Advanced Data Mining Assignment 2

Sharik Baig

20/11/2022

PART A

Question A 1: What is the key idea behind bagging? Can bagging deal both with high variance (overfitting) and high bias (underfitting)?

Bagging, also known as Bootstrap Aggregation, is an ensemble machine learning algorithm.It is a general procedure that can be used to reduce the variance for algorithms that have high variance (overfitting).So this method (overfitting) seems more viable option then high bias (underfitting).

Question A 2: Why bagging models are computationally more efficient when compared to boosting models with the same number of weak learners?

Since in Boosting, trees are developed consecutively where each tree is developed utilizing data from the recently developed trees, where in stowing, there is no successive development.

Question A 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?

No, this is not a viable approach because the purpose of setting up an ensemble model is to have diversity, and James has produced decision models that are identical to each other in the given example, which defeats the purpose of utilizing ensemble models. As a result, I would recommend rebuilding the model and decision trees to be independent of one another, as this would not improve performance in the given situation.

Question A 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?

Entropy for our data set: I(all _ data) = -[(9/16)log 2(9/16)+(7/16)log2(7/16)] = *0.9836*

The entropy of *small size* = *0.811278* & The entropy of *large size* = *0.954434*.

Using this formula, we can calculate the *Information Gain* to be *0.105843*.

Question A 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 too large and close to p, then we are almost choosing all attributes at each node, and as such, we may not get a proper diversity among different individual trees. On the other hand, if m is too small, each individual tree is likely not to be very predictive as we have limited each note to a very small fraction of attributes which may not be predictive.For random forests to be optimally set is critical since the core principle in random forests is that a random sample of predictors is utilized at each node, resulting in a more accurate predictor because not every node is similar.

Part B

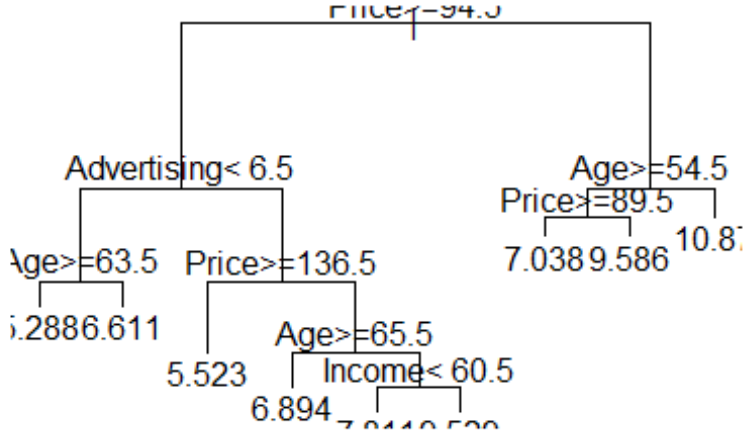
This part of the assignment involves building decision tree and random forest models to answer a number of questions. We will use the Carseats dataset that is part of the ISLR package.

Using the following libraries:

```
library(ISLR)
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(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-3

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

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

Question B 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?

" 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.

Question B 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?

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

We will anticipate Sales now that we have constructed our test set to run through our model.

```
Pred_sales_2 <- predict(Carseats_model_1, Test_Carseats)
Pred_sales_2

##       1
## 9.58625
```

The decision tree predicts that **9.58625** sales will occur with this particular record, according to our predict function.

Question B 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? (Make sure to set the random number generator seed to 123)

"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? "

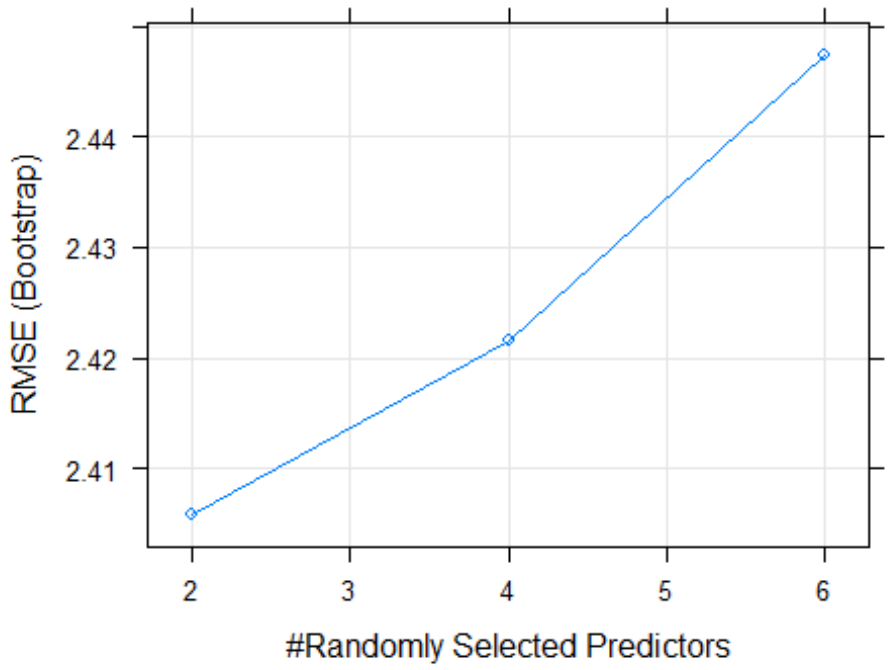
```
set.seed(123)
Model_forest <- train(Sales~., data = Carseats_Filtered, method = 'rf')

summary(Model_forest)

##                 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)
```

```
## 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)
```



This value is the best fit for mtry since **2** mtry has the lowest RMSE.

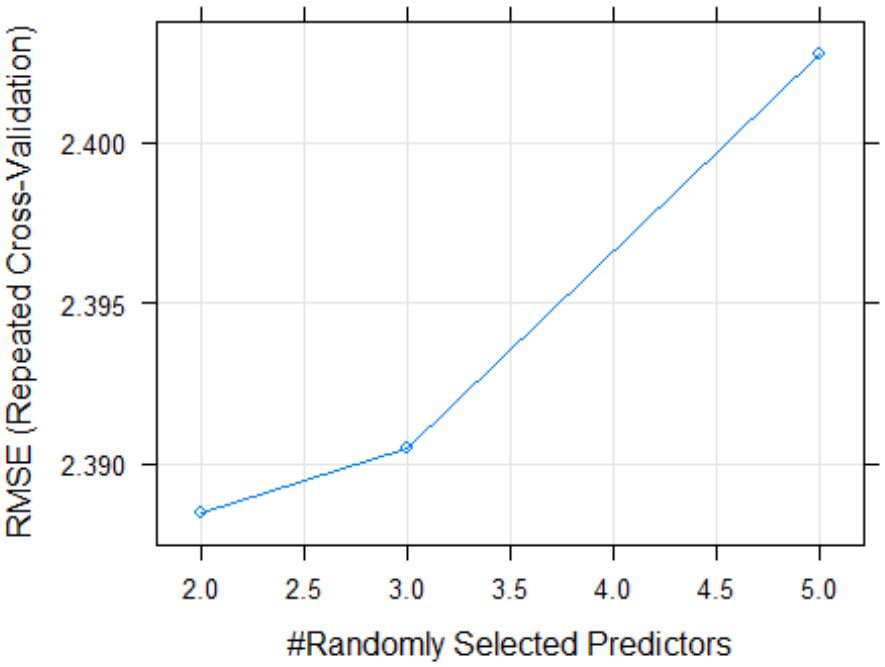
Question B 4: Customize the search grid by checking the model's performance for mtry values of 2, 3 and 5 using 3 repeats of 5-fold cross validation.

"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)
```



We still found that *2* mtry is the preferable mtry with the lowest RMSE of *2.388490* after checking mtry at 2,3, and 5 while utilizing 5-fold crossvalidation with 3 repeats.