P8106: Data Science II, Homework #4

Zachary Katz (UNI: zak2132)

4/13/2022

Contents

Question 1 **1**

Set-Up and Data Preprocessing 1

Part (a): Regression Tree 2

Minimum MSE Rule 2

1SE Rule 5

Comparison of Predictions 7

Part (b): Random Forest 8

Part (c): Boosting 8

Question 2 **8**

Set-Up and Data Preprocessing 8

Part (a): Classification Tree 8

Minimum MSE Rule 8

1SE Rule 11

Part (b): Boosting 13

Question 1

Set-Up and Data Preprocessing

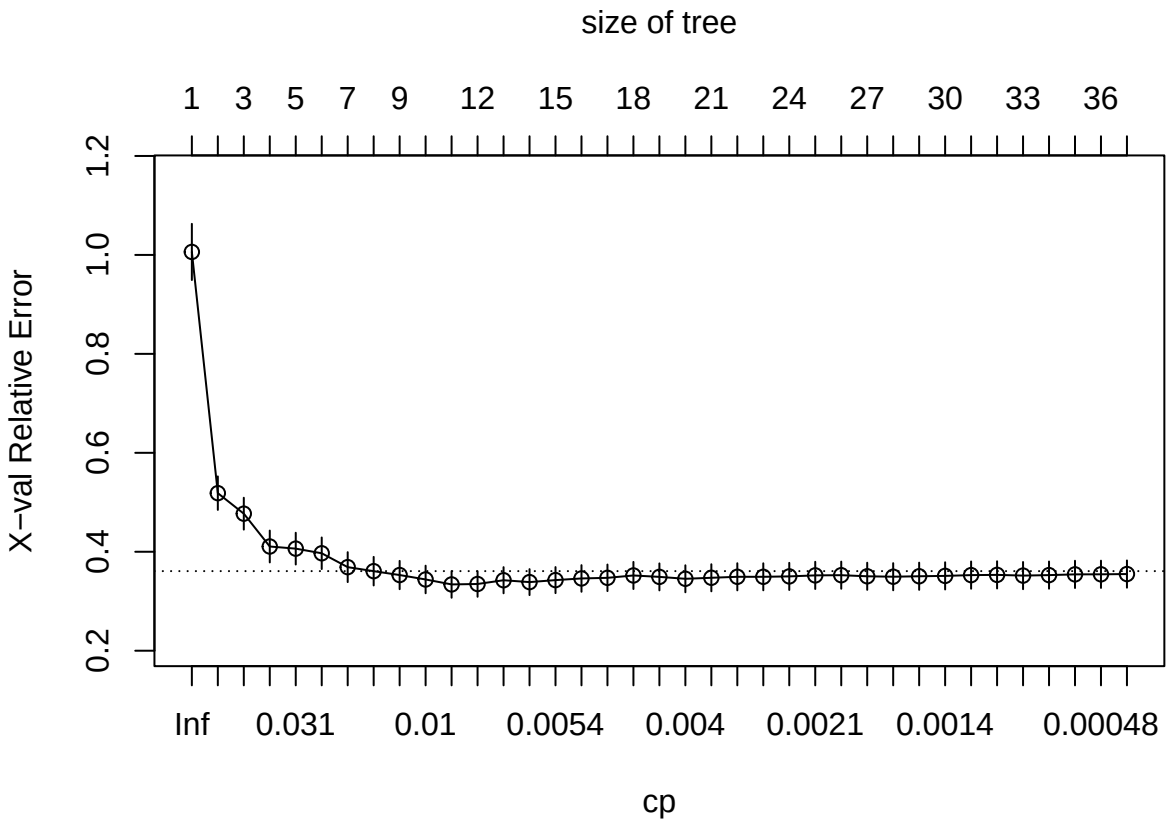
```
set.seed(2132)

# Load data, clean column names, eliminate rows containing NA entries
data = read_csv("./Data/College.csv") %>%
  janitor::clean_names() %>%
  na.omit() %>%
  relocate("outstate", .after = "grad_rate") %>%
  select(-college)
```

```r
# Partition data into training/test sets
indexTrain = createDataPartition(y = data$outstate,
                                 p = 0.8,
                                 list = FALSE)

training_df = data[indexTrain, ]

testing_df = data[-indexTrain, ]

# Create matrices for future analysis

# Training data
x_train = model.matrix(outstate~.,training_df)[, -1]
y_train = training_df$outstate

# Testing data
x_test <- model.matrix(outstate~.,testing_df)[, -1]
y_test <- testing_df$outstate
```

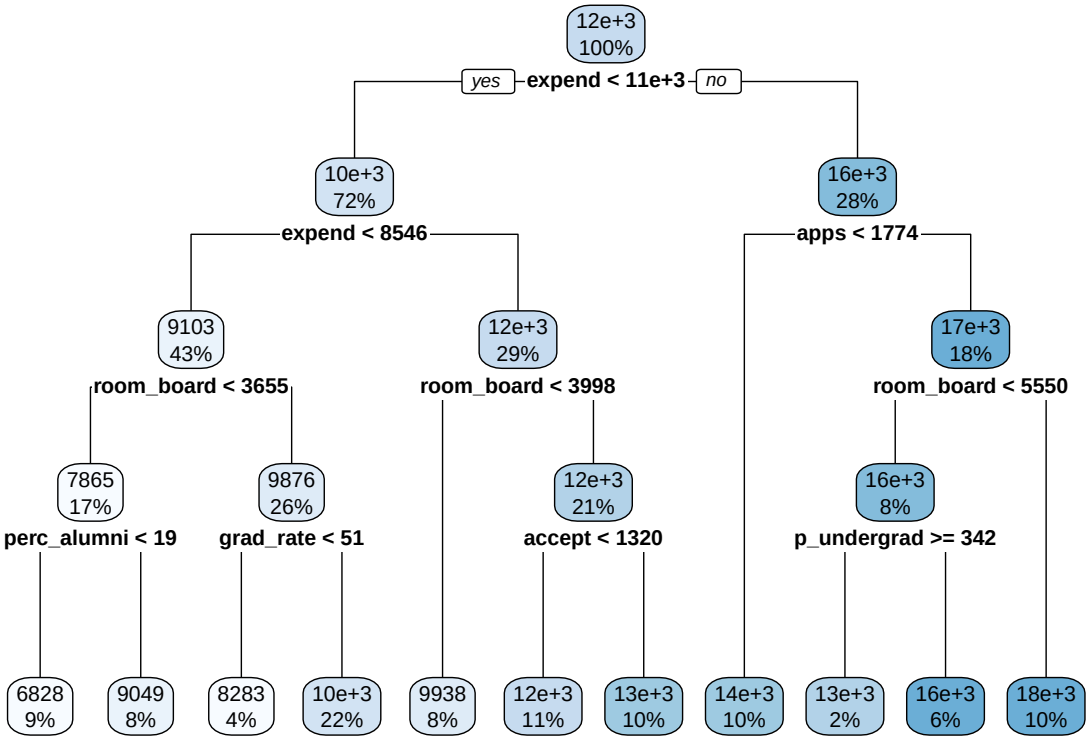
Part (a): Regression Tree

Minimum MSE Rule

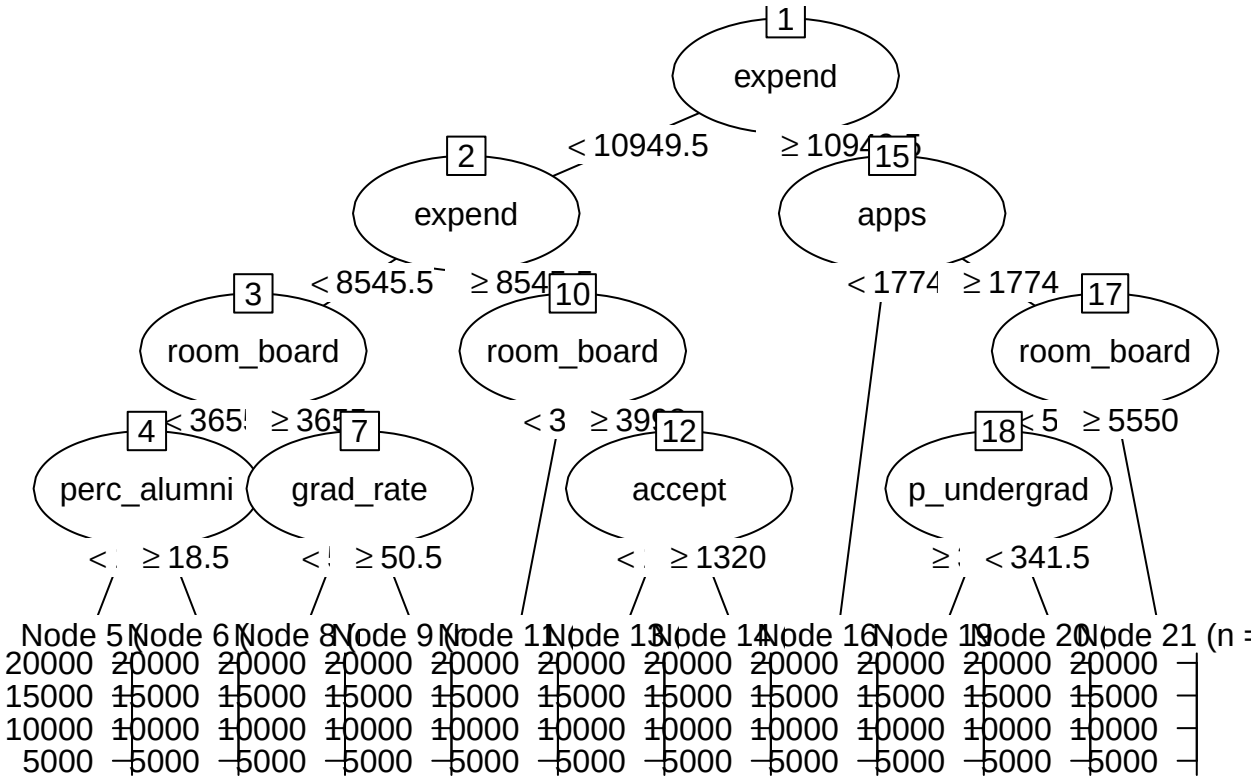
```r
# Build a regression tree on the training data to predict the response
set.seed(2132)

regression_tree = rpart(formula = outstate ~ . ,
               data = training_df, control = rpart.control(cp = 0))

# Cross-validation plot
regression_cptable = regression_tree$cptable
plotcp(regression_tree)
```




```
# Cost-complexity pruning
minimum_MSE = which.min(regression_cptable[,4])
final_regression_tree = prune(regression_tree, cp = regression_cptable[minimum_MSE,1])

# Plot of final tree
rpart.plot(final_regression_tree)
```



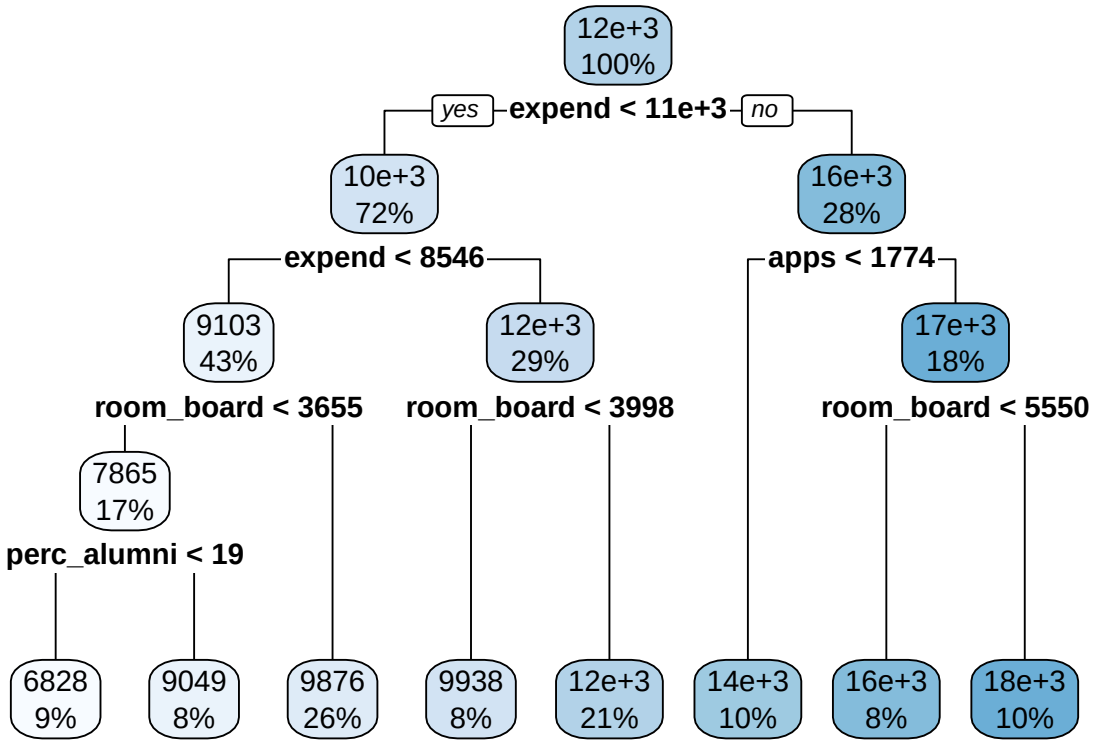

```
plot(as.party(final_regression_tree))
```



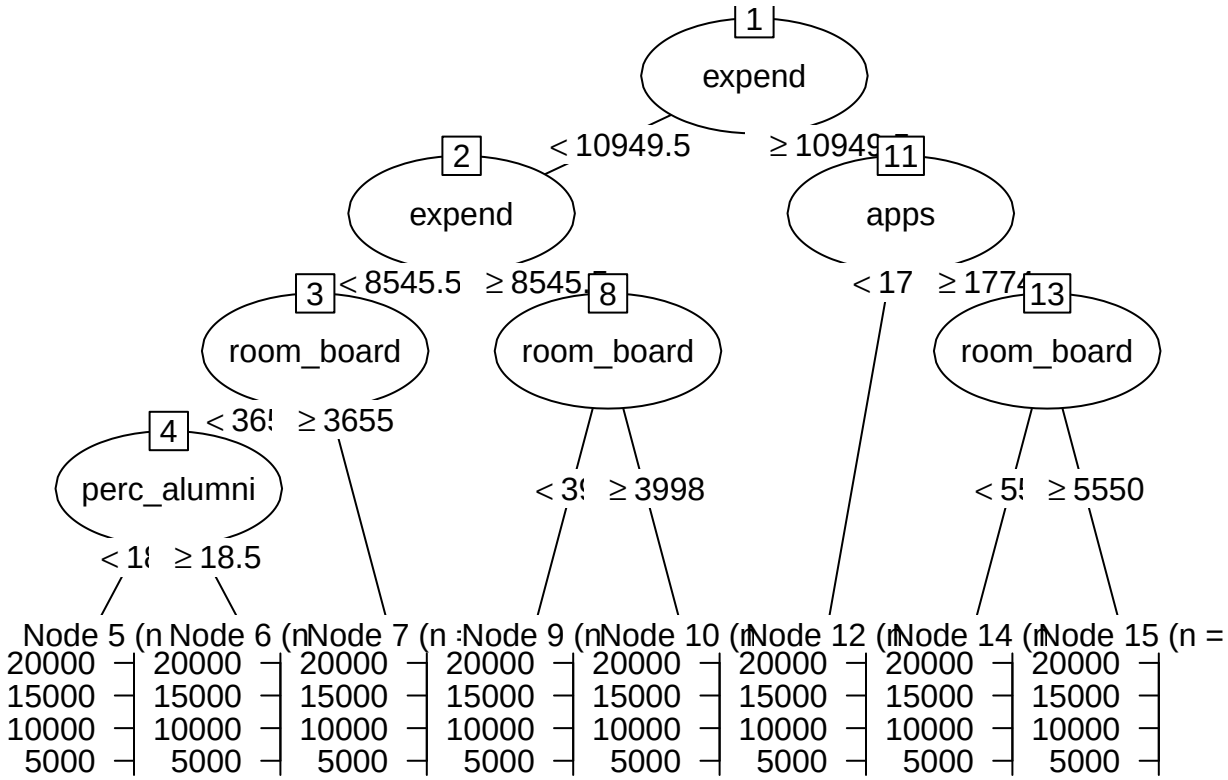

```
# Summary of final tree
# summary(final_regression_tree)
```

1SE Rule

```
# Alternatively, cost-complexity pruning using 1SE rule
final_regression_tree_1SE = prune(regression_tree, cp = regression_cptable[regression_cptable[,4]<regres

# Plot of 1SE tree
rpart.plot(final_regression_tree_1SE)
```




```
plot(as.party(final_regression_tree_1SE))
```



Comparison of Predictions

```
# For fun, compare predictions on first few observations in testing data set
reg_predict = predict(final_regression_tree, newdata = testing_df)
oneSE_predict = predict(final_regression_tree_1SE, newdata = testing_df)

# Compare predictions in data table
cbind(reg_predict, oneSE_predict) %>%
  as.data.frame() %>%
  head() %>%
  mutate(
    perc_diff = abs((reg_predict - oneSE_predict) * 100 / oneSE_predict)
  ) %>%
  knitr::kable(col.names = c("Prediction: Min MSE", "Prediction: 1SE", "Perc Diff"))
```

Prediction: Min MSE	Prediction: 1SE	Perc Diff
6827.90	6827.900	0.000000
11729.66	12488.737	6.078091
10194.94	9876.242	3.226919
10194.94	9876.242	3.226919
14146.09	14146.089	0.000000
10194.94	9876.242	3.226919

Part (b): Random Forest

TBD

Part (c): Boosting

TBD

Question 2

Set-Up and Data Preprocessing

```
set.seed(2132)

# Load data, clean column names, eliminate rows containing NA entries, factor outcome
data(OJ)
OJ_data = OJ %>%
  janitor::clean_names() %>%
  na.omit() %>%
  relocate("purchase", .after = "store") %>%
  mutate(
    purchase = as.factor(purchase)
  )

# Partition data into training/test sets (700 obs in training data)
OJ_indexTrain = createDataPartition(y = OJ_data$purchase,
                                    p = 0.653,
                                    list = FALSE)

OJ_training_df = OJ_data[OJ_indexTrain, ]

OJ_testing_df = OJ_data[-OJ_indexTrain, ]
```

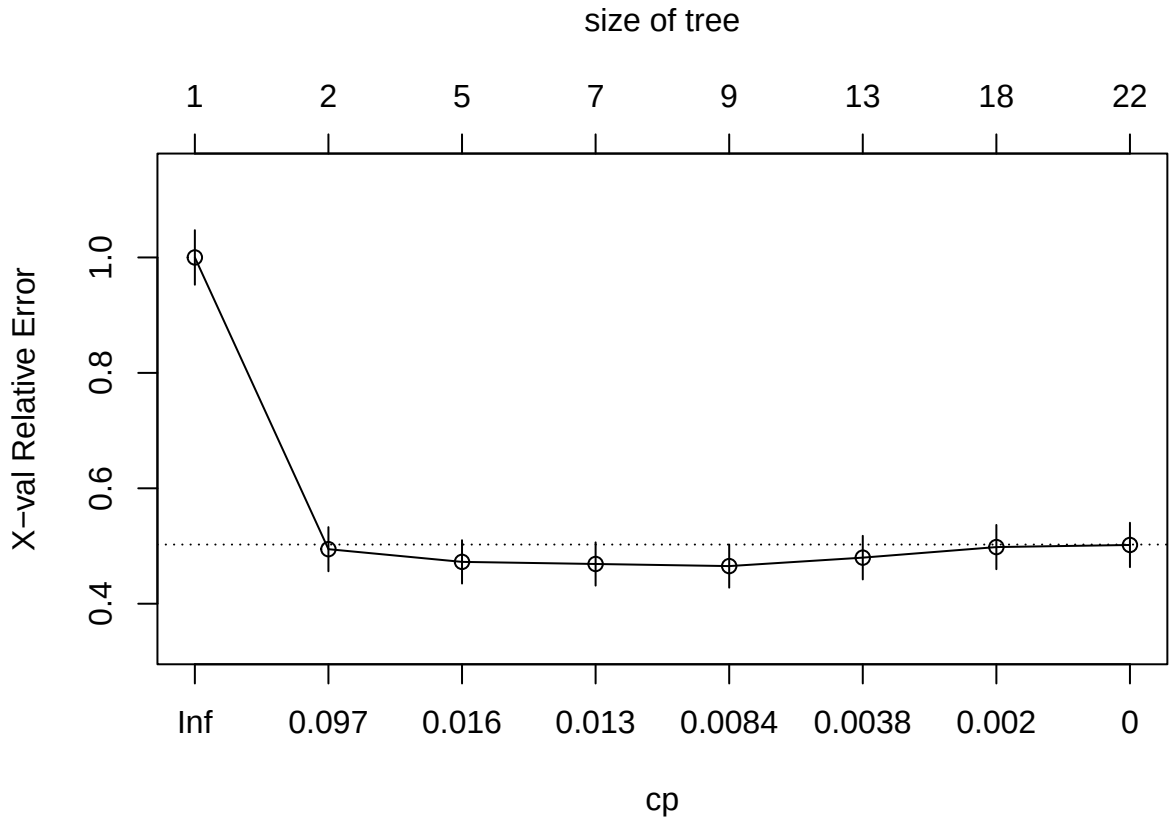
Part (a): Classification Tree

Minimum MSE Rule

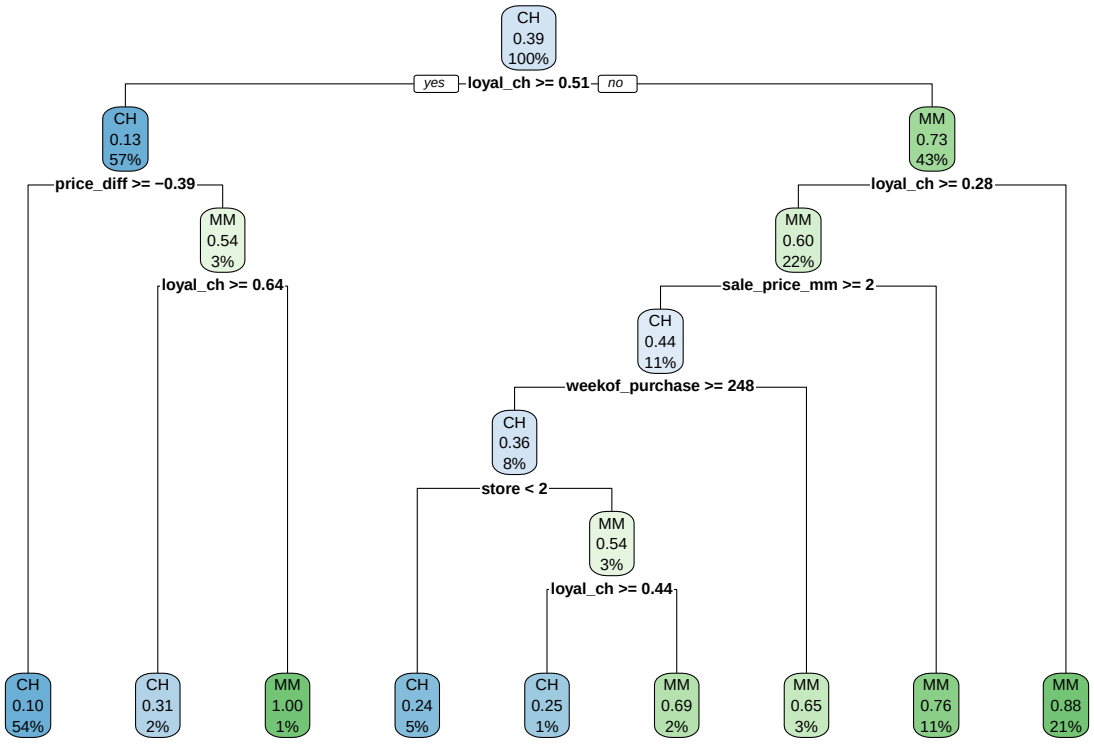
```
# Build classification tree using training data
set.seed(2132)

class_tree = rpart(formula = purchase ~ . ,
                   data = OJ_training_df,
                   control = rpart.control(cp = 0))

# Obtain cp table and plot vs cross-validation error
OJ_cp_table = printcp(class_tree)
```

```
##
## Classification tree:
## rpart(formula = purchase ~ ., data = OJ_training_df, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] list_price_diff loyal_ch        price_diff      sale_price_mm
## [5] store           store_id        weekof_purchase
##
## Root node error: 273/700 = 0.39
##
## n= 700
##
##          CP nsplit rel error  xerror     xstd
## 1 0.5164835      0   1.00000 1.00000 0.047270
## 2 0.0183150      1   0.48352 0.49451 0.038237
## 3 0.0146520      4   0.42491 0.47253 0.037575
## 4 0.0109890      6   0.39560 0.46886 0.037462
## 5 0.0064103      8   0.37363 0.46520 0.037348
## 6 0.0021978     12   0.34799 0.47985 0.037799
## 7 0.0018315     17   0.33700 0.49817 0.038344
## 8 0.0000000     21   0.32967 0.50183 0.038451
```

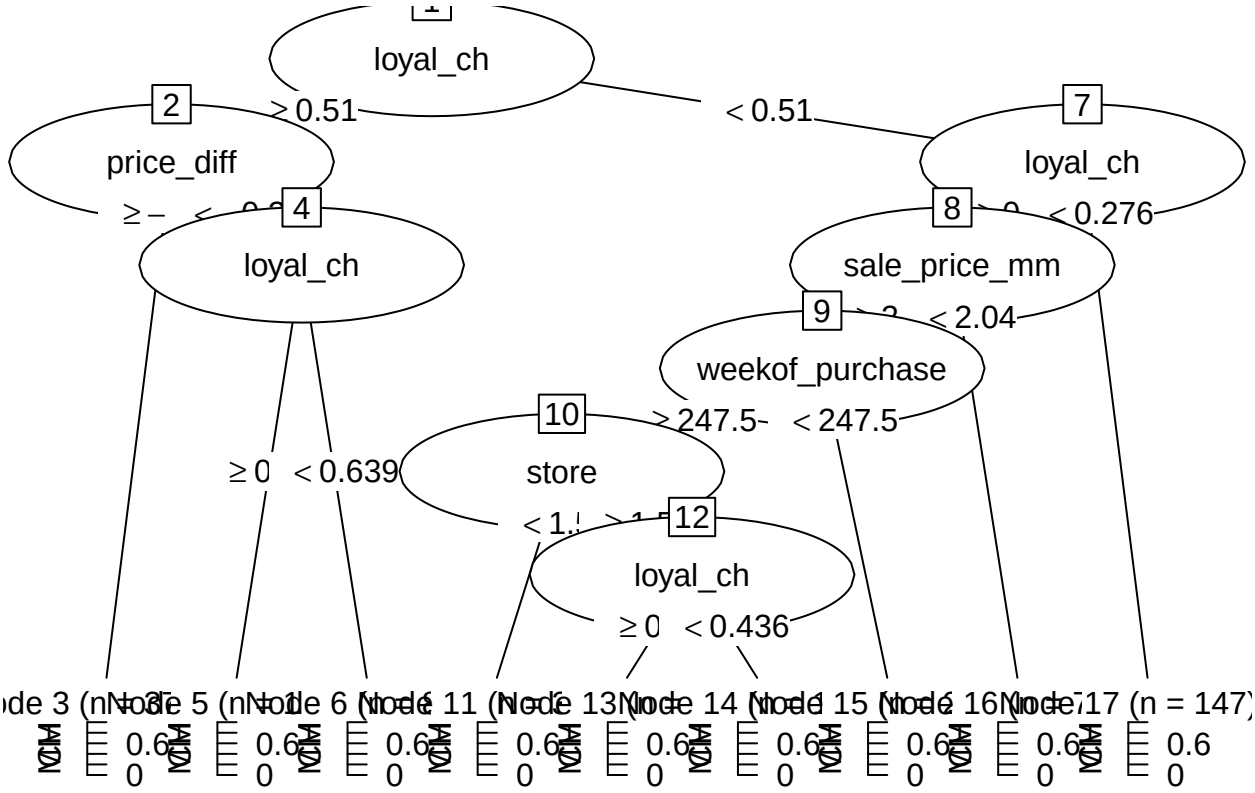
```
plotcp(class_tree)
```



```
# Obtain and plot final tree using min MSE rule
OJ_min_MSE = which.min(OJ_cp_table[,4])
final_class_tree = prune(class_tree, cp = OJ_cp_table[OJ_min_MSE,1])
rpart.plot(final_class_tree)
```

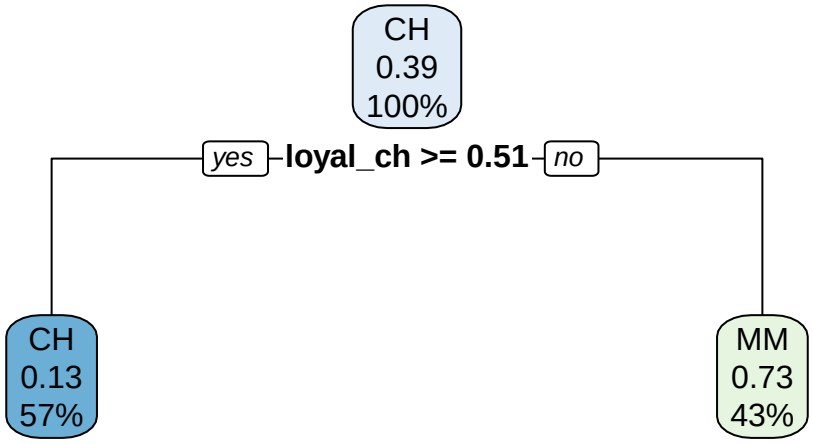



```
plot(as.party(final_class_tree))
```

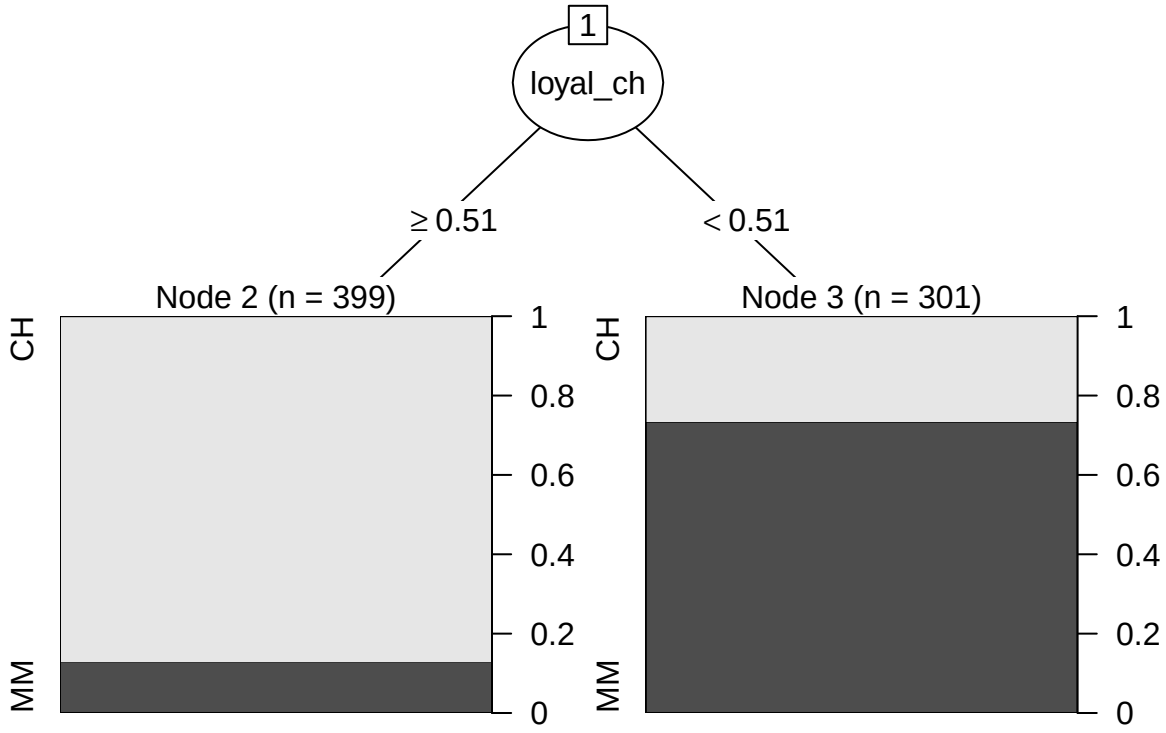


1SE Rule

```
# Obtain and plot final tree using 1SE rule
final_class_tree_1SE = prune(class_tree, cp = OJ_cp_table[OJ_cp_table[,4]<OJ_cp_table[OJ_min_MSE,4]+OJ_

# Plot of 1SE tree
rpart.plot(final_class_tree_1SE)
```




```
plot(as.party(final_class_tree_1SE))
```



Part (b): Boosting