Assignment 2

Part A

Q1

Ans: -

Bagging, or Bootstrap Aggregating, is a technique used to enhance the performance of machine learning models, especially decision trees, by reducing variance and improving predictive accuracy. This approach effectively addresses both high variance (overfitting) and, to some extent, high bias (underfitting) issues. Its primary goal is to combine multiple weak learners, each of which may individually exhibit high variance, to produce a model with reduced overall variance. These weak learners are typically of the same type and the process involves two key steps:

Bootstrapping: Bagging starts by creating multiple subsets or "bags" of the original dataset through a process called bootstrap sampling. This introduces randomness into the training process as each bag is formed by randomly selecting data points from the original dataset, often with replacement.

Aggregation: Once these bags are formed, individual base learners are trained on each bag. Predictions are then made for each data point in the validation set using these base learners. The final prediction is determined through aggregation, typically employing majority voting for classification problems, and averaging for regression problems.

In summary, bagging is a powerful ensemble learning technique that effectively addresses high variance and, to some extent, high bias by combining homogeneous weak learners. It significantly reduces overfitting and enhances predictive accuracy by introducing randomness and diversity into the training process.

Q2

Ans: -

Bagging models are computationally more efficient compared to boosting models when employing the same number of weak learners. The key factor contributing to this efficiency lies in the training process. In bagging, weak learners can be trained independently and in parallel, which optimizes resource utilization and speeds up the training process. On the other hand, boosting models rely on a sequential approach, where each weak learner's performance depends on the previous one. This sequential nature introduces computational overhead and can lead to slower model training. In summary, the parallel and independent training in bagging makes it a computationally more efficient choice, especially when compared to boosting models with an equivalent number of weak learners.

Q3

Ans: -

Ensemble model of boosting is helpful in the scenario. But it is important to understand the similarity between individual models and how it can affect the effectiveness of the ensemble. When selecting base models, if we opt for models with minimal bias (underfitting) but heightened variance (overfitting), our choice of aggregation method should prioritize variance reduction. Conversely, when working with base models characterized by reduced variance but amplified bias, our focus should be on employing an aggregation method that mitigates bias. If the similarity is primarily due to bias or a lack of complexity in the individual models, creating an ensemble may not yield significant improvements. In such cases, the ensemble might inherit the limitations of the base models.

On the other hand, if the similarity is because the individual models correctly capture the same underlying patterns, then an ensemble can still be effective in improving performance. By combining the errors and biases of individual trees, they often result in a more robust and accurate model.

However, it's essential to consider the nature of the data and the diversity of the models when creating the ensemble. The success of the ensemble depends on these factors. Additionally, the choice of ensemble method, such as Random Forest or AdaBoost, can influence the performance. A well-considered ensemble can potentially leverage the collective strength of the decision tree models and enhance their predictive capabilities.

Q4

Ans: -

Information gain = (Entropy of the parent node) – (average entropy of the child nodes)

Parent entropy = - [(7/16) * log2(7/16) + (9/16) * log2(9/16)] ≈ 0.9852

Small size entropy = - [(6/8) * log2(6/8) + (2/8) * log2(2/8)] = 0.8112.

Large size entropy = - [(3/8) * log2(3/8) + (5/8) * log2(5/8)] = 0.9533.

Average Entropy (Size) = (8/16) * 0.8112 + (8/16) * 0.9533 = 0.8822

As a result, the information gain = 0.9852 – 0.8852 = 0.1029

Q5

Ans: -

The "m" parameter in Random Forest models, which controls the number of attributes available at each split, is critical for model performance. Setting "m" too small may lead to underfitting and reduced diversity, while setting it too large can result in overfitting and a lack of diversity. The optimal "m" lies between these extremes, striking a balance to reduce overfitting while maintaining diversity. Cross-validation and setting "m" to the square root of the total features often work well in practice. Properly tuning "m" is essential for a well-generalizing ensemble.

Ref: -

https://www.analyticsvidhya.com/blog/2023/01/ensemble-learning-methods-bagging-boosting-and-stacking/

https://en.wikipedia.org/wiki/Bootstrap_aggregating

https://www.upgrad.com/blog/bagging-vs-boosting/

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

https://bradleyboehmke.github.io/HOML/random-forest.html

Part B

ktavva_2

Krishna Kumar Tavva

2023-11-12

```
library(ISLR)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
##
## 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 4.2.3
```

```
## Loaded glmnet 4.1-7
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
## Loading required package: lattice
```

```
library(rpart)
```

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

```
library(rpart.plot)
```

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

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

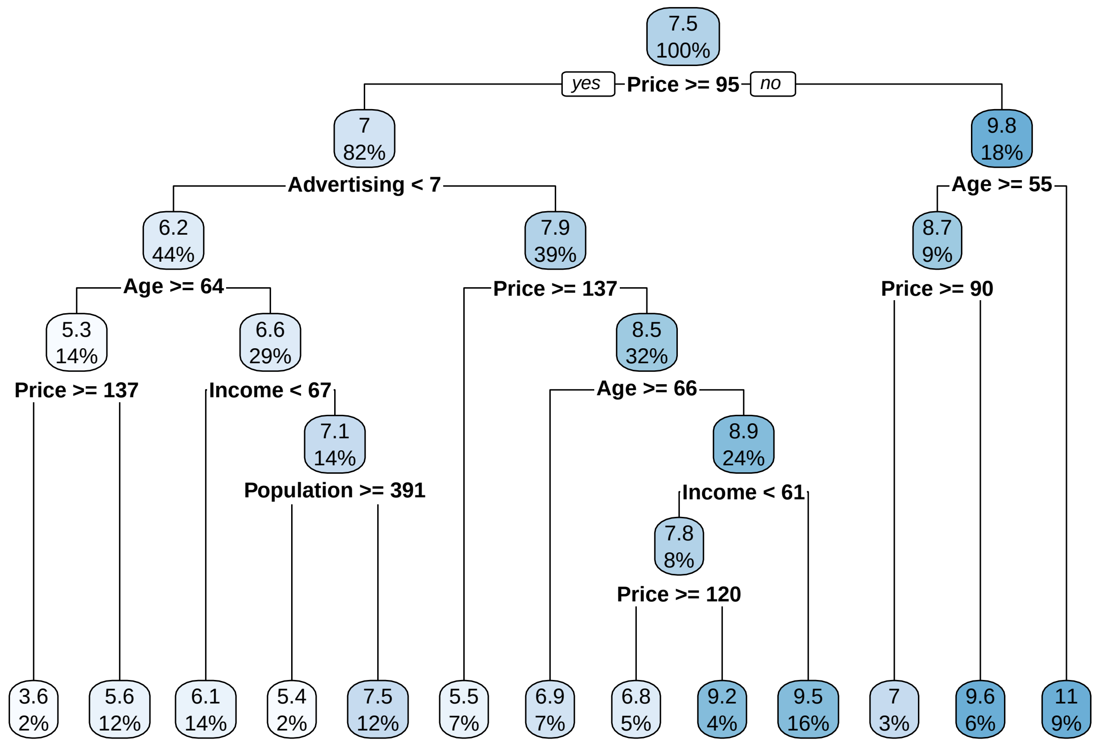
Q1

```
M1 <- rpart(Sales~., data=Carseats_Filtered)

summary(M1)
```

```
## Call:
## rpart(formula = Sales ~ ., data = Carseats_Filtered)
##   n= 400
##
##            CP nsplit rel error    xerror       xstd
## 1  0.14251535      0 1.0000000 1.0060857 0.06948663
## 2  0.08034146      1 0.8574847 0.9189063 0.06641354
## 3  0.06251702      2 0.7771432 0.8636415 0.06369040
## 4  0.02925241      3 0.7146262 0.8208908 0.05932561
## 5  0.02537341      4 0.6853738 0.8396510 0.06058309
## 6  0.02127094      5 0.6600003 0.8243796 0.05888116
## 7  0.02059174      6 0.6387294 0.7959830 0.05774471
## 8  0.01632010      7 0.6181377 0.7879925 0.05576583
## 9  0.01521801      8 0.6018176 0.7882243 0.05508520
## 10 0.01042023      9 0.5865996 0.8065791 0.05491741
## 11 0.01000559     10 0.5761793 0.8341392 0.05634047
## 12 0.01000000     12 0.5561681 0.8350256 0.05677185
##
## 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
```

```
#plot decision tree

rpart.plot(M1)
```



Q2

```
input <- data.frame(Sales=9, Price=6.54, Population=124, Advertising=0, Age=76,
                    Income= 110, Education=10)

M2 <- predict(M1, input)
```

#Estimated saled for this record is 9.586

Q3

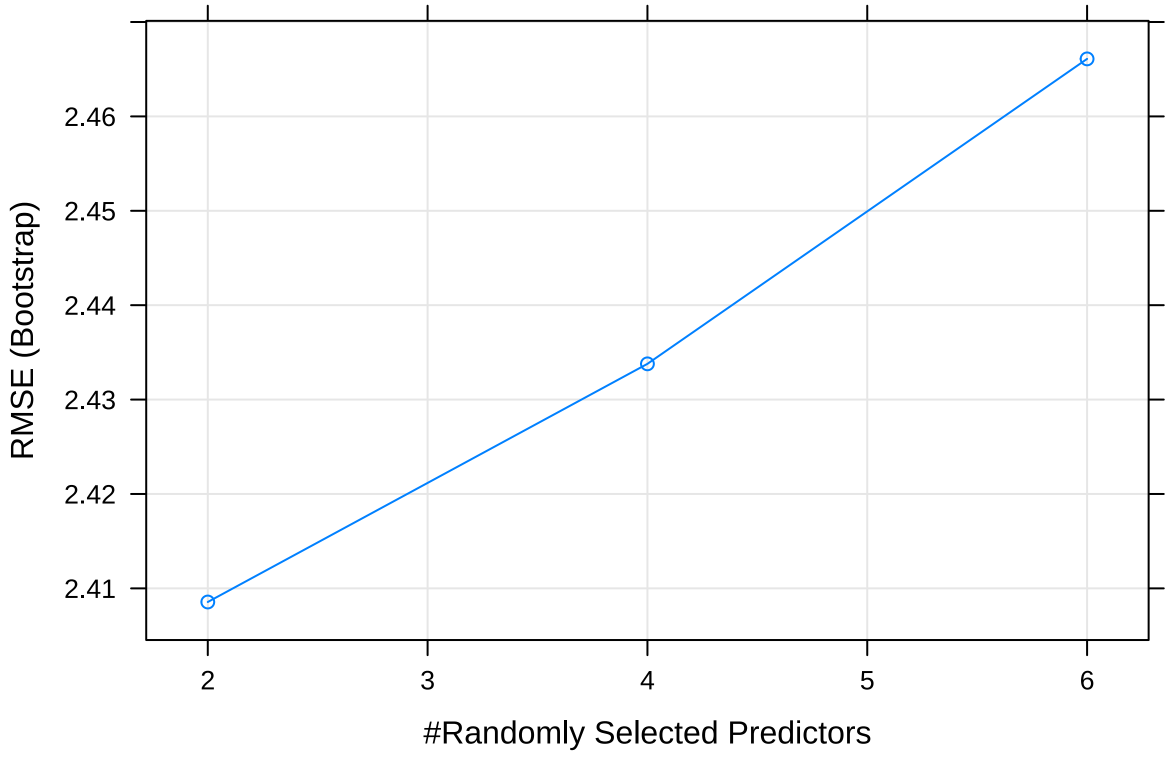
```
set.seed(1)

M3 <- train(Sales~., data=Carseats_Filtered, method= 'rf')

summary(M3)
```

```
##                    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(M3)
```

```
## 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     2.408556  0.2739281  1.920032
##   4     2.433785  0.2677241  1.935631
##   6     2.466094  0.2564218  1.964541
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
plot(M3)
```



#since 2 mtry is the lowest RMSE, this value yields the best performance.

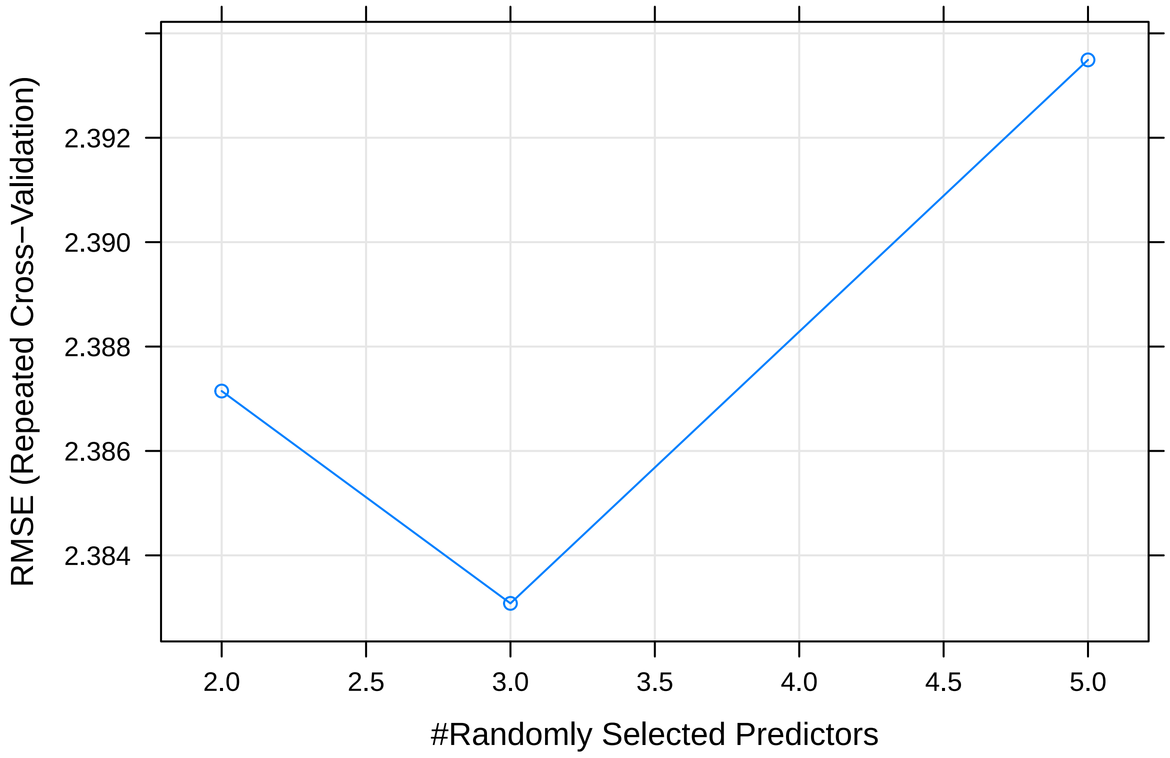
Q4

```
train_control <- trainControl(method="repeatedcv", number=5, repeats=3, search="grid")
tune_grid <- expand.grid(.mtry=c(2,3,5))
M4 <- train(Sales~., data=Carseats_Filtered, method='rf', tuneGrid=tune_grid, trControl=train_control)
print(M4)
```

```
## Random Forest
##
## 400 samples
##   6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 319, 320, 321, 320, 320, 321, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     2.387148  0.2904053  1.903958
##   3     2.383080  0.2944587  1.897428
##   5     2.393492  0.2908850  1.903436
```

```
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 3.
```

```
plot(M4)
```



#for mtry values of 2, 3 and 5 using 3 repeats of 5-fold cross validation, the final value used for the model was mtry = 2