Advanced Data Mining Assignment-2

Zachariah Alex

2023-04-04

PART A

QA1.What is the key idea behind bagging? Can bagging deal both with high variance(over-fitting) and high bias (under-fitting)?

Bagging a.k.a bootstrap aggregation is the process of training classifiers on different subset of random samples with replacement, of the data set.This is due to that fact that a single algorithm may not make the perfect prediction for a given data set.

Bagging can deal with the problem of high variance (over-fitting) but not high bias(under-fitting).

Since we are training each base learners on different samples of data set there is diversity in training samples.The base learners we use are weak learners and they are less likely to over-fit the training data compared to complex models and even if one base model over fits ,they will tend to over fit in a different way compared to the other base learner since others are trained on different samples of data.Therefore when we ensemble these base learners the probability of the model to over-fit will be very low This is not possible in the case of high bias.Each base learners are weak learners and there is a chance that these models might fail to capture the underlying patterns in the data.So even if we ensemble these models together the combined model will not be good enough to make predictions with high accuracy if underlying base learners are under-fitted.Therefore bagging can deal with over-fitting but not under-fitting.

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

Bagging is the process of training different models on different subset of training data independently and combining them to make the final prediction whereas in boosting different models are trained sequentially. Additionally bagging does not need a lot of data as the models are trained in parallel which gives a better computational efficiency compared to boosting. Each model in boosting is trained on the output or errors made by its predecessor so that the errors made by the first model or patterns unrecognized by the first model is rectified or captured by the second model.In boosting, models are fit iteratively so that the training of a given model at a given stage depends on the models fit at prior steps. When it comes to multiple models,it will become too computationally complex to fit sequentially when compared to bagging where models are trained and fit parallelly.Therefore we can say that bagging is more computationally efficient than boosting

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.

The two key requirement in creating an ensemble model is that the base learners should be better than a random model or should have some predictive power and the model should be independent of each other.Since all the base learners considered by James is of similar to each other there is no diversity among them and all the models will have similar outputs.Therefore even if we ensemble these model together this will not contribute in boosting the overall performance as base learners are not performing better than a random model and lack diversity between them.

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?

The measure of disorder or entropy of two classes is calculated by the following equation

$$Entropy = \sum_{i=1}^{n} -P(x=i).log_2.P(x=i),$$

where P(x=i) is the probability of class i

Information gain is the reduction in entropy or it helps to identify the best split for a decision tree which can reduce the entropy.

Information gain =Entropy(parent)-[average entropy(children)]

$$Entropy(parent) = (-9/16 * log_2 9/16) - (7/16 * log_2 7/16) = 0.988699$$

$$Entropy(children(large))[3 - edible/5 - nonedible] = (-3/8 * log_2 3/8) - (5/8 * log_2 5/8) = 0.954433$$

$$Entropy(children(small))[6 - edible/2 - nonedible] = (-6/8 * log_2 6/8) - (2/8 * log2/8) = 0.811277$$

$$Weighted[Average.Entropy](children = (8/16) * 0.954433 + (8/16) * 0.811277 = 0.882854$$

Information Gain = (Parent Entropy - average[child Entropy])

$$Information.Gain = 0.988699 - 0.882854 = 0.105845$$

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.

A random forest may consider the same attribute with highest predictive power or information gain for splitting at the top node and continue to behave similarly as we further split the child nodes.This will result in similar models as there is no diversity involved even if we use bagging.In order to address this issue in a random forest model,out of all the 'p' predictor variables we choose a set of random sampled predictors called 'm 'which is the number of attributes available for each split.Since all the features are not available for each node of each tree ,each tree will be different and this helps in incorporating diversity among different trees which is a key requirement for a efficient ensemble model.Therefore a random Forest model decides where to split based on a random selection of features

If m = p,it is just bagging and we are considering all the parameters for splitting at each node of each tree. If m parameter is too small ,then we are only considering very few attributes and each tree will not be having a good predictive power and this can lead to under-fitting. If m is parameter is too large ,we are considering almost all the attributes which can lead to similar trees and less diversity The default value for m in classification is the square root of p and minimum value of of node is one whereas in regression the default value for m is p/3 and minimum node size is five. In practical,the optimal value for m depends on the type of problem we are dealing with and should be considered as a tuning hyper-parameter.

PART B

```
knitr::opts_chunk$set(message = FALSE)
knitr::opts_chunk$set(warning = FALSE)
```

Loading Required Library

```
library(ISLR)
library(dplyr)
library(glmnet)
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
```

Filtering required attributes for our new data set

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

head(data)
```

```
##   Sales Price Advertising Population Age Income Education
## 1  9.50   120          11        276  42     73        17
## 2 11.22    83          16        260  65     48        10
## 3 10.06    80          10        269  59     35        12
## 4  7.40    97           4        466  55    100        14
## 5  4.15   128           3        340  38     64        13
## 6 10.81    72          13        501  78    113        16
```

QB1

```
model<-rpart (Sales~.,data=data, method = 'anova')

summary(model)
```

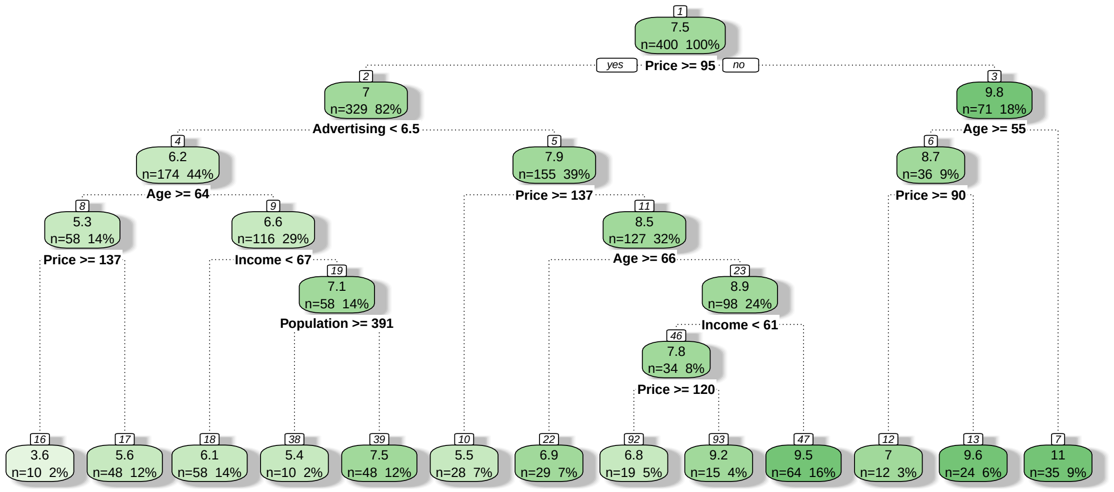
```
## Call:
## rpart(formula = Sales ~ ., data = data, method = "anova")
##   n= 400
##
##            CP nsplit rel error    xerror       xstd
## 1  0.14251535      0 1.0000000 1.0134303 0.06977157
## 2  0.08034146      1 0.8574847 0.9233810 0.06606936
## 3  0.06251702      2 0.7771432 0.8721170 0.06295329
## 4  0.02925241      3 0.7146262 0.7960983 0.05603114
## 5  0.02537341      4 0.6853738 0.8321664 0.05751408
## 6  0.02127094      5 0.6600003 0.8236392 0.05694477
## 7  0.02059174      6 0.6387294 0.8174191 0.05539030
## 8  0.01632010      7 0.6181377 0.8024554 0.05386180
## 9  0.01521801      8 0.6018176 0.7915028 0.05490226
## 10 0.01042023      9 0.5865996 0.8185036 0.05625182
## 11 0.01000559     10 0.5761793 0.8562661 0.05894232
## 12 0.01000000     12 0.5561681 0.8565394 0.05898521
##
## Variable importance
##       Price Advertising         Age      Income  Population   Education
##          49          18          16           8           6           3
##
## Node number 1: 400 observations,    complexity param=0.1425153
##   mean=7.496325, MSE=7.955687
##   left son=2 (329 obs) right son=3 (71 obs)
##   Primary splits:
##       Price       < 94.5  to the right, improve=0.14251530, (0 missing)
##       Advertising < 7.5   to the left,  improve=0.07303226, (0 missing)
##       Age         < 61.5  to the right, improve=0.07120203, (0 missing)
##       Income      < 61.5  to the left,  improve=0.02840494, (0 missing)
##       Population  < 174.5 to the left,  improve=0.01077467, (0 missing)
##
## Node number 2: 329 observations,    complexity param=0.08034146
##   mean=7.001672, MSE=6.815199
##   left son=4 (174 obs) right son=5 (155 obs)
##   Primary splits:
##       Advertising < 6.5   to the left,  improve=0.11402580, (0 missing)
##       Price       < 136.5 to the right, improve=0.08411056, (0 missing)
##       Age         < 63.5  to the right, improve=0.08091745, (0 missing)
##       Income      < 60.5  to the left,  improve=0.03394126, (0 missing)
##       Population  < 23    to the left,  improve=0.01831455, (0 missing)
##   Surrogate splits:
##       Population < 223   to the left,  agree=0.599, adj=0.148, (0 split)
##       Education  < 10.5  to the right, agree=0.565, adj=0.077, (0 split)
##       Age        < 53.5  to the right, agree=0.547, adj=0.039, (0 split)
##       Income     < 114.5 to the left,  agree=0.547, adj=0.039, (0 split)
##       Price      < 106.5 to the right, agree=0.544, adj=0.032, (0 split)
##
## Node number 3: 71 observations,    complexity param=0.02537341
##   mean=9.788451, MSE=6.852836
##   left son=6 (36 obs) right son=7 (35 obs)
##   Primary splits:
##       Age        < 54.5  to the right, improve=0.16595410, (0 missing)
##       Price      < 75.5  to the right, improve=0.08365773, (0 missing)
##       Income     < 30.5  to the left,  improve=0.03322169, (0 missing)
##       Education  < 10.5  to the right, improve=0.03019634, (0 missing)
##       Population < 268.5 to the left,  improve=0.02383306, (0 missing)
##   Surrogate splits:
##       Advertising < 4.5   to the right, agree=0.606, adj=0.200, (0 split)
##       Price       < 73    to the right, agree=0.592, adj=0.171, (0 split)
##       Population  < 272.5 to the left,  agree=0.592, adj=0.171, (0 split)
##       Income      < 79.5  to the right, agree=0.592, adj=0.171, (0 split)
##       Education   < 11.5  to the left,  agree=0.577, adj=0.143, (0 split)
##
## Node number 4: 174 observations,    complexity param=0.02127094
##   mean=6.169655, MSE=4.942347
##   left son=8 (58 obs) right son=9 (116 obs)
##   Primary splits:
##       Age         < 63.5  to the right, improve=0.078712160, (0 missing)
##       Price       < 130.5 to the right, improve=0.048919280, (0 missing)
##       Population  < 26.5  to the left,  improve=0.030421540, (0 missing)
##       Income      < 67.5  to the left,  improve=0.027749670, (0 missing)
##       Advertising < 0.5   to the left,  improve=0.006795377, (0 missing)
##   Surrogate splits:
##       Income     < 22.5  to the left,  agree=0.678, adj=0.034, (0 split)
##       Price      < 96.5  to the left,  agree=0.672, adj=0.017, (0 split)
##       Population < 26.5  to the left,  agree=0.672, adj=0.017, (0 split)
##
```

```
## Node number 5: 155 observations,    complexity param=0.06251702
##   mean=7.935677, MSE=7.268151
##   left son=10 (28 obs) right son=11 (127 obs)
##   Primary splits:
##       Price       < 136.5 to the right, improve=0.17659580, (0 missing)
##       Age         < 73.5  to the right, improve=0.08000201, (0 missing)
##       Income      < 60.5  to the left,  improve=0.05360755, (0 missing)
##       Advertising < 13.5  to the left,  improve=0.03920507, (0 missing)
##       Population  < 399   to the left,  improve=0.01037956, (0 missing)
##   Surrogate splits:
##       Advertising < 24.5  to the right, agree=0.826, adj=0.036, (0 split)
##
## Node number 6: 36 observations,    complexity param=0.0163201
##   mean=8.736944, MSE=4.961043
##   left son=12 (12 obs) right son=13 (24 obs)
##   Primary splits:
##       Price       < 89.5  to the right, improve=0.29079360, (0 missing)
##       Income      < 39.5  to the left,  improve=0.19043350, (0 missing)
##       Advertising < 11.5  to the left,  improve=0.17891930, (0 missing)
##       Age         < 75.5  to the right, improve=0.04316067, (0 missing)
##       Education   < 14.5  to the left,  improve=0.03411396, (0 missing)
##   Surrogate splits:
##       Advertising < 16.5  to the right, agree=0.722, adj=0.167, (0 split)
##       Income      < 37.5  to the left,  agree=0.722, adj=0.167, (0 split)
##       Age         < 56.5  to the left,  agree=0.694, adj=0.083, (0 split)
##
## Node number 7: 35 observations
##   mean=10.87, MSE=6.491674
##
## Node number 8: 58 observations,    complexity param=0.01042023
##   mean=5.287586, MSE=3.93708
##   left son=16 (10 obs) right son=17 (48 obs)
##   Primary splits:
##       Price      < 137   to the right, improve=0.14521540, (0 missing)
##       Education  < 15.5  to the right, improve=0.07995394, (0 missing)
##       Income     < 35.5  to the left,  improve=0.04206708, (0 missing)
##       Age        < 79.5  to the left,  improve=0.02799057, (0 missing)
##       Population < 52.5  to the left,  improve=0.01914342, (0 missing)
##
## Node number 9: 116 observations,    complexity param=0.01000559
##   mean=6.61069, MSE=4.861446
##   left son=18 (58 obs) right son=19 (58 obs)
##   Primary splits:
##       Income     < 67    to the left,  improve=0.05085914, (0 missing)
##       Population < 392   to the right, improve=0.04476721, (0 missing)
##       Price      < 127   to the right, improve=0.04210762, (0 missing)
##       Age        < 37.5  to the right, improve=0.02858424, (0 missing)
##       Education  < 14.5  to the left,  improve=0.01187387, (0 missing)
##   Surrogate splits:
##       Education   < 12.5  to the right, agree=0.586, adj=0.172, (0 split)
##       Age         < 58.5  to the left,  agree=0.578, adj=0.155, (0 split)
##       Price       < 144.5 to the left,  agree=0.569, adj=0.138, (0 split)
##       Population  < 479   to the right, agree=0.560, adj=0.121, (0 split)
##       Advertising < 2.5   to the right, agree=0.543, adj=0.086, (0 split)
```

```
##
## Node number 10: 28 observations
##   mean=5.522857, MSE=5.084213
##
## Node number 11: 127 observations,    complexity param=0.02925241
##   mean=8.467638, MSE=6.183142
##   left son=22 (29 obs) right son=23 (98 obs)
##   Primary splits:
##       Age         < 65.5  to the right, improve=0.11854590, (0 missing)
##       Income      < 51.5  to the left,  improve=0.08076060, (0 missing)
##       Advertising < 13.5  to the left,  improve=0.04801701, (0 missing)
##       Education   < 11.5  to the right, improve=0.02471512, (0 missing)
##       Population  < 479   to the left,  improve=0.01908657, (0 missing)
##
## Node number 12: 12 observations
##   mean=7.038333, MSE=2.886964
##
## Node number 13: 24 observations
##   mean=9.58625, MSE=3.834123
##
## Node number 16: 10 observations
##   mean=3.631, MSE=5.690169
##
## Node number 17: 48 observations
##   mean=5.632708, MSE=2.88102
##
## Node number 18: 58 observations
##   mean=6.113448, MSE=3.739109
##
## Node number 19: 58 observations,    complexity param=0.01000559
##   mean=7.107931, MSE=5.489285
##   left son=38 (10 obs) right son=39 (48 obs)
##   Primary splits:
##       Population  < 390.5 to the right, improve=0.10993270, (0 missing)
##       Price       < 124.5 to the right, improve=0.07534567, (0 missing)
##       Advertising < 0.5   to the left,  improve=0.07060488, (0 missing)
##       Age         < 45.5  to the right, improve=0.04611510, (0 missing)
##       Education   < 11.5  to the right, improve=0.03722944, (0 missing)
##
## Node number 22: 29 observations
##   mean=6.893793, MSE=6.08343
##
## Node number 23: 98 observations,    complexity param=0.02059174
##   mean=8.933367, MSE=5.262759
##   left son=46 (34 obs) right son=47 (64 obs)
##   Primary splits:
##       Income      < 60.5  to the left,  improve=0.12705480, (0 missing)
##       Advertising < 13.5  to the left,  improve=0.07114001, (0 missing)
##       Price       < 118.5 to the right, improve=0.06932216, (0 missing)
##       Education   < 11.5  to the right, improve=0.03377416, (0 missing)
##       Age         < 49.5  to the right, improve=0.02289004, (0 missing)
##   Surrogate splits:
##       Education < 17.5  to the right, agree=0.663, adj=0.029, (0 split)
##
```

```
## Node number 38: 10 observations
##   mean=5.406, MSE=2.508524
##
## Node number 39: 48 observations
##   mean=7.4625, MSE=5.381106
##
## Node number 46: 34 observations,    complexity param=0.01521801
##   mean=7.811471, MSE=4.756548
##   left son=92 (19 obs) right son=93 (15 obs)
##   Primary splits:
##       Price       < 119.5 to the right, improve=0.29945020, (0 missing)
##       Advertising < 11.5  to the left,  improve=0.14268440, (0 missing)
##       Income      < 40.5  to the right, improve=0.12781140, (0 missing)
##       Population  < 152   to the left,  improve=0.03601768, (0 missing)
##       Age         < 49.5  to the right, improve=0.02748814, (0 missing)
##   Surrogate splits:
##       Education   < 12.5  to the right, agree=0.676, adj=0.267, (0 split)
##       Advertising < 7.5   to the right, agree=0.647, adj=0.200, (0 split)
##       Age         < 53.5  to the left,  agree=0.647, adj=0.200, (0 split)
##       Population  < 240   to the right, agree=0.618, adj=0.133, (0 split)
##       Income      < 41.5  to the right, agree=0.618, adj=0.133, (0 split)
##
## Node number 47: 64 observations
##   mean=9.529375, MSE=4.5078
##
## Node number 92: 19 observations
##   mean=6.751053, MSE=3.378915
##
## Node number 93: 15 observations
##   mean=9.154667, MSE=3.273025
```

```
fancyRpartPlot(model)
```



Rattle 2023−Apr−09 13:16:49 ZAC

```
#The price attribute is at the root node for splitting
```

The price attribute is at the root node for splitting with a condition price greater than or equal to a particular value

QB2

```
#Estimated sales using the following record : Sales=9,Price=6.54,Population=124,Advertising=0,Age=76,In

#Creating a data frame with the following records

#Since we are considering Sales value as the target variable ,we will not include sales attribute in pr

pred_data = data.frame(Price=6.54 ,Population=124,Advertising=0,Age=76,Income= 110, Education= 10)


Estimated_sales<- predict(model,pred_data)

Estimated_sales
```

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

The predicted estimated sales value is 9.58625

QB3

```r
#Training a random forest model

set.seed(123)


model_2 <- train(Sales~., data= data,method = "rf")

#printing the model

print(model_2)
```

```
## 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
#mtry =2 has the least RMSE Value
```

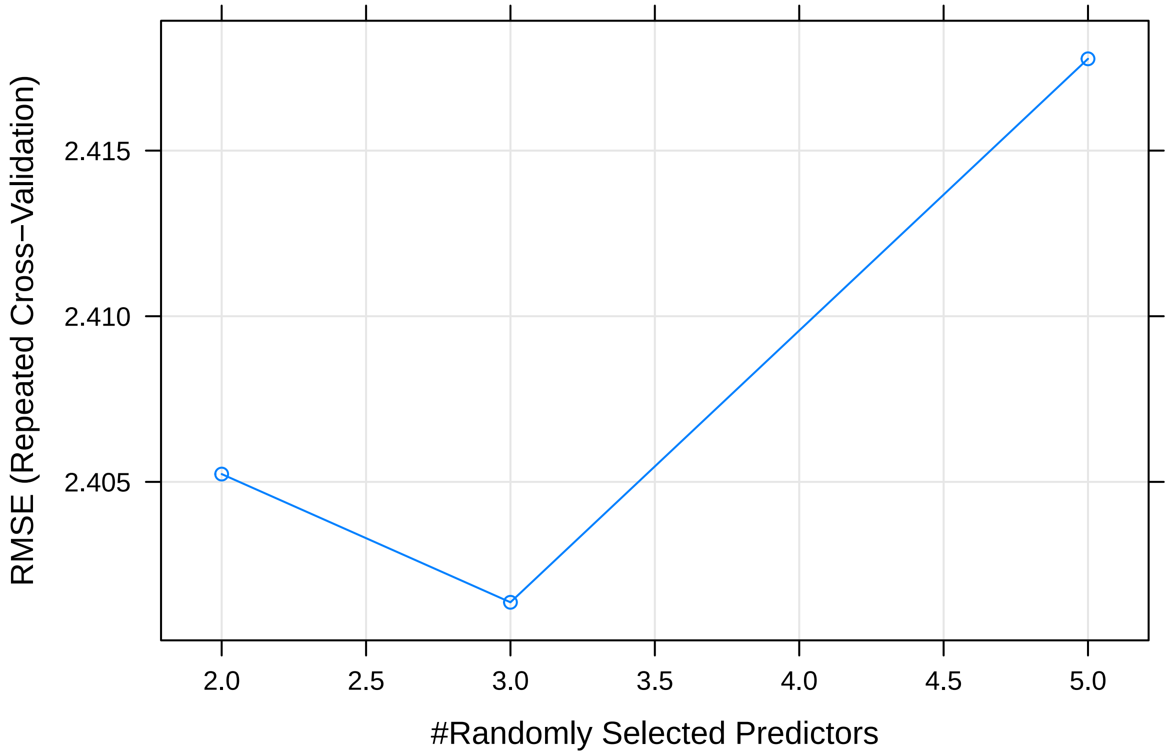
The best value for mtry=2 which has the least RMSE value

QB4

```r
set.seed(123)

#Customizing  with tuning parameters and  repeats of 5-fold cross validation.

custom <- trainControl(method="repeatedcv", number=5, repeats=3)

#defining mtry values in search grid

grids <- expand.grid(mtry=c(2,3,5))

g_search <- train(Sales~., data=data, method="rf",tuneGrid=grids, trControl=custom)

print(g_search)
```

```
## 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.
```

```
#Plotting

plot(g_search)
```



Optimal value for mtry=3 where RMSE is the least