ADM Assignment_2

ALLEN RICHARDS

2024-04-15

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

Combining multiple models that have been trained on different subsets of the training data is the main objective of bagging, also referred to as bootstrap aggregating, which aims to lower variance and improve machine learning models' stability and accuracy. For bagging to work, multiple bootstrap samples—random samples with replacement—are generated from the original training dataset. After that, a base model—which can be any model, though decision trees are often used—is trained using each bootstrap sample. In the end, predictions are either determined by a clear majority vote (in the case of classification) or by averaging the projected results of every initial model (in the case of regression).

High Variance-Overfitting: Since bagging is particularly effective at reducing variance, it mitigates overfitting. By enabling multiple models to be trained on different subsets of the data, bagging lessens the chance of overfitting and produces models that more closely resemble unseen data.

High Bias-Underfitting: While minimizing variance is bagging's primary objective, it may also partially mitigate high bias. By including more base models in the overall model and increasing its complexity, bagging can capture more intricate connections within the data itself. Still, if the base models are extremely biased and simplistic, bagging might not be enough to completely solve the underfitting issue.

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

In comparison to boosting with an equivalent number of weak learners, bagging's computational effectiveness is primarily due to its parallel nature. When simultaneous processing is possible, available computing resources are better utilized because each model is trained independently through bagging. Bagging also has the advantage of being computationally efficient due to its low data requirements. Boosting has a tendency to be less computationally efficient due to its sequential design, which uses the output of the previous model for each model's training and requires each subsequent attempt to fix any errors made by the previous one. The role of subsequent models is contingent upon that of the preceding model.

#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.

It might not be the best course of action in this situation to combine the similar decision tree models to create an ensemble model. A simple method of averaging or voting on the predictions of underperforming models may not yield much improvement if the models are very similar to one another and do not perform well individually. This is because ensemble methods are more successful at reducing variance, and the models are probably suffering from high bias (underfitting) rather than high variance (overfitting). The first step should be to try enhancing each individual model rather than assembling an ensemble. Some examples of this include switching to a base model type with less inherent bias, experimenting with hyperparameters, or thinking about boosting algorithms. Once the individual models are performing better, then an ensemble approach like bagging could be explored to potentially further improve the overall prediction accuracy.

#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? (15% of total points)

Information Gain = Entropy (Parent) - Weighted Average Entropy (Children) As we can see from the question, there are a total of 16 objects—9 of which are edible and 7 of which are not. The proportion of edible items to non-edible is 7/16 where as edible is 7/16 Parent Entropy = -(9/16) log2(9/16) -(7/16) log2(7/16) = 0.9887 After that, we can calculate the weighted average of the entropy for each of the two possible values for the “size” attribute: “small” and “large.” Entropy_small = -(6/8) * log2(6/8) -(2/8) * log2(2/8) = 0.8112781 Entropy_large = -(5/8) log2(5/8) -(3/8) log2(3/8) =0.954434 The substantial amount of information gain by splitting the dataset according to the “Size”Attribute. Information Gain = 0.9887 - (0.5 * 8112781 + 0.5 * 9.54434) = 0.105844

#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.

The m parameter, which represents the number of attributes randomly selected at each split in a Random Forest model, is an important hyperparameter that needs to be set optimally. If m is too small, the individual trees will have access to a limited set of features, resulting in weak trees and high bias in the ensemble. Conversely, if m is too large, the trees will be too similar, leading to high variance and overfitting. Finding the right balance for m is crucial for achieving good predictive performance in the Random Forest model, and is typically done through cross-validation and grid search. if “m” is set too high, the diversity of trees in the forest may increase, leading to underfitting. It is more difficult to identify which characteristics are most informative for each split because each tree is constructed from broader set of characteristics. If the model cannot identify the fundamental trends in the data, it may perform poorly on both the training set and new, untried data in this scenario.

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

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.3
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library(caret)
```

```
## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 4.3.2
```

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

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

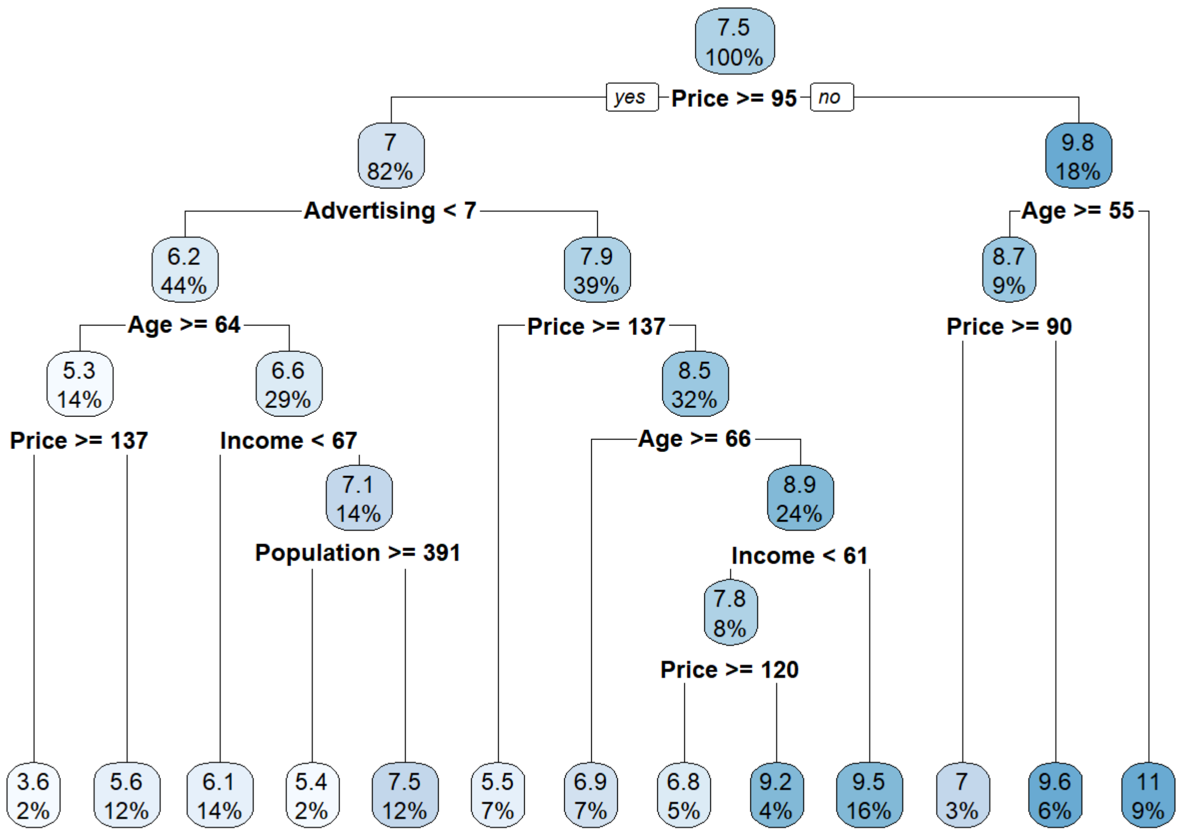
```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

```
Carseats_Filtered <- Carseats %>% select("Sales", "Price","Advertising","Population","Age","I
ncome","Education")
decision_tree_model<-rpart(Sales~.,data=Carseats_Filtered, method = 'anova')
summary(decision_tree_model)
```

```
## Call:
## rpart(formula = Sales ~ ., data = Carseats_Filtered, method = "anova")
##   n= 400
##
##            CP nsplit rel error    xerror       xstd
## 1  0.14251535      0 1.0000000 1.0051029 0.06942805
## 2  0.08034146      1 0.8574847 0.9012579 0.06444698
## 3  0.06251702      2 0.7771432 0.8412797 0.06262675
## 4  0.02925241      3 0.7146262 0.8069034 0.05894067
## 5  0.02537341      4 0.6853738 0.8014861 0.05695190
## 6  0.02127094      5 0.6600003 0.8106357 0.05852292
## 7  0.02059174      6 0.6387294 0.8090400 0.05907534
## 8  0.01632010      7 0.6181377 0.8107330 0.05895909
## 9  0.01521801      8 0.6018176 0.8213960 0.05884552
## 10 0.01042023      9 0.5865996 0.8290684 0.06006130
## 11 0.01000559     10 0.5761793 0.8586647 0.06078185
## 12 0.01000000     12 0.5561681 0.8558734 0.06082485
##
## 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
```

```
rpart.plot(decision_tree_model)
```



```
predicting_Q2=data.frame(Price=6.54 ,Population=124,Advertising=0,Age=76,Income= 110, Educati
on= 10)
sales_estimated_Q2 <-predict(decision_tree_model,predicting_Q2, method= "anova")
sales_estimated_Q2
```

```
##       1
## 9.58625
```

```
set.seed(123)
```

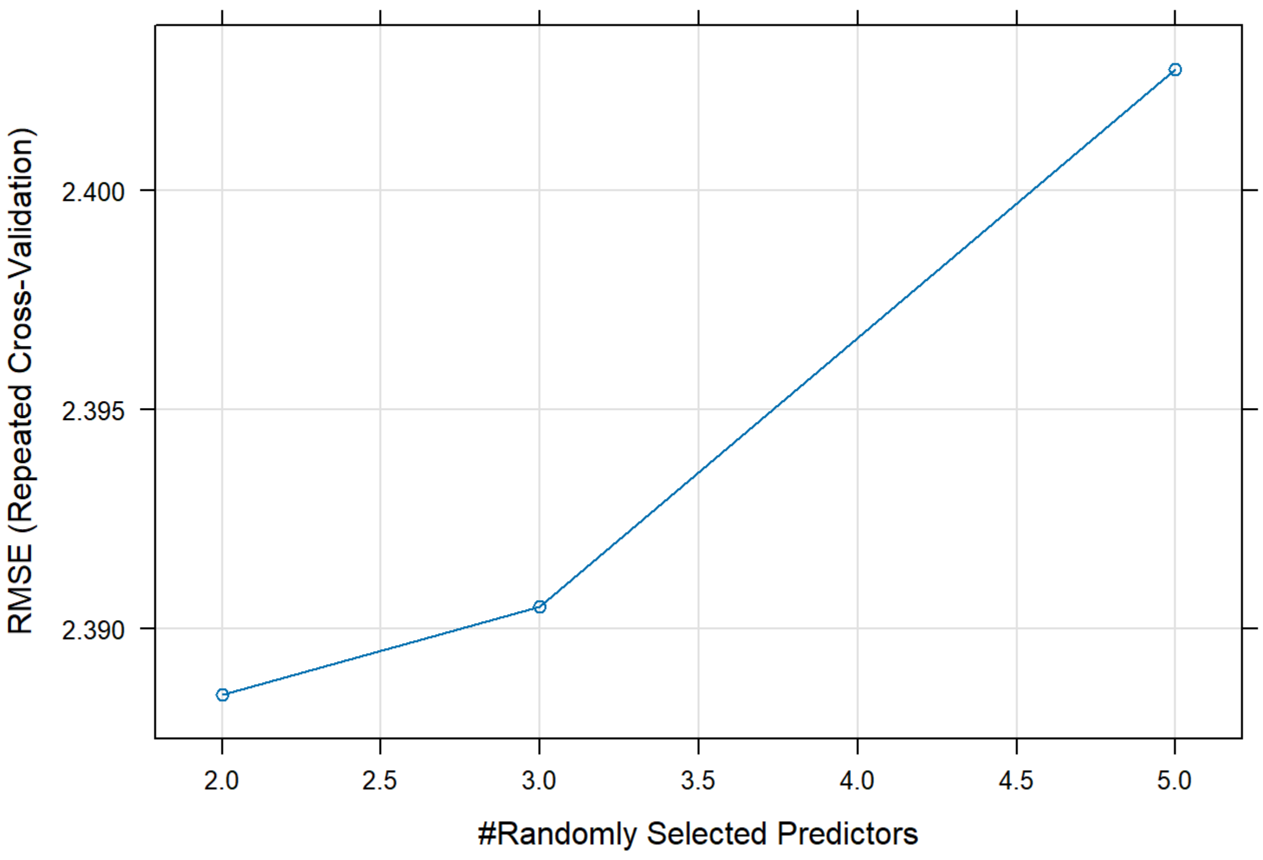
```
random_forest_model <- train(Sales~., data= Carseats_Filtered ,method = "rf")
print(random_forest_model)
```

```
## Random Forest
##
## 400 samples
##   6 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 400, 400, 400, 400, 400, 400, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     2.405819  0.2852547  1.926801
##   4     2.421577  0.2790266  1.934608
##   6     2.447373  0.2681323  1.953147
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
customize <- trainControl(method="repeatedcv", number=5, repeats=3)
new_grid_values <- expand.grid(mtry=c(2,3,5))
new_model<- train(Sales~., data=Carseats_Filtered, method="rf",tuneGrid=new_grid_values, trCo
ntrol=customize)
print(new_model)
```

```
## Random Forest
##
## 400 samples
##   6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 321, 320, 320, 320, 319, 320, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     2.388490  0.2902905  1.902942
##   3     2.390502  0.2898689  1.899672
##   5     2.402758  0.2869045  1.905036
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
plot(new_model)
```


RMSE (Repeated Cross-Validation)
2.400
2.395
2.390
2.0
2.5
3.0
3.5
4.0
4.5
5.0
#Randomly Selected Predictors