Part A

Please read the following questions carefully and answer each question.

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

Both Bagging and boosting are component of a sequence of statistical techniques referred to as ensemble methods. They are an ensemble machine learning Algorithm designed to reduce the variance for those algorithms that are identified to have high variance in a model.

The concept behind bagging is to combine the results of multiple models of weak learners to form a strong learner to be able to generalize the result (Kasturi, 2019) .

Bagging decreases only overfitting (high variance), but not underfitting (low variance) it has been established that even if individual classifiers are overfitted, they are likely to be overfitted in unique ways in such a way that the final ensemble reduces overfitting. However, If the individual classifiers are underfitted because of the same bias, the ensemble will equally be underfitted which means that bagging cannot deal with overfitting.

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

Although we know that Bagging and boosting are comparable in the sense that they are both ensemble techniques for which a set of weak learners are blended to create a strong learner that generate a better performance than a single learner, Bagging significantly decreases the model's variance relative to Boosting. The efficiency of Bagging over Boosting depends on the data, the simulation and the circumstances. Generally if the complexity of the single model is over-fitting, Bagging is more efficient and the best option for Boosting doesn't help to prevent over-fitting.

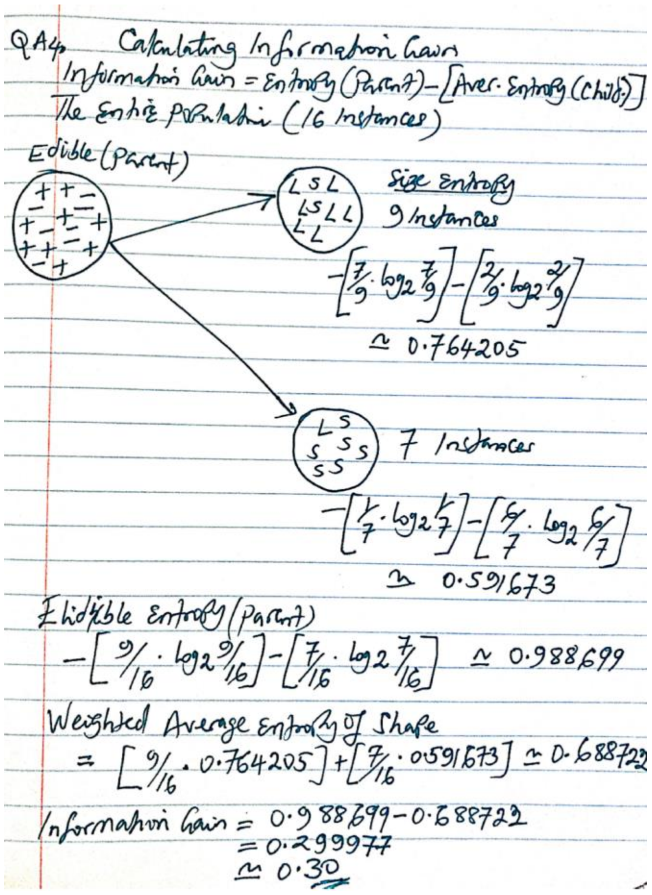
Bagging being a special case of the model averaging approach, the final predictions are defined by combination of the predictions from all the models to improve the stability and accuracy of machine learning algorithms (Soumya, 2019). Rather than training multiple classifiers independently, boosting works iteratively, however, in doing so, the new classifier might now get wrong cases that the earlier classifier got correct which makes Bagging computationally more efficient.

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.

Yes, reasons being that when we combined these three models, we now create an ensemble learning that helps us improve machine learning results by integrating several models. This approach enables the production of a much better predictive performance compared to 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 and these two have the potential to decrease the variance of a single estimate as they merge several estimates from different decision tree models. So, at the end of the day, the result would be a model with higher stability.

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?

Color	Size	Shape	Edible?
Yellow	Small	Round	+
Yellow	Small	Round	−
Green	Small	Irregular	+
Green	Large	Irregular	−
Yellow	Large	Round	+
Yellow	Small	Round	+
Yellow	Small	Round	+
Yellow	Small	Round	+
Green	Small	Round	−
Yellow	Large	Round	−
Yellow	Large	Round	+
Yellow	Large	Round	−
Yellow	Large	Round	−
Yellow	Large	Round	−
Yellow	Small	Irregular	+
Yellow	Large	Irregular	+



QA4. Calculating Information Gain

Information Gain = Entropy (Parent) − [Aver. Entropy (Child)]

The Entire Population (16 instances)

Edible (Parent)

+ + −
− + − +
+ − − +
+ + − +
− +
L S L
L S L L
L L
L S
S S S
S S

Size Entropy

9 Instances

$-\left[\frac{7}{9} \cdot \log_2 \frac{7}{9}\right] - \left[\frac{2}{9} \cdot \log_2 \frac{2}{9}\right]$

$\simeq 0.764205$

7 Instances

$-\left[\frac{1}{7} \cdot \log_2 \frac{1}{7}\right] - \left[\frac{6}{7} \cdot \log_2 \frac{6}{7}\right]$

$\simeq 0.591673$

Elidible Entropy (Parent)

$-\left[\frac{9}{16} \cdot \log_2 \frac{9}{16}\right] - \left[\frac{7}{16} \cdot \log_2 \frac{7}{16}\right] \simeq 0.988699$

Weighted Average Entropy of Shape

$= \left[\frac{9}{16} \cdot 0.764205\right] + \left[\frac{7}{16} \cdot 0.591673\right] \simeq 0.688722$

Information Gain $= 0.988699 - 0.688722$

$= 0.299977$

$\simeq 0.30$

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.

Random Forest models decide where to split based on a random selection of features. splitting at similar features at each node throughout, Random Forest models implement a level of differentiation because each tree based on m variables will split based on unique features.

In a scenario where we select attributes from any set of data, our aim is to find the best. We would now grow the trees to maximum depth and then combine the trees using votes or averaging with a view to predicting for unseen data.

The problem is a single decision tree is overly sensitive to data variations because it could easily be overfit to noise in the data. The Random Forest with only one tree will overfit to data because it is the equivalent to a single decision tree. If an analyst increases the number of trees to the Random Forest, then the propensity to overfitting should decline. However, as we begin increasing the number of trees the model does not get more complex since we would eventually average all the trees and the Random forests cannot overfit the data to number of trees.

We may note that 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. Similarly, if m is too small, each individual tree is most likely not to be very predictive as we have restricted each note to a very small fraction of attributes which may not be predictive. Therefore, we may conclude that the best m values for random Forest m parameters would depend on peculiar problem, and so should be treated as tuning parameters.

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 `Car seats` dataset that is part of the ISLR package (you need to install and load the library). We may also need the following packages: `caret`, `dplyr` and `glmnet`

Let's start by loading these libraries:

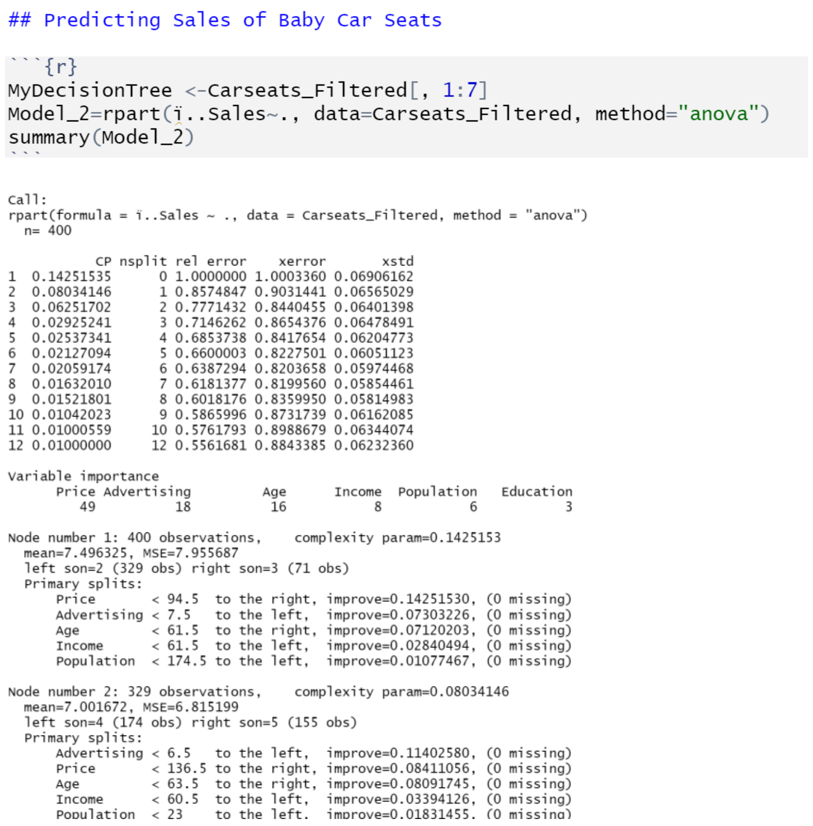
```
library(ISLR)

## Warning: package 'ISLR' was built under R version 4.0.3

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.0.2

## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading '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 4.0.2

## Loading required package: Matrix

## Loaded glmnet 4.0-2

library(caret)

## Warning: package 'caret' was built under R version 4.0.3

## Loading required package: lattice

## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 4.0.2
```

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_Filtered <- Carseats %>% select("Sales", "Price", "Advertising","Pop
ulation","Age","Income","Education")
```

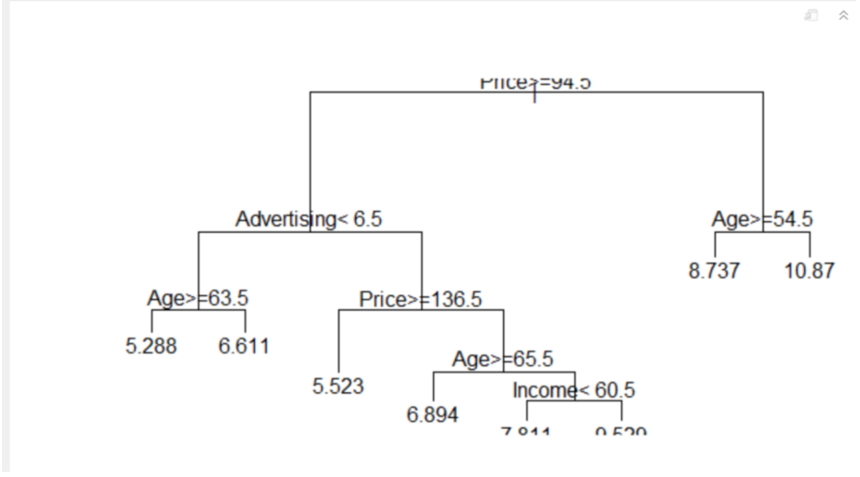
QB1. Build a decision tree regression model to predict Sales based on all other attributes ("Price", "Advertising", "Population", "Age", "Income" and "Education"). Which attribute is used at the top of the tree (the root node) for splitting? Hint: you can either plot () and text() functions or use the summary() function to see the decision tree rules.

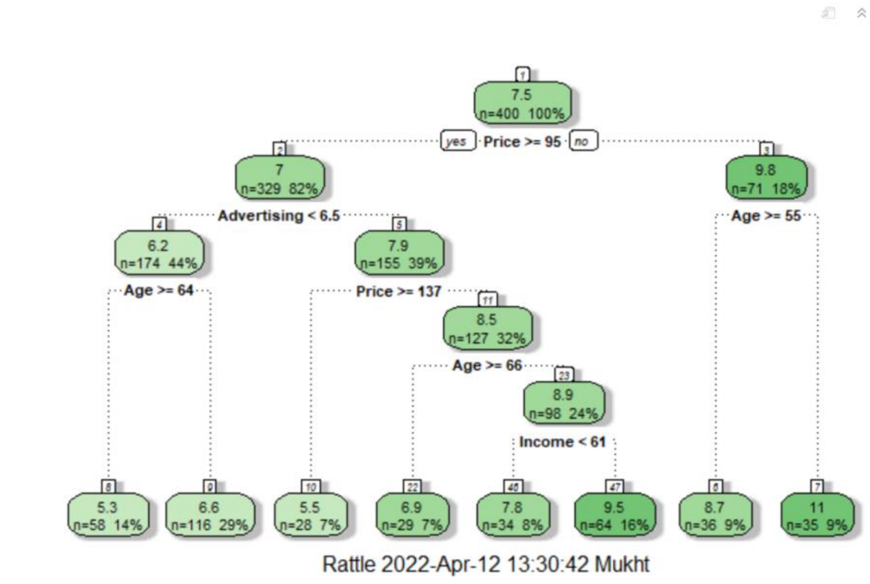


~~~
## Predicting Sales of Baby Car Seats

```{r}
MyDecisionTree <-Carseats_Filtered[, 1:7]
Model_2=rpart(ï..Sales~., data=Carseats_Filtered, method="anova")
summary(Model_2)
```
~~~

```
Call:
rpart(formula = ï..Sales ~ ., data = Carseats_Filtered, method = "anova")
  n= 400 

           CP nsplit rel error    xerror       xstd
1  0.14251535      0 1.0000000 1.0003360 0.06906162
2  0.08034146      1 0.8574847 0.9031441 0.06565029
3  0.06251702      2 0.7771432 0.8440455 0.06401398
4  0.02925241      3 0.7146262 0.8654376 0.06478491
5  0.02537341      4 0.6853738 0.8417654 0.06204773
6  0.02127094      5 0.6600003 0.8227501 0.06051123
7  0.02059174      6 0.6387294 0.8203658 0.05974468
8  0.01632010      7 0.6181377 0.8199560 0.05854461
9  0.01521801      8 0.6018176 0.8359950 0.05814983
10 0.01042023      9 0.5865996 0.8731739 0.06162085
11 0.01000559     10 0.5761793 0.8988679 0.06344074
12 0.01000000     12 0.5561681 0.8843385 0.06232360

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)
```

With plot() function

```
  {r}
##Less Complex
Model_2 = rpart (ï..Sales~.,data=Carseats_Filtered, method="anova", control =
rpart.control(minsplit = 60))
plot(Model_2)
text(Model_2)
```
```


Price>=94.5
Advertising< 6.5
Age>=54.5
8.737
10.87
Age>=63.5
Price>=136.5
5.288
6.611
5.523
Age>=65.5
Income< 60.5
6.894

With fancyRpartPlot()

```
  {r}
library(rattle)
library(rpart.plot)
fancyRpartPlot(Model_2)
```
```

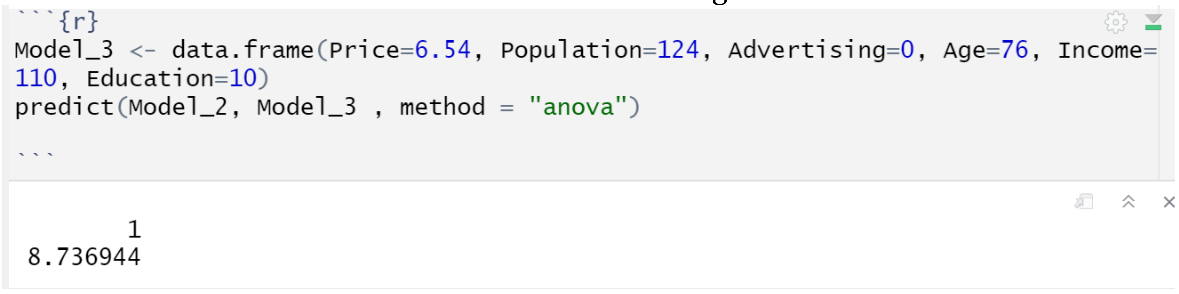


The Attribute used at the top of the tree is "Price" and it is of most variable importance (46) followed by Advertisement (18)

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?

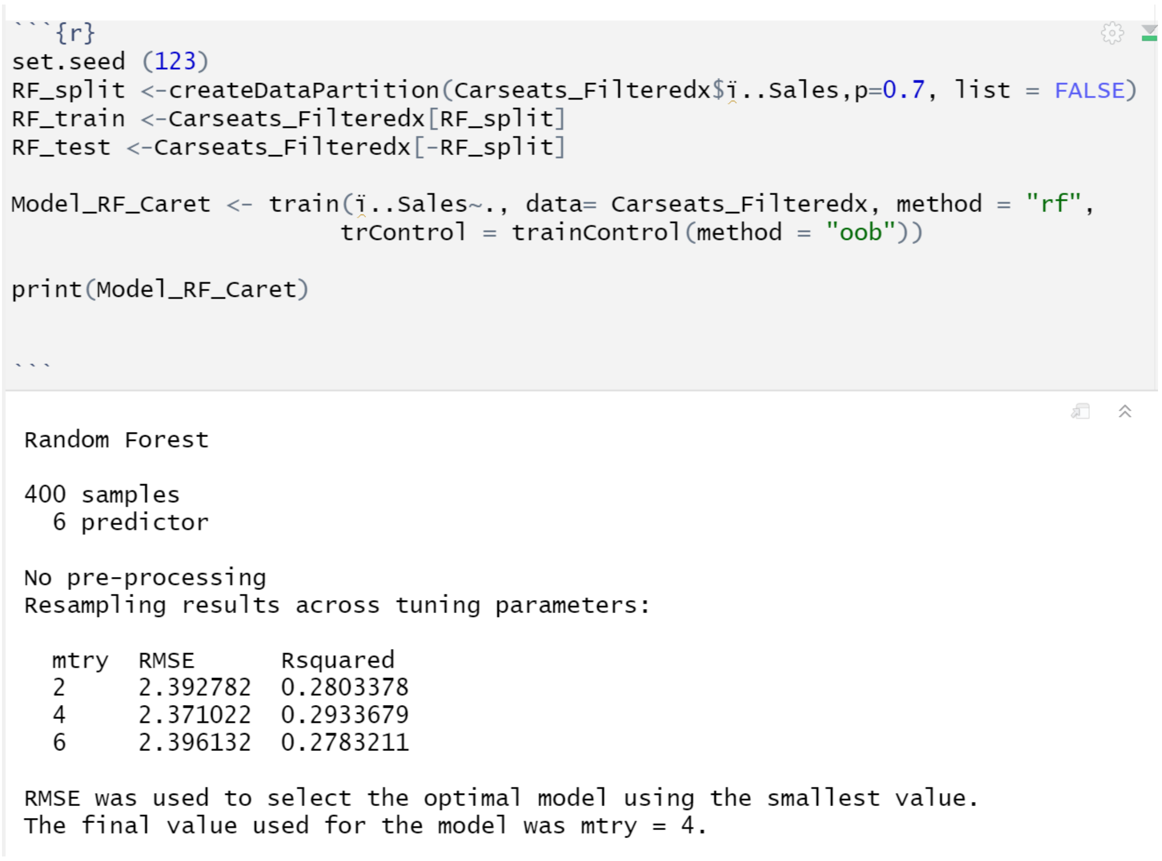


~~~
```{r}
Model_3 <- data.frame(Price=6.54, Population=124, Advertising=0, Age=76, Income=
110, Education=10)
predict(Model_2, Model_3 , method = "anova")

```
~~~

```
       1
8.736944
```

The estimated sales would be 8.736944 0r ≅ 9.0

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



~~~
```{r}
set.seed (123)
RF_split <-createDataPartition(Carseats_Filteredx$ï..Sales,p=0.7, list = FALSE)
RF_train <-Carseats_Filteredx[RF_split]
RF_test <-Carseats_Filteredx[-RF_split]

Model_RF_Caret <- train(ï..Sales~., data= Carseats_Filteredx, method = "rf",
                        trControl = trainControl(method = "oob"))

print(Model_RF_Caret)


```
~~~

```
Random Forest

400 samples
  6 predictor

No pre-processing
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared
  2     2.392782  0.2803378
  4     2.371022  0.2933679
  6     2.396132  0.2783211

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 4.
```

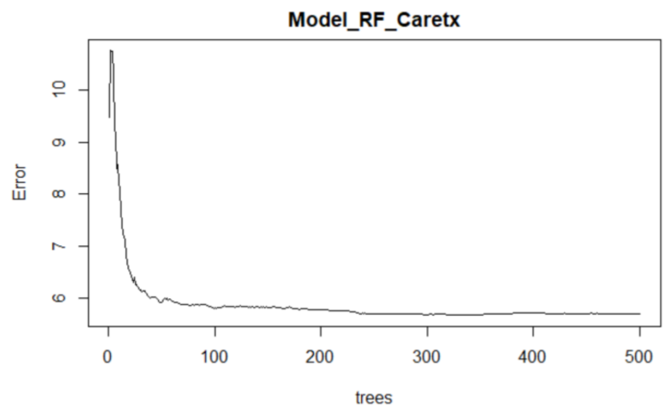
The mtry that gives the best performance is "4"

~~~
# for reproducibility
```{r}
set.seed(123)
# default RF model
Model_RF_Caretx <-randomForest(formula =ï..Sales~.,data = Carseats_Filteredx)
Model_RF_Caretx
```
~~~

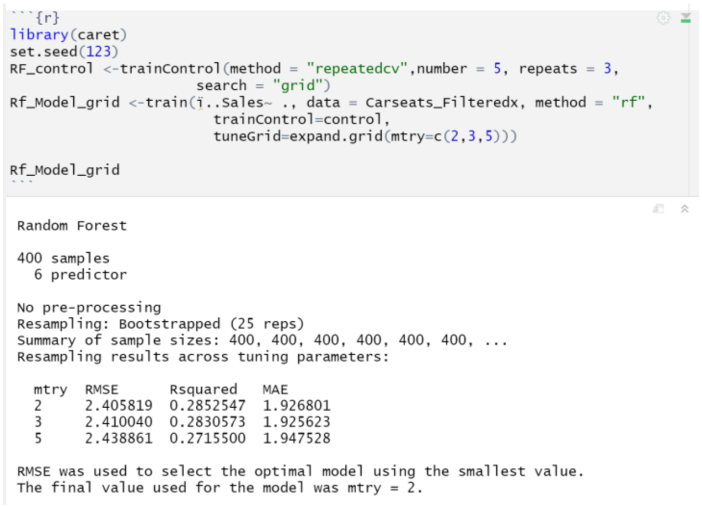
```

Call:
 randomForest(formula = ï..Sales ~ ., data = Carseats_Filteredx)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 5.698216
                    % Var explained: 28.38
```

The random forest model has 500 trees, which is the default setting, and 2 variables were tried at each split. This is our m parameter. The model seems to have an R squared value of 28.38%.

~~~
```{r}
plot(Model_RF_Caretx)
```
~~~



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.



~~~
```{r}
library(caret)
set.seed(123)
RF_control <-trainControl(method = "repeatedcv",number = 5, repeats = 3,
                         search = "grid")
Rf_Model_grid <-train(ï..Sales~ ., data = Carseats_Filteredx, method = "rf",
                       trainControl=control,
                       tuneGrid=expand.grid(mtry=c(2,3,5)))

Rf_Model_grid
```
~~~

```
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
  3     2.410040  0.2830573  1.925623
  5     2.438861  0.2715500  1.947528

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 2.
```

I used the search grid to explore the model performance. The final value used for the model that gives the best performance was mtry "2" with RMSE = 2.405819, R squared = 0.28525 47, and MAE = 1.926801

Reference

Kasturi, S. N. (2019). Difference between Bagging and Boosting? Retrieved from https://medium.com/swlh/difference-between-bagging-and-boosting-f996253acd22

Soumya. (2019). Bagging vs Boosting in Machine Learning. *GeeksforGeeks*. Retrieved from https://www.geeksforgeeks.org/bagging-vs-boosting-in-machine-learning/