Assignmennnn02-ADM

Chathurani Ekanayake

2023-11-12

Part A

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

Bagging, or bootstrap aggregating, is an ensemble learning technique that combines multiple models to improve the overall performance of the prediction. The key idea behind bagging is to reduce the variance of the individual models by averaging their predictions. This is particularly useful for models that are prone to overfitting, which is when the model performs well on the training data but poorly on unseen data

Can bagging deal with both high variance and high bias?

Yes, bagging can be used to deal with both high variance and high bias. For high variance (overfitting) models, bagging will reduce the variance without significantly affecting the bias. For high bias (underfitting) models, bagging may not improve the bias, but it may still improve the overall performance by reducing the variance.

In general, bagging is most effective for high variance (overfitting) models. However, it can still be beneficial for high bias models, especially if the bias is not too high.

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

Bagging and boosting are both ensemble learning techniques that combine multiple weak learners to improve the overall performance of the prediction.

Bagging is generally more computationally efficient than boosting because the individual weak learners in bagging can be trained in parallel. This is because the weak learners in bagging are independent of each other, and they do not need to be trained sequentially.As well as bagging is also more robust to outliers. So any outliers in the training data will have less impact on the overall performance of the model.

Boosting, on the other hand, requires the weak learners to be trained sequentially. This is because each weak learner is trained to correct the errors of the previous weak learners. As a result, boosting can be significantly more computationally expensive than bagging, especially for large datasets.

QA3. 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? Discuss your answer.

Combining James' decision tree models into an ensemble model could potentially improve the performance of predicting stock price movements, but it is not guaranteed. Ensemble methods have been shown to be effective in various machine learning tasks, including stock market prediction, by combining the strengths of multiple models to reduce overall error and improve generalizability.

However, in the case of James' decision tree models, which are currently performing no better than random guessing, it's crucial to address the underlying issue before attempting to combine them into an ensemble. If the individual models are not performing well, simply averaging or combining their predictions is unlikely to significantly improve the overall accuracy.

The fact that the decision tree models are very similar suggests that they may be overfitting to the training data. Overfitting occurs when a model learns the training data too closely and fails to generalize well to new data. This is often caused by having too many features or too complex models.

Once James has addressed the underlying issues with the individual decision trees, then he can consider creating an ensemble model. Ensemble methods can be more effective when the individual models have different strengths and weaknesses.

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?

Calculate the entropy of the original dataset based on the "Edible?" attribute.

Number of edible objects (+): 9 Number of non-edible objects (-): 7 Total number of objects: 16

Edible entropy (parent) Entropy (S) = - (p(+) * log2(p(+)) + p(-) * log2(p(-))) Entropy (S) = - [(9/16) * log2(9/16) + (7/16) * log2(7/16)] Entropy = 0.988699

the entropy of the small size = −(6/8·log2(6/8))−(2/8·log2(2/8))= 0.811278

large size entropy = −(3/8.log2(3/8))−(5/8·log2(5/8))= 0.954434

The average Entropy of the child is as follows:(0.811(8/16)) + (0.955(8/16))= 0.883

Information gain = Entropy(original dataset) - Weighted average of the entropy of the split datasets Information Gain = (0.988-0.883)= 0.105843

The information gain of 0.105843 indicates that the "Size" attribute is moderately useful for predicting whether an object is edible or not.This is a relatively small information gain, but

it is still significant. It means that we can expect to improve the accuracy of our model by using the "Size" attribute to predict whether an object is edible or not.

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.

Setting "m" too small: Underfitting: If "m" is set too small, it means that each decision tree in the random forest has access to only a limited subset of the features during its training process. This can lead to underfitting, where the trees are not able to capture the full complexity of the data, resulting in low predictive performance. Lack of diversity: One of the key strengths of random forests is their ability to create diverse decision trees through random feature selection. When "m" is too small, the trees in the ensemble become highly correlated, and the diversity is reduced. As a result, the ensemble's predictive power may decrease.

Setting "m" too large: Reduced decorrelation: If "m" is set too large, it may allow too many features to be available for each split in the decision trees. This can reduce the degree of decorrelation between the individual trees, which is one of the key mechanisms for improving prediction accuracy in random forests. Increased complexity: Larger "m" values can lead to decision trees with more complex structures, which may result in overfitting, especially when there is noise or irrelevant features in the data. Overfitting can lead to poor generalization to new, unseen data.

Part B

Loading required packages

```
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)

## Warning: package 'glmnet' was built under R version 4.2.3

## Loading required package: Matrix

## Loaded glmnet 4.1-7
```

```
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(ggplot2)
library(lattice)
library(Matrix)
library(rpart)
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.2.3
```

For this assignment, we only need the following attributes: “Sales”, “Price”, “Advertising”, “Population”, “Age”, “Income” and “Education”. The goal of the assignment is to build models to predict the sales of the carseats (“Sales” attribute) using the other attributes. We can use the dplyr select function to select these attributes.

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

QB1. 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? Hint: you can either plot () and text() functions or use the summary() function to see the decision tree rules.

Checking which attribute comes on the to of the tree

```
Model <-rpart(Sales~.,data = Carseats_filterd,method = 'anova')
summary(Model)

## Call:
## rpart(formula = Sales ~ ., data = Carseats_filterd, method = "anova")
##   n= 400
##
##            CP nsplit rel error    xerror       xstd
## 1  0.14251535      0 1.0000000 1.0091019 0.06956748
## 2  0.08034146      1 0.8574847 0.9178642 0.06212606
## 3  0.06251702      2 0.7771432 0.9281445 0.06564015
## 4  0.02925241      3 0.7146262 0.8548635 0.05961271
## 5  0.02537341      4 0.6853738 0.8549491 0.06034780
## 6  0.02127094      5 0.6600003 0.8273679 0.05874249
## 7  0.02059174      6 0.6387294 0.8336204 0.05837823
## 8  0.01632010      7 0.6181377 0.8239175 0.05550043
## 9  0.01521801      8 0.6018176 0.8222870 0.05430287
## 10 0.01042023      9 0.5865996 0.8300420 0.05616488
## 11 0.01000559     10 0.5761793 0.8465790 0.05562714
## 12 0.01000000     12 0.5561681 0.8562404 0.05668935
```

```
##
## 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)
##
```

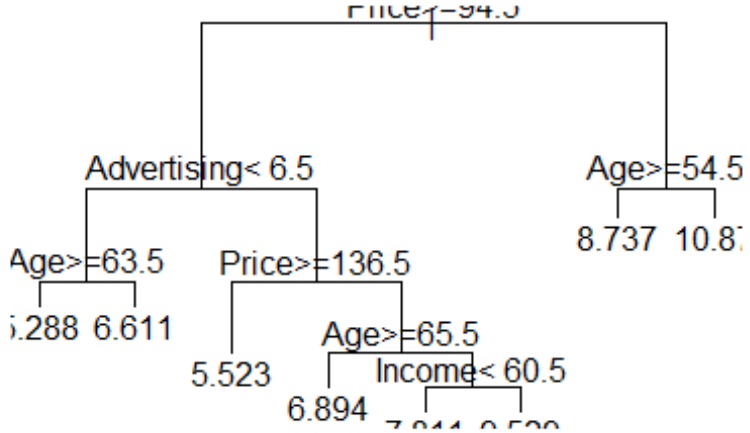
```
## Node number 9: 116 observations,    complexity param=0.01000559
##   mean=6.61069, MSE=4.861446
##   left son=18 (58 obs) right son=19 (58 obs)
##   Primary splits:
##       Income     < 67    to the left,  improve=0.05085914, (0 missing)
##       Population < 392   to the right, improve=0.04476721, (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
```

According to the above summary details most importabn attribute on deciding sales is "price".SO it is the attribute which should come on the top of the tree model. This can be proved by the following plots.

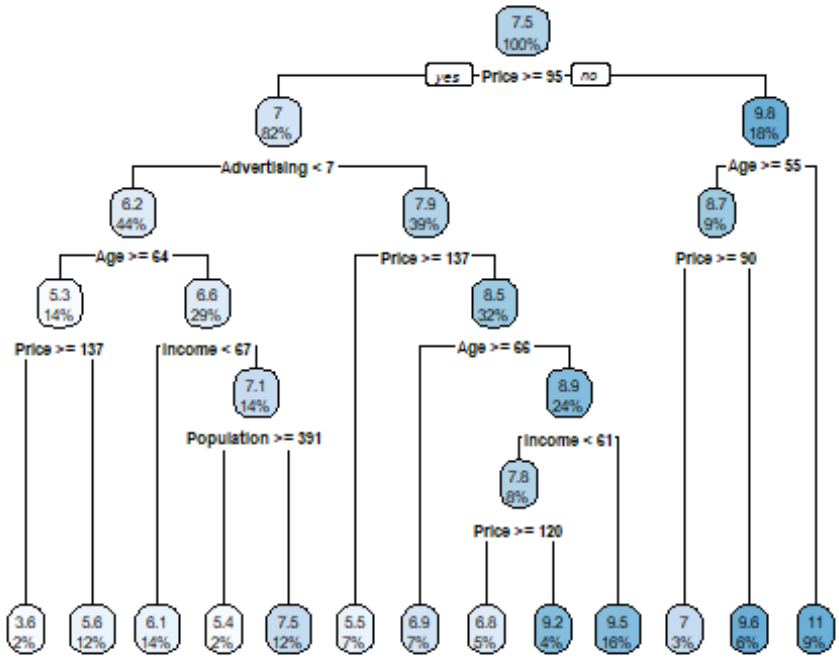
Less complex plot

```
Dec_Tree1 <- rpart(Sales~.,data = Carseats_filterd,method = "anova",control =
rpart.control(minsplit = 60))
plot(Dec_Tree1)
text(Dec_Tree1)
```



fancy RpartPlot

```
Dec_Tree2 <- rpart(Sales~.,data=Carseats_filterd,method='anova')
rpart.plot(Dec_Tree2)
```



QB2. 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?***

```
My_Data <-
data.frame(Price=6.54,Population=124,Advertising=0,Age=76,Income=110,Educatio
n=10)
Est_Sales<-predict(Dec_Tree2,My_Data)
Est_Sales

##       1
## 9.58625
```

According to the above information, the estimated sales for this record using decision tree model is 9.58625.

QB3. 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?

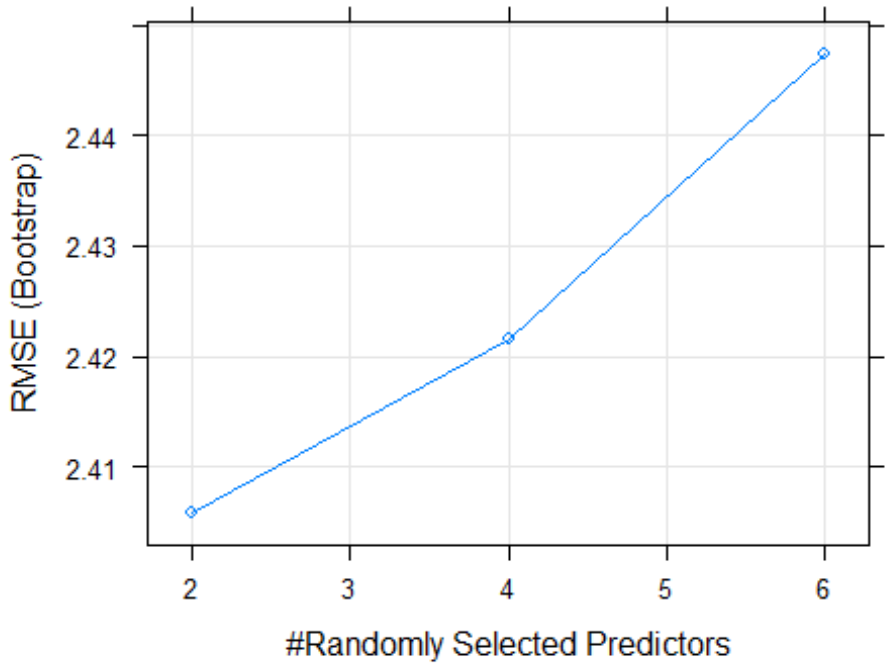
```
set.seed(123)
Ran_forest <-train(Sales~.,data = Carseats_filterd,method='rf')
summary(Ran_forest)

##                 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(Ran_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(Ran_forest)
```



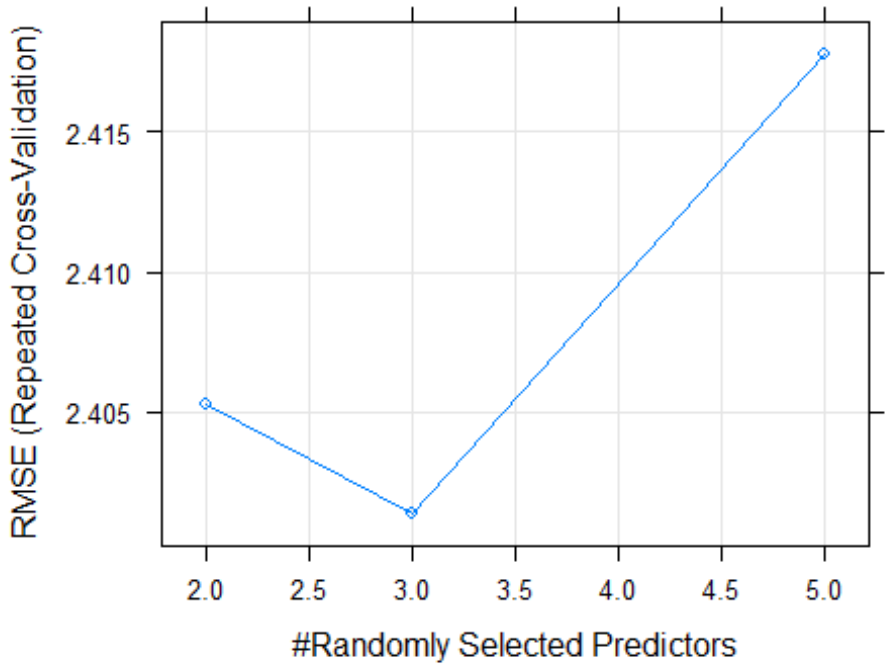
According to the above summary, RMSE is the lowest when mtry=2. Therefore, mtry 2 gives the best performance.

QB4. 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.

```
library(caret)
set.seed(123)
C_grid <- trainControl(method = "repeatedcv",number = 5,repeats = 3,search =
"grid")
C_grid2 <- expand.grid(.mtry=c(2,3,5))
RF_grid <-
train(Sales~.,data=Carseats_filterd,method="rf",tuneGrid=C_grid2,trControl=C_
grid)
print(RF_grid)
```

```
## 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.

plot(RF_grid)
```



According to the above summary, RMSE is the lowest when mtry=3. Therefore, mtry 3 gives the best performance.

References:

https://www.ibm.com/topics/bagging

https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/