# Machine Learning, Spring 2021
# Homework 9

Daniil Bulat, Jonas Josef Huwyler, Haochen Li,Giovanni Magagnin

5/13/2021

## Exercise 1

**We follow (and extend) the example in the book *Introduction to Statistical Learning* in Section 8.3.2, but with a different training data set, namely our standard choice:**

```
# Install packages required
library(MASS)

# Data Preparation
Data = Boston
set.seed(123)
Data = Data[sample(nrow(Data)),]
ntrain = floor(nrow(Data)*0.7)
Data.Train = Data[1:ntrain,]
Data.Test = Data[(ntrain+1):nrow(Data),]
```

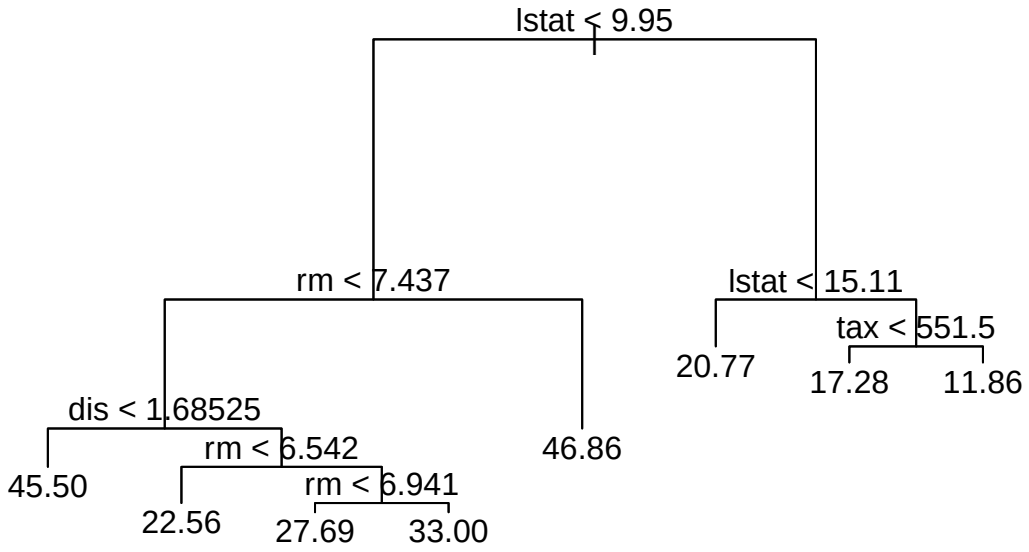
### Question 1

**Load the tree package/library. Use the tree command with the prespecified stopping criteria (i.e., do not specify anything in the command) to construct a (large) tree for the given (training) data. Plot it with the plot and the text command.**

```
# Load the tree package.
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.0.5
```

```
# Use the tree command with the prespecifed stopping criteria for the trainning data.
tree.Train = tree(medv~., data = Data.Train)
# Plot it
plot(tree.Train )
text(tree.Train )
```




## Question 2

**How many terminal nodes (leaves) do you get? Which of the input variables (features) have been used when building the tree?**

```
summary(tree.Train)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Data.Train)
## Variables actually used in tree construction:
## [1] "lstat" "rm"    "dis"   "tax"
## Number of terminal nodes:  8
## Residual mean deviance:  10.67 = 3693 / 346
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -18.0000  -2.0670   0.2369   0.0000   2.1390   9.9320
```

From both the plot and the summary command for the training model, we can find the following results:

- Number of terminal nodes: 8
- Input variables: “lstat” “rm” “dis” “tax”

## Question 3

**Use the predict command to get predictions for the test set. Compute the RSS for the test set.**

```r
# use the predict command to get predictions for the test set.
yhat1 = predict(tree.Train, newdata = Data.Test)
# Compute the RSS for the test set.
RSS1 = (Data.Test$medv - yhat1)%*%(Data.Test$medv - yhat1)
cat ("The RSS is ",RSS1)
```

```
## The RSS is  3263.481
```

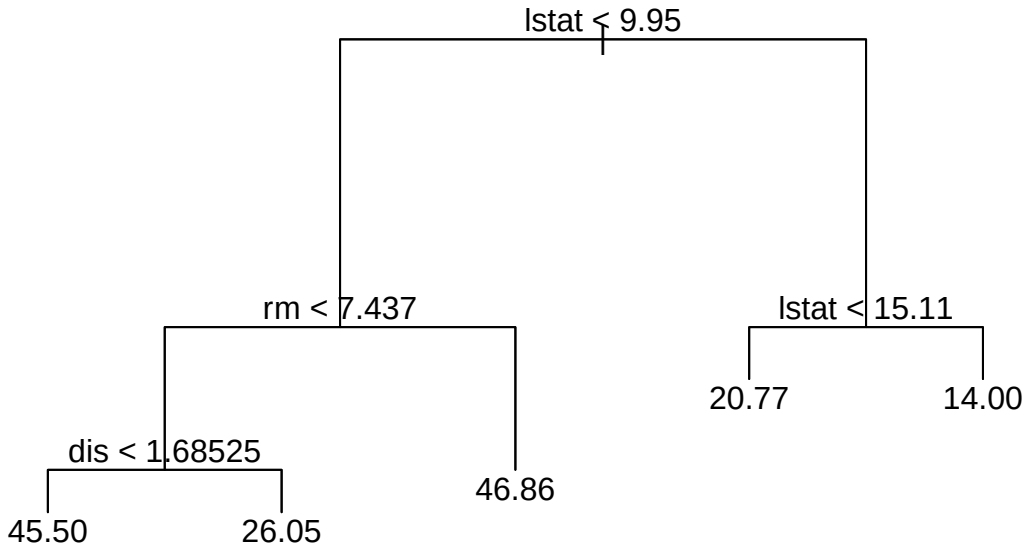
## Question 4

**Now prune the tree to get the best pruned subtree with five terminal nodes. Plot it. Which predictors have been used?**

```r
# Prune the tree to get the best pruned subtree with five terminal nodes.
p.tree.Train = prune.tree(tree.Train, best = 5)
# summary result
summary(p.tree.Train)
```

```
##
## Regression tree:
## snip.tree(tree = tree.Train, nodes = c(7L, 9L))
## Variables actually used in tree construction:
## [1] "lstat" "rm"    "dis"
## Number of terminal nodes:  5
## Residual mean deviance:  18.96 = 6617 / 349
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -18.0000  -2.8260  -0.4336   0.0000   2.6370  15.2500
```

```r
# Plot it.
plot(p.tree.Train)
text(p.tree.Train)
```




Again, we can find following results from both the plot and the summary results of the model:

- Number of terminal nodes: 5 (set by the pruned subtree model)
- Input predictors: "lstat" "rm" "dis"

## Question 5

**Use the predict command with the smaller tree to get predictions for the test set. Compute the RSS for the test set.**

```
# use the predict command with the  smaller tree to get predictions for the
# test set.
yhat2 = predict(p.tree.Train, newdata = Data.Test)

# Compute the RSS for the test set.
RSS2 = (Data.Test$medv - yhat2)%*%(Data.Test$medv - yhat2)
cat ("The RSS for the smaller tree is ",RSS2)
```

```
## The RSS for the smaller tree is  4290.135
```

## Question 6

**Load the *randomForest* package/library and run the command. How many trees did R used by default and how many variables were chosen at random for each split? What is the MSE? Run the command again - why do you get a slightly different result?**

```
# Load the randomForest package.
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```
## randomForest 4.6-14
```

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

```
# run the command randomForest(medv ., data = Data.Train)
forest.Train = randomForest(medv ~  ., data = Data.Train)
print(forest.Train)
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = Data.Train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 11.37757
##                     % Var explained: 86.2
```

```
# run the command again
randomForest(medv ~  ., data = Data.Train)
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = Data.Train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 11.47195
##                     % Var explained: 86.08
```

From the above results, we can find that R used 500 trees by default and 4 variables tried for each split. The corresponding MSE is 11.38 for the former one and 11.47 for the latter one.

The reason why that we get a slightly different results is because that random forest decorrelates the tree by the random argument so the results depend on the random seed we set at the beginning.

## Question 7

**Now use the predict command to use the random forest for predictions on the test set. Compute the test RSS and compare to the above resutls (normal tree and pruned tree).**

```
# use the predict command to use the random forest for predictions on the
# test set.
yhat3 = predict(forest.Train, newdata = Data.Test)
# Compute the RSS for the test set.
RSS3 = (Data.Test$medv - yhat3)%*%(Data.Test$medv - yhat3)
RSS = c(RSS1,RSS2,RSS3)
MODEL = c('Normal Tree','Pruned Tree','Random Forest')
results = data.frame(MODEL,RSS)
print(results)
```

```
##           MODEL      RSS
## 1   Normal Tree 3263.481
## 2   Pruned Tree 4290.135
## 3 Random Forest 1642.684
```

As we can find from the above table, the model of random forest reduces the RSS compared to single tree methods. In other words, the random forest improve the prediction accuracy efficiently.