626hw1

Q1 a. Report the training and validation sample MSEs for linear and polynomial models. Which of these models is the most accurate?

```
#import data
library(MASS)
library(ISLR)
attach(Boston)

set.seed(9)
num_obs = nrow(Boston)
train_index = sample(num_obs, size = trunc(0.50 * num_obs))# 50/50 split between test and train

train_data = Boston[train_index, ]# Training sample

test_data = Boston[-train_index, ]# Validation sample

##########################
###linear fitting model###
##########################
lm.fit = lm(medv  ~ rm, data = train_data)

# The estimated train MSE
mse_train = mean((train_data$medv - predict(lm.fit, train_data)) ^ 2)
mse_train
```

```
## [1] 37.75354
```

The estimated train MSE for the linear regression fit is 37.75354.

```
# The estimated test MSE
mse_test = mean((test_data$medv - predict (lm.fit, test_data)) ^ 2)
mse_test
```

```
## [1] 49.72474
```

The estimated test MSE for the linear regression fit is 49.72474.

```
##############################
###polynomial fitting model###
##############################

# quadratic function
lm.fit2 = lm(medv ~ poly(rm, 2), data = train_data)
```

```r
# The estimated train MSE
mse2_train = mean((train_data$medv - predict(lm.fit2, train_data)) ^ 2)
mse2_train
```

```
## [1] 31.55728
```

The estimated train MSE for the quadratic function fit is 31.55728.

```r
# The estimated test MSE
mse2_test = mean((test_data$medv -predict (lm.fit2, test_data)) ^ 2)
mse2_test
```

```
## [1] 44.94636
```

The estimated test MSE for the quadratic function fit is 44.94636.

```r
# cubic function
lm.fit3 = lm(medv ~ poly(rm, 3), data = train_data)

# The estimated train MSE
mse3_train = mean((train_data$medv - predict(lm.fit3, train_data)) ^ 2)
mse3_train
```

```
## [1] 30.90499
```

The estimated train MSE for the cubic function fit is 30.90499.

```r
# The estimated test MSE
mse3_test = mean((test_data$medv -predict (lm.fit3, test_data)) ^ 2)
mse3_test
```

```
## [1] 43.51928
```

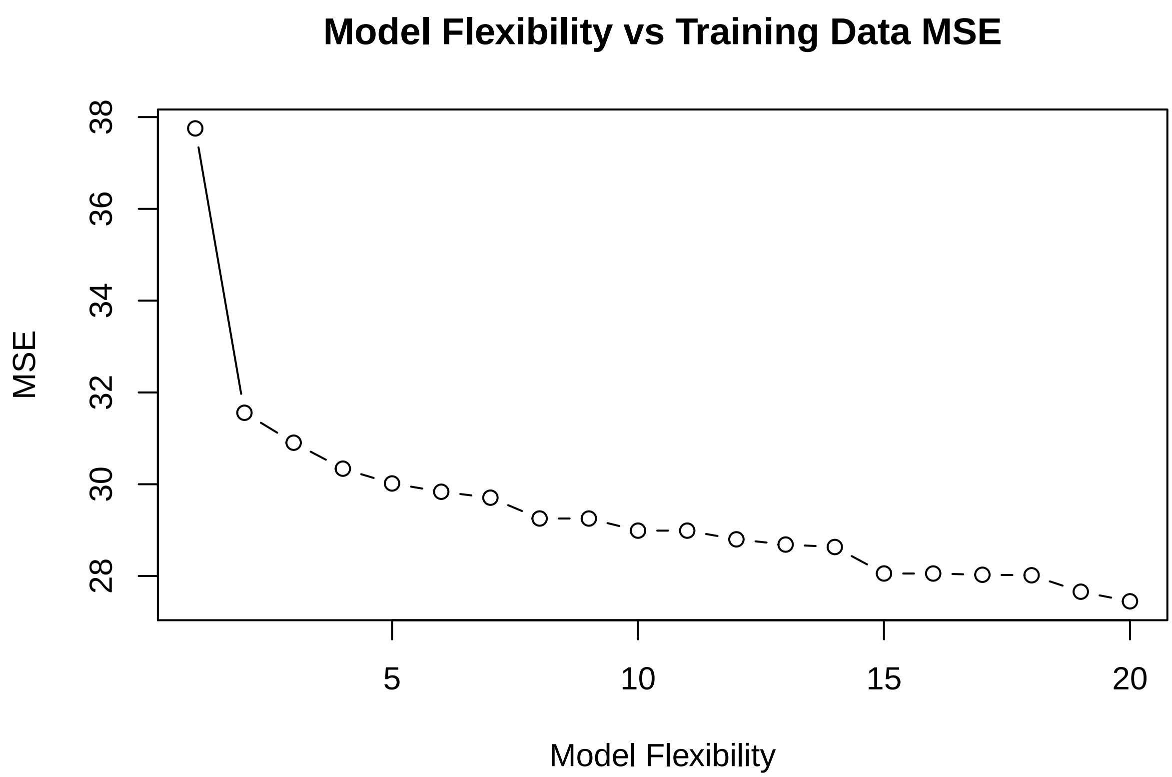
The estimated test MSE for the cubic function fit is 43.51928.

The polynomial fitting model is more accurate than the linear fitting model. We find that a model that predicts medv using a quadratic function of rm performs better than a model that involves only a linear function of rm for both the training and validation data. And there is little evidence in favor of a model that uses a cubic function of rm.

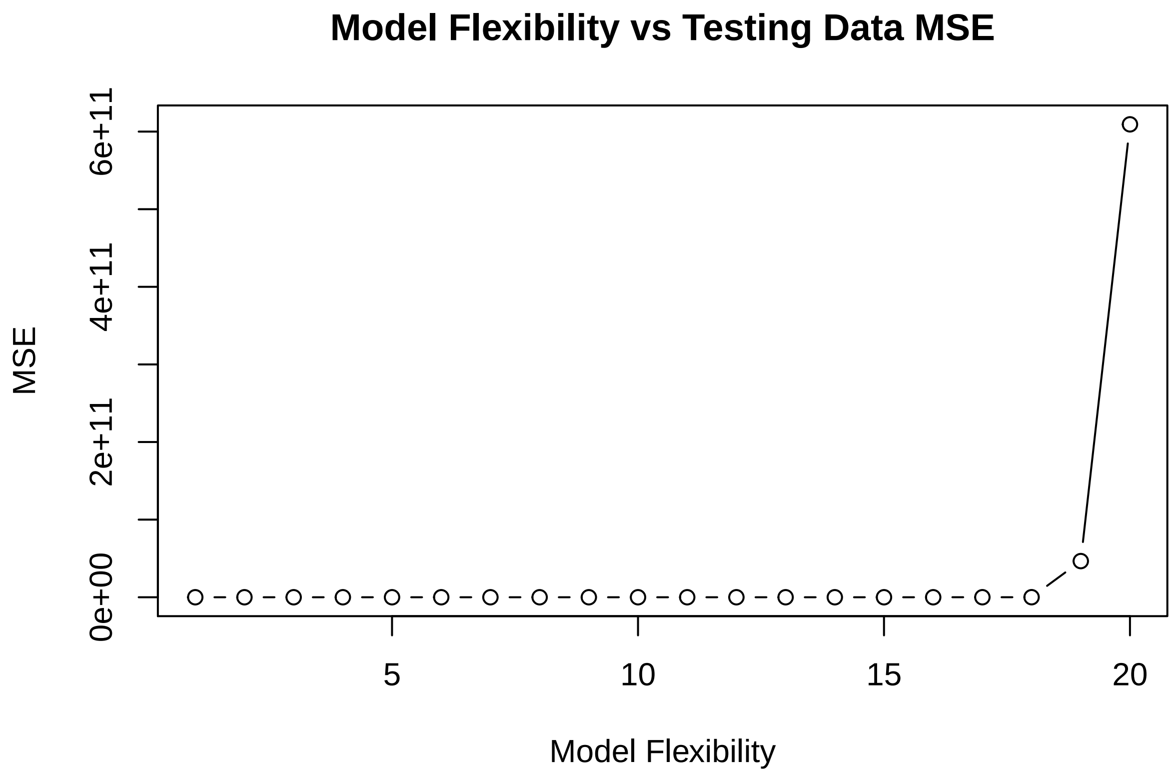
b. Construct a graph that illustrates the numbers reported in 1 as a function of the flexibility of the model.

```r
mse20.train = rep(0,20)
for (i in 1:20){
  lm.fit.i = lm(medv ~ poly(rm, i), data = train_data)
  mse20.train[i] = mean((train_data$medv - predict(lm.fit.i, train_data)) ^ 2)
}

mse20.test = rep(0,20)
```

```r
for (i in 1:20){
  lm.fit.i = lm(medv ~ poly(rm, i), data = train_data)
  mse20.test[i] = mean((test_data$medv - predict(lm.fit.i, test_data)) ^ 2)
}

plot(c(1:20), mse20.train, type = 'b', xlab = "Model Flexibility", ylab = "MSE",
     main = 'Model Flexibility vs Training Data MSE')
```




```r
plot(c(1:20), mse20.test, type = 'b', xlab = "Model Flexibility", ylab = "MSE",
     main = 'Model Flexibility vs Testing Data MSE')
```



c. Now repeat 1a, but using leave-one-out-cross-validation.

```
library(boot)
# Leave-One-Out Cross-Validation for the linear model fitting
glm.fit = glm(medv ~ rm, data = Boston)
cv.err = cv.glm(Boston,glm.fit)
cv.err$delta
```

```
## [1] 44.21666 44.21605
```

The cross-validation estimate for the test error is approximately 44.22.

```
# Leave-One-Out Cross-Validation to fit the linear model and polynomial model order from 2 to 5
cv.error = rep(0,5)
for (i in 1:5){
 glm.fit = glm(medv ~ poly(rm, i), data = Boston)
 cv.error[i] = cv.glm(Boston,glm.fit)$delta[1]
 }
cv.error
```

```
## [1] 44.21666 39.13461 38.24056 38.49876 36.14964
```

We can see a drop in the estimated test MSE between the linear and quadratic fits, and there is no clear improvement from using a higher-order polynomial.

d. Now repeat 1a, but using k-fold cross validation.

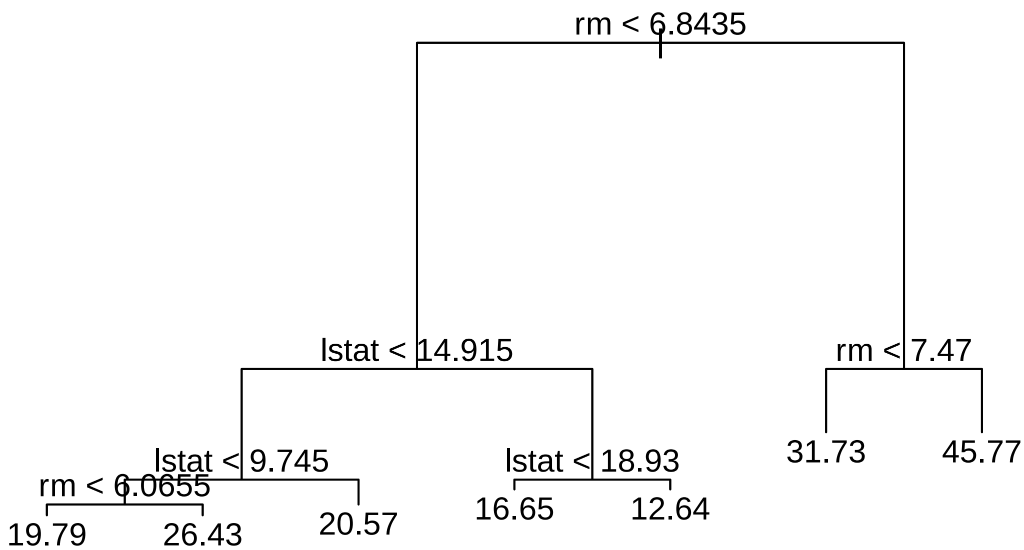
```
# K-fold Cross Validation to fit the linear model and polynomial model order from 2 to 9
cv.error.10 = rep(0, 10)
for (i in 1:10) {
glm.fit = glm(medv ~ poly(rm, i), data = Boston)
cv.error.10[i] = cv.glm(Boston, glm.fit, K = 10)$delta[1]
}
cv.error.10
```

```
##  [1] 44.11589 38.73139 37.98919 41.95685 36.26030 36.79426 38.53052 38.27325
##  [9] 69.48540 36.46831
```

There is not significant differences between the test errors of using cubic or higher-order polynomial terms than a quadratic fit.

Q2. a. Report a graph of the resulting regression tree and training and validation set MSEs.

```
library(tree)
set.seed(9)
num_obs = nrow(Boston)
train_index = sample(num_obs, size = trunc(0.50 * num_obs))# 50/50 split between test and train

train_data = Boston[train_index, ]# Training sample

test_data = Boston[-train_index, ]# validation sample

tree.boston = tree (medv ~ rm + age + lstat, train_data)

# Plot the resulting regression tree
window(plot(tree.boston), text(tree.boston, pretty=0))
```



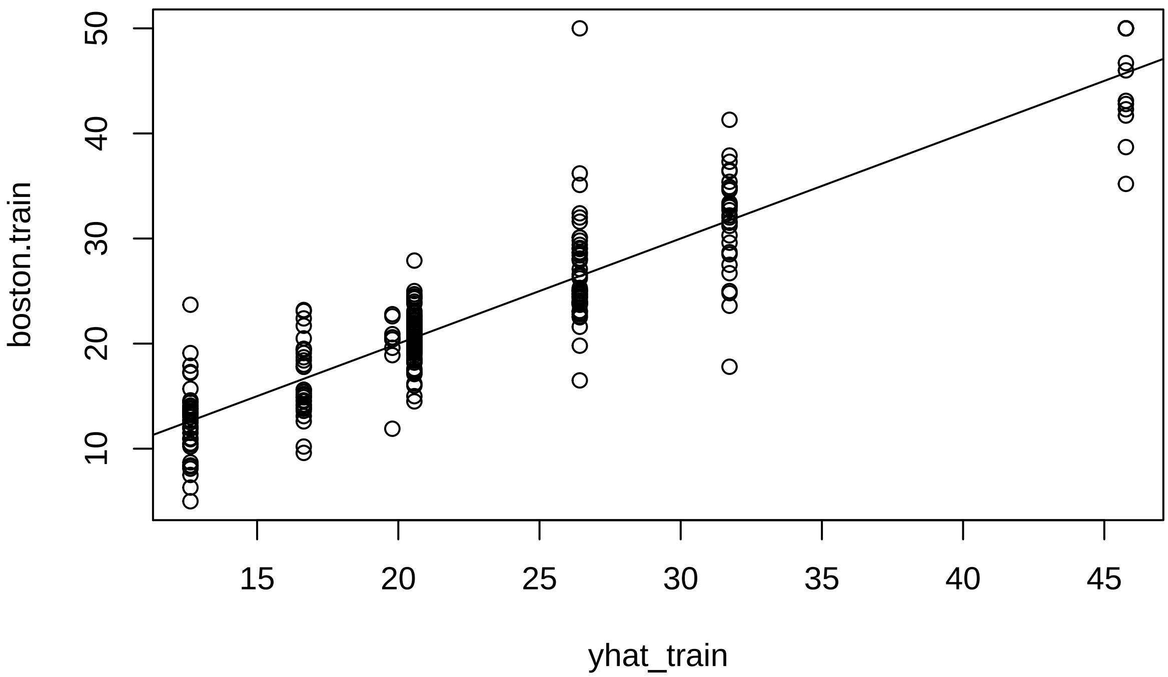

```
## $x
##  [1] 4.9375 3.3750 2.2500 1.5000 1.0000 2.0000 3.0000 4.5000 4.0000 5.0000
## [11] 6.5000 6.0000 7.0000
##
## $y
##  [1] 20830.661 10565.252  7083.453  6300.962  5961.146  5961.146  6300.962
##  [8]  7083.453  6775.959  6775.959 10565.252  8573.282  8573.282
##
## attr(,"tsp")
## [1] 1 2 1
```

```
# training set MSE
yhat_train = predict(tree.boston, newdata = train_data)
boston.train = train_data$medv
print(mean((yhat_train - boston.train) ^ 2))
```

```
## [1] 14.47305
```

The estimated test MSE is 14.47305.

```
plot(yhat_train, boston.train) + abline(0,1)
```



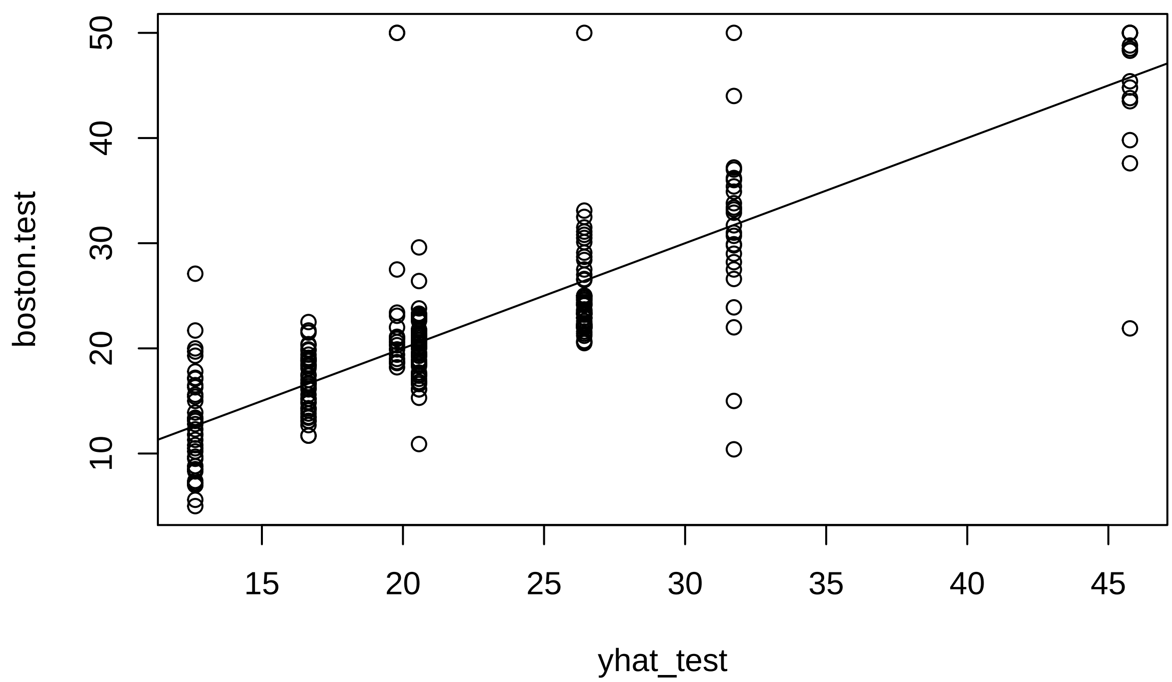

```
## integer(0)
```

```
# testing set MSE
yhat_test = predict(tree.boston, newdata = test_data)
boston.test = test_data$medv
print(mean((yhat_test - boston.test) ^ 2))
```

```
## [1] 29.16192
```

The estimated train MSE is 29.16192.

```
plot(yhat_test, boston.test) + abline(0,1)
```




```
## integer(0)
```

b. Provide a summary of the results and an assessment of which model is the most accurate.

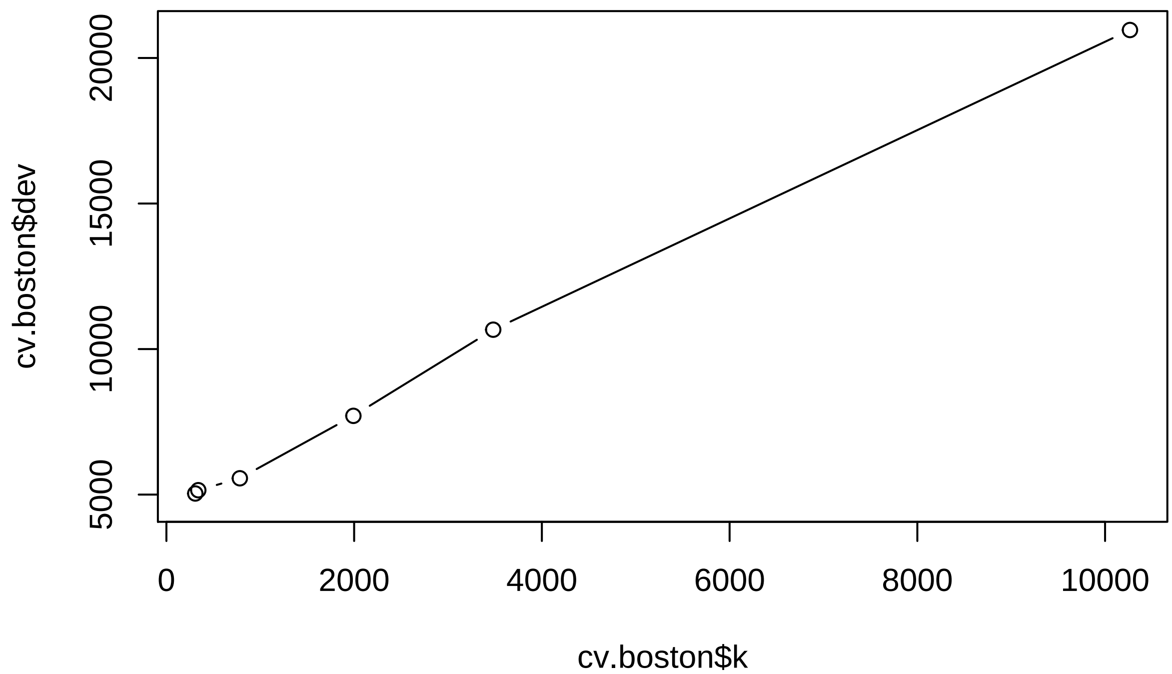
```
cv.boston = cv.tree(tree.boston)
cv.boston
```

```
## $size
## [1] 7 6 5 4 3 2 1
##
## $dev
## [1]  4714.662  5038.183  5151.019  5560.910  7705.970 10665.405 20961.182
##
## $k
## [1]       -Inf   307.4943   339.8163   782.4912  1991.9708  3481.7993 10265.4087
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

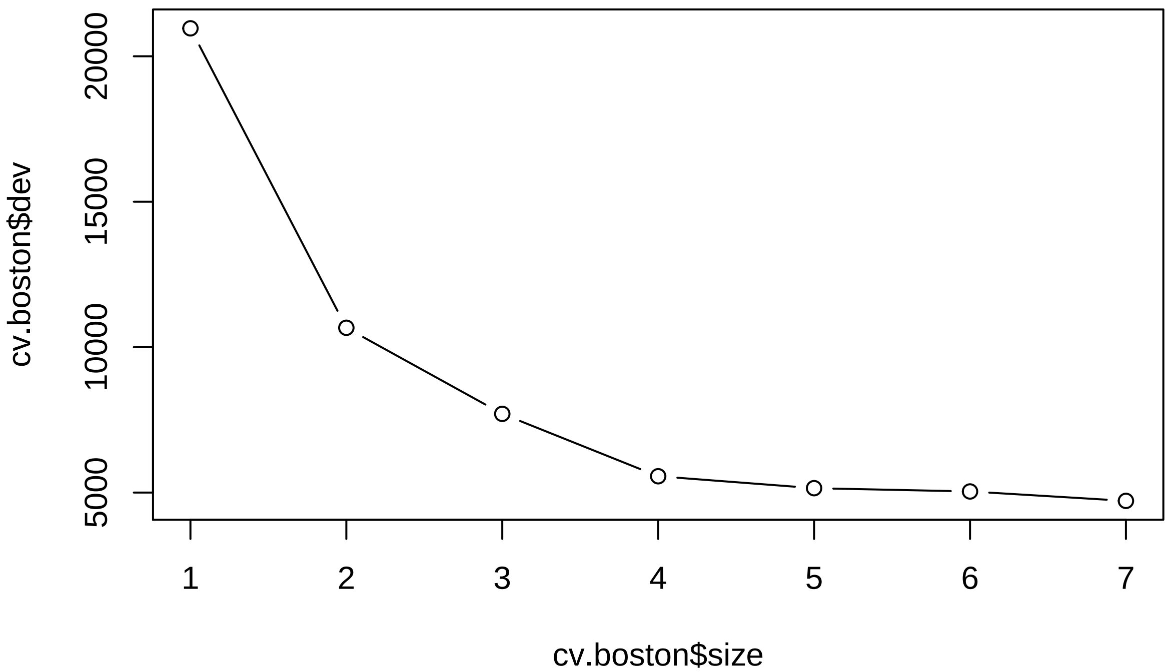
'dev' corresponds to the cross-validation error rate in this instance. 'k' represents the complexity penalty alpha. The tree with 7 terminal nodes results in the lowest cross-validation error rate, which is 4714.662.

The deviance decreases (accuracy increases) at a decreasing rate as the number of terminal nodes increases till 6 terminal nodes, and then the deviance continue to decrease (accuracy increases) at an increasing rate at 7 terminal nodes.

```
plot(cv.boston$k, cv.boston$dev, type="b")
```



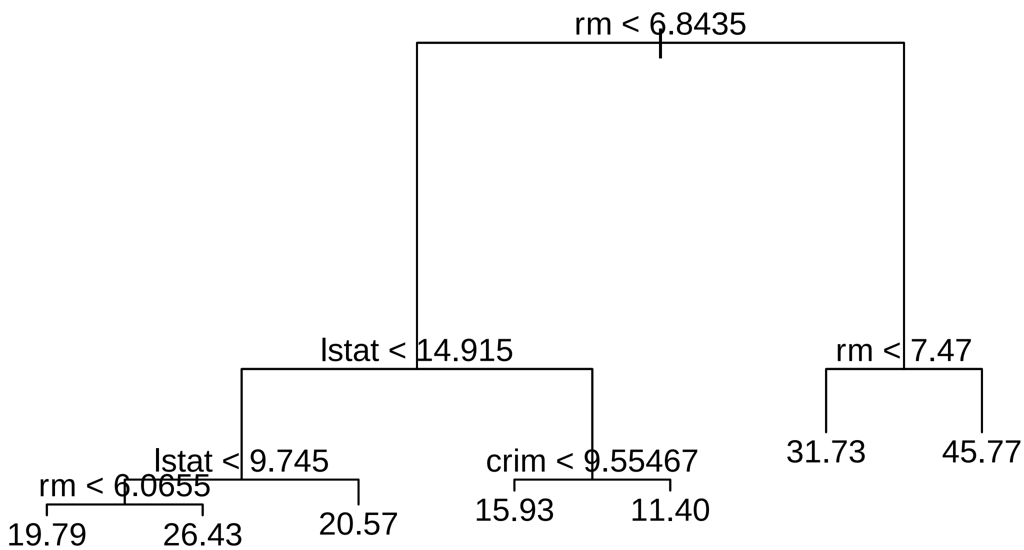

```
plot(cv.boston$size, cv.boston$dev, type='b')
```



From the first plot, a positive relationship between the complexity penalty alpha and the cross-validation error rate indicates there is a negative relationship between the complexity penalty alpha and the model accuracy. The second graph shows that the most complex tree is the best one.

c. Repeat 2a while allowing all available variables to be considered as predictors.

```
set.seed(9)
num_obs = nrow(Boston)
train_index = sample(num_obs, size = trunc(0.50 * num_obs))# 50/50 split between test and train

train_data = Boston[train_index, ]# Training sample

test_data = Boston[-train_index, ]# validation sample

tree.boston = tree (medv ~ ., train_data)

# Plot of the resulting regression tree
window(plot(tree.boston), text(tree.boston, pretty=0))
```



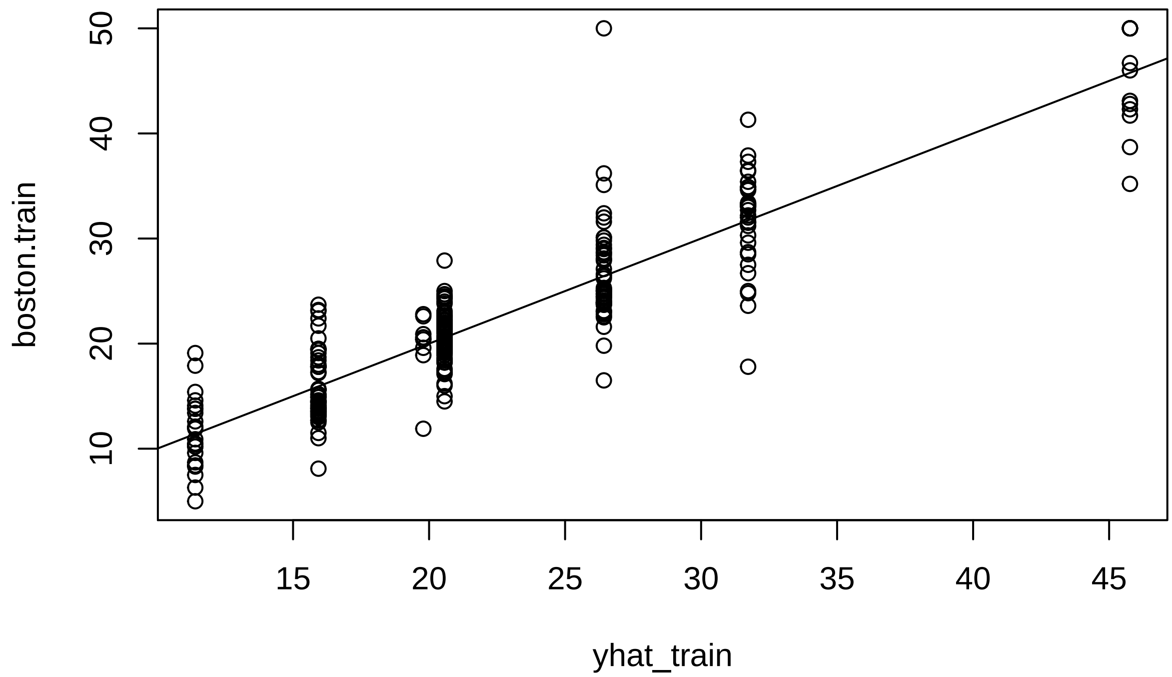

```
## $x
##  [1] 4.9375 3.3750 2.2500 1.5000 1.0000 2.0000 3.0000 4.5000 4.0000 5.0000
## [11] 6.5000 6.0000 7.0000
##
## $y
##  [1] 20830.661 10565.252  7083.453  6300.962  5961.146  5961.146  6300.962
##  [8]  7083.453  6737.196  6737.196 10565.252  8573.282  8573.282
##
## attr(,"tsp")
## [1] 1 2 1
```

```
# training set MSE
yhat_train = predict(tree.boston, newdata = train_data)
boston.train = train_data$medv
print(mean((yhat_train - boston.train) ^ 2))
```

```
## [1] 14.31983
```

The estimated test MSE is 14.31983.

```
plot(yhat_train, boston.train) + abline(0,1)
```



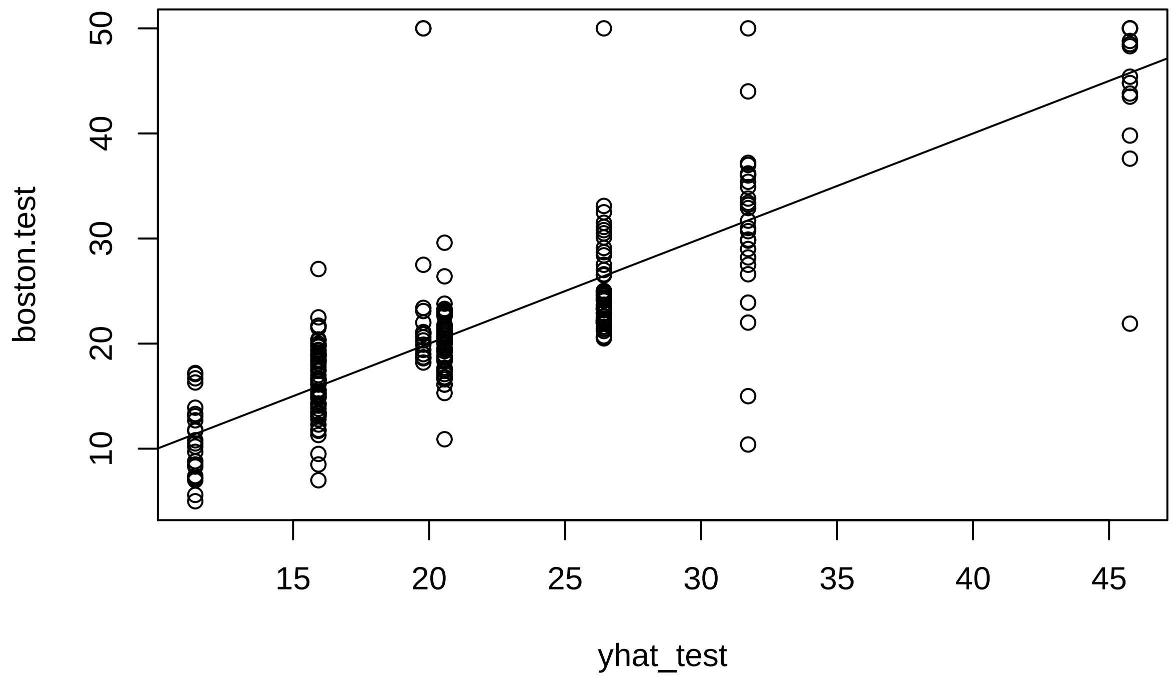

```
## integer(0)
```

```
# testing set MSE
yhat_test = predict(tree.boston, newdata = test_data)
boston.test = test_data$medv
print(mean((yhat_test - boston.test) ^ 2))
```

```
## [1] 28.46768
```

The estimated train MSE is 28.46768.

```
plot(yhat_test, boston.test) + abline(0,1)
```




```
## integer(0)
```