ISLR | Chapter 8 Exercises

Marshall McQuillen

10/10/2018

Conceptual

1

(sketch on following page)

2

Consider the regression setting; when each tree has depth one (meaning there is only one split), each tree will only consist of one split on one attribute, and the predicted response for those two groups will be the mean response (of the training data) of those two groups. Therefore, the base model will take the form $f_j(x_j) = \beta_j I(x_{ji} \leq t) + \beta'_j I(x_{ji} > t)$, where t is the value in the range of x_j that dictates the split (in order to minimize the RSS). Knowing this, one can walk through algorithm 8.2, as shown below:

$$\hat{f}(x) = 0 \tag{1}$$
$$f_1(x_1) = \beta_1 I(x_{1i} \leq t) + \beta'_1 I(x_{1i} > t) \tag{2}$$
$$\hat{f}(x) = \hat{f}(x) + \lambda f_1(x_1) \tag{3}$$
$$r_i = r_i - \lambda f_1(x_i) \tag{4}$$
$$f_2(x_2) = \beta_2 I(x_{2i} \leq t) + \beta'_2 I(x_{2i} > t) \tag{5}$$
$$\hat{f}(x) = \hat{f}(x) + \lambda f_2(x_2) \tag{6}$$
$$r_i = r_i - \lambda f_2(x_i) \tag{7}$$
$$\vdots \tag{8}$$
$$f_j(x_j) = \beta_j I(x_{ji} \leq t) + \beta'_j I(x_{ji} > t) \tag{9}$$
$$\hat{f}(x) = \hat{f}(x) + \lambda f_j(x_j) \tag{10}$$
$$r_i = r_i - \lambda f_j(x_i) \tag{11}$$
$$\vdots \tag{12}$$
$$f(x) = \sum_{j=1}^{P} f_j(X_j) \quad where \quad f_j(X_j) = \beta_j I(x_{ji} \leq t) + \beta'_j I(x_{ji} > t) \tag{13}$$
$$\tag{14}$$

Note that β_j is the mean of the response for all values of x_j that are less than or equal to t, and β'_j is the mean response for all values of x_j that are greater than t.

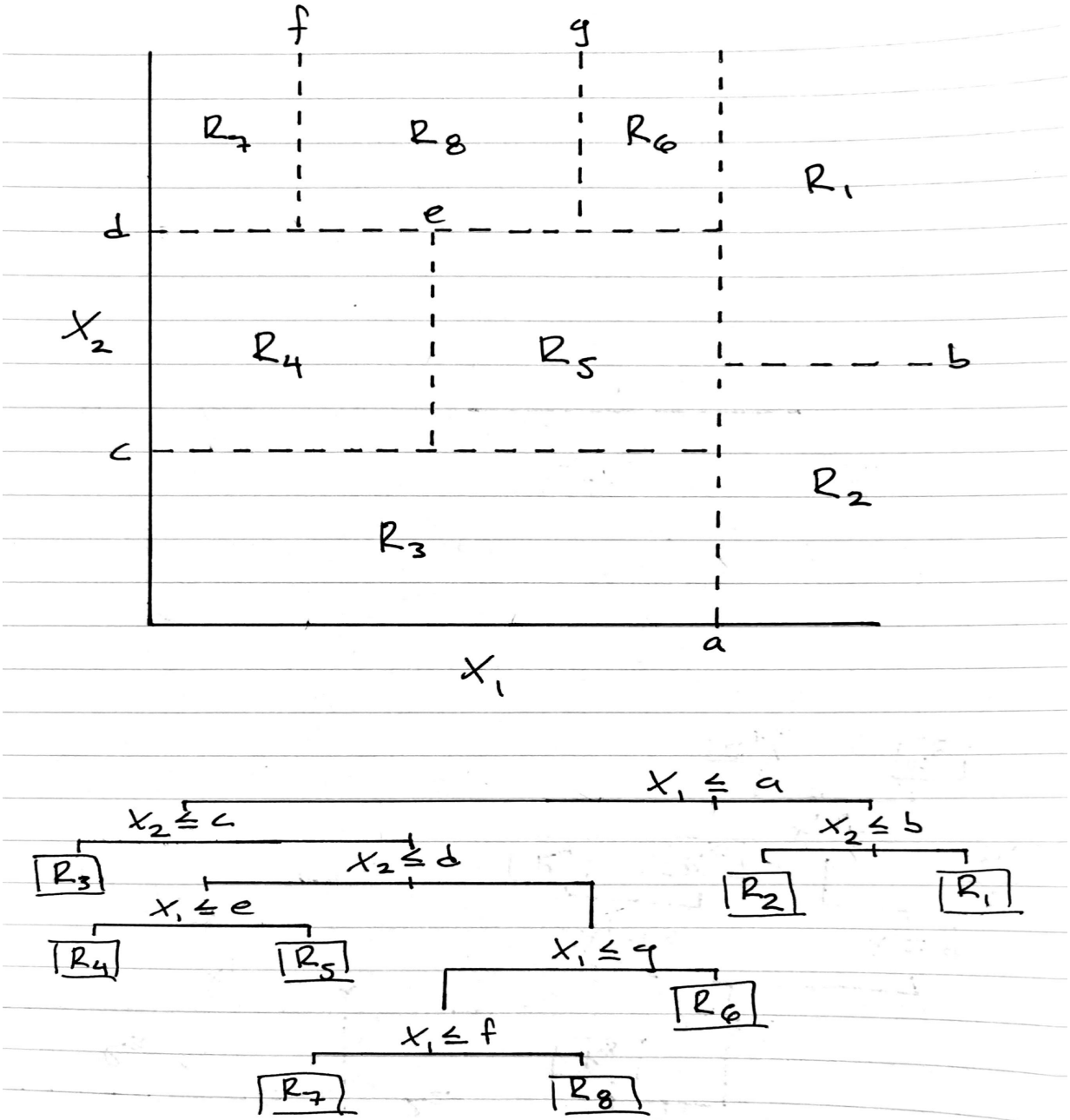
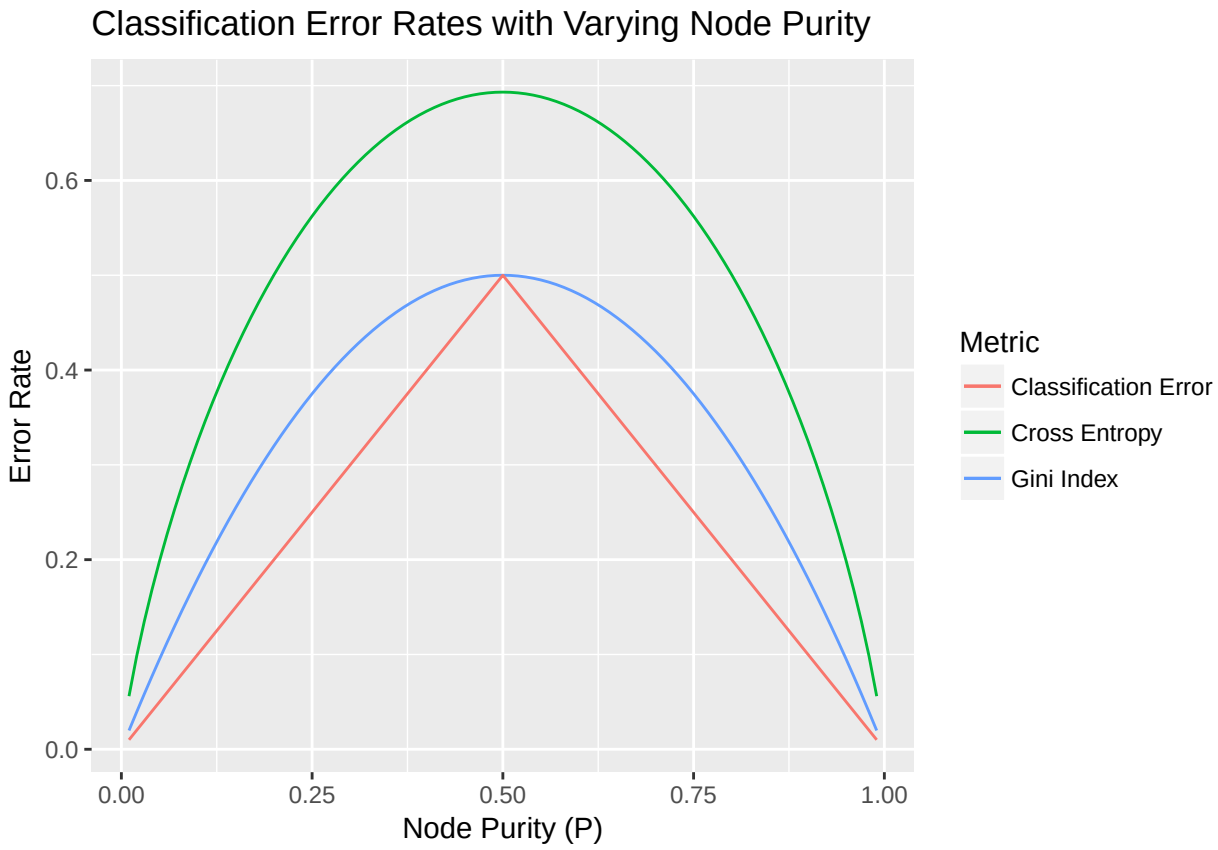


Figure 1: "Conceptual Exercise 1"

3

```r
suppressPackageStartupMessages(library(ggplot2))
pm1 <- seq(0.01, 0.99, 0.01)

# defining functions
two_class_gini <- function(prob) {
    return((prob * (1-prob)) + (prob * (1-prob)))
}

two_class_entropy <- function(prob) {
    return(-1*(prob*log(prob) + (1 - prob)*log(1 - prob)))
}

class_error <- function(prob) {
    return(1 - pmax(prob, (1 - prob)))
}

# error calculation
classification_error <- class_error(pm1)
gini_error <- two_class_gini(pm1)
entropy_error <- two_class_entropy(pm1)

# bring all data together for plotting
error_df <- data.frame(x = pm1,
                       gini = gini_error,
                       class = classification_error,
                       entropy = entropy_error)

# error plot
ggplot(error_df, aes(x = x, y = gini)) +
    geom_line(aes(color = 'Gini Index')) +
    geom_line(aes(y = entropy, color = 'Cross Entropy')) +
    geom_line(aes(y = class, color = 'Classification Error')) +
    guides(color = guide_legend(title = "Metric")) +
    ggtitle("Classification Error Rates with Varying Node Purity") +
    ylab("Error Rate") +
    xlab("Node Purity (P)")
```



4

(sketch on following page)

5

```
probs <- c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75)

# majority vote chooses red
majority_vote <- ifelse(probs >= 0.5, 'red', 'green')
table(majority_vote)
```

```
## majority_vote
## green   red
##     4     6
```

```
# average chooses green
ifelse(mean(probs) >= 0.5, 'red', 'green')
```

```
## [1] "green"
```

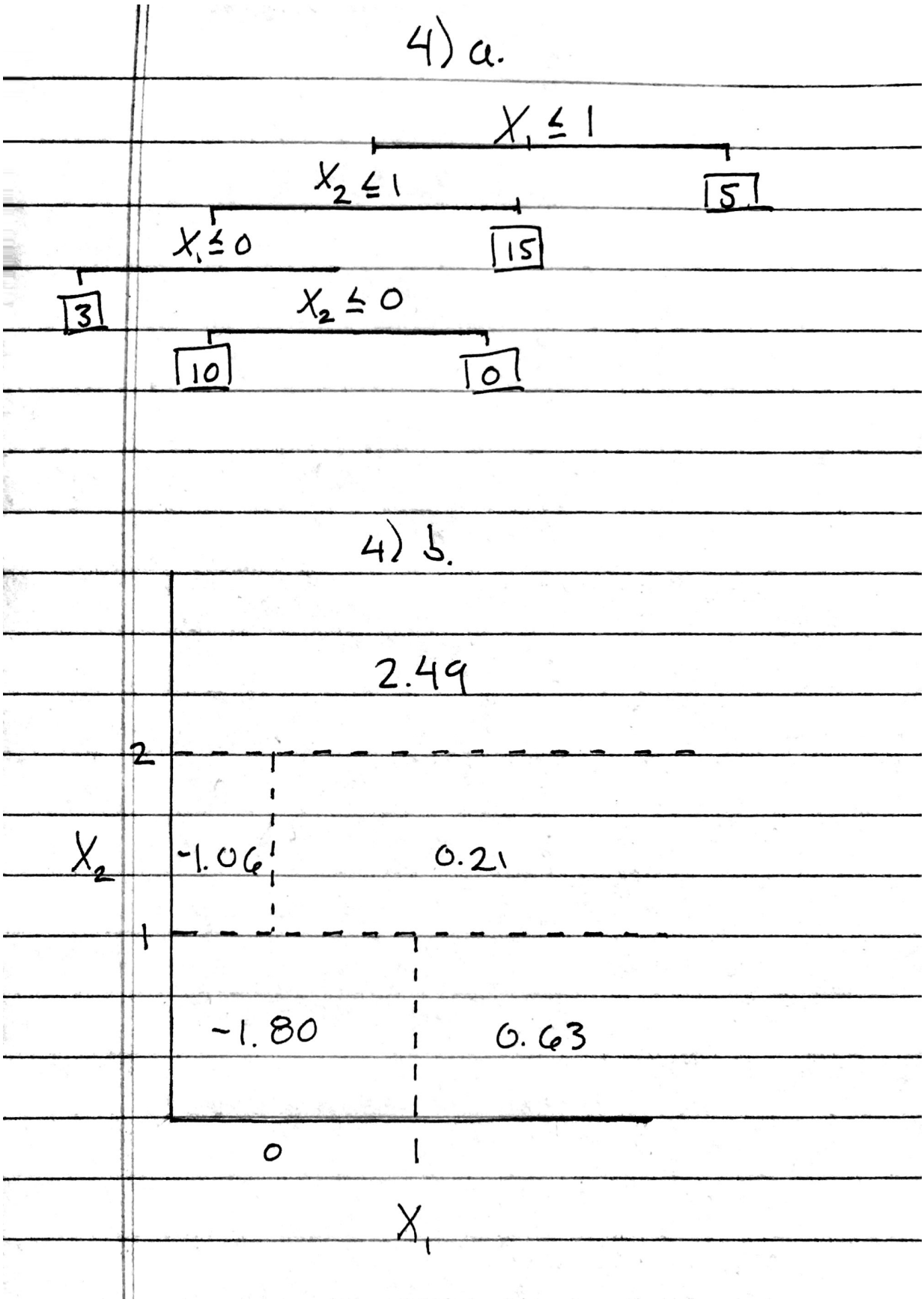


Figure 2: "Conceptual Exercise 4"

6

Broad Strokes - Use recursive binary splitting to continually split the data until there are fewer than x training observations in each terminal node, where x is predefined. The prediction for each terminal node will be the average of the response values of the training observations in that terminal node.

Detail - Starting with all the observations in the training set in one group, choose **the** value that, when split on, produces the lowest RSS among the two groups dictated by that split value.

- When the data set is split into two groups, the **single prediction** for the observations that are in group A will be the mean of the response values of the observations that are in group A, and the **single prediction** for the observations that are in group B will be the mean of the response values of the observations that are in group B.
- To calculate the (training) RSS of a particular split, sum the squared differences between the predicted value and the actual value of the response for all observations.

Now the data is parititioned into two groups. Repeat the same process again, treating each of those two groups as the *parent* group, which will be split into two *child* groups, leading to four total groups/divisions of the data set. Repeating again would lead to eight total division of the data set. Repeat this process until there are less than x training observations in each terminal node. Note that if only one of the *parent* groups has more than x training observations in it, but the other has less than x, only the group with more than x would be split further (in other words, the tree doesn't have to split evenly all the way down).

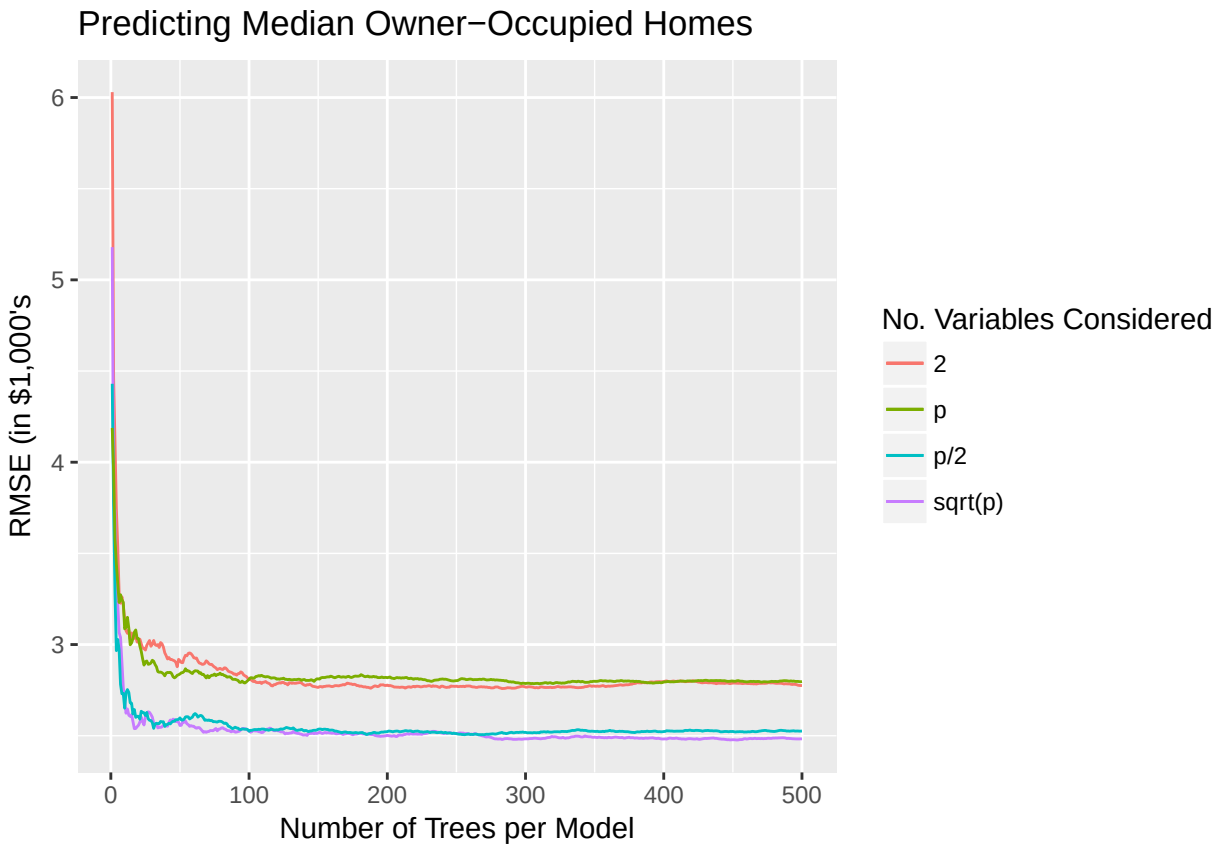
Applied

7

Looking at the plot below, it is clear that the testing error rate levels off after around 100 trees. Notably, as is characteristic of Random Forests, the classic "U" shape of the testing error is not present, showing a lack of overfitting as the number of trees in the ensemble increases.

```
# load libraries
suppressPackageStartupMessages(library(ISLR))
suppressPackageStartupMessages(library(MASS))
suppressPackageStartupMessages(library(randomForest))
attach(Boston)

set.seed(5)

# train test split
train <- sample(c(TRUE, FALSE), nrow(Boston), replace = TRUE)
X.train <- Boston[train, -14]
X.test <- Boston[-train, - 14]
y.train <- Boston[train, 14]
y.test <- Boston[-train, 14]

# modeling
rf.model.p <- randomForest(x = X.train,
                           y = y.train,
                           xtest = X.test,
                           ytest = y.test,
                           mtry = ncol(Boston) - 1,
```

```
                          ntree = 500)
rf.model.half_p <- randomForest(x = X.train,
                          y = y.train,
                          xtest = X.test,
                          ytest = y.test,
                          mtry = ceiling(ncol(Boston)/2),
                          ntree = 500)
rf.model.sqrt_p <- randomForest(x = X.train,
                          y = y.train,
                          xtest = X.test,
                          ytest = y.test,
                          mtry = ceiling(sqrt(ncol(Boston))),
                          ntree = 500)
rf.model.2 <- randomForest(x = X.train,
                          y = y.train,
                          xtest = X.test,
                          ytest = y.test,
                          mtry = 2,
                          ntree = 500)
# create dataframe
model.df <- data.frame(x = 1:500,
                       model_2 = rf.model.2$test$mse,
                       model_sqrt_p = rf.model.sqrt_p$test$mse,
                       model_half_p = rf.model.half_p$test$mse,
                       model_p = rf.model.p$test$mse)

# plot errors

ggplot(model.df, aes(x = x, y = model_2)) +
    geom_line(aes(y = sqrt(model_2), color = '2')) +
    geom_line(aes(y = sqrt(model_sqrt_p), color = 'sqrt(p)')) +
    geom_line(aes(y = sqrt(model_half_p), color = 'p/2')) +
    geom_line(aes(y = sqrt(model_p), color = 'p')) +
    guides(color = guide_legend(title = "No. Variables Considered")) +
    ggtitle("Predicting Median Owner-Occupied Homes") +
    xlab("Number of Trees per Model") +
    ylab("RMSE (in $1,000's")
```


Predicting Median Owner−Occupied Homes
RMSE (in $1,000's
6
5
4
3
0
100
200
300
400
500
Number of Trees per Model
No. Variables Considered
2
p
p/2
sqrt(p)