Assignment 2

Jacob Fabian

2023-04-08

#Q1 What is the key idea behind bagging? Can bagging deal both with high variance (overfitting) and high bias (underfitting)?

#A1 The main concept behind bagging is that we reduce the variance by randomly selecting samples of the training set and using the replacement, then each weak learner is fitted on each sample data. This is used to prevent overfitting and would not be an option for underfitting.

#Q2 Why bagging models are computationally more efficient when compared to boosting models with the same number of weak learners?

#A2 In Boosting, models are continuously grown and each model is grown using information from the previousl model, there is no sequential growth in bagging which makes boosting more efficient.

#Q3 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?

#A3 The big issue is James has made multiple decision tree models that are too similar. The ensemble method of boosting would help in this situation as the mistakes from previous models can be used to make the next models better. It is important to remember that having dissimilar trees are important as it helps create a more robust model. I would go back to the start and rebuild the decision trees to be further more different from each other by using a punishment for selecting a certain attribute at a specific level too many times.

#Q4 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?

#A4 Information gain = entropy(parent) - [avg. entropy (children)]

With the data provided the parent entropy is = 0.988699 small size entropy is = 0.811278 large size entropy is = 0.954434

We can determine by using the calculation that the Information Gain is 0.105843

#Q5 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.

#A5 If the parameter is too large then it wouldn't be different than using bagging and we wouldn't get diversity. If the parameter is set too small, each tree won't be very predictive since they will be constrained at each node to a small part of attributes. It is important to allow random forests to be optimally set as the key part in random forests so that at each node, a randomly sample of predictor are used so that not every node is similar and a more accurate predictor will be the result.

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(caret)
```

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

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

```
library(ISLR)
library(glmnet)
```

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

```
## Loaded glmnet 4.1-7
```

```
library(dplyr)
```

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

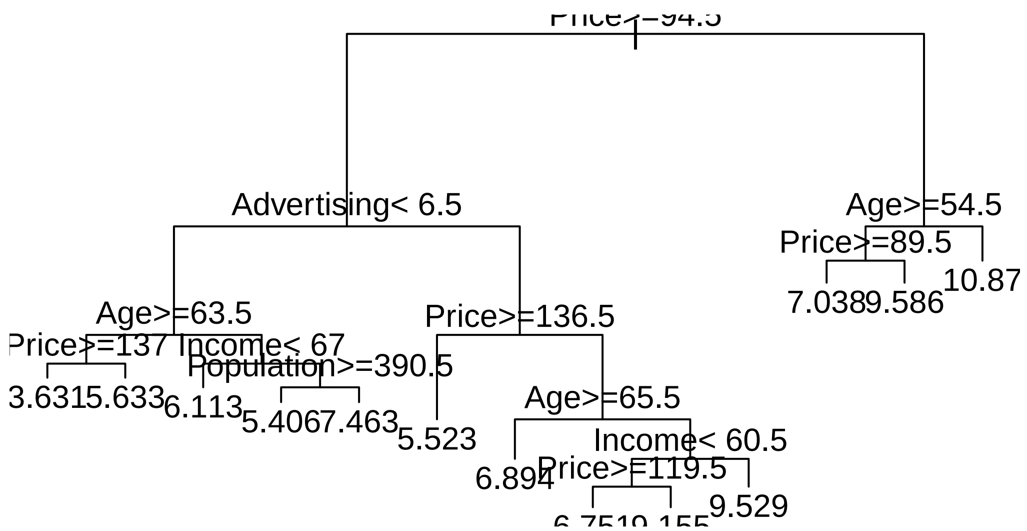
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

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

#Q1 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?

```
Model_1 <- rpart(Sales~., data=Carseats_Filtered, method = 'anova')
plot(Model_1)
text(Model_1)
```



Price greater than or equal to 94.5 is our root node for splitting

#Q2 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)
```

Predicting our sales since we have our test set to run with our model

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

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

After running our predict function, the decision tree indicates that 9.58625 sales will take place

#Q3 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?

```r
set.seed(123)
Model_forest_caret <- train(Sales~., data = Carseats_Filtered, method = 'rf')
```

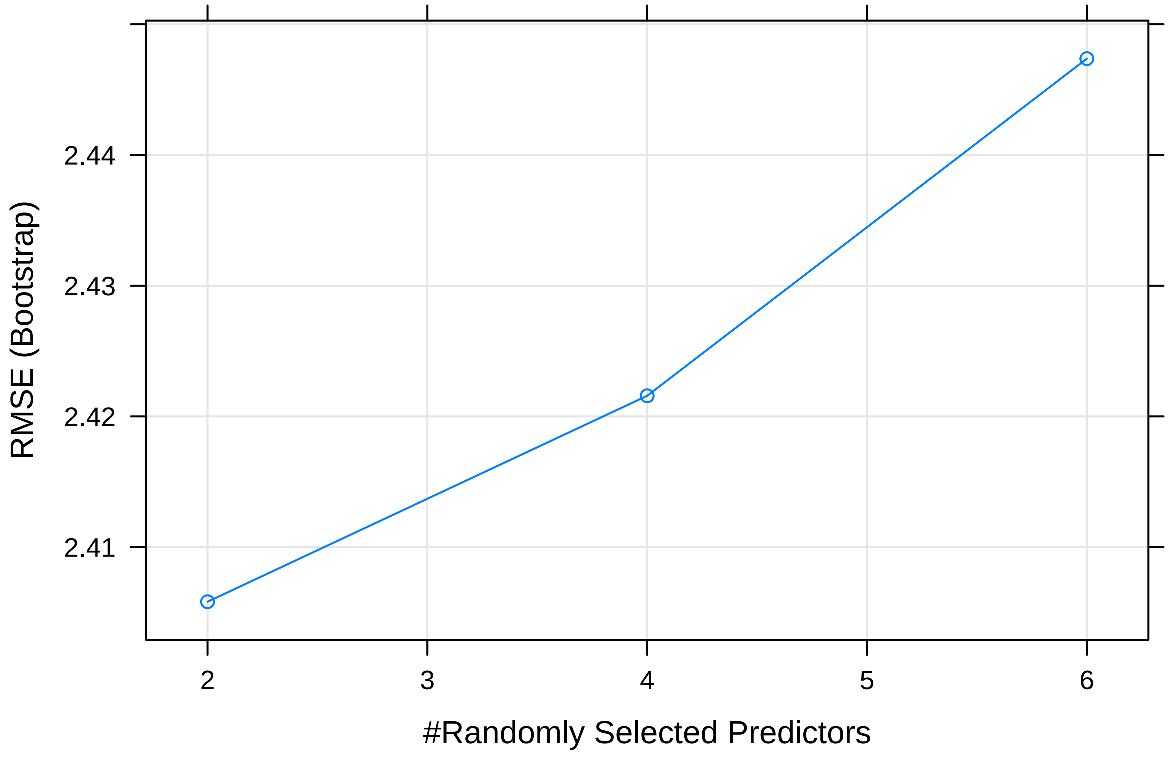
```r
summary(Model_forest_caret)
```

```
##                 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
```

```r
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.
```

```r
plot(Model_forest_caret)
```



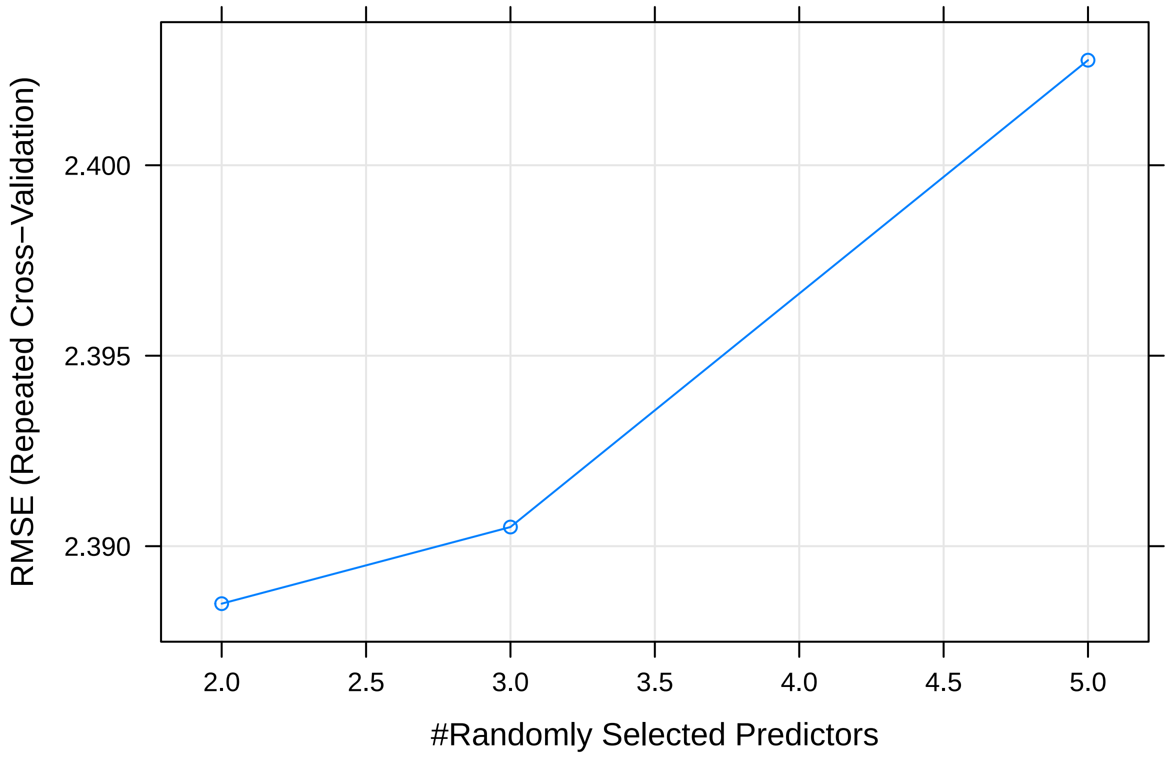
The mtry with the lowest RMSE is 2, this is the best fit for mtry

#Q4 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=contro
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)
```



While using 5 fold cross validation, and checking mtry at 2,3, and 5 with 3 repeats. We find that 2 mtry is the ideal mtry with the lowest RMSE of 2.388490