Assignment 2

2023-04-07

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

Ans. Bagging, also called bootstrap Aggregation, is an ensemble machine-learning algorithm. The key idea of bagging involves selecting multiple samples of the training set with replacement, thereby creating multiple samples of the same training dataset. The different samples of the training sets are trained, thus yielding different results for different samples. All the results from the different are aggregated to make a single prediction. This process can be repeated k-times until there is no further improvement.

Yes, the main reason to use the concept of bagging is to reduce high variance (overfitting). Since the individual classifiers are overfitted due to the model learning too much about the data. Bagging creates different samples with replacements, so the final ensemble model helps reduce the variance or overfitting. In case of high bias or underfitting, bagging doesn't perform the same because if the individual classifiers are under-fitting, the final ensemble model also under-fits.

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

Ans. The different training set samples are trained independently in bagging, then aggregated into the final mode until the model achieves a better result. Therefore, the model training in bagging is parallelized. In boosting, the models are trained sequentially based on the previous model's performance. The model creates new iterations of the training set based on the previous model's accuracy to reduce the previous model's error. Thus, increasing computational complexity for every iteration minimizes errors and improves accuracy.

Abhinav Thupili- athupili@kent.edu (811234108)

QA3. James is thinking of creating an ensemble model 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.

Ans. The concept of the ensemble model is to combine several individual models to improve overall performance. The ensemble model combines various strengths of different to improve accuracy. In the case of James, several decision tree models are weak. When such weak models are used to ensemble models, the model's output would also be weak, thus not providing any significant performance improvement.

Another reason the James ensemble cannot boost the performance is that one of the main drawbacks of the decision tree is that it is not fit for continuous numerical variables. James wants to predict the stock values, a continuous numerical variable.

QA4. 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?

Ans.

Information gain = entropy(parent) – [average entropy(children)]

Calculating **Size**(parent entropy):

$-\ 9/16.\log_2(9/16) - 7/16.\log_2(7/16)$

$-\ 9/16(-0.83) - 7/16(-1.1926)$

Parent(Size) entropy = **0.98**

Calculating "**Small" and "Large"** (Children entropy):

Calculating Child ("Small" entropy):

$-\ 6/8.\log_2(6/8) - 2/8.\log_2(2/8)$

$-\ 6/8(-0.415) - 2/8.\log_2(-2)$

Child ("Small" entropy) = **0.81**

Calculating Child ("Large" entropy):

$-3/8.\log_2(3/8) - 5/8.\log_2(5/8)$

$-3/8(-1.415) - 5/8(-0.6781)$

Child ("Large" entropy) = **0.95**

Weighted Average Entropy of Children = 8/16(0.81) + 8/16(0.95) = 0.88

Information gain = 0.98 – 0.88 = 0.10.

QA5. 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.

Ans. It is important to choose the optimal value for m as this creates diversity in the decision tree model while choosing the attribute at each split. Also, each decision tree is not restricted as it has options from the model. By selecting the m too small, the attributes selected at each split are too restricted. Thus, the decision tree model performance would be suboptimal. By selecting the m too large, the model has no diversity, as all the decision tree models are similar and only diversity is created by bootstrapping and nothing more.

```
library(ISLR)
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-7

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

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)
library(rattle)

## 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.

library(rpart.plot)
library(randomForest)

## randomForest 4.7-1.1

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

##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
##
##     importance

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

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

#Loading and selecting required columns from ISLR carseats dataset.

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

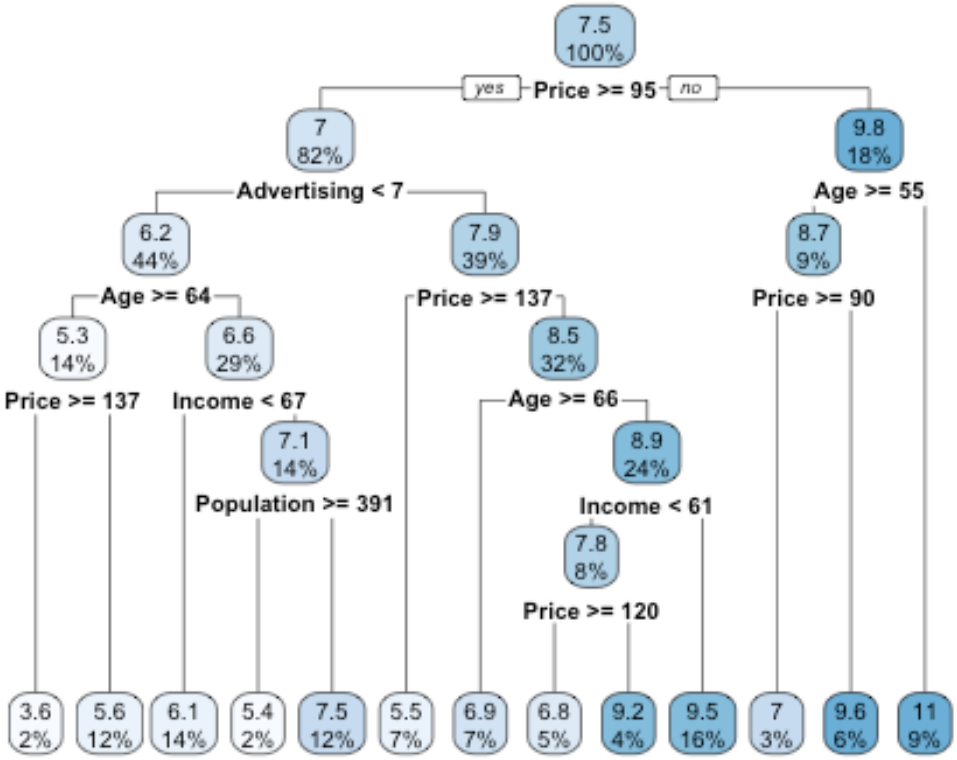
#QB1.Predicting Sales based on all attributes.

```
model_1<- rpart(Sales~., data = carseats, method = "anova")
print(model_1)

## n= 400
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 400 3182.27500  7.496325
##    2) Price>=94.5 329 2242.20000  7.001672
##      4) Advertising< 6.5 174  859.96840  6.169655
##        8) Age>=63.5 58  228.35070  5.287586
##         16) Price>=137 10   56.90169  3.631000 *
##         17) Price< 137 48  138.28890  5.632708 *
##        9) Age< 63.5 116  563.92770  6.610690
##         18) Income< 67 58  216.86830  6.113448 *
##         19) Income>=67 58  318.37860  7.107931
##           38) Population>=390.5 10   25.08524  5.406000 *
##           39) Population< 390.5 48  258.29310  7.462500 *
##      5) Advertising>=6.5 155 1126.56300  7.935677
##       10) Price>=136.5 28  142.35800  5.522857 *
##       11) Price< 136.5 127  785.25910  8.467638
##         22) Age>=65.5 29  176.41950  6.893793 *
##         23) Age< 65.5 98  515.75040  8.933367
##           46) Income< 60.5 34  161.72260  7.811471
##             92) Price>=119.5 19   64.19938  6.751053 *
##             93) Price< 119.5 15   49.09537  9.154667 *
##           47) Income>=60.5 64  288.49920  9.529375 *
##    3) Price< 94.5 71  486.55130  9.788451
##      6) Age>=54.5 36  178.59760  8.736944
##       12) Price>=89.5 12   34.64357  7.038333 *
```

```
##       13) Price< 89.5 24   92.01896  9.586250 *
##      7) Age< 54.5 35  227.20860 10.870000 *

rpart.plot(model_1)
```




```
#The attributes used at the top of Tree are Price, Advertising, Age.
```

#QB2.What will be the estimated Sales for this record using the decision tree model?

```
new_carseats<- data.frame("Sales" = 9, "Price" = 6.54, "Population" = 124,
"Advertising" = 0, "Age" = 76, "Income" = 110, "Education" = 10)
model_2<- predict(model_1, new_carseats, method = "anova")
model_2
```

```
##       1
## 9.58625

#The estimated Sales for this record is predicted by the Decision Tree model
is 9.58625.
```

#QB3.Using Caret function to train a random forest model.

```
set.seed(123)
model_3<- train(Sales~.,data=carseats, method= "rf")
print(model_3)
```

```
## 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     2.404492  0.2860284  1.925868
##   4     2.422719  0.2782951  1.935483
##   6     2.446748  0.2686232  1.952884
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

#The best performance of mtry value when using method ="rf" is 2, as the RMSE is 2.404492.

#QB4.Customizing the search grid by checking the model's performance for mtry values of 2, 3, 5 and 3-repeats of 5-fold cross validation.

```
set.seed(123)
model_4 <- trainControl(method = "repeatedcv", number = 5, repeats = 3,search
= "grid")
model_5 <- train(Sales~., data = carseats, method = "rf",trControl =
model_4,tuneGrid = expand.grid(mtry=c(2,3,5)))
model_5
```

```
## Random Forest
##
## 400 samples
##   6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 320, 321, 319, 320, 320, 319, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     2.405235  0.2813795  1.930855
##   3     2.401365  0.2858295  1.920612
##   5     2.417771  0.2821938  1.934886
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 3.
```

##The best performance of mtry value when using method ="rf" is 2,3,5 is 3, as the RMSE is 2.401365.