Assignment 2

Meghana Udiga

04/09/2023

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

#Bagging, often referred to as bootstrap aggregating, is the process of training many instances of the same model on different subsets of the training data, then pooling their predictions to boost a model's stability and accuracy.  
#Many bootstrap samples—subsets of the training data chosen at random with replacement—are produced from the training data as part of the bagging phase. Each bootstrap sample is then used to train a distinct instance of the model. The forecasts from all the models are then combined to provide the final prediction, which is frequently determined by taking the mean or majority vote.  
#Yes, bagging can help to deal with both high variance (overfitting) and high bias (underfitting) in a model.  
  
#By creating several copies of the same model that has been trained using different subsets of the training data, bagging reduces the variance of the predictions. Hence, overfitting is avoided. Overfitting, which occurs when a model is exceedingly complex and fits the training data too closely, might cause it to perform poorly when used with new, untrained data. By combining the predictions of several models trained on distinct subsets of the data, bagging reduces the risk of overfitting and produces predictions that are more accurate and dependable.  
#Because each model is trained on a slightly different subset of the data, bagging can also help to reduce the bias of the predictions. Underfitting, which occurs when a model is oversimplified and fails to understand the underlying patterns in the data, can cause it to perform poorly on both the training and test sets of data. The models can explore more of the feature space and identify more underlying patterns via bagging, which can help to reduce underfitting. It accomplishes this by training a large number of models on distinct data subsets.  
#Overall, bagging is a useful technique that can help to reduce a model's strong bias and high volatility, enhancing performance on new, untested data.

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

#A bagging model frequently has a greater computational efficiency than a boosting model with the same number of weak learners due to differences in the training processes between the two types of models.  
#Using different subsets of the training data and replacement, bagging is the process of training several instances of the same model individually. The ability to train every instance simultaneously can greatly speed up the training process. Due to their independence, the instances may also be taught using a variety of computational tools, which might further boost the process' efficiency.  
  
#Each student is given instructions to remedy the errors produced by the learners who came before them while weak learners are gradually schooled in boosting. This sequential training strategy may take longer than bagging since weak learners depend on others who came before them. Boosting can be more computationally expensive even if each weak learner is frequently more complex than the individual examples used in bagging.  
#Bagging models are computationally more efficient when compared to boosting models with the same number of weak learners since they train the instances independently and concurrently, as opposed to boosting models' sequential training procedure and potentially more complex 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?

#Several weak learners are integrated using ensemble procedures like bagging and boosting to create a more robust, accurate model. The idea behind ensemble approaches is that by integrating a number of imperfect models with various flaws, the faults will balance out and provide a more accurate model that is less prone to overfitting.  
  
#Nevertheless, if the separate models are very close to one another and are not outperforming a random model, integrating them is unlikely to produce an observable improvement in performance. It is possible that the ensemble model will perform poorly because of its higher level of complexity and propensity for overfitting.  
  
#There might be a variety of causes for the various decision tree models' subpar performance. For instance, the models could be very intricate and overfit to the training set of data. Decision trees have a tendency to overfit, especially when they are deep or the training set includes a large number of attributes. If so, utilizing ensemble approaches to integrate the models could not be advantageous and might perhaps make overfitting worse.  
  
#Another issue is that important correlations or qualities from the data are absent from the models. Decision trees have a tendency to be biased and can miss subtle or complex patterns in the data. If this is the case, merging the models might not be able to address the underlying issues or significantly improve performance.  
  
#When contemplating ensemble techniques, it is essential to make sure that the separate models are diverse and generate unique errors. This is done so that ensemble approaches, which aim to reduce forecast variance, may aggregate the results of numerous separate and distinct models. If the individual models are too similar or often make the same mistakes, combining the models could not produce a discernible performance boost.  
  
#Conclusion: Given that James' decision tree models are quite similar to one another and don't perform much better than a random model, it seems unlikely that combining them would significantly increase performance. He can experiment with other modeling techniques or look into more traits and data sources to improve the model's performance. Alternatively, he can investigate and address the reasons behind the poor performance of the decision tree models before considering ensemble methods.

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 highlights the importance of a certain feature vector attribute. Thus, the information gain of the size characteristic in this case is. It is 0.10578144 essential.

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.

#If the "m" parameter is set to an exceptionally large value that is near to the entire number of features ("p"), each node will select practically all characteristics, leading to a lack of diversity among various trees. The model will thereafter become too complex, have a large variance, and maybe overfit. Nevertheless, if the "m" value is set too low, the number of qualities that each node may represent will be restricted, which may make it more difficult for the decision tree to detect significant correlations between the features. As a result, the model will be biased and too simplistic, which might lead to underfitting.  
  
#As a result, it's critical to set the "m" parameter in Random Forest models to an appropriate value. The best value for "m" is dependent on the specific problem at hand and is best thought of as a tuning parameter. In general, a good place to start is to set "m" to the square root of the number of features for classification problems and to the logarithm base 2 of the number of features for regression problems. It is necessary to experiment with different values of "m" in order to find the optimum one that strikes a balance between bias and variety. It's important to keep in mind that Random Forest models are designed to utilize the diversity of unique decision trees, and that attaining this goal by setting "m" to an appropriate value can help.

#LOADING THE REQUIRED LIBRARIES INTO THE PROJECT.   
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.

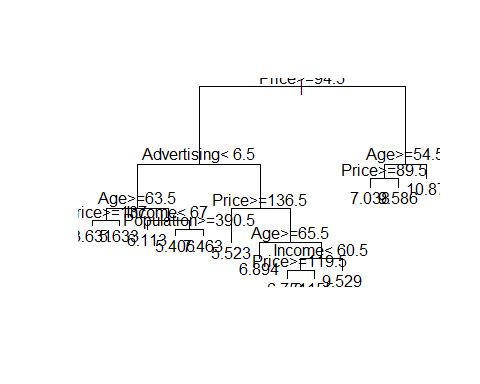
#LOADING THE REQUIRED LIBRARIES INTO THE PROJECT.   
library(rpart)  
library(rpart.plot)

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

carsdata <- Carseats\_Filtered  
MODEL1 = rpart(Sales~.,data=carsdata, method='anova')  
#SUMMARY OF THE MODEL 1  
summary(MODEL1)

## Call:  
## rpart(formula = Sales ~ ., data = carsdata, method = "anova")  
## n= 400   
##   
## CP nsplit rel error xerror xstd  
## 1 0.14251535 0 1.0000000 1.0029933 0.06914770  
## 2 0.08034146 1 0.8574847 0.9205641 0.06565627  
## 3 0.06251702 2 0.7771432 0.9088317 0.06475136  
## 4 0.02925241 3 0.7146262 0.8004412 0.05648838  
## 5 0.02537341 4 0.6853738 0.8156100 0.05838915  
## 6 0.02127094 5 0.6600003 0.8303489 0.05944487  
## 7 0.02059174 6 0.6387294 0.8170714 0.05898005  
## 8 0.01632010 7 0.6181377 0.7995355 0.05703027  
## 9 0.01521801 8 0.6018176 0.8298316 0.05756965  
## 10 0.01042023 9 0.5865996 0.8463973 0.05811728  
## 11 0.01000559 10 0.5761793 0.8783895 0.05913567  
## 12 0.01000000 12 0.5561681 0.8694888 0.05889828  
##   
## 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 MODEL1  
plot(MODEL1)  
text(MODEL1)

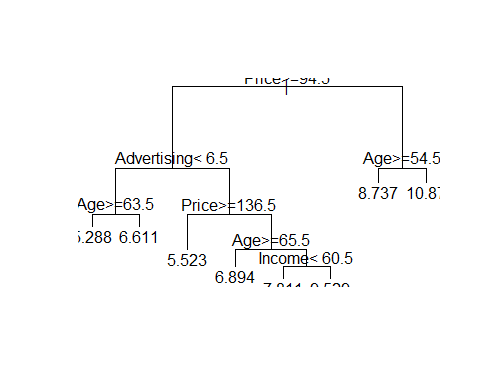
 #The attribute that is at the top of the tree is Price.

#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?

MODEL2 = rpart(Sales~.,data=carsdata, method='anova', control = rpart.control(minsplit = 60 ))  
#SUMMARY OF THE MODEL 2  
summary(MODEL2)

## 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 1.0062816 0.06965681  
## 2 0.08034146 1 0.8574847 0.9444504 0.06843644  
## 3 0.06251702 2 0.7771432 0.8942798 0.06479818  
## 4 0.02925241 3 0.7146262 0.8265741 0.06057394  
## 5 0.02537341 4 0.6853738 0.8587857 0.05883287  
## 6 0.02127094 5 0.6600003 0.8533863 0.05850406  
## 7 0.02059174 6 0.6387294 0.8442623 0.05822882  
## 8 0.01000000 7 0.6181377 0.8142170 0.05533802  
##   
## 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

#PLOTTING THE MODEL2  
plot(MODEL2)  
text(MODEL2)



N\_MODEL <- data.frame(Price=6.54, Population=124, Advertising=0, Age=76, Income= 110, Education=10)  
predict(MODEL1, newdata=N\_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)  
  
Train\_Data = Carseats\_Filtered[train,]  
Test\_Data = 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)  
TREE2 <- train(Sales~.,  
 method = "rf",  
 data = Train\_Data,  
 trControl = Train\_2,  
 tuneGrid=tunegrid1  
 )  
print(TREE2)

## 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 THE MTRY VALUE OF 3.  
mtry = 3  
Train\_2 <- trainControl(method = "repeatedcv", number = 5, repeats = 3)  
tunegrid <- expand.grid(.mtry=mtry)  
TREE2 <- train(Sales~.,  
 method = "rf",  
 data = Train\_Data,  
 trControl = Train\_2,  
 tuneGrid=tunegrid  
 )  
print(TREE2)

## 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 THE MTRY VALUE OF 5.  
mtry = 5  
Train\_2 <- trainControl(method = "repeatedcv", number = 5, repeats = 3)  
tunegrid <- expand.grid(.mtry=mtry)  
TREE2 <- train(Sales~.,  
 method = "rf",  
 data = Train\_Data,  
 trControl = Train\_2,  
 tuneGrid=tunegrid  
 )  
print(TREE2)

## 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