Assignment 2

Venu Dodda

04/08/2023

Question 1: What is the key idea behind bagging? Can bagging deal both with high variance (overfitting) and high bias (under fitting)?

#The main goal of bagging, also known as bootstrap aggregating, is to increase a model's stability and accuracy by training several instances of the same model on various subsets of the training data, then combining their predictions.As part of the bagging procedure, many bootstrap samples—subsets of the training data selected at random with replacement—are created from the training data. After that, a different instance of the model is trained using each bootstrap sample. The final prediction is then created by combining all of the models' projections, often by taking the average or majority vote.  
#Yes, bagging can help to deal with both high variance (overfitting) and high bias (underfitting) in a model.  
#Bagging decreases the variance of the predictions by building numerous copies of the same model trained on various subsets of the training data. This prevents overfitting. A model can underperform when applied to fresh, untrained data due to overfitting, which happens when it is very sophisticated and fits the training data too closely. Bagging decreases the danger of overfitting by integrating the predictions of many models trained on various subsets of the data, resulting in more reliable and precise predictions.  
#Bagging can also aid in lowering the bias of the predictions because each model is trained on a slightly different subset of the data. A model might perform badly on both the training and test sets of data due to underfitting, which happens when it is overly simplistic and fails to recognize the underlying patterns in the data. Bagging enables the models to explore more of the feature space and capture more of the underlying patterns, which can assist to decrease underfitting. It does this by training numerous models on various subsets of the data.  
#Overall, bagging is an effective method that can assist to lower a model's high volatility and high bias, improving performance on fresh, untested data.

Question 2: Why bagging models are computationally more efficient when compared to boosting models with the same number of weak learners?

#Due to the distinctions in the training procedures between boosting and bagging models, a bagging model often has a higher computational efficiency when compared to a boosting model with the same number of weak learners.  
#Bagging is the process of training numerous instances of the same model separately using various subsets of the training data and replacement. The training process can be considerably sped up by being able to train every instance simultaneously. The instances may also be taught using various computational resources because they are independent, which can increase the process' efficiency even further.  
#While weak learners are educated progressively in boosting, each learner is instructed to correct the mistakes made by the learners that came before them. Since that each weak learner depends on the one before them, this sequential training method may take longer than bagging. Although each weak learner is often more sophisticated than the individual examples used in bagging, boosting can also be more computationally costly. Because to the independent and concurrent training of the instances in bagging, as opposed to the sequential training process and possibly more sophisticated learners in boosting, bagging models are computationally more efficient when compared to boosting models with the same number of weak learners.

Question 3: 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?

#With ensemble approaches like bagging and boosting, numerous weak learners are combined to produce a stronger, more precise model. The theory behind ensemble techniques is that by combining a number of weak models with different types of mistakes, the errors will balance out and the resultant model will be more accurate and less prone to overfitting. Nevertheless, integrating them is unlikely to result in a noticeable boost in performance if the individual models are highly similar to one another and are not outperforming a random model. Due to the ensemble model's increased complexity and the risk of overfitting, it can potentially result in worse performance.There might be a number of reasons why the individual decision tree models are not performing adequately. The models may be too complicated and overfit to the training set of data, for example. Overfitting is a problem with decision trees, especially when they are deep or the training set contains a lot of characteristics. If so, integrating the models using ensemble techniques could not be beneficial and might potentially make overfitting worse.Another possibility is that the models are missing key relationships or characteristics from the data. Decision trees can be biased and sometimes fail to recognize subtle or complicated links in the data. If this is the case, integrating the models might not solve the fundamental problems and might not considerably boost the performance. In general, it is crucial to make sure that the individual models are varied and commit distinct mistakes when considering ensemble approaches. This is so that ensemble techniques may combine the predictions of several independent and different models in order to lower the variance of the forecasts. Combining the models may not result in a noticeable performance gain if the separate models are too similar or consistently make the same mistakes.In conclusion, it seems doubtful that merging James' decision tree models will considerably improve performance if they are very close to one another and don't perform any better than a random model. To increase the performance of the model, he can like to experiment with various modeling approaches or look into additional characteristics and data sources. Instead, he can look for and fix the causes of the decision tree models' subpar performance before contemplating ensemble approaches.

Question 4: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?

#Entropy for our data set: I(all \_ data) = -[(9/16)log 2(9/16)+(7/16)log2(7/16)] = 0.98361  
#The entropy of small size = 0.811278 & The entropy of large size = 0.954434.  
#Using this formula, we can calculate the Information Gain to be 0.105843.  
#Information gain reveals the significance of a specific feature vector property. So, in this instance, the size attribute's information gain is. It's 0.10578144 crucial.

Question 5: 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.

#Each node will choose almost all characteristics if the "m" parameter is set to an extremely large value that is close to the total number of features ("p"), resulting in a lack of variety across different trees. As a result, the model will grow overly complicated and have a high variance, which might result in overfitting. On the other hand, if the "m" parameter is set too low, the amount of characteristics that can be represented by each node will be constrained, which may make it harder for the decision tree to capture significant correlations between the features. As a result, the model will be too simple and biased, which might cause underfitting. Consequently, in Random Forest models, it's crucial to adjust the "m" parameter to a suitable value. The ideal value for "m" depends on the particular issue at hand and, in actuality, should be seen as a tuning parameter. Generally speaking, a reasonable place to start is to set "m" to the logarithm base 2 of the number of features for regression issues and to the square root of the number of features for classification problems. To discover the ideal value that balances bias and variation, it is required to experiment with various values of "m." It's crucial to remember that Random Forest models are made to take use of the variety of individual decision trees, and that achieving this aim by setting "m" to an ideal number can assist.

#Loading the Required Packages   
library(ISLR)

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

library(dplyr)

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

##   
## 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

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

## Loaded glmnet 4.1-7

library(caret)

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

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.2.2

## Loading required package: lattice

#Using dplyr to select sales, price, advertising, population, age, income, and education.  
Carseats\_Filtered <- 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.

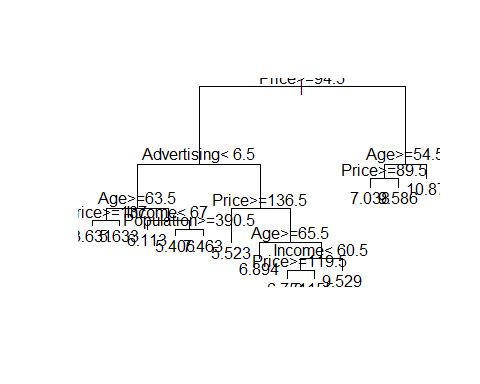
#Loading the Required Packages  
library(rpart)  
library(rpart.plot)

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

carsdata <- Carseats\_Filtered  
M1 = rpart(Sales~.,data=carsdata, method='anova')  
#Summary of the Model 1  
summary(M1)

## Call:  
## rpart(formula = Sales ~ ., data = carsdata, method = "anova")  
## n= 400   
##   
## CP nsplit rel error xerror xstd  
## 1 0.14251535 0 1.0000000 1.0053049 0.06947696  
## 2 0.08034146 1 0.8574847 0.9091155 0.06460930  
## 3 0.06251702 2 0.7771432 0.8789991 0.06440239  
## 4 0.02925241 3 0.7146262 0.8279740 0.05882521  
## 5 0.02537341 4 0.6853738 0.8131840 0.05618621  
## 6 0.02127094 5 0.6600003 0.8034948 0.05388906  
## 7 0.02059174 6 0.6387294 0.7884291 0.05359054  
## 8 0.01632010 7 0.6181377 0.7814260 0.05303058  
## 9 0.01521801 8 0.6018176 0.7946862 0.05341794  
## 10 0.01042023 9 0.5865996 0.7865215 0.05227201  
## 11 0.01000559 10 0.5761793 0.8007423 0.05353502  
## 12 0.01000000 12 0.5561681 0.7982033 0.05320929  
##   
## 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

#Plotting the M1   
plot(M1)  
text(M1)

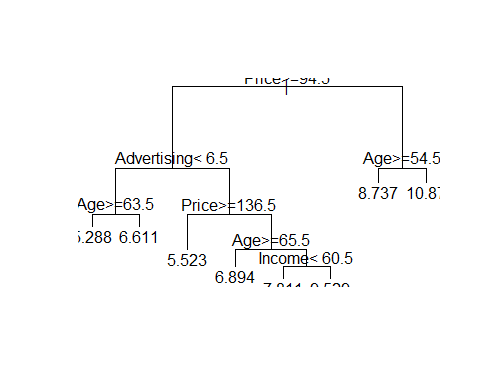
 #The attribute that is at the top of the tree is Price.

#Question #2: 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?

M\_2 = rpart(Sales~.,data=carsdata, method='anova', control = rpart.control(minsplit = 60 ))  
summary(M\_2)

## Call:  
## rpart(formula = Sales ~ ., data = carsdata, method = "anova",   
## control = rpart.control(minsplit = 60))  
## n= 400   
##   
## CP nsplit rel error xerror xstd  
## 1 0.14251535 0 1.0000000 1.0035604 0.06934091  
## 2 0.08034146 1 0.8574847 0.8960742 0.06435476  
## 3 0.06251702 2 0.7771432 0.8508622 0.06245334  
## 4 0.02925241 3 0.7146262 0.8241223 0.05997959  
## 5 0.02537341 4 0.6853738 0.8049088 0.05587229  
## 6 0.02127094 5 0.6600003 0.8090596 0.05744325  
## 7 0.02059174 6 0.6387294 0.7844632 0.05577398  
## 8 0.01000000 7 0.6181377 0.7572251 0.05375345  
##   
## Variable importance  
## Price Advertising Age Income Population Education   
## 49 20 18 7 4 2   
##   
## 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 < 174.5 to the left, improve=0.01484238, (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 < 77.5 to the right, improve=0.08080275, (0 missing)  
## Population < 268.5 to the left, improve=0.02383306, (0 missing)  
## Income < 57 to the left, improve=0.02353594, (0 missing)  
## Education < 12.5 to the right, improve=0.02237407, (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)  
## Income < 67.5 to the left, improve=0.027749670, (0 missing)  
## Population < 326 to the right, improve=0.020525710, (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 < 65.5 to the right, improve=0.07915291, (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  
## mean=8.736944, MSE=4.961043   
##   
## Node number 7: 35 observations  
## mean=10.87, MSE=6.491674   
##   
## Node number 8: 58 observations  
## mean=5.287586, MSE=3.93708   
##   
## Node number 9: 116 observations  
## mean=6.61069, MSE=4.861446   
##   
## 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 < 405 to the left, improve=0.01719030, (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 46: 34 observations  
## mean=7.811471, MSE=4.756548   
##   
## Node number 47: 64 observations  
## mean=9.529375, MSE=4.5078

plot(M\_2)  
text(M\_2)



New\_Model <- data.frame(Price=6.54, Population=124, Advertising=0, Age=76, Income= 110, Education=10)  
predict(M1, newdata=New\_Model)

## 1   
## 9.58625

#The estimated sales for this record using a decision tree model is 9.5862.

#Question 3: 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?

size = floor(0.70\*nrow(Carseats\_Filtered))  
size

## [1] 280

set.seed(123)  
train = sample(seq\_len(nrow(Carseats\_Filtered)), size = size)  
  
TrainData = Carseats\_Filtered[train,]  
TestData = Carseats\_Filtered[-train,]  
rf\_tree <- train(Sales~., data = Carseats\_Filtered, method = "rf")  
print(rf\_tree)

## 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 2.432085 0.2777412 1.957961  
## 4 2.449238 0.2728587 1.965729  
## 6 2.474999 0.2626543 1.985643  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 2.

#The mtry that gives the best performance is the 2nd mtry.

#Question 4: 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

#Using the mtry value of 2  
mtry = 2  
Train\_2 <- trainControl(method = "repeatedcv", number = 5, repeats = 3)  
tunegrid1 <- expand.grid(.mtry=mtry)  
tree\_2 <- train(Sales~.,  
 method = "rf",  
 data = TrainData,  
 trControl = Train\_2,  
 tuneGrid=tunegrid1  
 )  
print(tree\_2)

## Random Forest   
##   
## 280 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold, repeated 3 times)   
## Summary of sample sizes: 224, 224, 224, 224, 224, 224, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 2.433514 0.2623944 1.951287  
##   
## Tuning parameter 'mtry' was held constant at a value of 2

#Using mtry value of 3  
mtry = 3  
Train\_2 <- trainControl(method = "repeatedcv", number = 5, repeats = 3)  
tunegrid <- expand.grid(.mtry=mtry)  
tree\_2 <- train(Sales~.,  
 method = "rf",  
 data = TrainData,  
 trControl = Train\_2,  
 tuneGrid=tunegrid  
 )  
print(tree\_2)

## Random Forest   
##   
## 280 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold, repeated 3 times)   
## Summary of sample sizes: 224, 224, 224, 224, 224, 224, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 2.411611 0.2755602 1.927487  
##   
## Tuning parameter 'mtry' was held constant at a value of 3

#Using mtry value of 5  
mtry = 5  
Train\_2 <- trainControl(method = "repeatedcv", number = 5, repeats = 3)  
tunegrid <- expand.grid(.mtry=mtry)  
tree\_2 <- train(Sales~.,  
 method = "rf",  
 data = TrainData,  
 trControl = Train\_2,  
 tuneGrid=tunegrid  
 )  
print(tree\_2)

## Random Forest   
##   
## 280 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold, repeated 3 times)   
## Summary of sample sizes: 224, 224, 224, 224, 224, 224, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 2.431127 0.2711105 1.963302  
##   
## Tuning parameter 'mtry' was held constant at a value of 5