Homework 4

Daniel Dulaney

October 25, 2020

```r
library(tidyverse)
library(tidymodels)
library(ISLR)
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.0.3
```

```r
library(kableExtra)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.3
```

8.4 drawn by hand and attached above.

8.5

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

sum(probs > .5)
```

```
## [1] 6
```

```r
sum(!(probs > .5))
```

```
## [1] 4
```

Majority ultimately classifies red.

```r
mean(probs)
```

```
## [1] 0.45
```

Average probabililty ultimately classifies green.

8.8

(a)

```r
carseats <- as_tibble(Carseats)

split <- initial_split(carseats)

carseats_train <- training(split)
carseats_test <- testing(split)
```

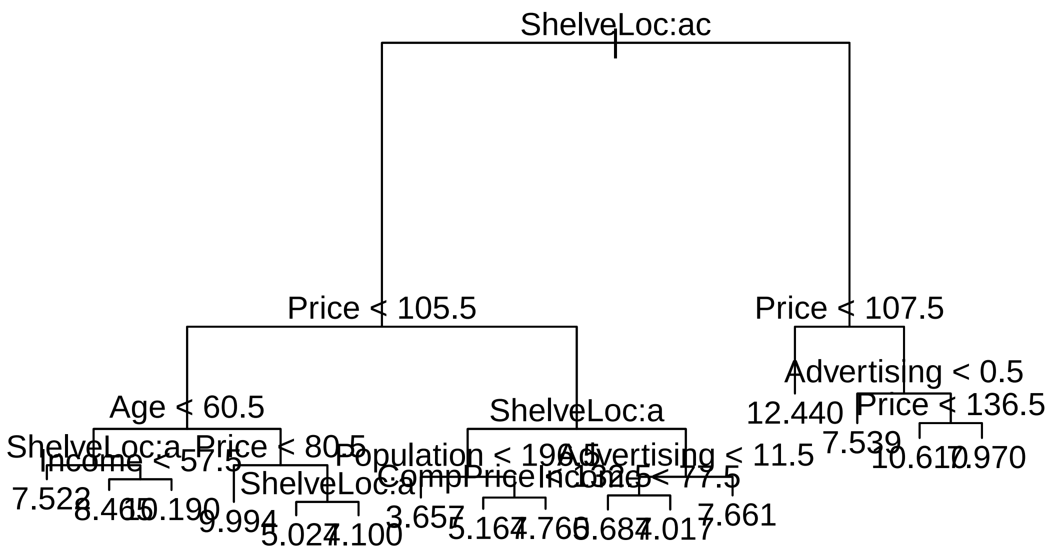
(b)

```r
tree_carsets <- tree(Sales ~ ., data = carseats_train)

summary(tree_carsets)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = carseats_train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Income"      "Population"
## [6] "CompPrice"   "Advertising"
## Number of terminal nodes:  16
## Residual mean deviance:  2.765 = 785.2 / 284
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4.24100 -1.16200  0.02045  0.00000  1.09800  4.25300
```

```r
plot(tree_carsets)
text(tree_carsets)
```




```r
preds_carsets <- predict(tree_carsets, carseats_test)

# calculate MSE
mean((carseats_test$Sales - preds_carsets)^2)
```

```
## [1] 4.597958
```

MSE is 5.11

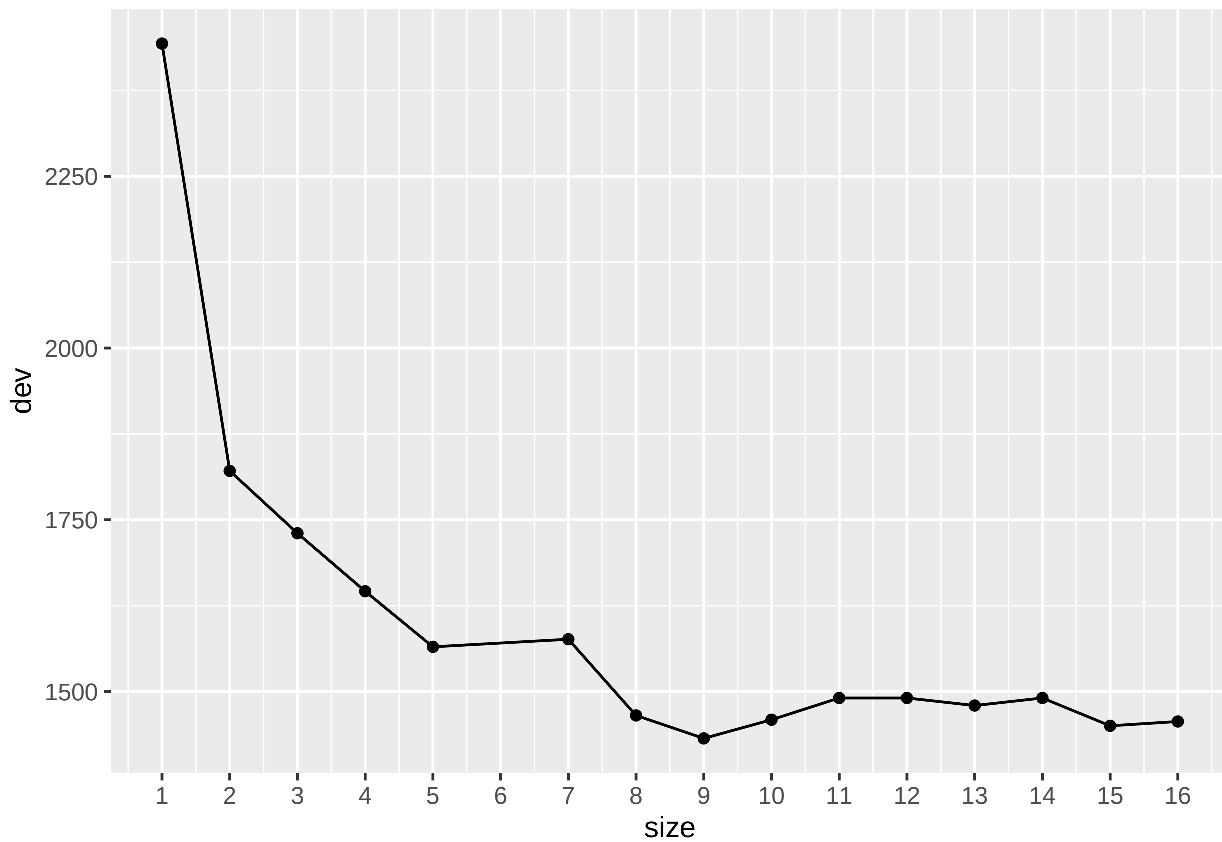
(c)

```r
cv_carseats <- cv.tree(tree_carsets, FUN = prune.tree)

cv_metrics <- cbind(cv_carseats$dev, cv_carseats$size, cv_carseats$k) %>%
  as_tibble() %>%
  rename(dev = V1, size = V2, k = V3)
```

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_repair` is
## Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

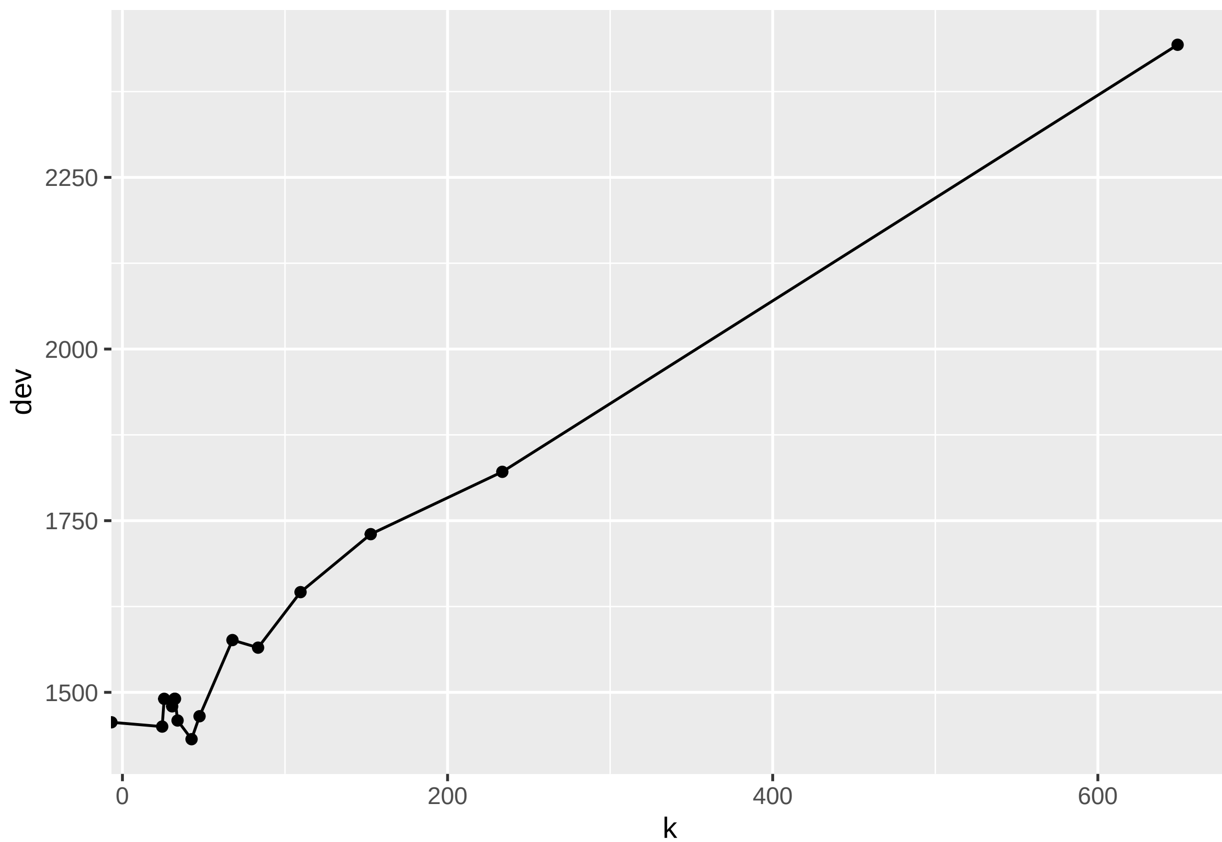
```r
# plot dev vs size
cv_metrics %>%
  ggplot(aes(size, dev)) +
  geom_path() +
  geom_point() +
  scale_x_continuous(breaks = seq(1, 16))
```




```r
cv_metrics %>%
  arrange(dev) %>%
  select(size, dev) %>%
  kable() %>%
  kable_styling()
```

size	dev
9	1431.808
15	1450.153
16	1456.423
10	1458.988
8	1465.226
13	1479.644
12	1490.601
11	1490.601
14	1490.615
5	1565.120
7	1576.145
4	1645.957
3	1730.413
2	1821.193
1	2443.149

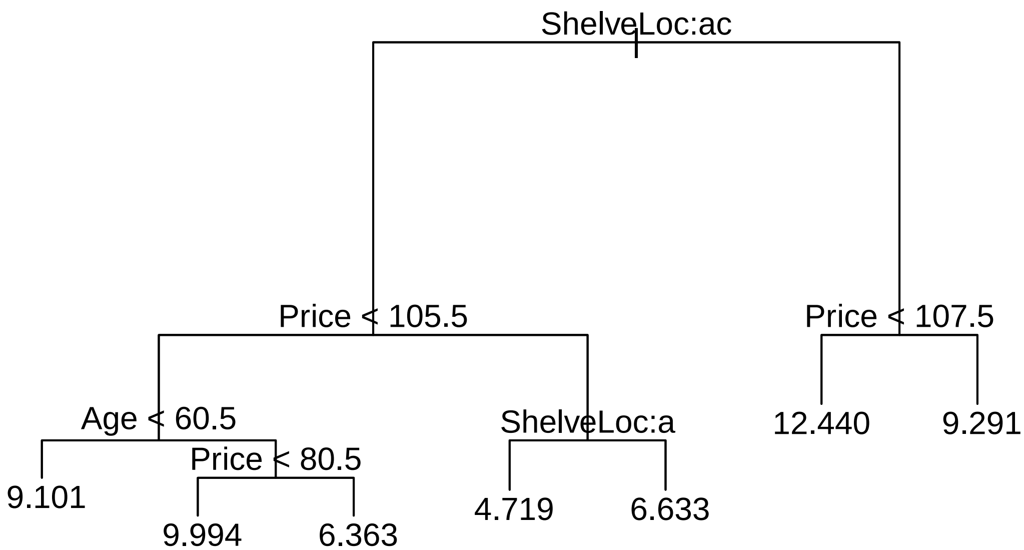
```
# plot dev vs k
cv_metrics %>%
  ggplot(aes(k, dev)) +
  geom_path() +
  geom_point()
```



The best size appears to be 6.

```r
pruned_carseats <- prune.tree(tree_carsets, best = 6)

plot(pruned_carseats)
text(pruned_carseats)
```




```r
preds_pruned <- predict(pruned_carseats, carseats_test)
mean((carseats_test$Sales - preds_pruned)^2)
```

```
## [1] 4.527111
```

MSE not improved by pruning, higher at 5.3 vs 5.1 before.

(d)

```r
bag_carseats <- randomForest(Sales ~ ., data = carseats_train, mtry = 10, ntree = 500, importance = T)
preds_bag <- predict(bag_carseats, carseats_test)
mean((carseats_test$Sales - preds_bag)^2)
```

```
## [1] 2.281601
```

```r
importance(bag_carseats)
```

```
##                %IncMSE IncNodePurity
## CompPrice   30.0877308    201.056382
## Income      13.5291325    151.607555
## Advertising 18.3254185    172.235576
## Population  -2.0913376     86.018943
## Price       69.4001545    685.188520
## ShelveLoc   82.5353954    793.907256
## Age         18.1992826    198.666229
## Education   -0.4424069     56.242492
## Urban       -3.3351715      8.383944
## US           4.6808841     11.898710
```

Test MSE is 2.56, and the most important variables are listed above. 3 most important are `CompPrice`, `Income`, and `Advertising`.

(e)

```r
# everything the same as (d) but mtry = 5 now for RF instead of bagging
rf_carseats <- randomForest(Sales ~ ., data = carseats_train, mtry = 5, ntree = 500, importance = T)
preds_rf <- predict(rf_carseats, carseats_test)
mean((carseats_test$Sales - preds_rf)^2)
```

```
## [1] 2.479093
```

```r
importance(rf_carseats)
```

```
##               %IncMSE IncNodePurity
## CompPrice   22.234624     200.56917
## Income      10.867437     167.00352
## Advertising 19.367243     197.23497
## Population   2.236126     117.95800
## Price       53.356812     629.28738
## ShelveLoc   64.915517     708.54880
## Age         18.486754     221.14174
## Education    2.369627      75.43932
## Urban       -2.526406      11.54964
## US           6.104067      25.28383
```

MSE is slightly higher at 2.60, while the 3 most important variables remain the same. In this example, lowering m worsened test MSE.

8.9

(a)

```r
oj <- as_tibble(ISLR::OJ)

oj_split <- initial_split(oj, prop = 800/nrow(oj))
oj_train <- training(oj_split)
oj_test <- testing(oj_split)
```

(b)

```r
tree_oj <- tree(Purchase ~ ., data = oj_train)

summary(tree_oj)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = oj_train)
## Variables actually used in tree construction:
## [1] "LoyalCH"     "PriceDiff"   "SalePriceMM" "DiscCH"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7112 = 563.2 / 792
## Misclassification error rate: 0.1575 = 126 / 800
```

Training error rate is 18.1%, while the variables used in construction were `LoyalCH`, `PriceDiff`, `ListPriceDiff`, `PctDiscMM`, and `StoreID`. There are 9 terminal nodes.

(c)

```r
tree_oj
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1057.00 CH ( 0.62625 0.37375 )
##    2) LoyalCH < 0.461965 279  295.60 MM ( 0.22222 0.77778 )
##      4) LoyalCH < 0.051325 59    0.00 MM ( 0.00000 1.00000 ) *
##      5) LoyalCH > 0.051325 220  261.70 MM ( 0.28182 0.71818 )
##       10) PriceDiff < 0.31 172  181.70 MM ( 0.22093 0.77907 ) *
##       11) PriceDiff > 0.31 48   66.54 CH ( 0.50000 0.50000 ) *
##    3) LoyalCH > 0.461965 521  453.60 CH ( 0.84261 0.15739 )
##      6) LoyalCH < 0.764572 250  300.20 CH ( 0.71200 0.28800 )
##       12) PriceDiff < -0.165 38   43.80 MM ( 0.26316 0.73684 ) *
##       13) PriceDiff > -0.165 212  216.50 CH ( 0.79245 0.20755 )
##         26) SalePriceMM < 2.125 124  152.80 CH ( 0.69355 0.30645 )
##           52) DiscCH < 0.115 110  141.80 CH ( 0.65455 0.34545 ) *
##           53) DiscCH > 0.115 14    0.00 CH ( 1.00000 0.00000 ) *
##         27) SalePriceMM > 2.125 88   43.81 CH ( 0.93182 0.06818 ) *
##      7) LoyalCH > 0.764572 271   85.62 CH ( 0.96310 0.03690 ) *
```

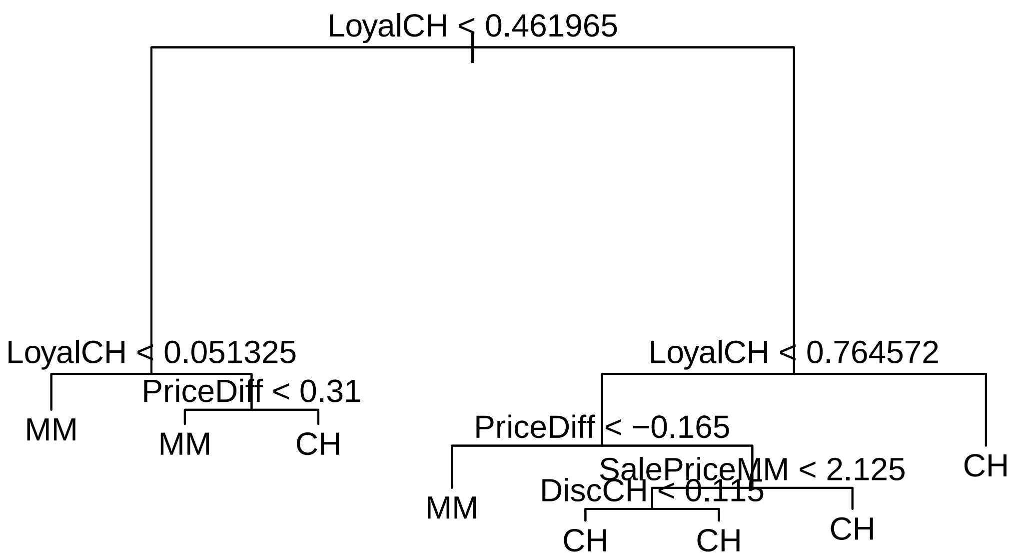
Looking at node 7), the last from the output above.

The split variable is `LoyalCH`, and the split value is > .76. There are 246 points in this subsection of the tree. The * at the end indicates this is a terminal node.

The prediction at this point is `Purchase` = `CH`, which is the case for 96% of points in this node.

(d)

```r
plot(tree_oj)
text(tree_oj)
```



Can see that `LoyalCH` is by far the most important variable in this tree model. The top nodes are all `LoyalCH` indicators.

On the left of this graph, for instance, we see classification of `MM` for any `LoyalCH` < .06, or any `LoyalCH` between .06 and .27 if the `PriceDiff` > .05.

(e)

```r
preds_oj <- predict(tree_oj, oj_test, type = "class")
table(oj_test$Purchase, preds_oj)
```

```
##      preds_oj
##        CH  MM
##   CH 129  23
##   MM  32  86
```

Test error rate is $21+8/270 = 10.7\%$

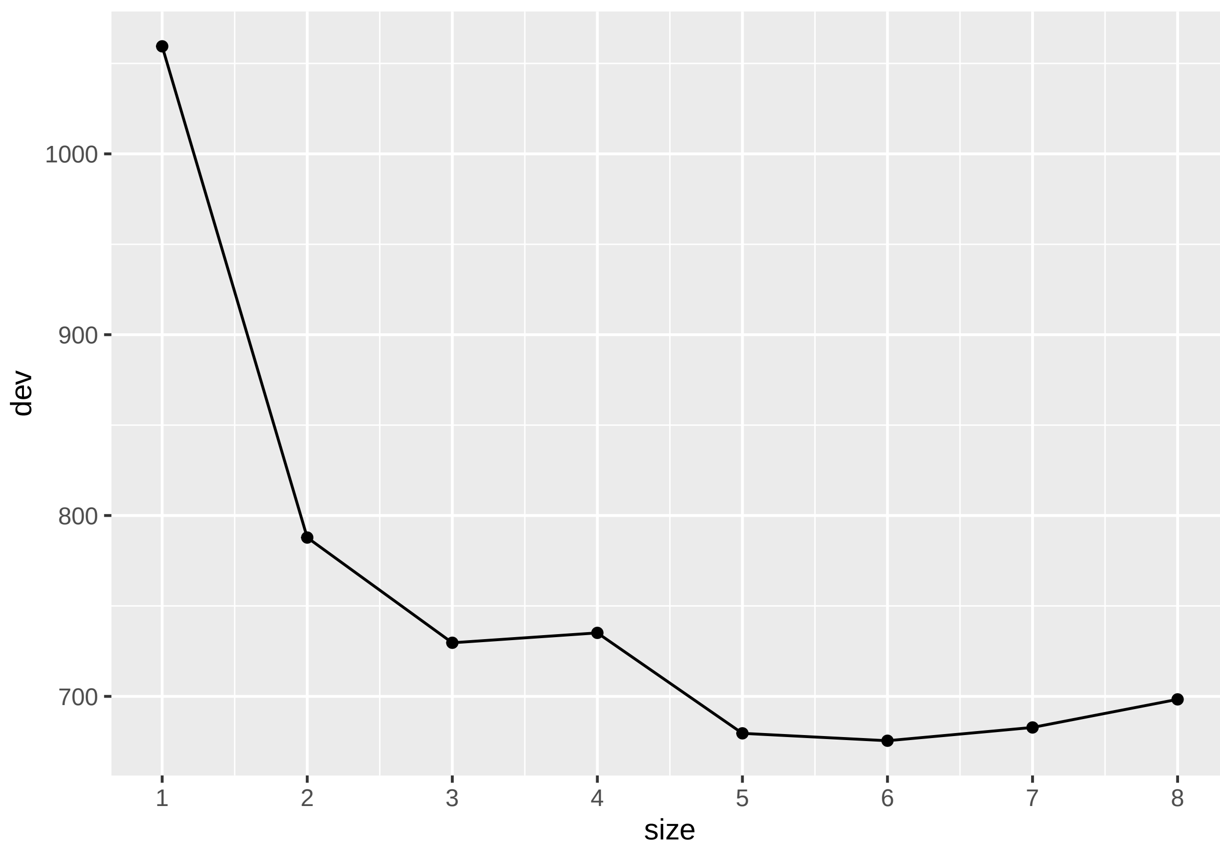
(f)

```r
cv_oj = cv.tree(tree_oj, FUN = prune.tree)

cv_metrics <- cbind(cv_oj$dev, cv_oj$size, cv_oj$k) %>%
  as_tibble() %>%
  rename(dev = V1,
         size = V2,
         k = V3)
```

(g)

```r
ggplot(aes(size, dev), data = cv_metrics) +
  geom_path() +
  geom_point() +
  scale_x_continuous(breaks = 1:9)
```



(h)

Size 9, at `dev` = 729.

(i)

```r
oj_pruned <- prune.tree(tree_oj, best = 9)
```

```
## Warning in prune.tree(tree_oj, best = 9): best is bigger than tree size
```

(j)

```r
summary(oj_pruned)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = oj_train)
## Variables actually used in tree construction:
## [1] "LoyalCH"     "PriceDiff"   "SalePriceMM" "DiscCH"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7112 = 563.2 / 792
## Misclassification error rate: 0.1575 = 126 / 800
```

Error rate of 145/800 is exactly the same as the un-pruned model before.

(k)

```r
preds_oj_pruned <- predict(oj_pruned, oj_test, type = "class")
table(oj_test$Purchase, preds_oj_pruned)
```

```
##     preds_oj_pruned
##       CH  MM
##   CH 131  21
##   MM  34  84
```

Same exact test error rate of 29/270.