MATH 189 HW8

Zijian Su
Zelong Zhou
Xiangyi Lin

Last Updated: March 10, 2023

Concrete contributions

All problems were done by Zijian Su, Zelong Zhou, Xiangyi Lin. All contributing equally to this assignment. Everyone put in enough effort.

Overview

In this problem, we revisit the Boston Housing Price dataset (Boston.csv). The description of the dataset can be found in the lecture slides. The response variable is median house price (medv). Analyze this dataset using tree-based methods through the following steps.

Packages

```
#install.packages("rmarkdown")
#install.packages("randomForest")
#install.packages("tree")
#tinytex::install_tinytex()
#install.packages("scatterplot3d")
```

Question 1

Randomly divide the dataset into a training set and a test set with equal or nearly equal sizes.

Answer:

read and divide

```
# read data
data <- read.csv("Boston.csv")
data <- data[,2:15]

set.seed(1)
break_point <- sample(1:nrow(data), floor((0.5)*nrow(data))-1)
```

```r
#Randomly divide the dataset into a training set and a test
train_data = data[break_point, ]
test_data = data[-break_point, ]
head(train_data)
```

```
##         crim zn indus chas   nox    rm  age    dis rad tax ptratio     bk lstat
## 505  0.10959  0 11.93    0 0.573 6.794 89.3 2.3889   1 273    21.0 393.45  6.48
## 324  0.28392  0  7.38    0 0.493 5.708 74.3 4.7211   5 287    19.6 391.13 11.74
## 167  2.01019  0 19.58    0 0.605 7.929 96.2 2.0459   5 403    14.7 369.30  3.70
## 129  0.32543  0 21.89    0 0.624 6.431 98.8 1.8125   4 437    21.2 396.90 15.39
## 418 25.94060  0 18.10    0 0.679 5.304 89.1 1.6475  24 666    20.2 127.36 26.64
## 471  4.34879  0 18.10    0 0.580 6.167 84.0 3.0334  24 666    20.2 396.90 16.29
##     medv
## 505 22.0
## 324 18.5
## 167 50.0
## 129 18.0
## 418 10.4
## 471 19.9
```

```r
head(test_data)
```

```
##      crim   zn indus chas   nox    rm  age    dis rad tax ptratio     bk lstat
## 2 0.02731  0.0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0.0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0.0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0.0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0.0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
## 7 0.08829 12.5  7.87    0 0.524 6.012 66.6 5.5605   5 311    15.2 395.60 12.43
##   medv
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
## 7 22.9
```

```r
# size
nrow(train_data)
```

```
## [1] 252
```

```r
nrow(test_data)
```

```
## [1] 254
```

Question 2

Fit the training set with a single regression tree using all predictors. Use CV to select a subtree size and apply the tree prune to obtain a subtree. Plot, respectively, the large tree and the selected subtree. Use the test set to calculate prediction MSEs for both the large tree and selected subtree.

Answer:

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.1.3
```

```
set.seed(1)
tree_ = tree(medv~., data = train_data)
summary(tree_)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = train_data)
## Variables actually used in tree construction:
## [1] "rm"    "lstat" "crim"  "age"
## Number of terminal nodes:  7
## Residual mean deviance:  10.28 = 2517 / 245
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10.1800  -1.7770  -0.1775   0.0000   1.9230  16.5800
```

Cross validation and pruning

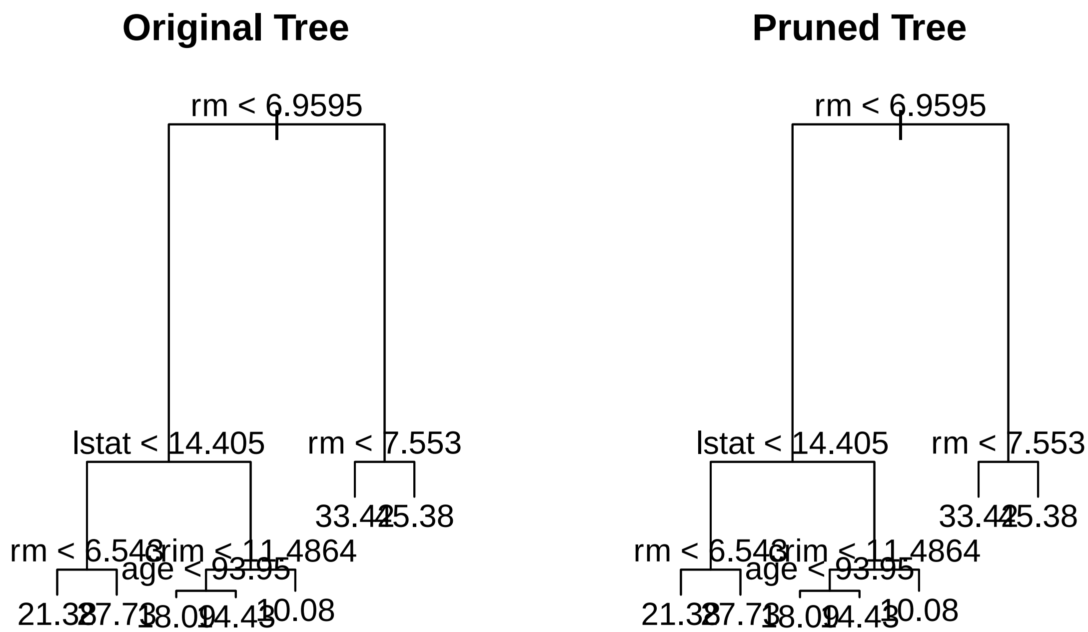
```
set.seed(1)
# Use CV to select a subtree size
cv_t <- cv.tree(tree_, K=10)
cv_size <-  cv_t$size[which.min(cv_t$dev)]

# Apply the tree prune to obtain a subtree
prune_t = prune.tree(tree_, best=cv_size)
```

Plot trees

```
par(mfrow = c(1, 2))
plot(tree_)
text(tree_, pretty=0)
title(main = "Original Tree")

plot(prune_t)
text(prune_t, pretty = 0)
title(main = "Pruned Tree")
```



Prediction MSEs

```r
set.seed(1)
pred_tree = predict(tree_, newdata = test_data)
pred_prune = predict(prune_t, newdata = test_data)

# prediction MSEs for both
mse_tree = mean((pred_tree - test_data$medv)^2)
mse_prune = mean((pred_prune - test_data$medv)^2)

sprintf("MSE for single tree   : %0.2f", mse_tree)
```

```
## [1] "MSE for single tree   : 35.36"
```

```r
sprintf("MSE for pruned subtree: %0.2f", mse_prune)
```

```
## [1] "MSE for pruned subtree: 35.36"
```

Question 3

Fit the training set with bagging and random forests. For both methods, use 100 trees. For random forests, set m=4. Again, use the test set to calculate prediction MSEs for bagging and random forests. Compare these results with those obtained for one single regression tree.

Answer:

training set with bagging

```
library(randomForest)

set.seed(1)
# use 100 trees.
bagging <- randomForest(medv~., data = train_data, mtry=13, importance=TRUE, ntree=100)

#calculating MSE
bagging_pred <- predict(bagging, newdata = test_data)
bagging_mse <- mean((bagging_pred - test_data$medv)^2)
sprintf("MSE for bagging: %0.2f", bagging_mse)
```

```
## [1] "MSE for bagging: 23.58"
```

training set with random forests

```
set.seed(1)
random_f <- randomForest(medv ~ ., data=train_data, mtry=4, importance=TRUE, ntree=100)

# MSE for random forests
random_f_pred <- predict(random_f, newdata=test_data)
random_f_mse <- mean((test_data$medv - random_f_pred)^2)
sprintf("MSE for random Forest: %0.2f", random_f_mse)
```

```
## [1] "MSE for random Forest: 19.15"
```

Compare

```
mse <- data.frame(Method =c("Single Regression Tree", "Pruned Regression Tree", "Bagging", "Random Fore

mse
```

```
##                   Method      MSE
## 1 Single Regression Tree 35.35694
## 2 Pruned Regression Tree 35.35694
## 3                Bagging 23.57915
## 4         Random Forests 19.14659
```

By comparing different mses, we can see that both Bagging and Random Forests reduce the error. Therefore, we can consider them to be better choices. In addition, if a bad m is selected in Random Forests, its mse may be larger than Bagging.