c is a bit higher with value of $22.98794.

lower <- creditBalanceMean - 2\*bootSEval  
upper <- creditBalanceMean + 2\*bootSEval  
CI <- c(lower, upper)   
CI

## [1] 475.5484 564.4816

t.test(Credit$Balance, conf.level = 0.95)

##   
## One Sample t-test  
##   
## data: Credit$Balance  
## t = 22.621, df = 399, p-value < 2.2e-16  
## alternative hypothesis: true mean is not equal to 0  
## 95 percent confidence interval:  
## 474.8224 565.2076  
## sample estimates:  
## mean of x   
## 520.015

From the results we can see that they are very similar/values are very close to each other.

# Question 3

library(tree)  
library(gbm)

## Loaded gbm 2.1.8

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

setwd("H:/Documents/STAT318/Assignments/Assignment3")  
training = read.csv("carseatsTrain.csv")  
testing = read.csv("carseatsTest.csv")

library(tree)  
library(gbm)  
library(randomForest)  
library(MASS)

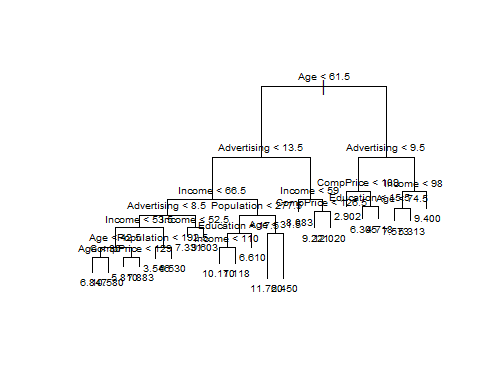
##   
## Attaching package: 'MASS'

## The following object is masked from 'package:ISLR2':  
##   
## Boston

setwd("H:/Documents/STAT318/Assignments/Assignment3")  
training = read.csv("carseatsTrain.csv")  
testing = read.csv("carseatsTest.csv")  
  
#a)  
train\_tree <- tree(Sales~., data=training)

## Warning in tree(Sales ~ ., data = training): NAs introduced by coercion

plot(train\_tree)  
text(train\_tree,  
 cex=0.6)



trainingMSE <- (mean((predict(train\_tree,  
 newdata=training) - training$Sales) ^ 2))

## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion

testingMSE <- (mean((predict(train\_tree,  
 newdata=testing) - testing$Sales) ^ 2))

## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion

trainingMSE

## [1] 4.224765

testingMSE

## [1] 9.758961

From the tree we can see that there are 22 terminal nodes. We can also see that Age and Advertising are the most useful predictors as age is at the top and advertising is in the second level.

The training MSE is 4.224765 and the testing MSE is 9.758961.

#b)  
prune <- cv.tree(train\_tree)

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

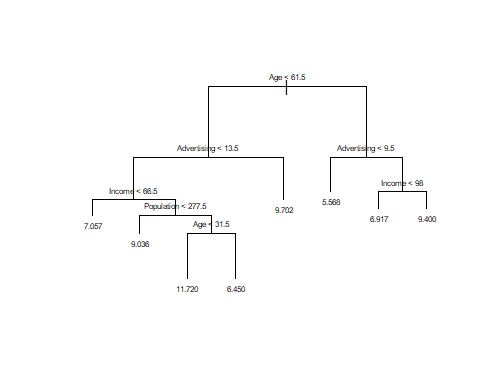
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

prune

## $size  
## [1] 22 21 20 19 18 17 15 13 11 10 8 7 4 3 2 1  
##   
## $dev  
## [1] 2751.276 2653.155 2653.155 2605.795 2579.758 2579.758 2580.755 2519.878  
## [9] 2482.008 2482.008 2461.860 2390.998 2225.892 2264.497 2281.425 2306.561  
##   
## $k  
## [1] -Inf 23.48884 23.54946 28.77311 32.44689 32.60524 32.75497  
## [8] 37.40522 38.43426 38.54855 41.61580 43.01071 69.00835 89.56740  
## [15] 109.43976 185.42861  
##   
## $method  
## [1] "deviance"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

tree\_prune <- prune.tree(train\_tree, best=8)  
plot(tree\_prune)  
text(tree\_prune,  
 cex=0.5)



pruneMSE <- (mean((predict(tree\_prune,   
 newdata=testing) - testing$Sales)^2))

## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion

pruneMSE

## [1] 7.004235

When we compare the pruned tree MSE and the non pruned tree MSE we can see that the pruned tree MSE (7.004235) is less than the non pruned tree (9.758961). So therefore the pruned tree does perform better.

#c)  
randForest <- randomForest(Sales~.,data=training,importance=TRUE)  
  
RF\_train\_MSE <- (mean((predict(randForest, newdata=training) - training$Sales)^2))  
RF\_test\_MSE <- (mean((predict(randForest, newdata=testing) - testing$Sales)^2))  
  
RF\_train\_MSE

## [1] 1.135293

RF\_test\_MSE

## [1] 5.01192

bagged <- randomForest(Sales~.,data=training,mtry=9,importance=TRUE)  
  
bagged\_train\_MSE <- (mean((predict(bagged, newdata=training) - training$Sales)^2))  
bagged\_test\_MSE <- (mean((predict(bagged, newdata=testing) - testing$Sales)^2))  
  
bagged\_train\_MSE

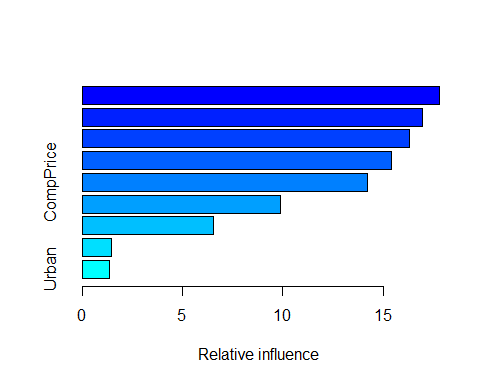
## [1] 0.9717669

bagged\_test\_MSE

## [1] 5.016608

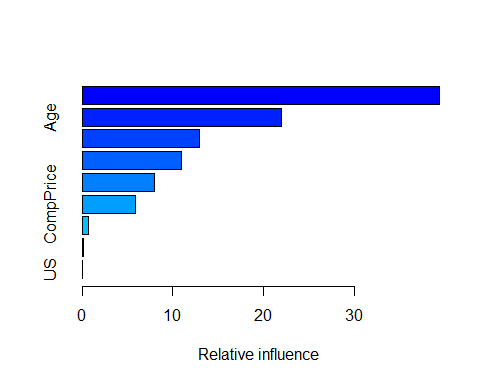
The random forest training MSE had a value of 1.157876 and the random forest testing MSE had a value of 4.981077. The bagged training MSE is 0.9528967 and the bagged testing MSE is 4.968944. Because bagging has smaller MSE values in both training and testing decorrelating was not a good strategy for this problem.

train <- sample(1:nrow(training), nrow(training)/2)  
  
training$ShelveLoc <- as.factor(training$ShelveLoc)  
training$Urban <- as.factor(training$Urban)  
training$US <- as.factor(training$US)  
  
boost\_tree <- gbm(Sales~.,  
 data=training[train,],  
 distribution="gaussian",  
 n.trees=5000,  
 interaction.depth=4,  
 shrinkage=0.05)  
  
  
summary(boost\_tree)



## var rel.inf  
## Population Population 17.825007  
## Age Age 16.949765  
## Income Income 16.326230  
## ShelveLoc ShelveLoc 15.412011  
## CompPrice CompPrice 14.207624  
## Advertising Advertising 9.906896  
## Education Education 6.523474  
## US US 1.477072  
## Urban Urban 1.371920

MSE\_calc <- function(x) {  
 MSE\_train\_calc <- (mean((predict(x,   
 newdata=training) - training$Sales)^2))  
 MSE\_test\_calc <- (mean((predict(x,   
 newdata=testing) - testing$Sales)^2))  
   
 return(c(MSE\_train\_calc, MSE\_test\_calc))  
}  
  
boost\_tree <- gbm(Sales~.,  
 data=training[train,],  
 distribution="gaussian",  
 n.trees=1000,  
 interaction.depth=1,  
 shrinkage=0.005)  
  
  
summary(boost\_tree)



## var rel.inf  
## ShelveLoc ShelveLoc 39.45783907  
## Age Age 21.99640548  
## Income Income 12.92454429  
## Population Population 11.00032774  
## Advertising Advertising 7.96804931  
## CompPrice CompPrice 5.87930201  
## Education Education 0.63417656  
## Urban Urban 0.10466617  
## US US 0.03468938

boost\_train\_MSE <- MSE\_calc(boost\_tree)[1]

## Using 1000 trees...  
##   
## Using 1000 trees...

## Warning in predict.gbm(x, newdata = testing): NAs introduced by coercion

boost\_test\_MSE <- MSE\_calc(boost\_tree)[2]

## Using 1000 trees...  
##   
## Using 1000 trees...

## Warning in predict.gbm(x, newdata = testing): NAs introduced by coercion

boost\_train\_MSE

## [1] 4.875997

boost\_test\_MSE

## [1] 6.918933

After trying different depths, shrinkage and number of trees I have found that 1000 trees with depth 1 and shrinkage gave me a training MSE of 4.311855 and testing MSE of 6.474855 which was my best one. When I tested different values I found that increasing the depth, number of trees and shrinkage would increase the MSE values.

1. I believe bagging model performed the best as it had a testing MSE of 4.968944 which was the smallest. The Sales were the most important for the predictors.