Individual Assignment 2

Tejasvini Mavuleti

2022-11-19

ADVANCED DATA MINING & PREDICTIVE ANALYTICS

INDIVIDUAL ASSIGNMENT 2

PART A

#QA1) What is the key idea behind bagging? Can bagging deal both with high variance (overfitting) and high bias (underfitting)?

The concept of bagging is about using multiple base learners, trained separately with a random sample from the training set through a voting or an averaging approach to produce a more stable and accurate model. Bagging is also known as Bootstrap aggregating, and in simple terms, it is used to deal with bias-variance trade-offs and reduces the variance of a prediction model.

In decision trees, it is used when our goal is to reduce the variance of a decision tree. Here the idea is to create several subsets of data from training samples chosen randomly with replacements. Now, each collection of subset data is used to train their decision trees. And in random forest models, it acts as an ensemble algorithm that fits multiple models on different subsets of a training data set, then combines the predictions from all models. Random forest is an extension of bagging that also randomly selects subsets of features used in each data sample. We can't improve the model's predictive force by increasing the training set's size. Instead, it decreases the variance and narrowly tunes the prediction to an expected outcome. These multisets of data are used to train multiple models. As a result, we end up with an ensemble of different models. The average of all the predictions from different models is used. This is more robust than a model. Prediction can be the average of all the predictions given by the different models in the regression case. In the case of classification, the majority vote is taken into consideration.

We also learned that bagging is one of the best ways to reduce over fitting but could be useless for under fitting problems.

Bagging on high variance models: The variance of the model will be reduced without increasing the bias. However, the performance of this model will be better, so bagging is recommended.Bagging on high-bias models: The model's accuracy will always drop compared to the model we could have obtained without bagging. The bagging accuracy increases as the number of bagged models increases. As n reaches infinity, the accuracy of the bagged model will be equal to the accuracy of the direct model. As the model's accuracy always stays the same in the case of biased models, the use of bagging is dissuaded.

It is also known that it reduces variance, so it enormously benefits high-variance classifiers. As the prediction is an average of many classifiers, you obtain a mean score and variance. It can be interpreted as the uncertainty of the prediction. Especially in regression tasks, such uncertainties are otherwise hard to get. A low bias and a low variance are the two most fundamental features expected for a model, although they most often vary in opposite directions. Indeed, to be able to solve a problem, we want our model to have enough degrees of freedom to resolve the underlying complexity of the data we are working with. Still, we also want it to have a few degrees of freedom to avoid high variance and be more robust. This is the well-known bias-variance trade-off.

QA2) Why bagging models are computationally more efficient when compared to boosting models with the same number of weak learners?

The most significant advantage of Bagging is that multiple weak learners can work better than a single strong learner. It provides stability and increases the machine learning algorithm's accuracy, which is used in statistical classification and regression. In addition, it helps in reducing variance, i.e., it avoids overfitting. We call weak learners or base models that can be used as building blocks for designing more complex models by combining several. Most of the time, these basic models perform poorly because they have a high bias - a low degree of freedom model, for example, or because they have too much variance to be robust (high degree of freedom models. Then, the idea of ensemble methods is to combine several of them to create a strong learner or ensemble model that achieves better performances.

Bagging is a model averaging approach; the final predictions are defined by combining the projections from all the models to improve the stability and accuracy of machine learning algorithms, rather than training multiple classifiers independently, boosting works. In doing so, the new classifier might now get the wrong cases that the earlier classifier got correct, making Bagging computationally more efficient.

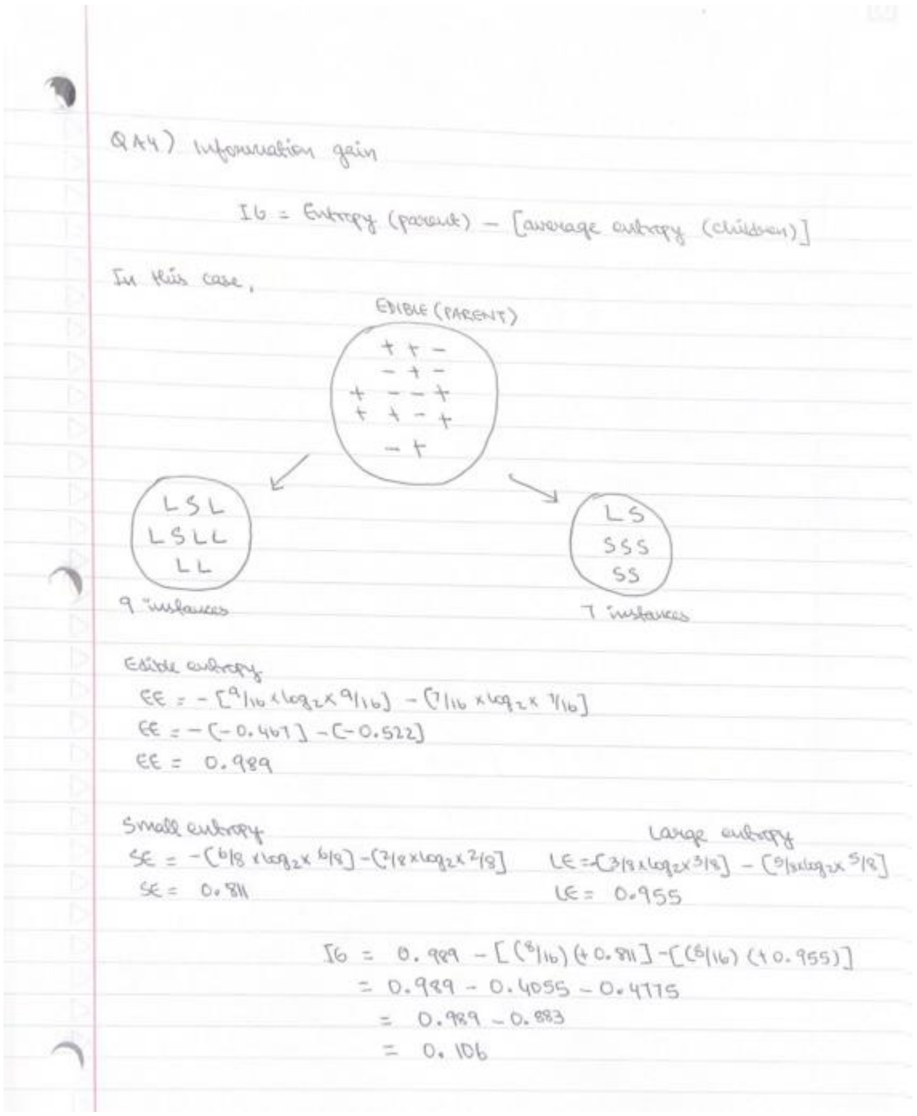
QA3) Do you think creating an ensemble model by combining these tree models can boost the performance? Discuss your answer.

In this case, James is thinking of creating several decision tree models that are similar to each other. It could be useful if he uses the ensemble method of boosting to re-use the failed iterations and improve them in the next step. James must try to make sure that the decision trees are different to develop a much more robust model. This approach enables the production of a much better predictive performance than a single model to predict the state of a given stock in the next week.

The best-known ensemble learning Algorithms are bagging and boosting. These two can potentially decrease the variance of a single estimate as they merge several estimates from different decision tree models. However, for ensemble models to be able to work as planned, the base learners should be better than a random model and be independent of each other. Neither is the case in the above example; hence, the above example would not boost performance. So, the result would be a model with higher stability. I think I would suggest he start over and begin from scratch to see if there is any change in the results and find if any specific attribute repeats too many times and is the purpose of reducing the model performance.

QA4) What would be the information gain for splitting the dataset based on the “Size” attribute?

Information gain calculates the reduction in entropy or surprise from transforming a data set in some way. It is commonly used in the construction of decision trees from a training data set, by evaluating the information gain for each variable, and selecting the variable that maximizes the information gain, which in turn minimizes the entropy and best splits the data set into groups for effective classification. In this case, the objects are color, size, shape and the parent category is edible/ or not edible.



QA4) Information gain

IG = Entropy (parent) − [average entropy (children)]

In this case,

EDIBLE (PARENT)

+ + −
− + −
+ − − +
+ + − +
− +

LSL
LSLL
LL

9 instances

LS
SSS
SS

7 instances

Edible entropy

$EE = -[9/16 \times \log_2 \times 9/16] - [7/16 \times \log_2 \times 7/16]$

$EE = -[-0.467] - [-0.522]$

$EE = 0.989$

Small entropy

$SE = -[6/8 \times \log_2 \times 6/8] - [7/8 \times \log_2 \times 2/8]$

$SE = 0.811$

Large entropy

$LE = -[3/8 \times \log_2 \times 3/8] - [5/8 \times \log_2 \times 5/8]$

$LE = 0.955$

$$
\begin{aligned}
IG &= 0.989 - [(8/16)(+0.811)] - [(8/16)(+0.955)] \\
&= 0.989 - 0.4055 - 0.4775 \\
&= 0.989 - 0.883 \\
&= 0.106
\end{aligned}
$$

The Information Gain is 0.106

QA5) Discuss the implications of setting this parameter too small or too large.

It is important to use Random Forest models to decide where and how to split the data based on the selection features. Splitting the data into nodes implements a level of differentiation because each tree based on x variables that split based on unique features. In such cases, we must select attributes from a data set and ensure it gives out the best performance. Now, we develop the individual tree with the maximum depth and cluster the trees using weights and averages to predict the performance of the unseen data. The problem is that a single decision tree is overly sensitive to data variations because it could easily overfit noise in the data. The Random Forest with only one tree will overfit to data because it is the equivalent of a single decision tree. If an analyst increases the number of trees in the Random Forest, then the propensity to overfitting should decline. Therefore, this differentiation allows a robot ensemble to generate a much more accurate predictor. And the reason behind selecting an optimal parameter is to reduce the diversity of each individual tree. If it is too low, then the individual trees will not be predictive. So, the best optimal m for this kind of Random Forest model, m parameters would depend on a problem that stands out. Therefore it should be used as mere tuning parameters.

Part B

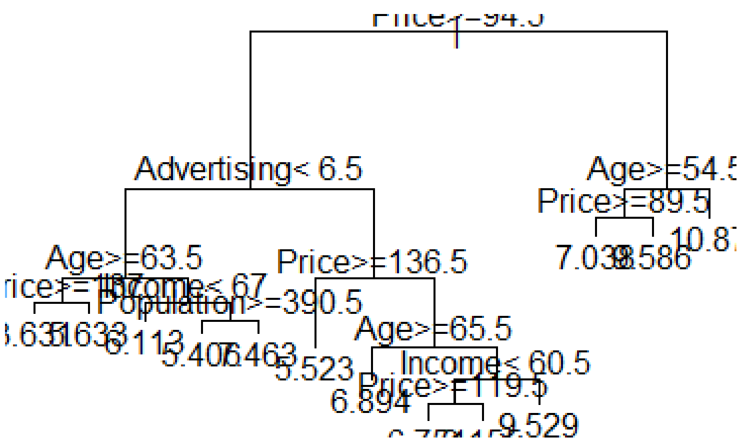
Building Decision Tree and Random Forest Models

```
library(caret)

## Warning: package 'caret' was built under R version 4.2.1

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.2.1

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.2.1

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(ISLR)

## Warning: package 'ISLR' was built under R version 4.2.1

library(glmnet)

## Warning: package 'glmnet' was built under R version 4.2.1

## Loading required package: Matrix

## Loaded glmnet 4.1-4

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

```
## Warning: package 'rpart.plot' was built under R version 4.2.2
```

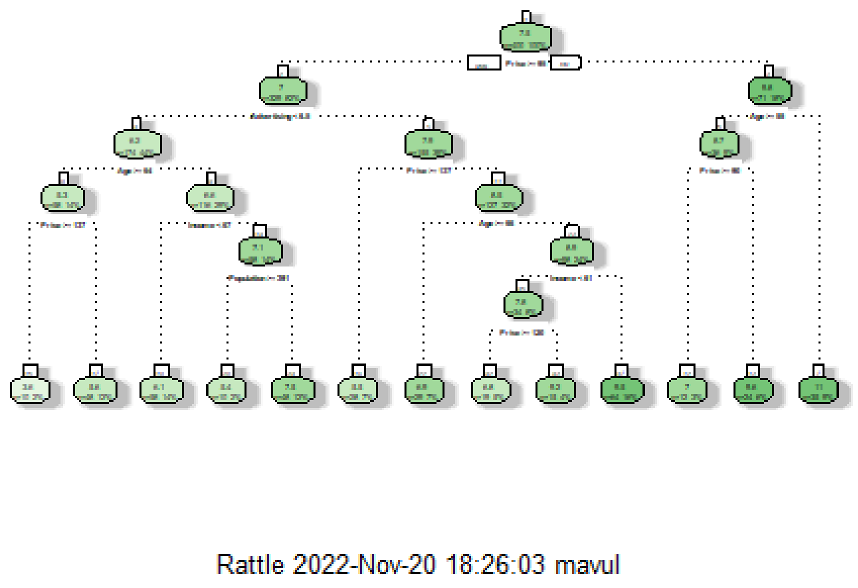
```
Carseats_Filtered <- Carseats %>% select("Sales", "Price", "Advertising", "Po
pulation", "Age", "Income", "Education")
```

QB1) Building a decision tree regression model to predict sales



The root node for splitting for this model is Price. And the result shows that price is greater than or equal to 94.5.

```
library(rattle)

## Warning: package 'rattle' was built under R version 4.2.2

## Loading required package: tibble

## Loading required package: bitops

##
## Attaching package: 'bitops'

## The following object is masked from 'package:Matrix':
##
##     %&%
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

fancyRpartPlot(Predict_model)
```



In this Rpart Plot, the top most variable importance is price and the comes the advertisement.

QB2) Estimating the sales using the decision tree model

Given - Sales=9, Price=6.54, Population=124, Advertising=0, Age=76, Income= 110, Education=10

```
Sales <- c(9)
Price <- c(6.54)
Population <- c(124)
Advertising <- c(0)
Age <- c(76)
Income <- c(110)
Education <- c(10)
Estimate_model <- data.frame(Sales,Price,Population,Advertising,Age,Income,Ed
ucation)
```

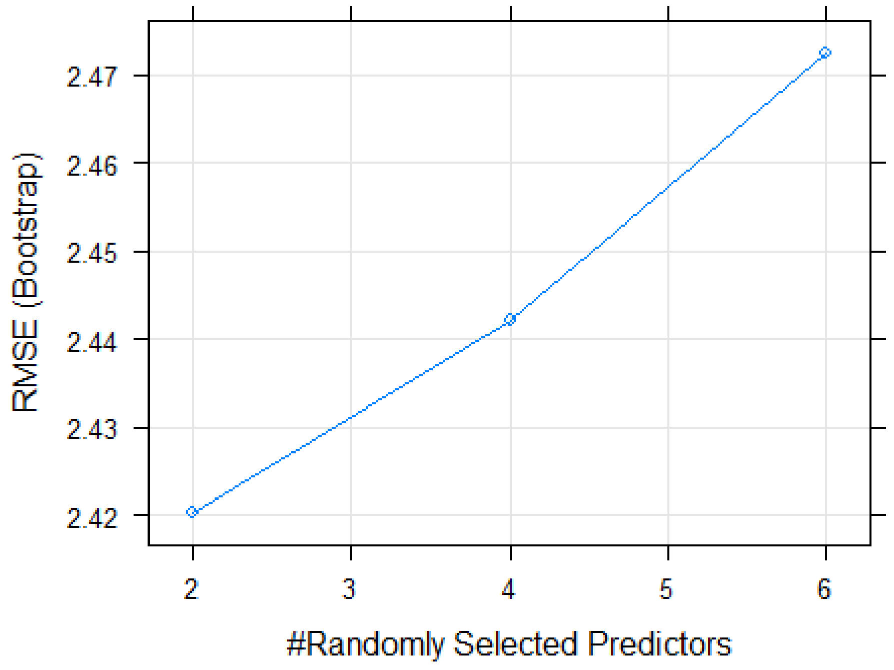
In the next step, we need to run our test set - estimate model to predict the accurate sales value.

```
Predict_Sales_Model <- predict(Predict_model, Estimate_model)
Predict_Sales_Model

##       1
## 9.58625
```

Using the decision tree model, the predicted sales value is 9.59 when we use the given record of sales = 9.

QB3) Using Caret function and Random Forest

```
set.seed(64037)
Caret_random_forest_model <- train(Sales~., data = Carseats_Filtered, method
= 'rf')

summary(Caret_random_forest_model)

##                  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(Caret_random_forest_model)

## 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     2.420356  0.2748871  1.930059
##   4     2.442112  0.2689776  1.946698
##   6     2.472423  0.2576078  1.969846
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.

plot(Caret_random_forest_model)
```

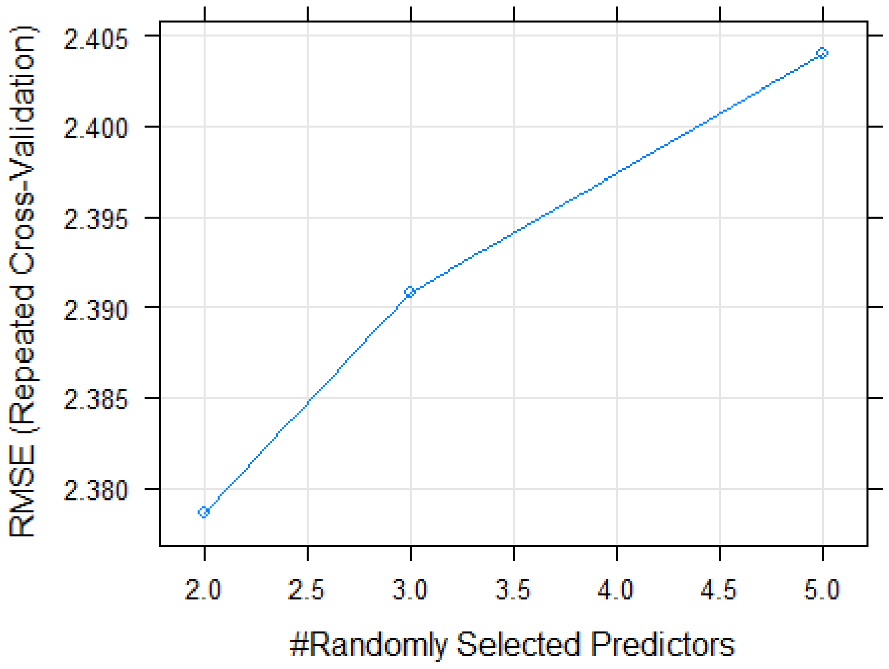


The plot clearly shows that the lowest RMSE is at 2 mtry and this means that 2 is the best performance in this model.

QB4 Checking the model's performance for mtry values of 2, 3 and 5 using 3 repeats of 5-fold cross validation

```
Train_controller <- trainControl(method="repeatedcv", number=5, repeats=3, se
arch="grid")
tunegrid <- expand.grid(.mtry=c(2,3,5))

New_grid <- train(Sales~., data=Carseats_Filtered, method="rf", tuneGrid=tune
grid,trControl=Train_controller)
print(New_grid)
```

```
## Random Forest
##
## 400 samples
##   6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 320, 320, 321, 320, 319, 321, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     2.378629  0.2973694  1.903298
##   3     2.390821  0.2914276  1.906833
##   5     2.403998  0.2892133  1.921023
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.

plot(New_grid)
```



The new grid shows that even after checking mtry at 2,3, and 5 with 5-fold cross validaion and 3 repeats, 2 mtry is still the best spot for mtry with the lowest RMSE of 2.378629.

Conclusion

Bagging is an approach that tries to stabilize the predictions from trees. Rather that fitting a single tree to the data, we will bootstrap the data and for each bootstrap sample, we will fit a tree. The final prediction is an average of all the predictions of the trees, if the outcome is continuous, or the class that was most common among the predictions, if the outcome is categorical and the downside of this approach is that we loose the interpret ability of a single tree. This was one of the unique ways to interpret and see if there is any change using decision trees and randomly selecting the predictors for the best out of the best performing models.