Homework 8 Solutions

Instructions

Due: 5pm on Wednesday, November 10th

1. Add your name between the quotation marks on the author line in the YAML above.
2. Compose your answer to each problem between the bars of red stars.
3. Commit your changes frequently.
4. Be sure to knit your .Rmd to a .pdf file.
5. Push both the .pdf and the .Rmd file to your repo on the class organization before the deadline.

Theory

Problem 1

Based on ISLR Exercise 8.1

Draw an example (of your own invention) of a partition of two-dimensional feature space that could result from recursive binary splitting. Your example should contain at least 6 regions. Draw the decision tree corresponding to this partition. Be sure to label all aspects of your figures, including the regions, the splitting points, and so forth.

There are a number of ways to add your "drawing" to your .Rmd file. You could...

1. Draw the figure by hand, take a picture/scan the figure, and then include in your .Rmd file using `include_graphics(...)`
2. Create a digital figure using a digital drawing application of your choice and include the resulting image suing `include_graphics()`.
3. Use `ggplot2` and `geom_segment` to manually create a partition plot.

Answers will obviously vary. Things to look for in the predictor space diagram:

- All regions should be rectangular and all decision boundaries parallel to x or y axis
- Partition boundaries should stop once they reach a perpendicular boundary (i.e. boundary intersections should look like T rather than +)
- Each region should be labeled with its final prediction (can either be for regression or classification)
- Decision boundary lines should have x- or y-intercepts clearly labeled (i.e. the location of the cut should be included)

Things to look for in the tree:

- All splits should be binary
- There should be at least 6 terminal nodes (and there should be a node for each region in the predictor space diagram)

- Each decision point should list a condition (i.e. $X_1 < 15.2$) involving exactly 1 of the predictors. The left branch should correspond to this condition being true, and the right branch should correspond to the condition being false
- The final predictions at the terminal nodes of the tree should match the predictions from the the predictor space diagram

Here's an example of a diagram-tree pair (it's not necessary for them to include the actual data points, but its fine if they do)

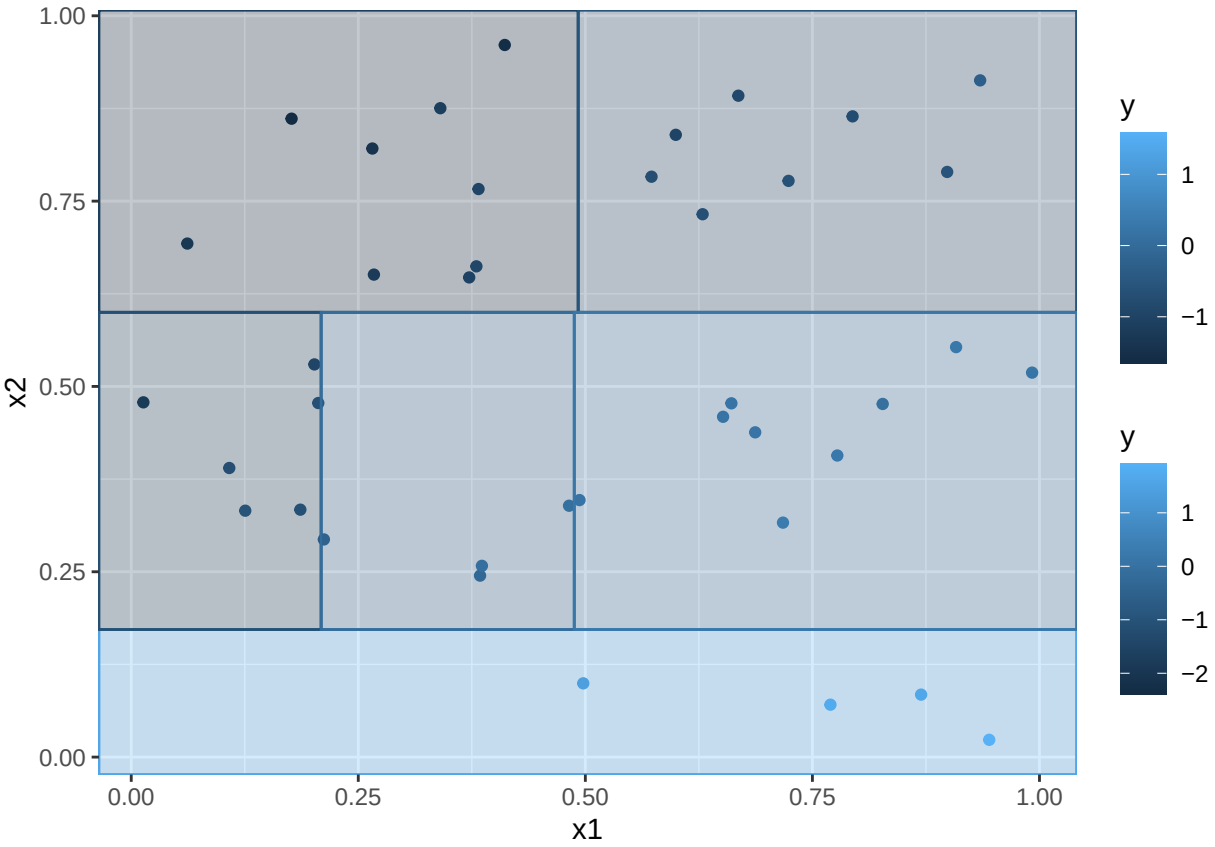
```
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.6.2
```

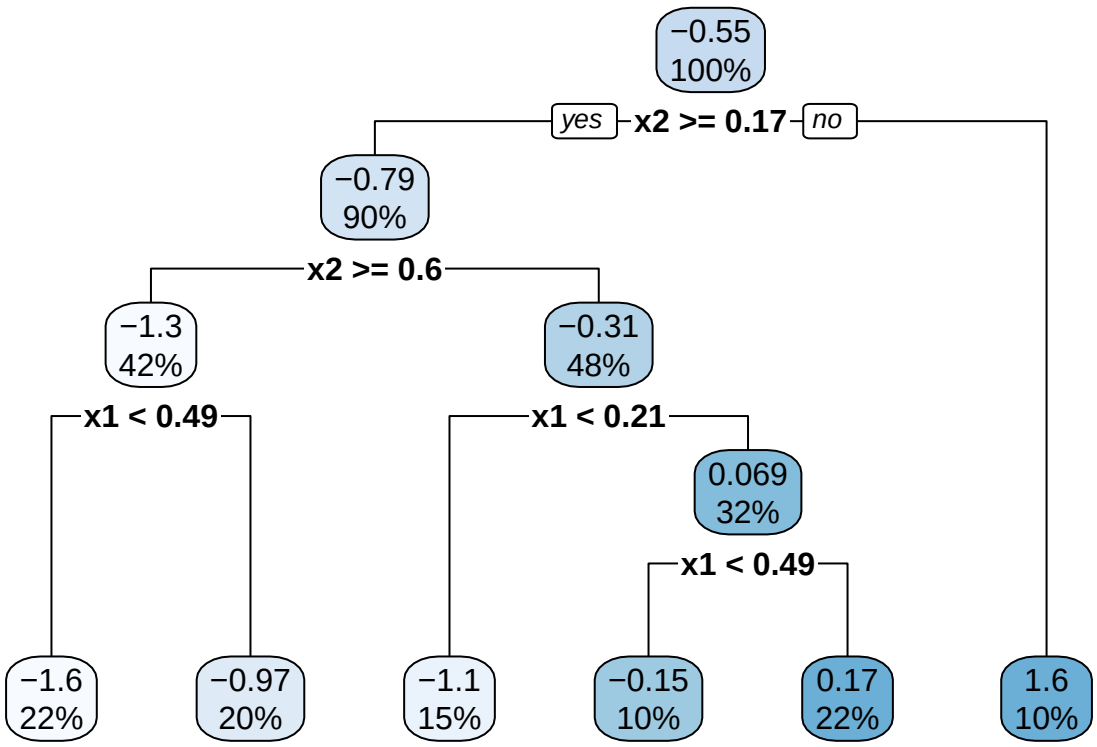
```
library(parttree)
set.seed(1)
x1<-runif(40)
x2<-runif(40)
e<-rnorm(40,0,.25)
y<-2*x1 - 3*x2 + e

fake_data<-data.frame(x1,x2,y)
fake_tree<-rpart(y~., data = fake_data, control = rpart.control(cp = 0, minsplit = 12))

fake_data %>% ggplot(aes(x = x1, y = x2, color = y))+geom_point()+geom_parttree(data = fake_tree, aes(f
```




```
rpart.plot(fake_tree)
```



Applied

Problem 2

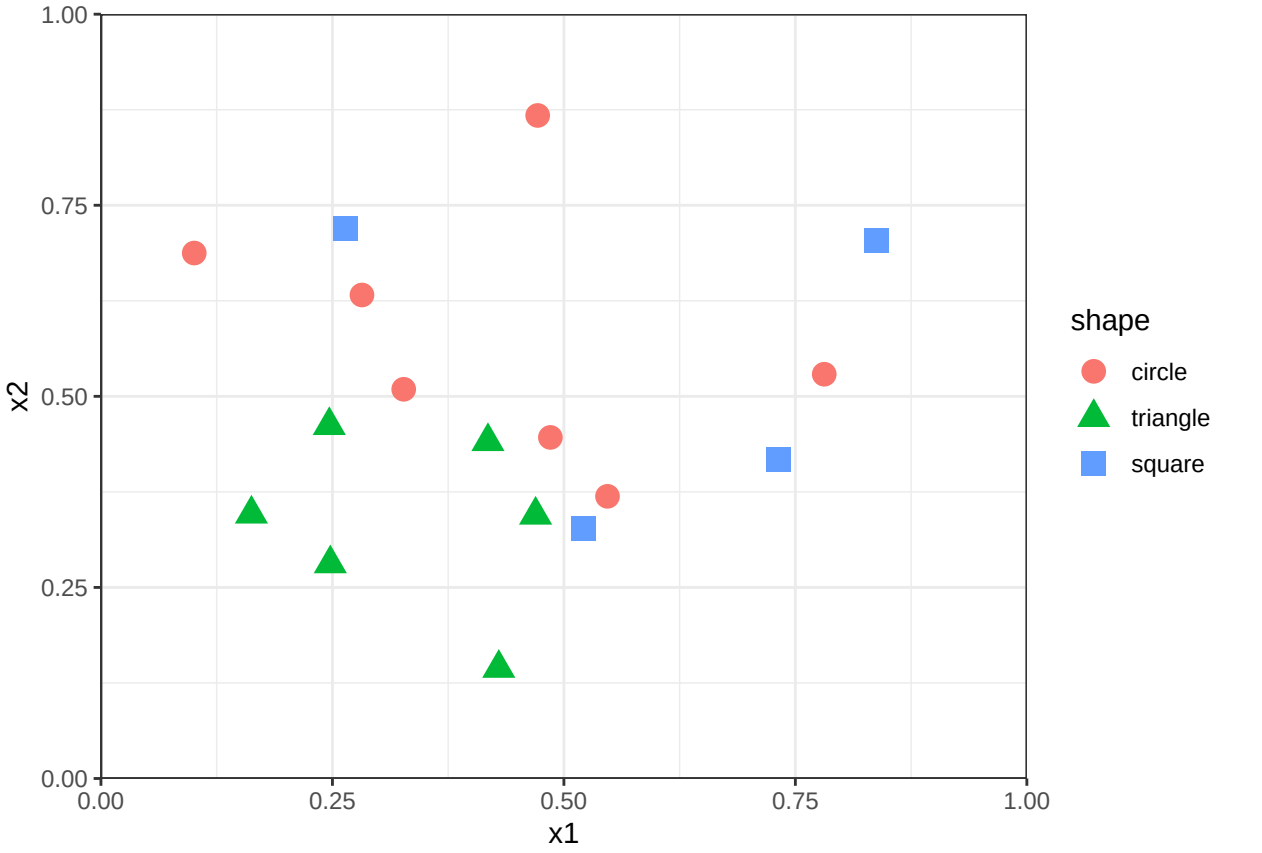
In Friday's class, we estimated by eye the first split in a classification tree for the `shapes` data set constructed and plotted using the code chunk below. Now we'll check to see if our graphical intuition agrees with that of the full classification tree algorithm.

```
set.seed(100)
n <- 6
circles <- data.frame(
  shape = "circle",
  x1 = runif(n+1, 0.05,.95),
  x2 = runif(n+1, .25,.95 )
)
triangles <- data.frame(
  shape = "triangle",
  x1 = runif(n, 0.05,.6),
  x2 = runif(n,  0.05,.6)
)
squares <- data.frame(
  shape = "square",
  x1 = runif(n-2, 0.0,.95),
  x2 = runif(n-2, 0.1,.75)
)
shapes <- rbind(circles, triangles, squares) %>% mutate(shape = factor(shape, levels = c("circle", "tri

g<- ggplot(shapes, aes(x = x1, y = x2, col = shape, shape = shape)) +
```

```
  geom_point(size = 4) +
  scale_x_continuous(expand = c(0, 0) , limits = c(0, 1)) +
  scale_y_continuous(expand = c(0, 0), limits = c(0, 1)) +
  theme_bw()

g
```



a. Use the `rpart` package in R to fit a tree **of depth 2 or 3** to this data set, making splits based on the *Gini index* (the default for `rpart`). *You will need to change the minimum split to something more appropriate, since the default is 20 will produce a tree with no splits.* Plot the resulting tree.

b. Use `geom_segment` in `ggplot2` to recreate the plot above with partition boundaries (the `ggplot` above is saved as the object `g`. You can recreate the plot in your answer just by adding multiple `geom_segment()` layers to `g` graphic, i.e. `g + geom_segment()...`)

c. Two commonly suggested splits for this data are to make a horizontal split around $X_2 \approx 0.5$ and a vertical split around $X_1 \approx 0.5$. Was either of these the first split decided upon by your classification tree?

d. What is the benefit of the second split in the tree?

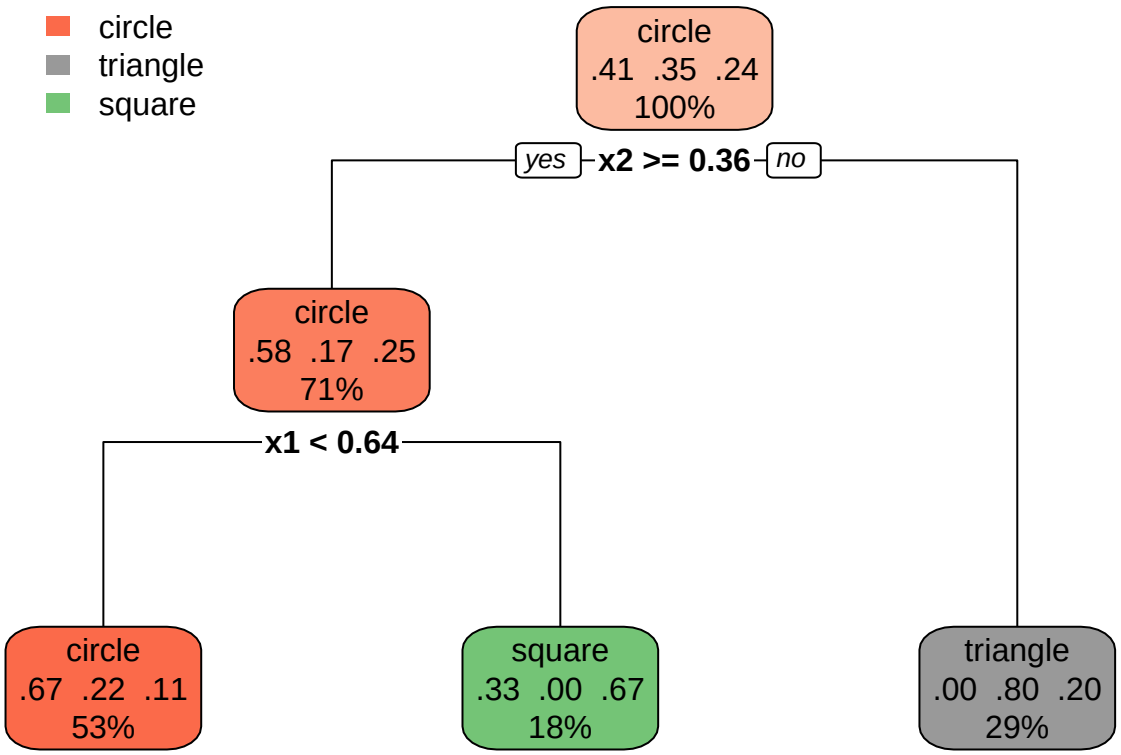
e. Which class would this model predict for the new observation with $X_1 = 0.21, X_2 = 0.56$?

f. Which classes are easiest to predict based your tree? Which classes are hard to predict or distinguish between?

---

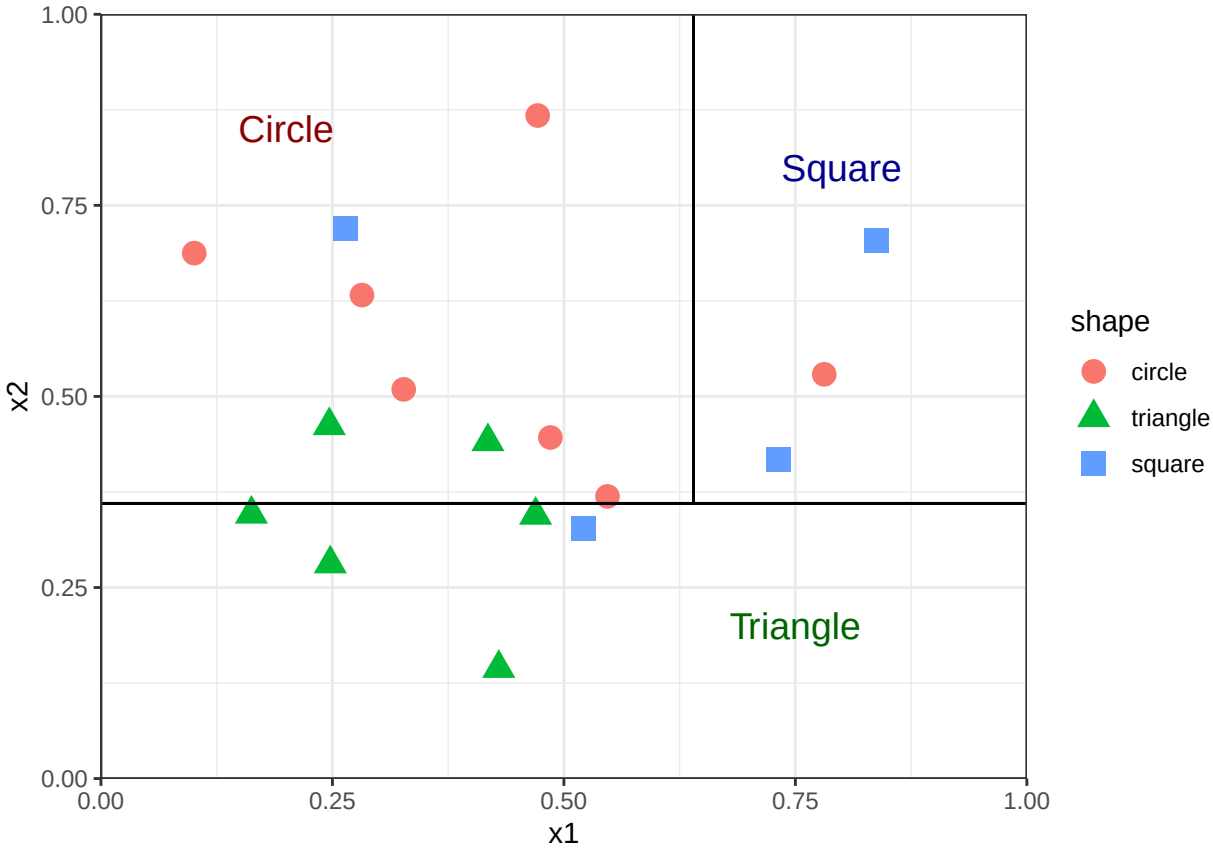
a. Choosing a minimum split between 11 and 17 will produce 2 terminal nodes and depth 2, while choosing minimal split between 5 and 10 will produce 3 terminal nodes and depth 3. Either is fine.

```
full_tree <- rpart(shape~., data = shapes, control = rpart.control(minsplit = 5))
rpart.plot(full_tree)
```



b. We set the limits of the `geom_segment` lines using the decision boundaries given the tree above (note that the segment separating the circle and square region extends from $y = 1$ down to $y = .36$, rather than all the way to $y = 0$.)

```
g +
  geom_segment(x = 0, xend = 1, y = .36, yend = .36,color = "black")+
  geom_segment(x = .64, xend = .64, y = .36, yend = 1,color = "black")+
  annotate(geom = "text", x = .75, y = .2, label = "Triangle", color = "darkgreen", size = 5)+
  annotate(geom = "text", x = .8, y = .8, label = "Square", color = "darkblue", size = 5)+
  annotate(geom = "text", x = .2, y = .85, label = "Circle", color = "darkred", size = 5)
```



c. The first split suggested by the tree was $X_2 \geq 0.36$, which resulted in some of the triangles being misclassified as circles.

d. The second split allows us to distinguish between circles and squares, leading to greater node purity.

e. The point $X_1 = .21$ and $X_2 = .56$ lies in the "circle" region of the partition plot, and so our tree would predict "circle" for this observation.

f. The tree has the easiest time predicting Triangles, since the triangle node has the greatest node purity. On the other hand, the model has a hard time distringuishing between circles and squares.

Problem 3

Based on ISLR Exercise 8.9

This problem uses the `OJ` (as in Orange Juice, not the 1990s murder trial defendant) data set from the `ISLR2` package. If you haven't used `ISLR2` before, you may need to first install the package by running the code `install.packages("ISLR2")`.

Load the data by running the following code chunk:

```
#If you haven't used ISLR2 before, you may need to first install
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 3.6.2
```

```
data(OJ)
```

The data set contains information for 18 variables on 1070 purchases of either Citrus Hill or Minute Maid Orange Juice.

a. Split the data into a training and test set.

b. Fit a tree to the training data, with `Purchase` as the response and all other variables as predictors.

c. Create a plot of the tree.

d. What is the training error rate? How many terminal nodes does the tree have?

e. Give a 2 - 3 sentence description of the decision rules used to predict whether a customer purchases Citrus Hill or Minute Maid. Your description should be aimed at a non-statistical audience.

f. Make predictions on the test data and produce a confusion matrix of the results. What is the test error rate?

g. Plot the cross-validated error rates for the tree, and determine the optimal size for the tree.

h. Create a pruned tree corresponding to the size you selected in the previous part. Plot the pruned tree.

i. Compare the training error rates between the pruned and unpruned trees. Which is higher? Explain why this occurs.

j. Compare the test error rates between the pruned and unpruned trees. Which is higher. Explain why this occurs.

---

a.

```
set.seed(10)
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 3.6.2
```

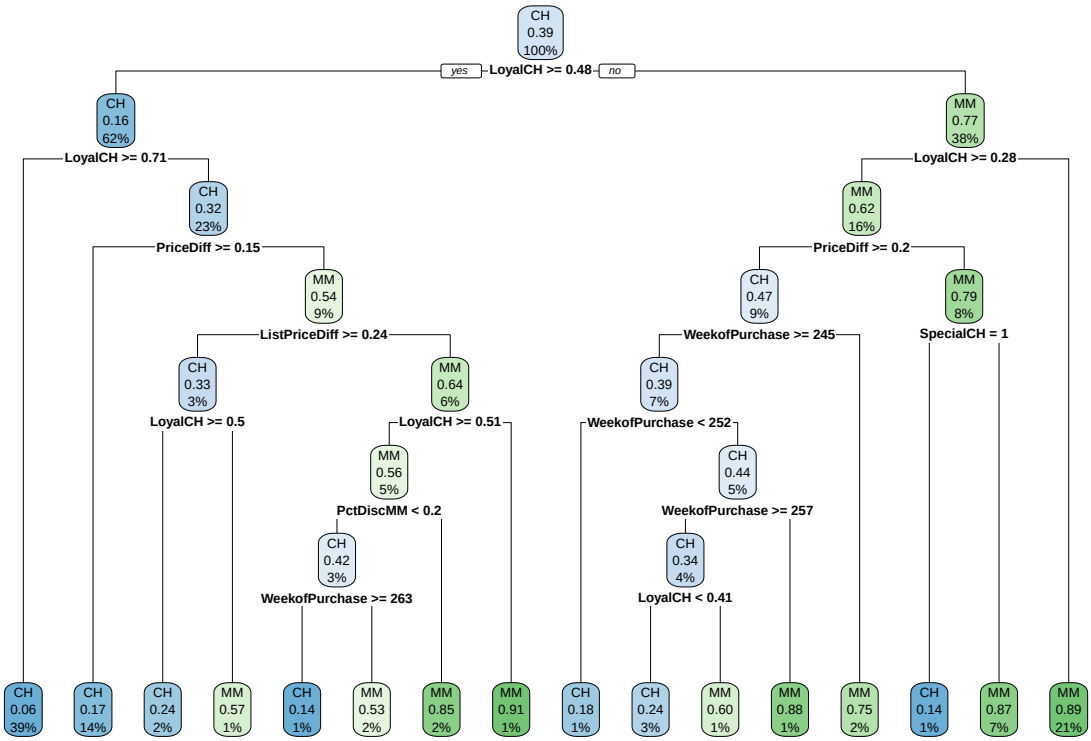
```
OJ_initial <- initial_split(OJ, strata = Purchase)
OJ_train <- training(OJ_initial)
OJ_test <- testing(OJ_initial)
```

b. We will grow the full tree by setting the cp threshold to 0.

```
library(rpart)
OJ_tree <- rpart(Purchase ~., data = OJ_train, control = rpart.control(cp = 0))
```

c.

```
library(rpart.plot)
rpart.plot(OJ_tree)
```



d. The relative training error for the full tree corresponds to the `rel error` entry for tree with 20 splits, which is 0.32907. The baseline, the root node error for the simple tree with no splits, is 0.38979. And so the training error rate for full tree is $0.39 * 0.33 = 0.128$.

```
0.38979*0.32907
```

```
## [1] 0.1282682
```

```
printcp(OJ_tree)
```

```
##
## Classification tree:
## rpart(formula = Purchase ~ ., data = OJ_train, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] ListPriceDiff  LoyalCH        PctDiscMM      PriceDiff      SpecialCH
## [6] WeekofPurchase
##
## Root node error: 313/803 = 0.38979
##
## n= 803
##
##          CP nsplit rel error  xerror     xstd
## 1 0.5271565      0   1.00000 1.00000 0.044154
## 2 0.0149095      1   0.47284 0.49840 0.035819
## 3 0.0127796      4   0.42812 0.50160 0.035906
## 4 0.0095847      8   0.37380 0.47604 0.035195
## 5 0.0063898     10   0.35463 0.47284 0.035103
## 6 0.0031949     13   0.33546 0.46965 0.035011
## 7 0.0000000     15   0.32907 0.50479 0.035992
```

e. Since the first split is made along the `LoyalCH` variable, it appears brand loyalty is the most important

feature for predicting purchase brand. In particular, if brand loyalty for Citrus Hills is high, it is more likely than not that the customer will purchase Citrus Hills. After taking brand loyalty into account, for individauls with mixed brand loyalty (i.e `LoyalCH` between about 0.25 and 0.75), the price difference between Minute Maid and Citrus Hills is the next most important factor.

f. The test error rate is 0.2059925

```
library(yardstick)
```

```
## Warning: package 'yardstick' was built under R version 3.6.2
```

```
## For binary classification, the first factor level is assumed to be the event.
## Use the argument `event_level = "second"` to alter this as needed.
```

```
##
## Attaching package: 'yardstick'
```

```
## The following object is masked from 'package:readr':
##
##     spec
```

```
OJ_preds <- predict(OJ_tree, OJ_test, type = "class")
OJ_probs <- predict(OJ_tree, OJ_test, type = "prob")[,2] #prob of MM

results_OJ <- data.frame(obs = OJ_test$Purchase, preds = OJ_preds, probs = OJ_probs)

conf_mat(results_OJ, truth = obs, estimate = preds)
```

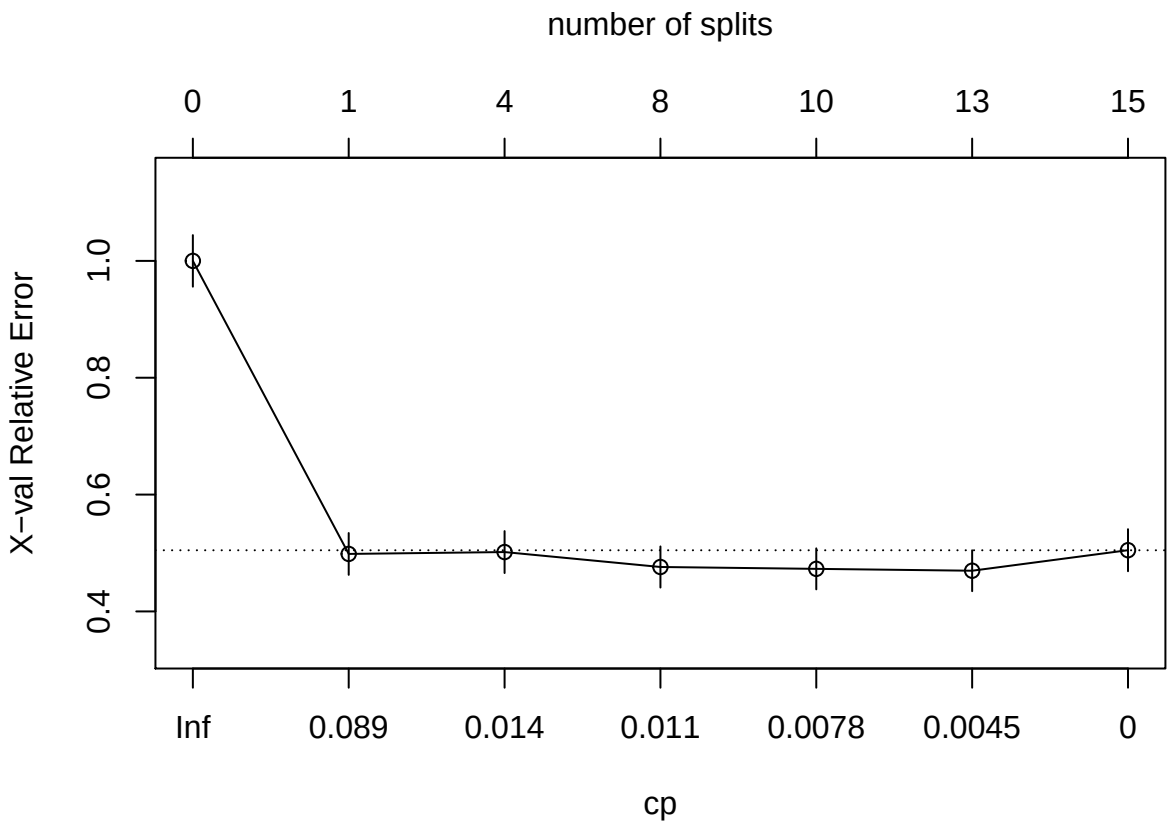
```
##           Truth
## Prediction  CH  MM
##         CH 136  28
##         MM  27  76
```

```
acc <- accuracy(results_OJ, truth = obs, estimate = preds) %>% pull(.estimate)
1-acc
```

```
## [1] 0.2059925
```

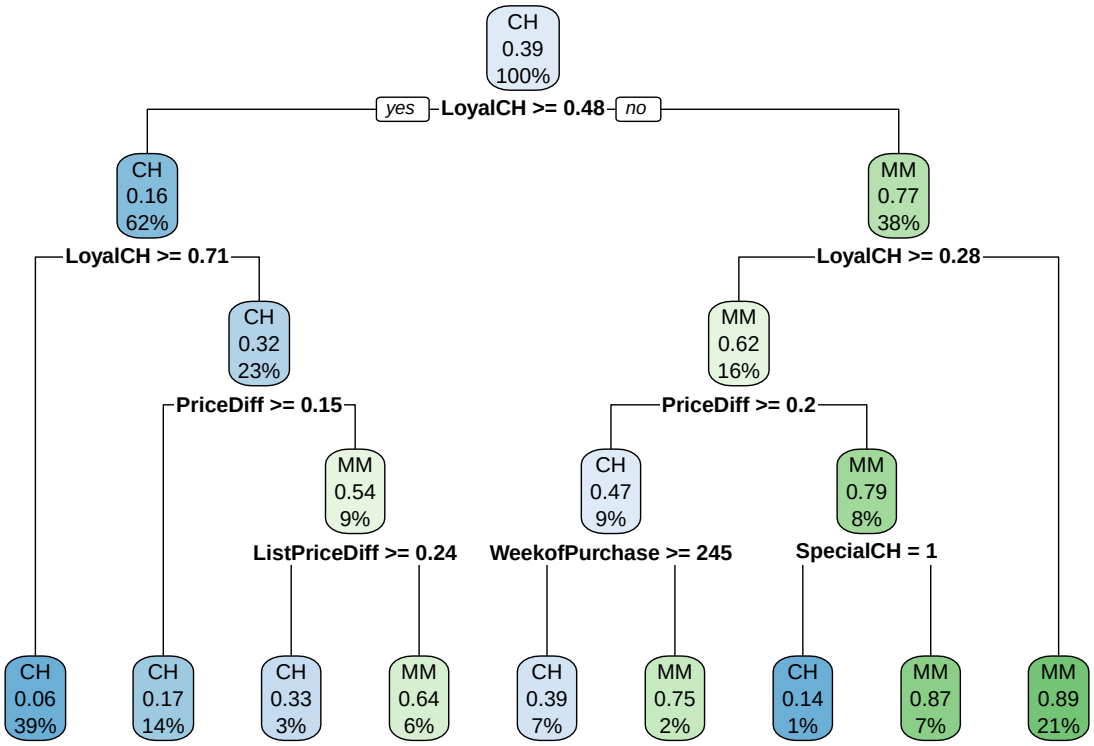
g. Based on cross-validation, it appears a tree with 8 splits is optimal, corresponding to a CP threshhold of 0.011.

```
plotcp(OJ_tree, upper = "splits")
```



h.

```
pruned_OJ_tree <- prune(OJ_tree, cp = 0.011)
rpart.plot(pruned_OJ_tree)
```



i. The relative error training error for the pruned tree is 0.37, compared to the relative error of 0.33 for the full tree. Both have the same baseline error of 0.3879, which indicates that the pruned tree has **higher** training error rate than the full tree. Of course, this makes sense, since deeper trees are better able to adapt to traing data.

```
printcp(pruned_OJ_tree)
```

```
##
## Classification tree:
## rpart(formula = Purchase ~ ., data = OJ_train, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] ListPriceDiff  LoyalCH        PriceDiff      SpecialCH      WeekofPurchase
##
## Root node error: 313/803 = 0.38979
##
## n= 803
##
##         CP nsplit rel error  xerror     xstd
## 1 0.527157      0   1.00000 1.00000 0.044154
## 2 0.014909      1   0.47284 0.49840 0.035819
## 3 0.012780      4   0.42812 0.50160 0.035906
## 4 0.011000      8   0.37380 0.47604 0.035195
```

j. Not surprisingly, the pruned tree has higher accuracy than the full tree. However, the difference in accuracy is relatively small (and could very well just be due to the random test-training split. Perhaps this should be further verified by cross-validation!)

```
new_OJ_preds <- predict(pruned_OJ_tree, OJ_test, type = "class")
new_OJ_probs <- predict(pruned_OJ_tree, OJ_test, type = "prob")[,2] #prob of MM

results_OJ <- results_OJ %>% mutate(model = "full tree") %>%
  rbind(data.frame(obs = OJ_test$Purchase,
                   preds = new_OJ_preds,
                   probs = new_OJ_probs,
                   model = "pruned tree"))

results_OJ %>% group_by(model) %>%  accuracy(truth = obs, estimate = preds)
```

```
## # A tibble: 2 x 4
##   model       .metric  .estimator .estimate
##   <chr>       <chr>    <chr>          <dbl>
## 1 full tree   accuracy binary         0.794
## 2 pruned tree accuracy binary         0.816
```

Probelm 4

For this exercise, we will use the `College` data from the ISLR package. Familiarize yourself with this dataset before performing analysis. We will attempt to predict the `Outstate` variable, which indicates the college's out-of-state tutition rate in 1995. Load the data with the following code:

```
library(ISLR2)
data("College")
```

a. Split the data into a training and test set.

b. Create the following **four** models on the training data and tune relevant hyperparameters using cross-validation.

- The full linear model
- A well-tuned LASSO model
- A well-tuned KNN model
- A well-tuned decision tree

c. Calculate the test RMSE for each model and compare. Which model performs best on test data? What features seem most important for determining out-of-state tuition?

---

a.

```
College_initial <- initial_split(College)
College_train <- training(College_initial)
College_test <- testing(College_initial)
```

b.

Linear Model

```
college_lm <- lm(Outstate ~ ., data = College_train)
college_preds_lm <- predict(college_lm, College_test)
```

LASSO

The `Private` variable appears to be most influential in the model (which matches conventional wisdom: Private universities are far more expensive than public ones, since public universities are partially subsidized by state funding).

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.6.2
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 3.6.2
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

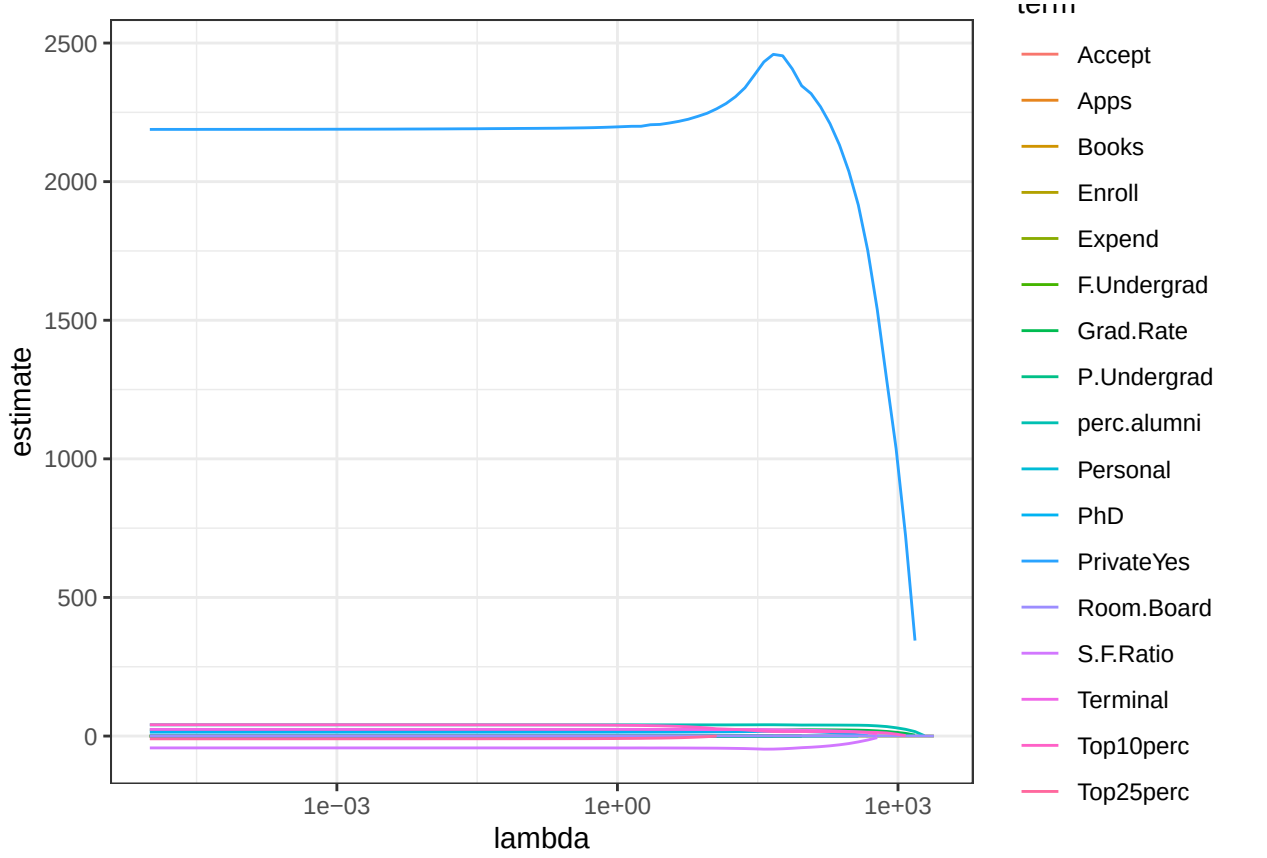
```
## Loaded glmnet 4.1-1
```

```
x_train <- model.matrix(Outstate ~ ., data = College_train)[,-1]
y_train <- College_train$Outstate
grid <- 10^seq(-5, 5, length = 100)
College_lasso <- glmnet(x_train, y_train, lambda = grid, alpha = 1)
```

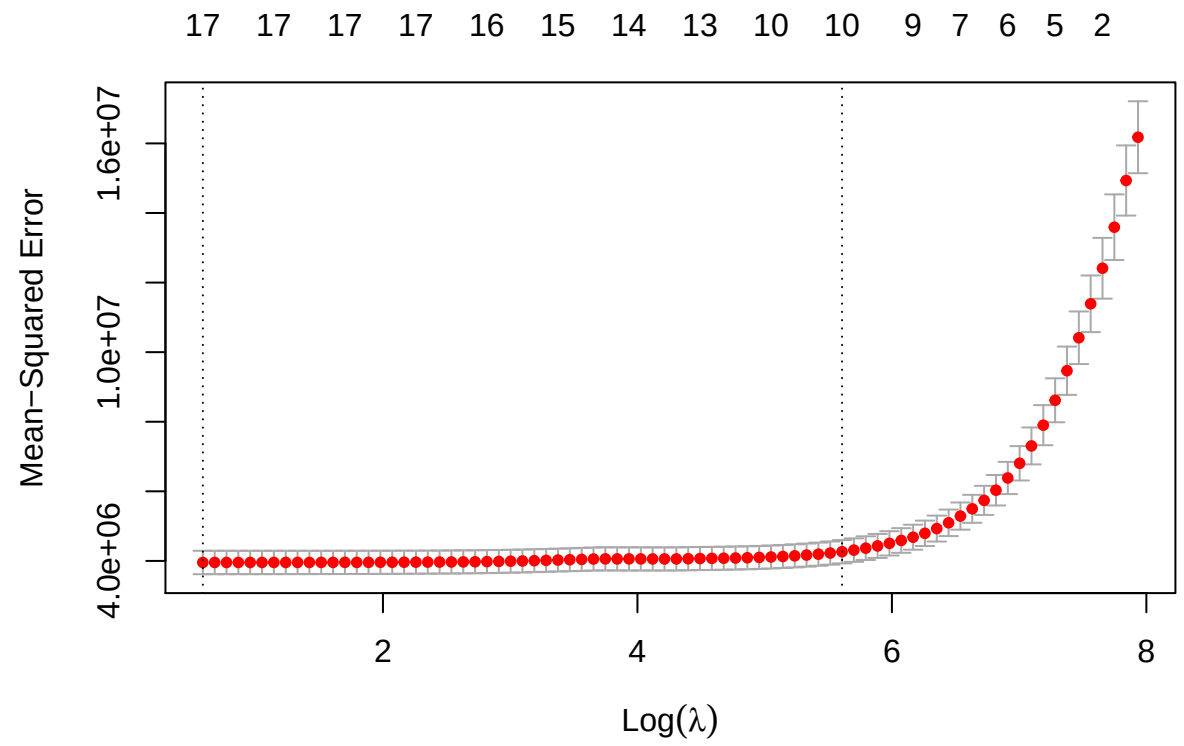
```
library(broom)
```

```
## Warning: package 'broom' was built under R version 3.6.2
```

```
tidy_lasso <- tidy(College_lasso) %>% filter(term != "(Intercept)")
tidy_lasso %>% ggplot(aes(x = lambda, y = estimate, color = term))+geom_line()+theme_bw()+scale_x_log10
```




```
set.seed(1)
College_lasso_cv <- cv.glmnet(x_train, y_train, alpha = 1)
plot(College_lasso_cv)
```



```
best_l <- College_lasso_cv$lambda.1se
```

```
x_test <- model.matrix(Outstate ~., data = College_test)[,-1]
y_test <- College_test$Outstate

college_preds_lasso <- predict.glmnet(College_lasso, newx=x_test, s = best_l)
```

KNN

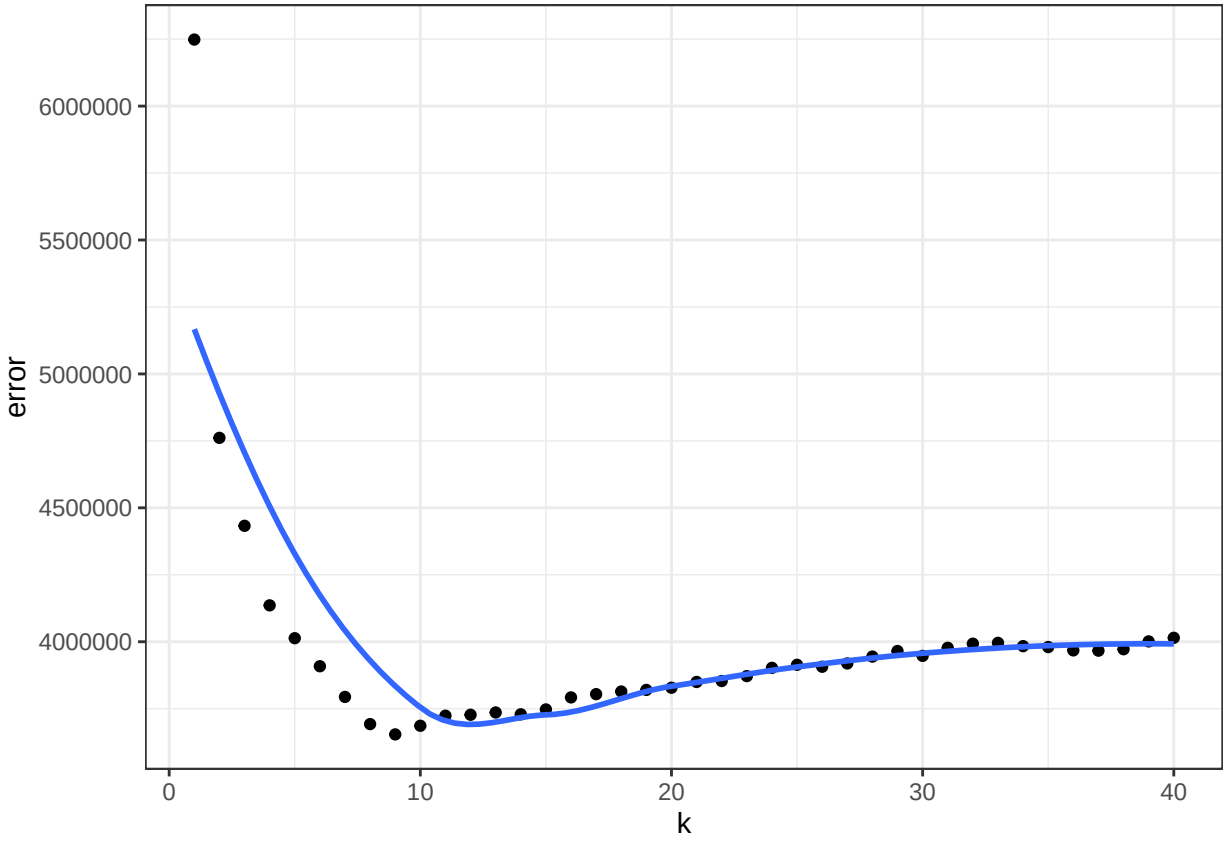
```
library(kknn)

College_knn_tune <- train.kknn(Outstate~., data = College_train, kmax = 40, kernel = "rectangular")

best_k <- College_knn_tune$best.parameters$k
```

```
as.data.frame(College_knn_tune$MEAN.SQU) %>%
  rownames_to_column(var = "k") %>%
  mutate(error = rectangular, k = as.numeric(k)) %>%
ggplot( aes(x = k, y = error))+geom_point()+geom_smooth(se = F)+theme_bw()
```

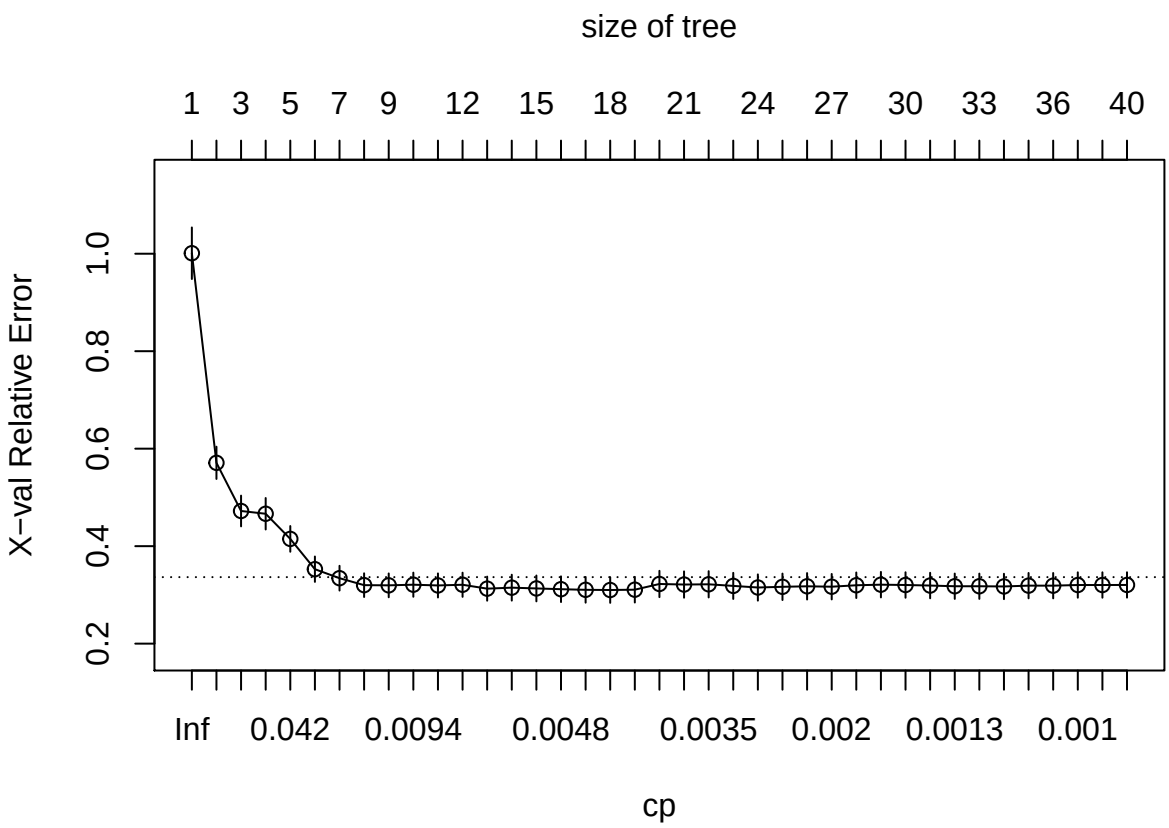
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
college_preds_knn <- kknn(Outstate ~., train = College_train, test = College_test, k = best_k, kernel =
```

Regression Tree

```
College_tree <- rpart(Outstate ~., data = College_train, control = rpart.control(cp = .001))
plotcp(College_tree)
```




```
best_cp <- 0.01
College_tree_pruned <- prune(College_tree, cp = 0.01)
```

```
college_preds_tree <- predict(College_tree_pruned, College_test)
```

c. We'll create a results data frame for each model, and the use `rmse` from `yardstick`. We see that the KNN model performed best, followed by the decision tree, the LASSO model, and finally the linear model. Based on the decision tree, `Expend` (instructional expenditure per student) and `Private` (whether the college is public or private) are the two variables with greatest importance.

```
#Variable Importance, rescaled to sum to 100
College_tree_pruned$variable.importance/sum(College_tree_pruned$variable.importance)*100
```

```
##      Expend     Private   S.F.Ratio   Top10perc         PhD    Terminal
## 24.64678988  9.17705810  8.10519847  7.86745306  7.55709410  7.21620545
##   Top25perc F.Undergrad      Enroll      Accept P.Undergrad        Apps
##  6.03421372  5.89920602  5.26074628  4.61472540  4.52077015  3.71280086
##  Room.Board   Grad.Rate perc.alumni    Personal
##  3.24776246  1.05915362  1.01142545  0.06939698
```

```
results<-data.frame(
  obs = College_test$Outstate,
  linear = college_preds_lm,
  lasso = as.numeric(college_preds_lasso),
  knn = college_preds_knn,
  tree = college_preds_tree
) %>% pivot_longer(!obs, names_to = "model", values_to = "preds")

results %>% group_by(model) %>% rmse(truth = obs, estimate = preds) %>% arrange(.estimate)
```

```
## # A tibble: 4 x 4
##   model  .metric .estimator .estimate
##   <chr>  <chr>   <chr>          <dbl>
## 1 knn    rmse    standard       1950.
## 2 tree   rmse    standard       2037.
## 3 linear rmse    standard       2043.
## 4 lasso  rmse    standard       2149.
```
