Assignment_2_rspake1

Part A

In this section I will be answering the questions asked in Part A.

Question 1

“What is the key idea behind bagging? Can bagging deal both with high variance (overfitting) and high bias (underfitting)?”

- The Key concept behind bagging (bootstratp aggregation) is that we are reducing variance by selecting random samples of the training set using replacement (any given value may appear multiple times or not at all) and then each weak learner is fit on each data sample. This method is used to fight overfitting (variance) and would not be a viable option to tackle underfitting.

Question 2

“Why bagging models are computationally more efficient when compared to boosting models with the same number of weak learners?”

- Because in Boosting, trees are grown sequentially where each tree is grown using information from the previously grown trees, where in bagging, there is no sequential growth.

Question 3

“James is thinking of creating an ensemble mode to predict whether a given stock will go up or down in the next week. He has trained several decision tree models but each model is not performing any better than a random model. The models are also very similar to each other. Do you think creating an ensemble model by combining these tree models can boost the performance? Discuss your answer.”

- The main issue here is that James has created several decision tree models that are too similar to each other. Using the ensemble method of boosting is helpful in this case as the failures from the previous iterations can be used to make the next iterations better. However, it is also important to remember that having trees that are not similar is important as it helps us develop a more robust model. As such, I would recommend going back to the beginning as well and rebuilding the decision trees to be more different from each other by implementing a penalty for choosing a specific attribute at a certain level too many times.

Question 4

“Consider the following Table that classifies some objects into two classes of edible (+) and non- edible (-), based on some characteristics such as the object color, size and shape. What would be the Information gain for splitting the dataset based on the “Size” attribute? "

- central to this question, we must remember that Information Gain = entropy(parent)-[avg. entropy(children)]

Using the data provided, we find that the **parent** entropy is **0.988699** the entropy of the **small size** is **0.811278** and **large size** entropy is **0.954434**.

Using this calculation, we can determine that the **Information Gain** is **0.105843**.

Question 5

“Why is it important that the m parameter (number of attributes available at each split) to be optimally set in random forest models? Discuss the implications of setting this parameter too small or too large.”

- If m is set too large (all predictors p) then it would be no different than just using bagging and should thus be avoided as we will not get proper diversity. Alternatively if m is too small, each tree will not be very predictive as they will be too constrained at each node to a very small fraction of attributes. Allowing random forests to be optimally set is important as the key concept in random forests is that at each node, a random sample of predictors are used so that not every node is similar (as in bagging) and will result in a more accurate predictor.

Part B

This part of the assignment involves building decision tree and random forest models to answer a number of questions. We will use the Carseats dataset that is part of the ISLR package.

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.1.2
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.1.2
```

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

```
## Loaded glmnet 4.1-3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.2
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
## Loading required package: lattice
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.1.3
```

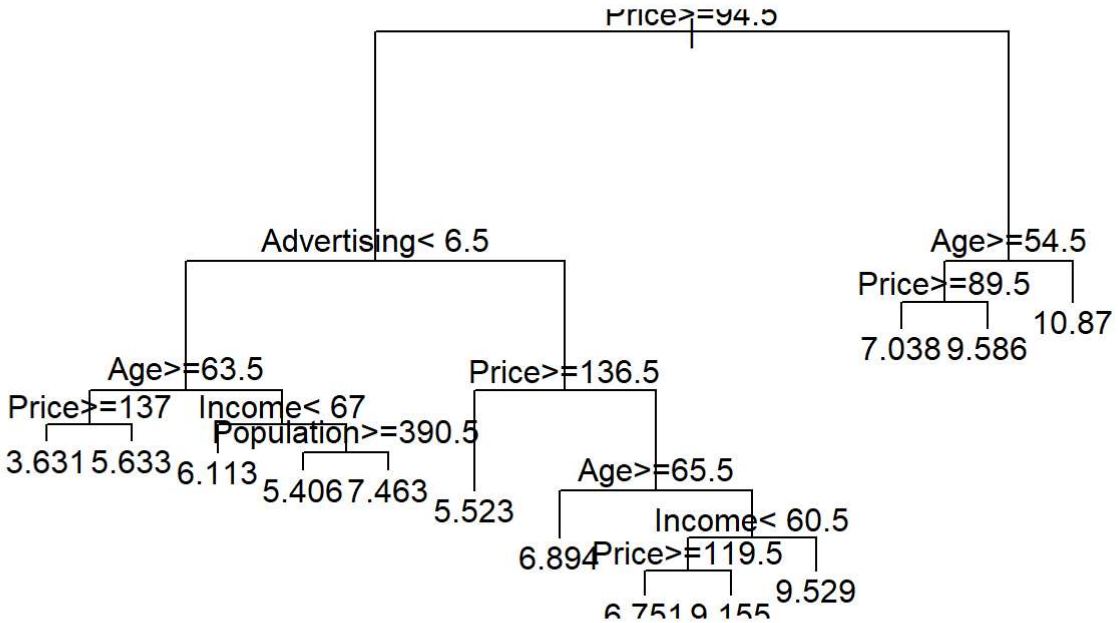
```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.1.3
```

```
Carseats_Filtered <- Carseats %>% select("Sales", "Price", "Advertising", "Population", "Age",
"Income", "Education")
```

Question 1

" Build a decision tree regression model to predict Sales based on all other attributes (“Price”, “Advertising”, “Population”, “Age”, “Income” and “Education”). Which attribute is used at the top of the tree (the root node) for splitting?"



Based off of what we see above, **Price Greater than or equal to 94.5** is our root node for splitting.

Queston 2

“Consider the following input: Sales=9, Price=6.54, Population=124, Advertising=0, Age=76, Income= 110, Education=10 What will be the estimated Sales for this record using the decision tree model?”

```
Sales <- c(9)
Price <- c(6.54)
Population <- c(124)
Advertising <- c(0)
Age <- c(76)
Income <- c(110)
Education <- c(10)

Test <- data.frame(Sales,Price,Population,Advertising,Age,Income,Education)
```

Now that we have created our test set to run through our model, we will predict Sales.

```
Pred_sales_2 <- predict(Model_1, Test)
Pred_sales_2
```

```
##       1
## 9.58625
```

According to our predict function, the decision tree indicates that **9.58625** sales will take place with this given record.

Question 3

“Use the caret function to train a random forest (method=’rf’) for the same dataset. Use the caret default settings. By default, caret will examine the “mtry” values of 2,4, and 6.
Recall that mtry is the number of attributes available for splitting at each splitting node. Which mtry value gives the best performance? "

```
set.seed(123)

Model_forest_caret <- train(Sales~., data = Carseats_Filtered, method = 'rf')
```

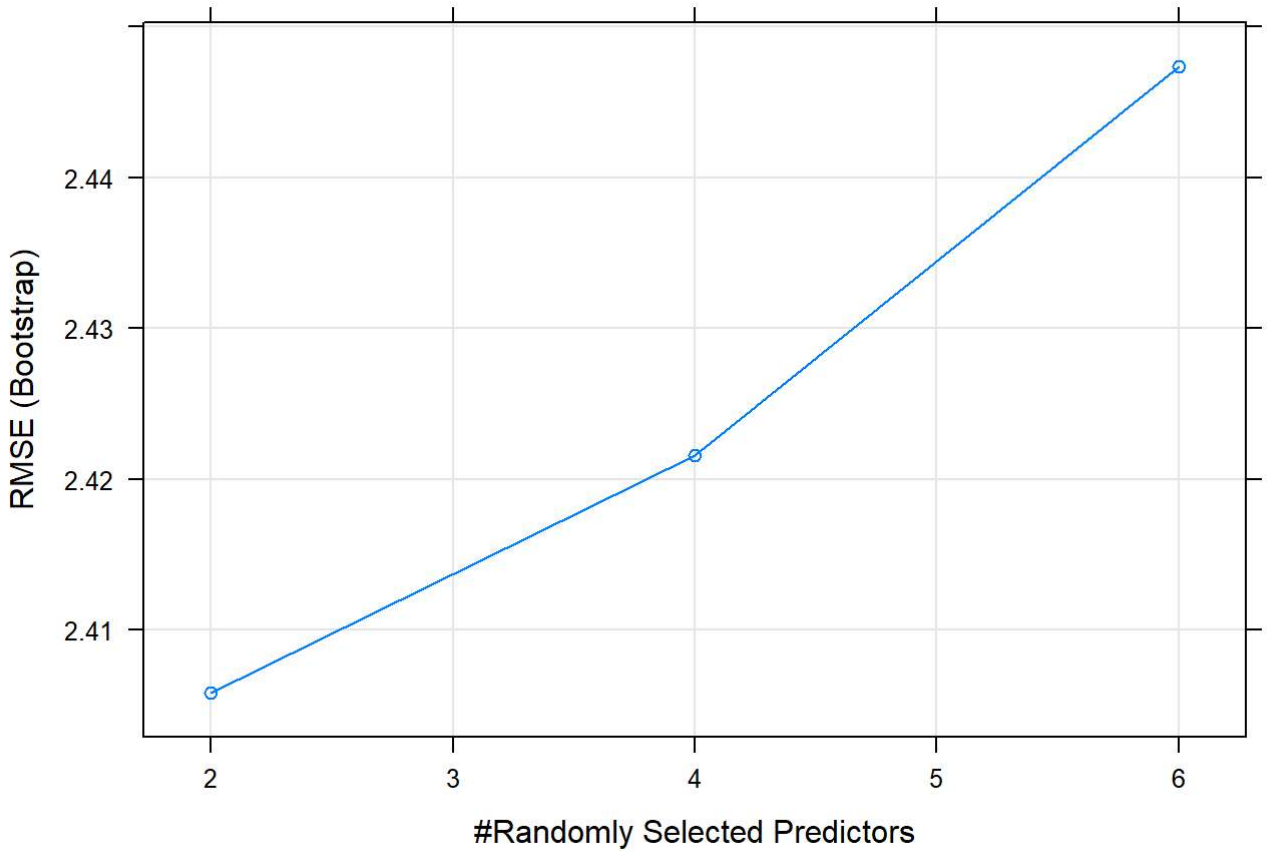
```
summary(Model_forest_caret)
```

```
##                 Length Class      Mode
## call              4    -none-     call
## type              1    -none-     character
## predicted       400    -none-     numeric
## mse             500    -none-     numeric
## rsq             500    -none-     numeric
## oob.times       400    -none-     numeric
## importance        6    -none-     numeric
## importanceSD      0    -none-     NULL
## localImportance   0    -none-     NULL
## proximity         0    -none-     NULL
## ntree             1    -none-     numeric
## mtry              1    -none-     numeric
## forest           11    -none-     list
## coefs             0    -none-     NULL
## y               400    -none-     numeric
## test              0    -none-     NULL
## inbag             0    -none-     NULL
## xNames            6    -none-     character
## problemType       1    -none-     character
## tuneValue         1    data.frame list
## obsLevels         1    -none-     logical
## param             0    -none-     list
```

```
print(Model_forest_caret)
```

```
## Random Forest
##
## 400 samples
##   6 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 400, 400, 400, 400, 400, 400, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     2.405819  0.2852547  1.926801
##   4     2.421577  0.2790266  1.934608
##   6     2.447373  0.2681323  1.953147
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
plot(Model_forest_caret)
```



Since **2** mtry has the lowest RMSE, this value is the best fit for mtry.

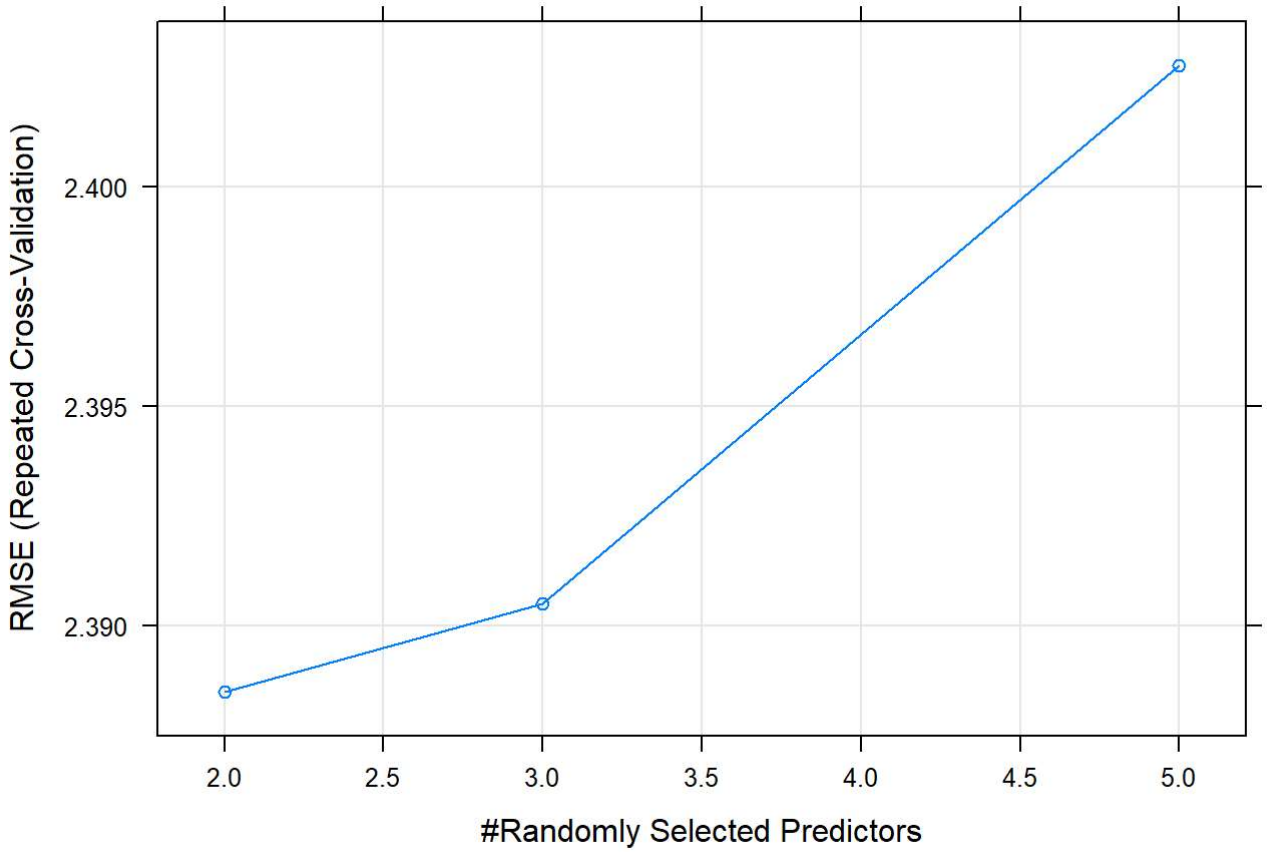
Question 4

"Customize the search grid by checking the model's performance for mtry values of 2, 3 and 5 using 3 repeats of 5-fold cross validation."

```
control <- trainControl(method="repeatedcv", number=5, repeats=3, search="grid")
tunegrid <- expand.grid(.mtry=c(2,3,5))
rf_gridsearch <- train(Sales~., data=Carseats_Filtered, method="rf", tuneGrid=tunegrid,trControl
=control)
print(rf_gridsearch)
```

```
## Random Forest
##
## 400 samples
##   6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 321, 320, 320, 320, 319, 320, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     2.388490  0.2902905  1.902942
##   3     2.390502  0.2898689  1.899672
##   5     2.402758  0.2869045  1.905036
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
plot(rf_gridsearch)
```



After checking mtry at 2,3, and 5 while using 5-fold crossvalidaion with 3 repeats, we still find that **2** mtry is the preferred mtry with the lowest RMSE of **2.388490**.