Assignment 3

Lam

2024-05-01

Question 1

Load the libraries and read in the dataset

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.3.3
```

```
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.3.3
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

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

```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

```
## Loading required package: rpart
```

```
## Warning: package 'rpart' was built under R version 4.3.3
```

```
theme_set(theme_minimal())
```

```
data("OJ")
```

```
OJ %>% str()
```

```
## 'data.frame':    1070 obs. of  18 variables:
##  $ Purchase      : Factor w/ 2 levels "CH","MM": 1 1 1 2 1 1 1 1 1 1 ...
##  $ WeekofPurchase: num  237 239 245 227 228 230 232 234 235 238 ...
##  $ StoreID       : num  1 1 1 1 7 7 7 7 7 7 ...
##  $ PriceCH       : num  1.75 1.75 1.86 1.69 1.69 1.69 1.69 1.75 1.75 1.75 ...
##  $ PriceMM       : num  1.99 1.99 2.09 1.69 1.69 1.99 1.99 1.99 1.99 1.99 ...
##  $ DiscCH        : num  0 0 0.17 0 0 0 0 0 0 0 ...
##  $ DiscMM        : num  0 0.3 0 0 0 0 0.4 0.4 0.4 0.4 ...
##  $ SpecialCH     : num  0 0 0 0 0 0 1 1 0 0 ...
##  $ SpecialMM     : num  0 1 0 0 0 1 1 0 0 0 ...
##  $ LoyalCH       : num  0.5 0.6 0.68 0.4 0.957 ...
##  $ SalePriceMM   : num  1.99 1.69 2.09 1.69 1.69 1.99 1.59 1.59 1.59 1.59 ...
##  $ SalePriceCH   : num  1.75 1.75 1.69 1.69 1.69 1.69 1.69 1.75 1.75 1.75 ...
##  $ PriceDiff     : num  0.24 -0.06 0.4 0 0 0.3 -0.1 -0.16 -0.16 -0.16 ...
##  $ Store7        : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 2 2 2 2 2 ...
##  $ PctDiscMM     : num  0 0.151 0 0 0 ...
##  $ PctDiscCH     : num  0 0 0.0914 0 0 ...
##  $ ListPriceDiff : num  0.24 0.24 0.23 0 0 0.3 0.3 0.24 0.24 0.24 ...
##  $ STORE         : num  1 1 1 1 0 0 0 0 0 0 ...
```

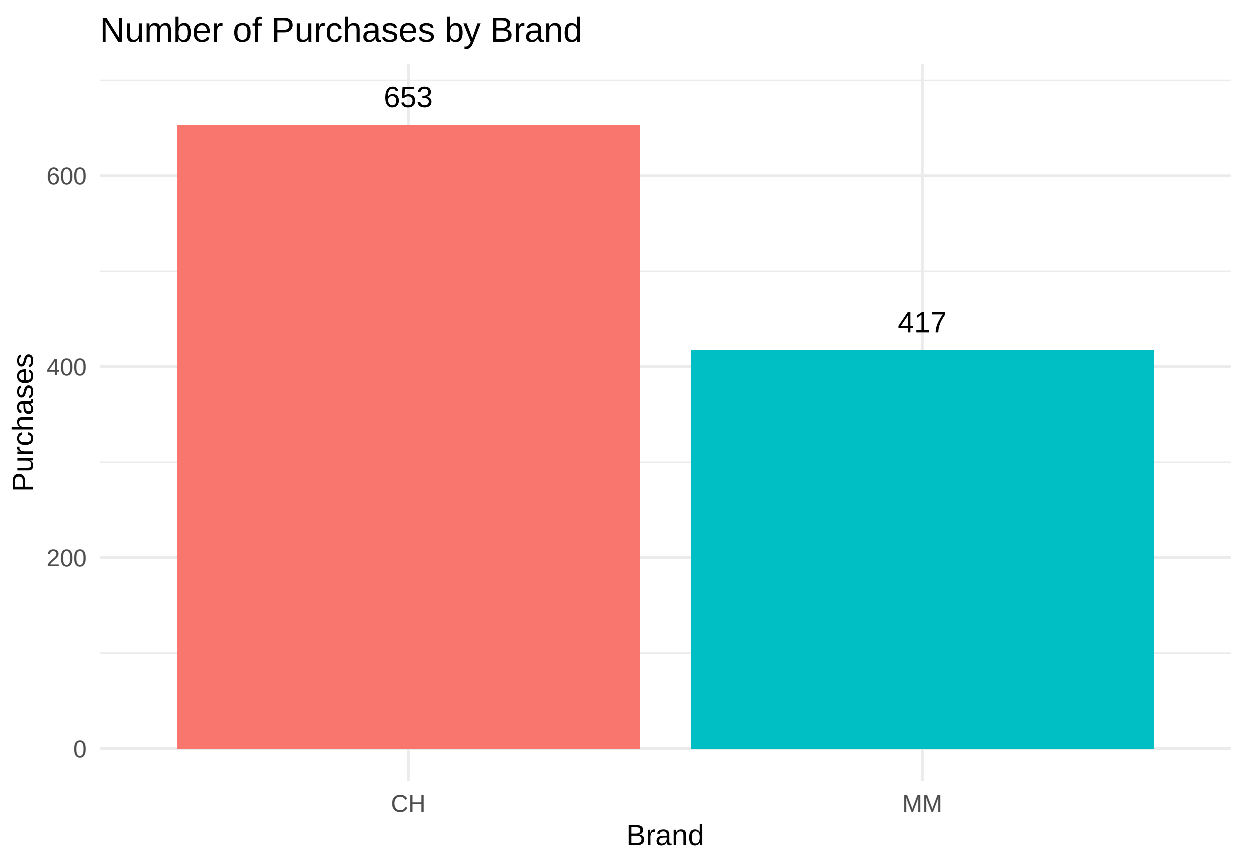
a. EDA

Question: Is one brand bought more often than the other one?

```
p <- ggplot(data=OJ, aes(x = Purchase)) +
  geom_bar(aes(fill=Purchase), show.legend = FALSE) +
  labs(
    title = 'Number of Purchases by Brand',
    x = 'Brand',
    y = 'Purchases'
  )

p_dt <- layer_data(p)

p + annotate(geom='text', label=p_dt$count, x=p_dt$x, y=p_dt$y+30)
```



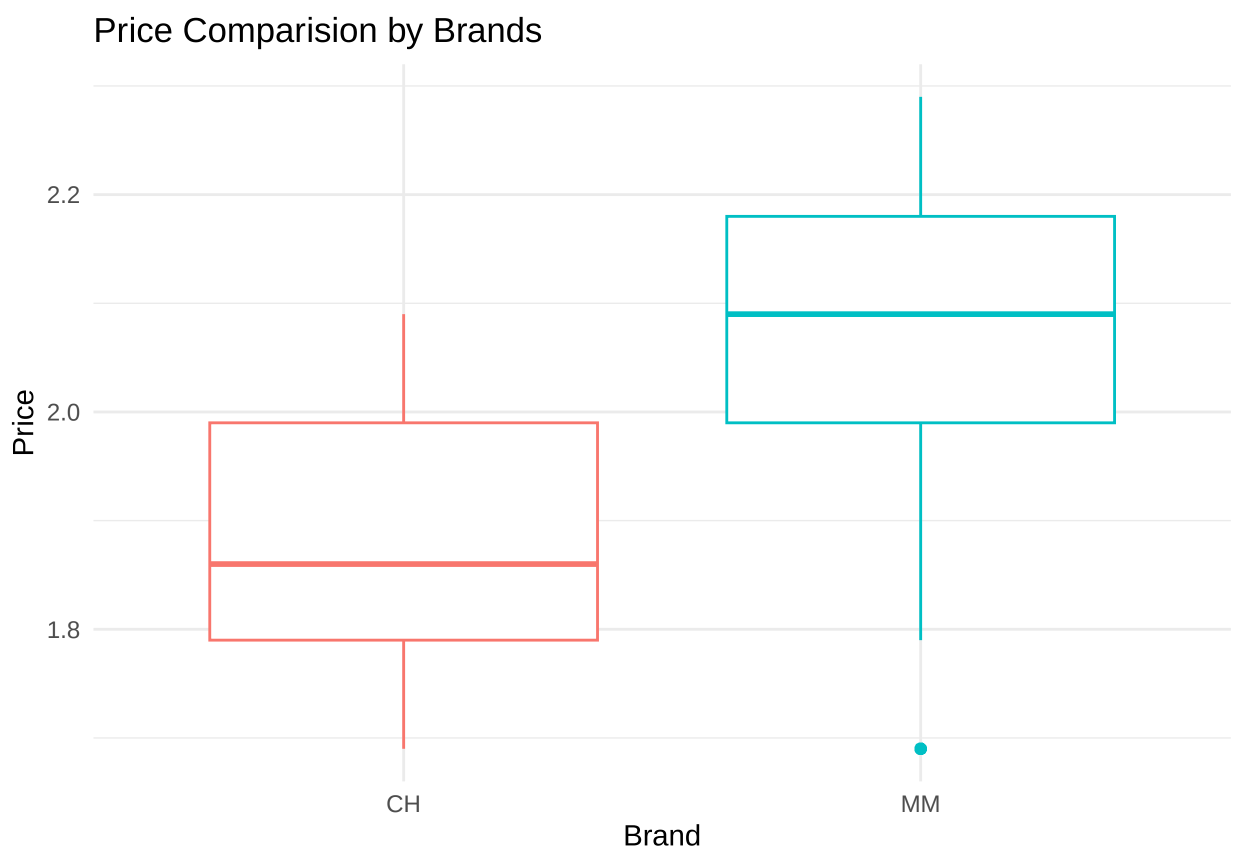
```r
# Number of purchases by Brands by percentage
table(OJ$Purchase) %>% prop.table()
```

```
##
##        CH        MM
## 0.6102804 0.3897196
```

61% of the total purchases in this dataset belongs to `CH` and only 38% of the `MM` purchases. Therefore, **Citrus Hill** seems to be more popular and bought more frequently than **Minute Maid**.

Question: Which brand tends to be more expensive?

```r
OJ %>% select(PriceCH, PriceMM) %>%
  melt(measure.vars = c('PriceCH', 'PriceMM')) %>%
  ggplot() +
  geom_boxplot(aes(x=variable, y=value, colour=variable),show.legend = FALSE) +
  labs(
    x = 'Brand',
    y = 'Price',
    title = 'Price Comparision by Brands'
  ) +
  scale_x_discrete(labels = c('CH', 'MM'))
```



```
summary(OJ$PriceCH)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.690   1.790   1.860   1.867   1.990   2.090
```

```
summary(OJ$PriceMM)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.690   1.990   2.090   2.085   2.180   2.290
```

From the side-by-side boxplot we can clearly see that `MM` is more expensive than `CH`. The bold horizontal line represents the average price of the brand. Most of the purchased price of `CH` is between 1.7 to 2.1 with the **average price of 1.86** while for `MM` is between 1.7 to 1.9 with **average price of 2.08.**

So on average, Minute Maid is more expensive than Citrus Hill.

b. Train, test split

```
set.seed(1113)
OJ <- OJ %>% mutate(id = row_number())
train <-  OJ %>% sample_frac(0.8)
test <- anti_join(OJ, train, by='id')
```

```
train <- train %>% select(c(-id))
test <- test %>% select(c(-id))
```

```
train %>% dim()
```

```
## [1] 856  18
```

```
test %>% dim()
```

```
## [1] 214  18
```

c. Classification Tree

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.3.3
```

```
set.seed(1)
# Fit the tree
tree_clf <- tree(Purchase~., data=train)
summary(tree_clf)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = train)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "ListPriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.741 = 628.4 / 848
## Misclassification error rate: 0.1694 = 145 / 856
```

Decision tree Misclassification error rate is 0.1694.

```
# Record the error rate
training_mis <- c(0.1694)
```

Using 10 folds cross validation to prune it

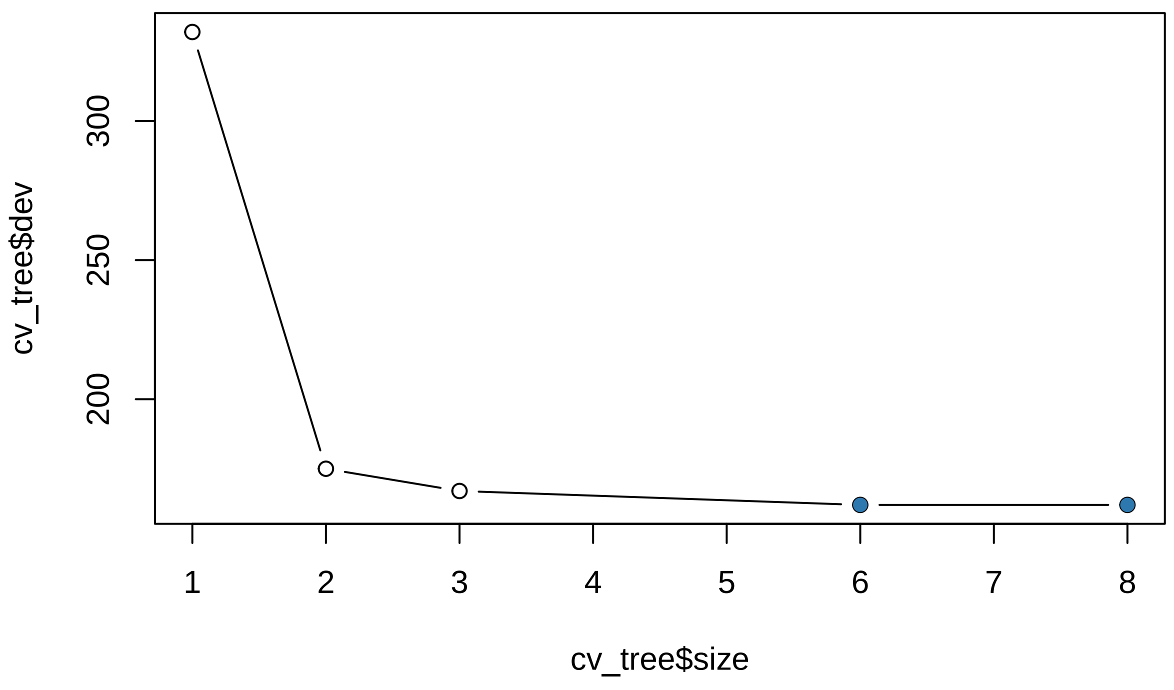
```
cv_tree <- cv.tree(tree_clf, FUN=prune.misclass, K = 10)

cv_tree
```

```
## $size
## [1] 8 6 3 2 1
##
## $dev
## [1] 162 162 167 175 332
##
## $k
## [1]       -Inf   0.000000   4.333333  13.000000 161.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cv_tree$size, cv_tree$dev, type = 'b')

x_vals <- cv_tree$size[which(cv_tree$dev == min(cv_tree$dev))]
y_vals <- rep(min(cv_tree$dev), length(x_vals))
points(x_vals, y_vals ,col = "#2E78B0", pch = 16)
```



It turns out that the optimal terminal nodes are 6 and 8 which has the lowest cross validated error. For interpretability and simplixity we will prune it to size 6 tree.

```
# Prune
tree_clf_pruned <- prune.misclass(tree_clf, best = 6)
tree_clf_pruned %>% summary()
```

```
##
```

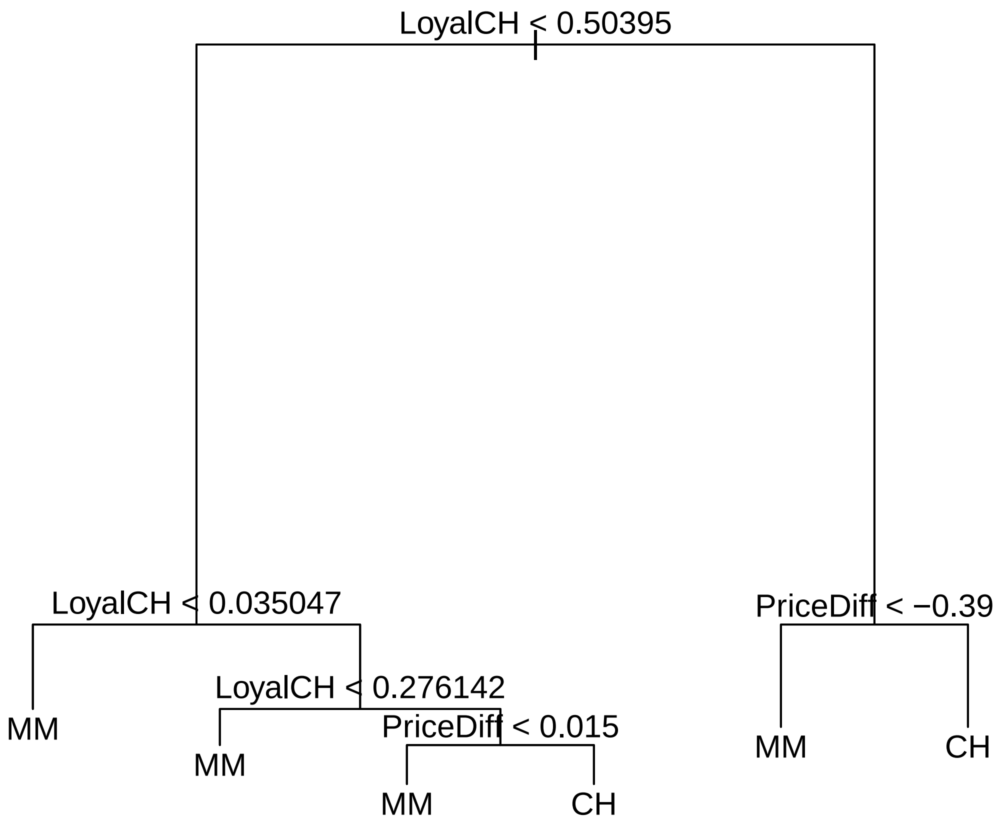
```
## Classification tree:
## snip.tree(tree = tree_clf, nodes = 7L)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  6
## Residual mean deviance:  0.8102 = 688.7 / 850
## Misclassification error rate: 0.1694 = 145 / 856
```

The pruned tree has misclassification error rate of 0.1694.

```
training_mis <- c(training_mis, 0.1694)
```

Visualize the tree

```
plot(tree_clf_pruned)
text(tree_clf_pruned, pretty = 0)
```



The feature `LoyalCH` (Customer brand loyalty for CH) seems to be very important to our model. This is to say that brand loyalty is an important factor on purchase decision.

From the tree we can say that in general, if a customer is somewhat not loyal to Citrus Hill and the price different is small, they tend to make a purchase on Minute Maid. And if a customer is definitely not loyal to Citrus Hill they will purchase Minute Maid and vice versa.

d. Bagging

Bagging is simply a special case of random forest where m = p (where p is the number of predictors and m is the number of variables randomly sampled as candidates at each split). So we can use **randomForest** library and set `mtry` = 17 to perform bagging.

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(2)
bagged_clf <- randomForest(Purchase~., mtry=17 ,data = train, importance = TRUE)

bagged_clf
```

```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = train, mtry = 17,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 17
##
##         OOB estimate of  error rate: 19.28%
## Confusion matrix:
##     CH  MM class.error
## CH 440  84   0.1603053
## MM  81 251   0.2439759
```

```
# Calculate error rate for bagged model
bagged_pred <- predict(bagged_clf, newdata = train)

table(bagged_pred, train$Purchase)
```

```
##
## bagged_pred  CH  MM
##          CH 522   7
##          MM   2 325
```

```
# bagging misclassification error
(2+7) / nrow(train)
```

```
## [1] 0.01051402
```

Note that the misclassification error for our bagged model is extremely small. This is because the model is trained on bootstraped samples of the training set. And with enough trees (500 in our case), it can fit the data very well.

Generally, it is more sensible to use OOB (out of bag) misclassification error when evaluating the model performance. In our case estimated **OOB error is 19.28%**.

```r
training_mis <- c(training_mis, 0.1928)
```

e. Random Forest

```r
set.seed(3)
# Default mtry for Random Forest Classifier is sqrt(p) = 4.12
rf_clf <- randomForest(Purchase~., data = train)

rf_clf
```

```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 17.99%
## Confusion matrix:
##     CH  MM class.error
## CH 454  70   0.1335878
## MM  84 248   0.2530120
```

```r
rf_pred <- predict(rf_clf, newdata = train)
table(rf_pred, train$Purchase)
```

```
##
## rf_pred  CH  MM
##      CH 507  32
##      MM  17 300
```

```r
# training misclassified error rate
(17 + 32 ) / nrow(train)
```

```
## [1] 0.05724299
```

The training error rate is **0.05** and OOB estimated error rate is **17.99%.**

```r
training_mis <- c(training_mis, 0.1799)
```

f. Boosting

```r
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.3.3
```

```
## Loaded gbm 2.1.9
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.c
```

Encoding the response variable

```
train$Purchase_coded <- recode(train$Purchase, 'CH' = 0, 'MM' = 1)
test$Purchase_coded <- recode(test$Purchase, 'CH' = 0, 'MM' = 1)
```

```
set.seed(4)

gbm_clf <- gbm(Purchase_coded~.-Purchase, data=train, distribution = 'bernoulli', n.trees = 500)

gbm_clf
```

```
## gbm(formula = Purchase_coded ~ . - Purchase, distribution = "bernoulli",
##     data = train, n.trees = 500)
## A gradient boosted model with bernoulli loss function.
## 500 iterations were performed.
## There were 17 predictors of which 17 had non-zero influence.
```

```
gbm_prob_train <- predict(gbm_clf, type = 'response', n.trees = 500)
gbm_pred_train <- ifelse(gbm_prob_train < 0.5, 0, 1)
mean(gbm_pred_train != train$Purchase_coded)
```

```
## [1] 0.1331776
```

GBM has training misclassification rate of **0.133**

```
training_mis <- c(training_mis, 0.133)
```

g. Models Comparison

```
# Function to put models error rate into a data frame for easy plotting
create_error_df <- function(predictions_list, test_data){
  error_list <- numeric(length(predictions_list))
  for (i in seq_along(predictions_list)) {
  if (names(predictions_list[i]) == 'GBM') {
    error_list[i] <- mean(predictions_list[[i]] != test_data$Purchase_coded)
  }
  else(
    error_list[i] <- mean(predictions_list[[i]] != test_data$Purchase)
  )
  }
  df <- data.frame(Model = names(predictions_list), Misclassification_Rate = error_list)
  return(df)
}

# compute predictions on test set
tree_pred <- predict(tree_clf, newdata = test, type='class')
tree_pruned_pred <- predict(tree_clf_pruned, newdata = test, type='class')
```

```r
bagged_pred <- predict(bagged_clf, newdata = test)
rf_pred <- predict(rf_clf, newdata = test)

gbm_prob <- predict(gbm_clf, newdata = test, type='response', n.trees = 500)
gbm_pred <- ifelse(gbm_prob < 0.5, 0, 1)
```

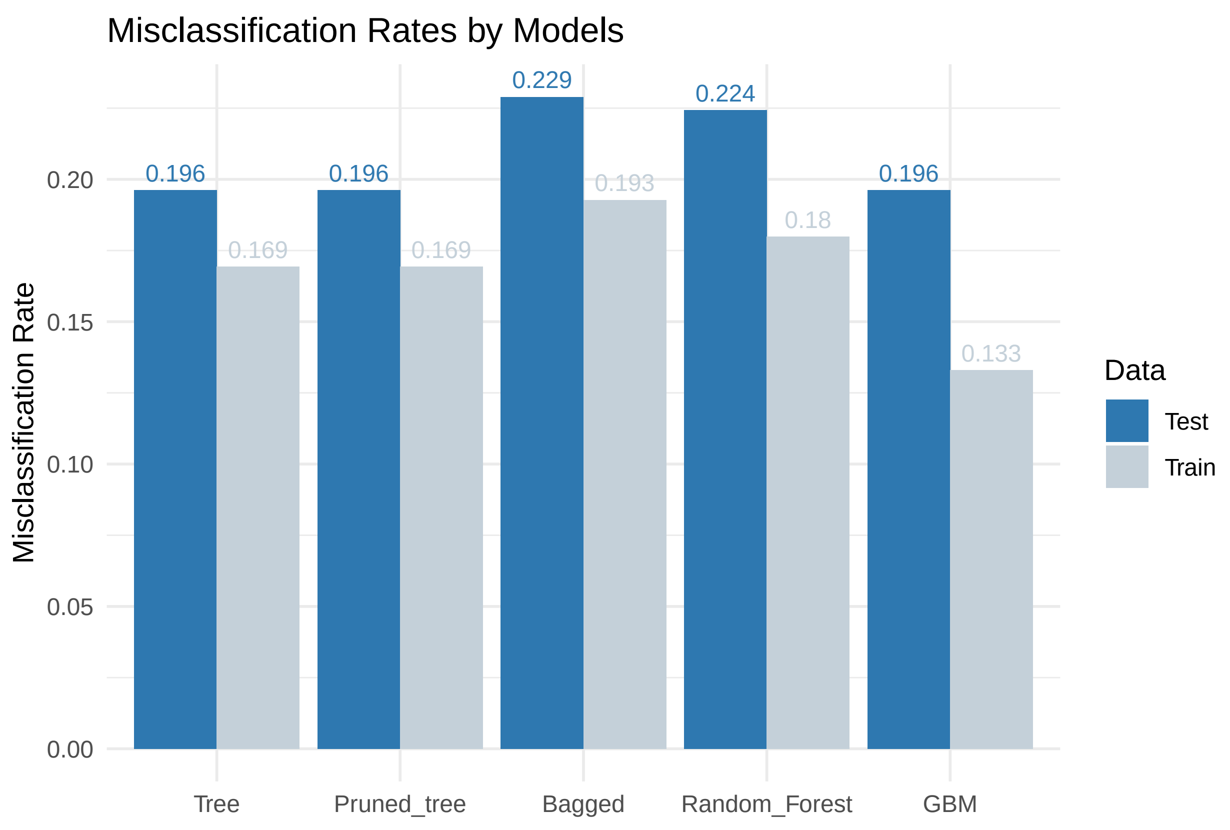
```r
# Put models predictions in a list
predictions_list <- list(Tree = tree_pred, Pruned_tree = tree_pruned_pred, Bagged = bagged_pred, Random_

# Testing error data frame
models <- c('Tree', 'Pruned_tree', 'Bagged', 'Random_Forest', 'GBM')
plot_data <- create_error_df(predictions_list, test)
plot_data$Model <- factor(plot_data$Model, levels = models)

# Training error data frame
df <- data.frame(Model = factor(models, levels = models), Misclassification_Rate = training_mis)
```

```r
# Join the two data frame
joined_df <- left_join(plot_data, df, by = join_by(Model))
colnames(joined_df) <- c('Model', 'Test', 'Train')
joined_df %>% pivot_longer(cols = - Model, names_to = 'Data', values_to = 'Misclassification_Rate') -> j

# Visualize the results
colour_scale <- c("#2E78B0", "#C4D0D9")
joined_df %>% ggplot(aes(x = Model, y = Misclassification_Rate, fill=Data)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  geom_text(aes(label = round(Misclassification_Rate, 3), color=Data),
            position = position_dodge(width = .9),
            vjust = -0.5, size = 3) +
  scale_fill_manual(values = colour_scale) +
  scale_colour_manual(values =colour_scale) +
  labs(title = "Misclassification Rates by Models", x = "", y = "Misclassification Rate")
```



Observation

- On training data, GBM performs best with error rate of 0.133.
- However, on test data, to our surprise, Decision Tree, Pruned Decision Tree (Size 6) and GBM performs equally good with error rate of 0.196.
- On both train and test set, Bagging and Random Forest seem to have worse performance compared to other models.

```
knitr::include_graphics("assignment_3_q2.pdf")
```

```
knitr::include_graphics("assignment_3_q2.pdf")
```

```
knitr::include_graphics("assignment_3_q2.pdf")
```

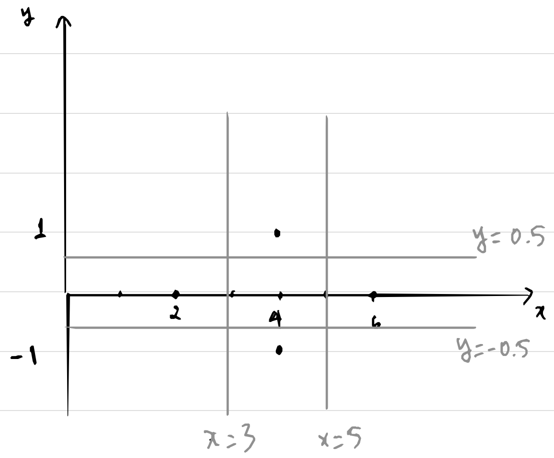
```
knitr::include_graphics("assignment_3_q2.pdf")
```

```
knitr::include_graphics("assignment_3_q2.pdf")
```

Question 2.

a)

First we will try all possible splits for all features.
Split is labeled at midpoint of two adjacent values.
(For example $x=3$ is midpoint of $x=2$ & $x=4$)



x	y	z
2	0	10
4	1	4
6	0	4
4	-1	-5

* Split at $x=3$

$\hat{z}^T = (10 \quad 1 \quad 1 \quad 1)$

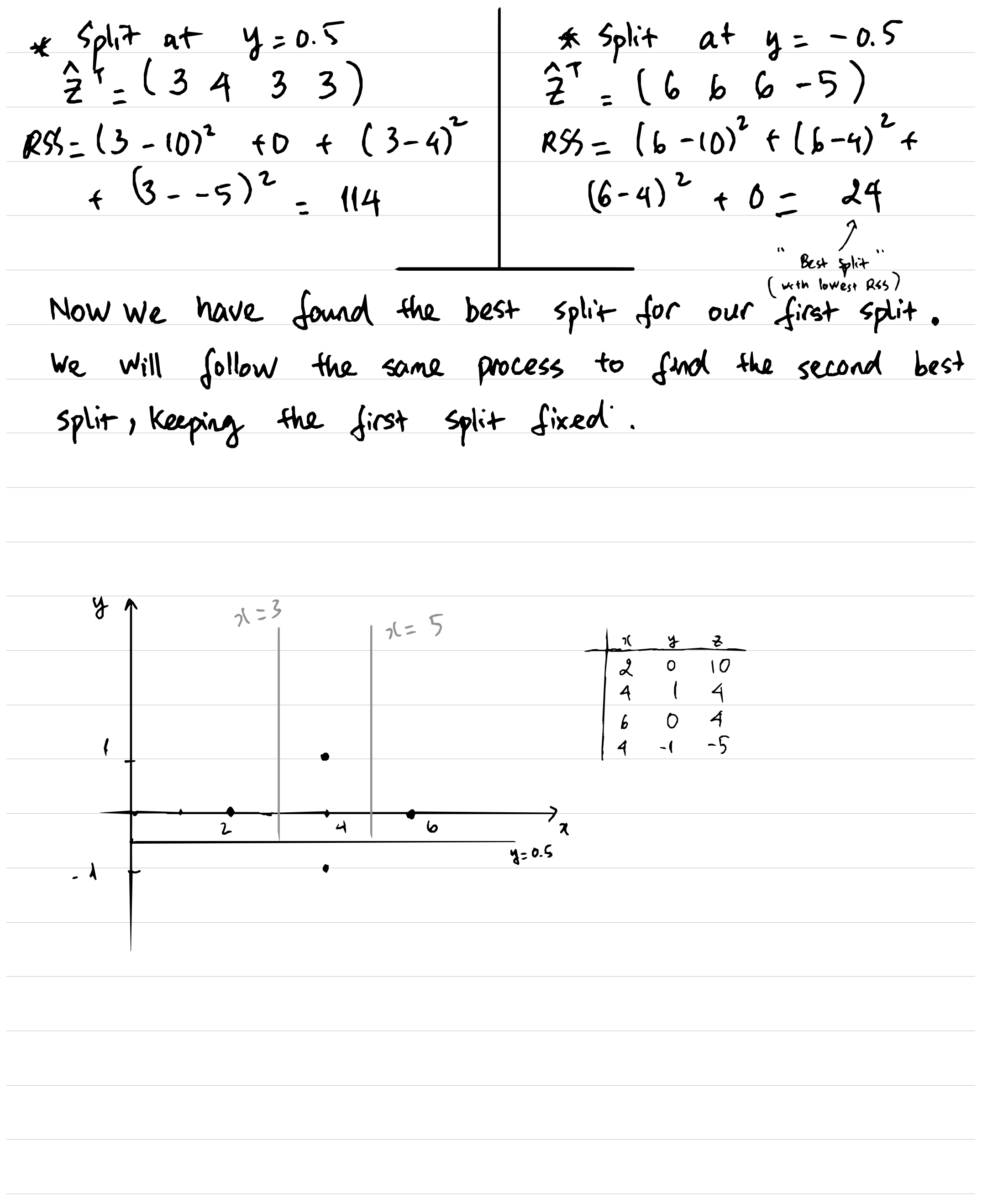
$RSS = 0 + (1-4)^2 + (1-4)^2 + (1--5)^2 = 54$

* Split at $x=5$

$\hat{z}^T = (3 \quad 3 \quad 4 \quad 3)$

$RSS = (3-10)^2 + (3-4)^2 + 0 + (3--5)^2 = 114.$

Figure 1: page 1



* Split at $y = 0.5$

$\hat{z}^T = (3\ 4\ 3\ 3)$

$RSS = (3-10)^2 + 0 + (3-4)^2 + (3 - -5)^2 = 114$

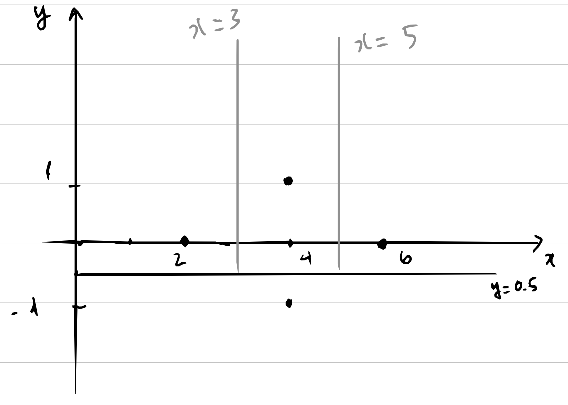
* Split at $y = -0.5$

$\hat{z}^T = (6\ 6\ 6\ -5)$

$RSS = (6-10)^2 + (6-4)^2 + (6-4)^2 + 0 = 24$

" Best split "
(with lowest RSS)

Now we have found the best split for our first split. We will follow the same process to find the second best split, keeping the first split fixed.



x	y	z
2	0	10
4	1	4
6	0	4
4	-1	-5

Figure 2: page 2

* Split at $y = -0.5$ & $x = 3$

$\hat{Z}^T = (10 \quad 4 \quad 4 \quad -5)$

$RSS = (10-10)^2 + (4-4)^2 + (4-4)^2 + (-5--5)^2$

$= 0$. ← "Best split"

* Split at $y = -0.5$ & $x = 5$

$\hat{Z}^T = (7 \quad 7 \quad 4 \quad -5)$

$RSS = (7-10)^2 + (7-4)^2 + (4-4)^2 + (-5--5)^2$

$= 18$

b)

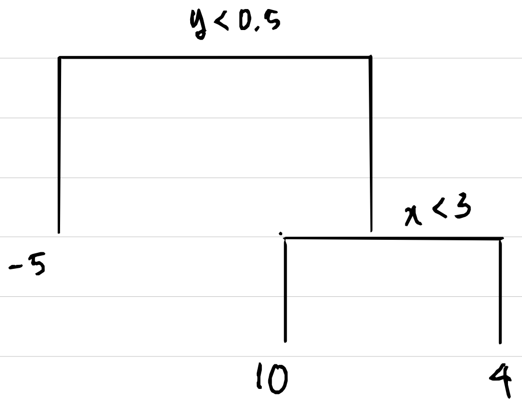
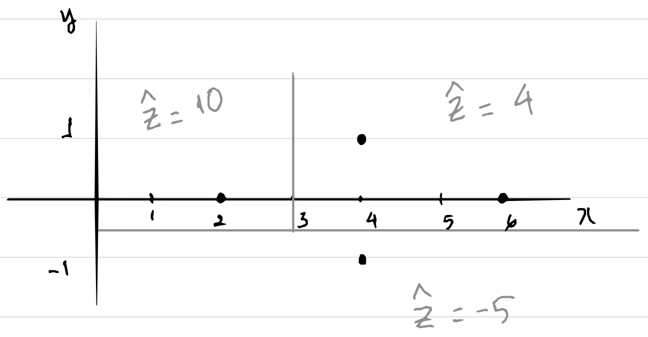


Figure 3: page 3



x	y	z
2	0	10
4	1	4
6	0	4
4	-1	-5

c)

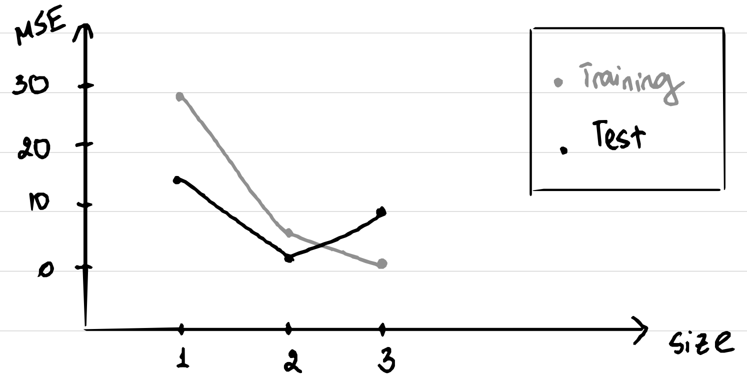
$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

- size 1: $\hat{z} = 3.25$
  $\Rightarrow$ MSE (traing) = 28.68
  $\Rightarrow$ MSE (test) = 14.06

- size 2: $\hat{z} = 6$
  MSE (traing) = 6
  MSE (test) = 1

- size 3: $\hat{z} = 10$
  MSE (traing) = 0
  MSE (test) = 9

Figure 4: page 4



Observation:

- On training set, tree with 3 terminal nodes performs best
- On test set tree with 2 terminal nodes has best performance.
- This indicates that as the size increases, the model has adapted more closely to the training data.

As the flexibility increases, the variance also increases. This explains why, with the new (unseen) data, the more complex model has wrose performance. This is known as overfitting.

Figure 5: page 5