Øving 7

2022-05-30

Problem 1

a) Provide a detailed explanation of the algorithm that is used to fit a regression tree. What is different for a classification tree?

We divide the predictor space into J distinct and non-overlapping regions $R_1, R_2...R_J$. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j. The goal is to find R_j such that the RSS is minimized.

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})$$

Where $\hat{y}_{R_j})$ is the response for the training observations within the j‘th box.

For classification trees we predict a qualitative response rather than a quantitative one. We predict that each observation belongs to the most commonly occurring class of training observations in th region to which it belongs. We can NOT use RSS for binary splits.

b) What are the advantages and disadvantages of regression and classification trees?

Advantages:

```
• Simple and useful for interpretation.
• Easier to explain than regression.
• More closely mirrors human decision making.
• Bagging, random forests and boosting methods grow multiple trees, which results in improvements for pr
• Displayed graphically and there interpretable.
• Trees can easily handle qualitative predictors without the need to create dummy variables.
```

Disadvantages:

```
• Bagging, random forests and boosting methods grow gives loss in interpretation.
• Not competitive with the best supervised learning methods in terms of prediction accuracy.
• Trees generally do not have the same level of predictive accuracy as other regression and classificati
```

c) What is the idea behind bagging and what is the role of the bootstrap? How do random forests improve that idea?

Bootstrap aggregation, or bagging is a procedure for reducing the variance of a statistival learning method.

Averaging a set of observations reduces the variance by a factor 1/n. Thi is not practical because we generally don‘t have access to multiple training sets. In stead, we can bootstrap.

We generate B different bootstrapped training sets. We then train our method on the bth bootstrapped training set in order to get the estimated f_b(x), the prediction at point x. We then average all the predictors.

Random forests provide an improvement over bagged trees by way of a small tweak that decorrelated the trees. This reduces the variance when we average the trees.

As in bagging, we build a number of decision trees on bootstrapped training samples.

When building these decision trees, each time a split in a tree is considered, a random selection of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.

A fresh selection of m predictors is taken at each split, and typically we choose m = sqrt(p). That is the number of predictors considered at each split.

d)
e)

Problem 2

a) Split the data into a training and a test set,

```
library(ISLR)

data("Carseats")

set.seed(4268)

n = nrow(Carseats)

train = sample(1:n, 0.7 * nrow(Carseats), replace = F)
test = (1:n)[-train]

Carseats.train = Carseats[train,]
Carseats.test = Carseats[-train,]

head(Carseats)
```

```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age Education
## 1  9.50       138     73          11        276   120       Bad  42        17
## 2 11.22       111     48          16        260    83      Good  65        10
## 3 10.06       113     35          10        269    80    Medium  59        12
## 4  7.40       117    100           4        466    97    Medium  55        14
## 5  4.15       141     64           3        340   128       Bad  38        13
## 6 10.81       124    113          13        501    72       Bad  78        16
##   Urban  US
## 1   Yes Yes
## 2   Yes Yes
## 3   Yes Yes
## 4   Yes Yes
## 5   Yes  No
## 6    No Yes
```

b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

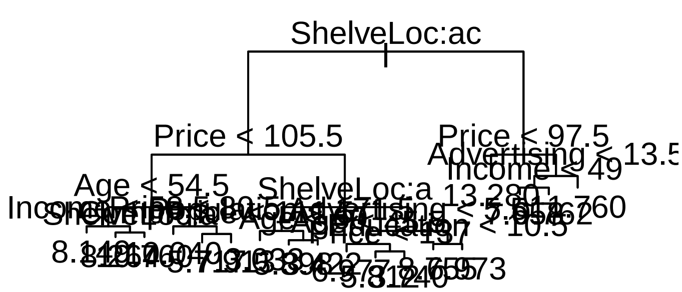
```r
#install.packages("tree")
library(tree)

tree.mod = tree(Sales ~ ., data = Carseats.train)

summary(tree.mod)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats.train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Income"      "CompPrice"
## [6] "Population"  "Advertising" "Education"
## Number of terminal nodes:  18
## Residual mean deviance:  2.609 = 683.6 / 262
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.74000 -1.12400 -0.06522  0.00000  1.06800  4.47200
```

```r
plot(tree.mod)
text(tree.mod)
```



It seems as the shelveloc and the price are the two most important variables in predicting the sales for our dataset, Age and advertising seems to be quite important as well.

Now we check the test MSE.

```r
preds = predict(tree.mod, newdata = Carseats.test)

MSE = mean((Carseats.test$Sales - preds)^2)
MSE
```

```
## [1] 4.585249
```

c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

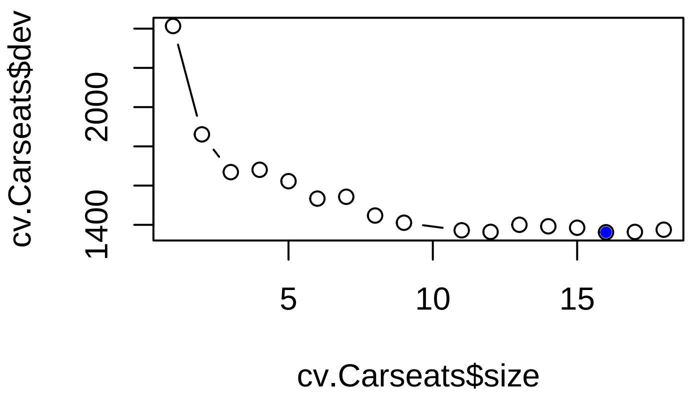
```
set.seed(4268)

cv.Carseats = cv.tree(tree.mod, FUN = prune.tree, K = 10)

tree.min = which.min(cv.Carseats$dev)

best = cv.Carseats$size[tree.min]

plot(cv.Carseats$size, cv.Carseats$dev, type = "b")
points(best, cv.Carseats$dev[tree.min], col = "blue", pch = 20)
```



We observe that the trees of the size 11 and 12 have similar results as the 16th. We might choose 11 for a simpler tree.

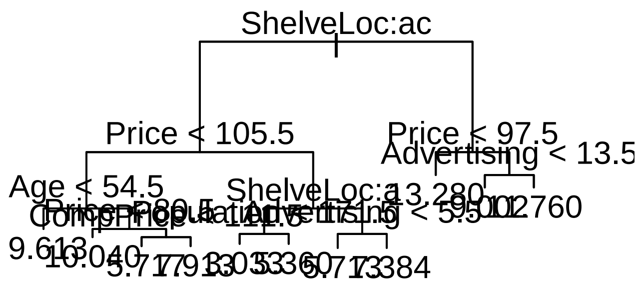
```
pruned_tree = prune.tree(tree.mod, best = 11)
pruned_preds = predict(pruned_tree, newdata = Carseats.test)

pruned_MSE = mean((pruned_preds-Carseats.test$Sales)^2)
pruned_MSE
```

```
## [1] 4.378499
```

Which is a lower test MSE than the first tree. We plot the tree to view it.

```
plot(pruned_tree)
text(pruned_tree)
```



d) Use the bagging approach with 500 trees in order to analyze the data. What test MSE do you obtain? Use the importance() function to determine which variables are most important.

```r
#install.packages("randomForest")
library(randomForest)

bag.Carseats = randomForest(Sales ~ ., data=Carseats.train, ntree = 500, importance = TRUE)

bag_preds = predict(bag.Carseats, newdata = Carseats.test)

bag_MSE = mean((Carseats.test$Sales-bag_preds)^2)
bag_MSE
```

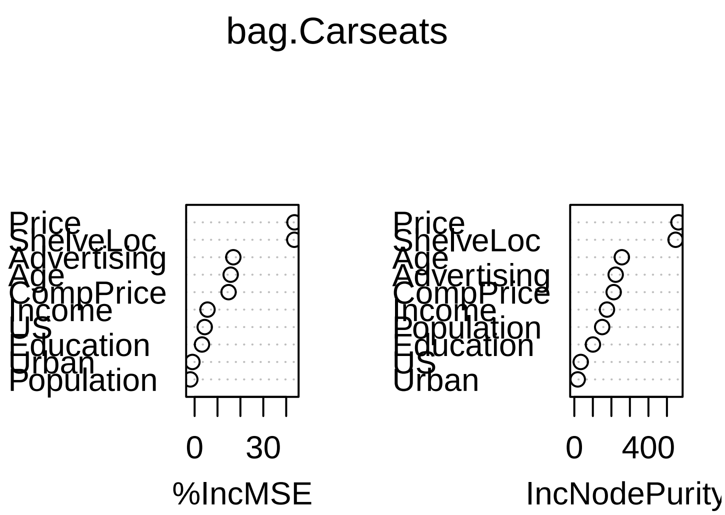
```
## [1] 2.205642
```

We observe after bagging that the test MSE is even lower.

```r
importance(bag.Carseats)
```

```
##                %IncMSE IncNodePurity
## CompPrice   14.8283028     212.50052
## Income       5.6307470     176.16937
## Advertising 16.9012567     223.23017
## Population  -1.8783109     150.45803
## Price       43.5966216     562.11112
## ShelveLoc   43.5261787     546.62779
## Age         15.7561119     256.52949
## Education    3.2204370     100.40560
## Urban       -0.9570597      18.34805
## US           4.4134142      34.63795
```

```r
varImpPlot(bag.Carseats)
```



We observe that Shelveloc, Price, age and advertising are the most important variables.

e) Use random forests to analyze the data. Include 500 trees and select 3 variables for each split. What test MSE do you obtain? Use the importance() function to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

```r
rf.Carseats = randomForest(Sales ~ ., data = Carseats.train,  mtry = 3, ntree = 500, importance = TRUE)

yhat.rf = predict(rf.Carseats, newdata= Carseats.test)

rf_MSE = mean((yhat.rf - Carseats.test$Sales)^2)
rf_MSE
```

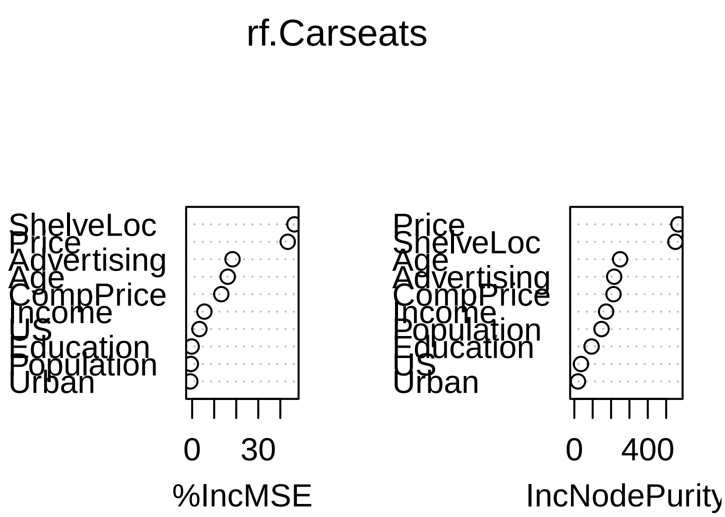
```
## [1] 2.25897
```

We use p/m = 10/3 trees (about 3 trees). We get an MSE a little larger than Bagging.

```r
importance(rf.Carseats)
```

```
##                %IncMSE IncNodePurity
## CompPrice   13.2355917     213.40899
## Income       5.5487005     173.08169
## Advertising 18.3241456     217.00013
## Population  -0.5880682     147.97481
## Price       43.3832973     566.91128
## ShelveLoc   46.4146575     550.24111
## Age         16.1467391     249.21306
## Education   -0.2206834      94.00016
## Urban       -0.8309524      19.95026
## US           3.3052482      36.29111
```

```
varImpPlot(rf.Carseats)
```



f) Finally use boosting with 500 trees, an interaction depth d = 4 and a shrinkage factor $\lambda = 0.1$, which is the default for the gbm() function. Compare MSE to the other methods.

```
#install.packages("gbm")
library(gbm)

r.boost = gbm(Sales ~ ., data = Carseats.train, distribution = "gaussian", n.tree = 500, interaction.de

boost_preds = predict(r.boost, newdata = Carseats.test)

boost_MSE = mean((boost_preds-Carseats.test$Sales)^2)
boost_MSE
```

```
## [1] 1.929048
```

We observe that the MSE is even lower than for the other methods.

g) What is the effect of the number of trees (ntree) on the test error. Plot the test MSE as a function of ntree for both the bagging and the random forest method.

Problem 3

b) Create a training set and a test set for the dataset.

```
#install.packages("kernlab")

library(kernlab)
data(spam)
n = nrow(spam)
```

```
train_index = sample(1:n, 0.7 * nrow(spam), replace = F)

spam.train = spam[train_index,]
spam.test = spam[-train_index,]
```

c) Fit a tree to the training data with type as the response and the rest of the variables as predictors. Study the results by using the summary() functuon. Also create a plot of the tree. How many terminal nodes does it have?

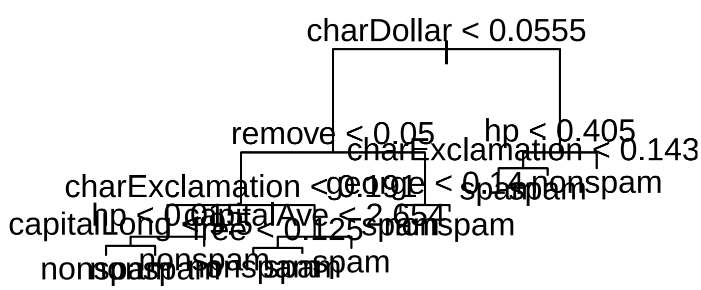
```
library(tree)

tree_model = tree(type ~ ., data= spam.train)
summary(tree_model)
```

```
##
## Classification tree:
## tree(formula = type ~ ., data = spam.train)
## Variables actually used in tree construction:
## [1] "charDollar"      "remove"          "charExclamation" "hp"
## [5] "capitalLong"     "capitalAve"      "free"            "george"
## Number of terminal nodes:  11
## Residual mean deviance:  0.5184 = 1664 / 3209
## Misclassification error rate: 0.09317 = 300 / 3220
```

We have 13 terminal nodes.

```
plot(tree_model)
text(tree_model)
```



d) Predict the response on the test data. What is the misclassification rate?

```
tree_preds = predict(tree_model, newdata = spam.test, type = "class")

missclass_table = table(tree_preds, spam.test$type)

error_rate = 1-sum(diag(missclass_table))/sum(missclass_table)
error_rate
```

```
## [1] 0.0948588
```

e)