Individual Project

Pratyush Rohilla

2024-08-04

```r
library(rpart)
library(rpart.plot)
library(MASS)
library(caret)
```

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

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

```r
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(gbm)
```

```
## Loaded gbm 2.2.2
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/
gbm-developers/gbm3
```

```r
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ lubridate 1.9.3     ✓ tibble    3.2.1
## ✓ purrr     1.0.2     ✓ tidyr     1.3.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::combine()       masks randomForest::combine()
## ✕ dplyr::filter()        masks stats::filter()
## ✕ dplyr::lag()           masks stats::lag()
## ✕ purrr::lift()          masks caret::lift()
## ✕ randomForest::margin() masks ggplot2::margin()
## ✕ dplyr::select()        masks MASS::select()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(dplyr)
library(BART)
```

```
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
##
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
```

```r
library(tree)
library(cluster)
```

reading of austin dataset and making another column as loglatestprice from latestprice, also dropping streetadress and description

```r
austin_data <- read.csv("austinhouses.csv")

# Include all predictors except 'streetAddress' and 'description'
austin_data <- austin_data %>%
  select(-streetAddress, -description)

# Add a new column 'logLatestPrice' based on 'latestPrice'
austin_data <- austin_data %>%
  mutate(logLatestPrice = log(latestPrice))

# Display the first few rows of the modified dataset
head(austin_data)
```

```
##   zipcode latitude longitude garageSpaces hasAssociation hasGarage hasSpa
## 1   78717 30.49564 -97.79787            0           TRUE     FALSE  FALSE
## 2   78717 30.48878 -97.79490            2           TRUE      TRUE  FALSE
## 3   78725 30.23315 -97.58732            2          FALSE      TRUE  FALSE
## 4   78725 30.23824 -97.57833            2           TRUE      TRUE  FALSE
## 5   78726 30.42646 -97.85929            2           TRUE      TRUE  FALSE
## 6   78726 30.42596 -97.85841            0           TRUE     FALSE  FALSE
##   hasView      homeType yearBuilt latestPrice latest_saledate latest_salemonth
## 1   FALSE Single Family      2008       400.0      2020-01-10                1
## 2   FALSE Single Family      2013       549.9      2018-03-13                3
## 3   FALSE Single Family      1999       240.0      2020-12-31               12
## 4   FALSE Single Family      2012       200.0      2018-01-30                1
## 5    TRUE Single Family      2004       875.0      2020-11-09               11
## 6   FALSE Single Family      2005       830.0      2019-09-17                9
##   latest_saleyear numOfPhotos numOfAccessibilityFeatures numOfAppliances
## 1            2020          20                          0               3
## 2            2018          69                          0               4
## 3            2020          10                          0               4
## 4            2018          33                          0               5
## 5            2020          38                          0               8
## 6            2019          37                          0               4
##   numOfParkingFeatures numOfPatioAndPorchFeatures numOfSecurityFeatures
## 1                    2                          0                     0
## 2                    3                          0                     0
## 3                    2                          2                     0
## 4                    2                          0                     0
## 5                    2                          4                     1
## 6                    1                          1                     3
##   numOfWaterfrontFeatures numOfWindowFeatures numOfCommunityFeatures
## 1                       0                   0                      0
## 2                       0                   0                      0
## 3                       0                   0                      0
## 4                       0                   0                      0
## 5                       0                   0                      0
## 6                       0                   1                      0
##   lotSizeSqFt livingAreaSqFt avgSchoolDistance avgSchoolRating avgSchoolSize
## 1      7666.0           2228          1.900000        8.333333          1481
## 2      8494.0           3494          3.300000        7.666667          1259
## 3      5183.0           1534          1.800000        3.000000          1457
## 4      8145.0           1652          1.966667        3.000000          1457
## 5     30056.4           3402          2.066667        7.000000          1277
## 6     19166.4           3573          2.000000        7.000000          1277
##   MedianStudentsPerTeacher numOfBathrooms numOfBedrooms numOfStories
## 1                       16              2             3            1
## 2                       14              5             4            2
## 3                       13              3             3            1
## 4                       13              2             3            1
## 5                       16              4             4            2
## 6                       16              5             4            2
##   logLatestPrice
## 1       5.991465
## 2       6.309736
## 3       5.480639
## 4       5.298317
## 5       6.774224
## 6       6.721426
```

```
print(colnames(austin_data))
```

```
##  [1] "zipcode"                    "latitude"
##  [3] "longitude"                  "garageSpaces"
##  [5] "hasAssociation"             "hasGarage"
##  [7] "hasSpa"                     "hasView"
##  [9] "homeType"                   "yearBuilt"
## [11] "latestPrice"                "latest_saledate"
## [13] "latest_salemonth"           "latest_saleyear"
## [15] "numOfPhotos"                "numOfAccessibilityFeatures"
## [17] "numOfAppliances"            "numOfParkingFeatures"
## [19] "numOfPatioAndPorchFeatures" "numOfSecurityFeatures"
## [21] "numOfWaterfrontFeatures"    "numOfWindowFeatures"
## [23] "numOfCommunityFeatures"     "lotSizeSqFt"
## [25] "livingAreaSqFt"             "avgSchoolDistance"
## [27] "avgSchoolRating"            "avgSchoolSize"
## [29] "MedianStudentsPerTeacher"   "numOfBathrooms"
## [31] "numOfBedrooms"              "numOfStories"
## [33] "logLatestPrice"
```

printing out categorical variables

```
str(austin_data)
```

```
## 'data.frame':    6784 obs. of  33 variables:
##  $ zipcode                   : int  78717 78717 78725 78725 78726 78726 78725 78725 78744 78726 ...
##  $ latitude                  : num  30.5 30.5 30.2 30.2 30.4 ...
##  $ longitude                 : num  -97.8 -97.8 -97.6 -97.6 -97.9 ...
##  $ garageSpaces              : int  0 2 2 2 2 0 0 0 2 2 ...
##  $ hasAssociation            : logi  TRUE TRUE FALSE TRUE TRUE TRUE ...
##  $ hasGarage                 : logi  FALSE TRUE TRUE TRUE TRUE FALSE ...
##  $ hasSpa                    : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ hasView                   : logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
##  $ homeType                  : chr  "Single Family" "Single Family" "Single Family" "Single Family" ...
##  $ yearBuilt                 : int  2008 2013 1999 2012 2004 2005 2000 2009 2016 1988 ...
##  $ latestPrice               : num  400 550 240 200 875 ...
##  $ latest_saledate           : chr  "2020-01-10" "2018-03-13" "2020-12-31" "2018-01-30" ...
##  $ latest_salemonth          : int  1 3 12 1 11 9 6 3 3 3 ...
##  $ latest_saleyear           : int  2020 2018 2020 2018 2020 2019 2019 2019 2018 2020 ...
##  $ numOfPhotos               : int  20 69 10 33 38 37 24 26 31 8 ...
##  $ numOfAccessibilityFeatures: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ numOfAppliances           : int  3 4 4 5 8 4 4 4 7 2 ...
##  $ numOfParkingFeatures      : int  2 3 2 2 2 1 1 1 2 2 ...
##  $ numOfPatioAndPorchFeatures: int  0 0 2 0 4 1 0 0 0 2 ...
##  $ numOfSecurityFeatures     : int  0 0 0 0 1 3 1 0 0 2 ...
##  $ numOfWaterfrontFeatures   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ numOfWindowFeatures       : int  0 0 0 0 0 1 0 0 0 1 ...
##  $ numOfCommunityFeatures    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ lotSizeSqFt               : num  7666 8494 5183 8145 30056 ...
##  $ livingAreaSqFt            : int  2228 3494 1534 1652 3402 3573 2035 1304 1907 2484 ...
##  $ avgSchoolDistance         : num  1.9 3.3 1.8 1.97 2.07 ...
##  $ avgSchoolRating           : num  8.33 7.67 3 3 7 ...
##  $ avgSchoolSize             : int  1481 1259 1457 1457 1277 1277 1457 1457 1532 1275 ...
##  $ MedianStudentsPerTeacher  : int  16 14 13 13 16 16 13 13 12 15 ...
##  $ numOfBathrooms            : num  2 5 3 2 4 5 3 2 3 3 ...
##  $ numOfBedrooms             : int  3 4 3 3 4 4 3 3 3 4 ...
##  $ numOfStories              : int  1 2 1 1 2 2 2 1 2 2 ...
##  $ logLatestPrice            : num  5.99 6.31 5.48 5.3 6.77 ...
```

```
# Identify categorical variables based on the number of unique values
categorical_vars <- sapply(austin_data, function(x) is.factor(x) || length(unique(x)) < 10)
categorical_vars <- names(austin_data)[categorical_vars]

# Print the categorical variables
print("Categorical variables in the dataset:")
```

```
## [1] "Categorical variables in the dataset:"
```

```
print(categorical_vars)
```

```
##  [1] "hasAssociation"             "hasGarage"
##  [3] "hasSpa"                     "hasView"
##  [5] "homeType"                   "latest_saleyear"
##  [7] "numOfAccessibilityFeatures" "numOfParkingFeatures"
##  [9] "numOfPatioAndPorchFeatures" "numOfSecurityFeatures"
## [11] "numOfWaterfrontFeatures"    "numOfWindowFeatures"
## [13] "numOfCommunityFeatures"     "numOfBedrooms"
## [15] "numOfStories"
```

checking the correlation between loglatest price and numoofphotos.

```
correlation <- cor(austin_data$numOfPhotos, austin_data$latestPrice, use = "complete.obs")
print(correlation)
```

```
## [1] 0.1815243
```

- Since we have a very weak correlation between the latest price and numOfPhotos, it is better to drop this column. also clubbing it with other columns would not increase the significance of it.
- Also since the homeType has only one type of value which is “Single Family” its not telling us anything about the latest price and is redundant throughout the whole data set so we will drop it.
- I will also drop hasGarage column since its a true and false values also the noofgarage column signifies the same thing if its false by the number 0 so it has no meaning and is only presenting redundant information.

```
austin_data <- austin_data %>%
select(-numOfPhotos)
```

```
austin_data <- austin_data %>%
select(-homeType)
```

```
austin_data <- austin_data %>%
select(-hasGarage)
```

Calculate the age of the property

```
current_year <- as.numeric(format(Sys.Date(), "%Y"))
austin_data <- austin_data %>%
  mutate(property_age = current_year - yearBuilt)
```

Calculate the time since the last sale

```
current_year <- as.numeric(format(Sys.Date(), "%Y"))
austin_data <- austin_data %>%
  mutate(time_since_last_sale = current_year - latest_saleyear)
```

removing yearbuilt after calculating property_age column

```
austin_data <- austin_data %>%
select(-yearBuilt)
```

- Since we are already capturing the age of property and the time since last sale, along with this we also have sale month and sale year the latest sale date column has no significance, so we will drop it.

removing latest_salesdate and latest_salesyear

```
austin_data <- austin_data %>%
select(-latest_saledate)
```

```
austin_data <- austin_data %>%
select(-latest_saleyear)
```

printing the num rows in my dataset

```
num_rows <- nrow(austin_data)
print(num_rows)
```

```
## [1] 6784
```

Convert binary columns to numeric

```
austin_data <- austin_data %>%
  mutate(
    hasAssociation = as.numeric(hasAssociation),
    hasSpa = as.numeric(hasSpa),
    hasView = as.numeric(hasView)
  )
```

```
austin_data <- austin_data %>%
  mutate(combined_features = hasAssociation + hasSpa + hasView)
```

- Since we have made an another single column by the name of group_mean_price by associating “hasAssociation”, “hasSpa”, “hasView” columns we will drop these columns now.

after making the groupmeanprice column with has-columns, i am dropping these 3 columns

```
austin_data <- austin_data %>%
select(-hasAssociation, -hasSpa, -hasView)
```

Perform k-means clustering

```
set.seed(123)
k <- 7  # Number of clusters
kmeans_result <- kmeans(austin_data[, c("latitude", "longitude")], centers = k)
```

Add the cluster assignments to the dataset

```
austin_data$location_cluster <- kmeans_result$cluster
```

Create a season feature

```
austin_data <- austin_data %>%
  mutate(season = case_when(
    latest_salemonth %in% c(12, 1, 2) ~ "Winter",
    latest_salemonth %in% c(3, 4, 5) ~ "Spring",
    latest_salemonth %in% c(6, 7, 8) ~ "Summer",
    latest_salemonth %in% c(9, 10, 11) ~ "Fall"
  ))
```

Convert season to a factor

```
austin_data$season <- factor(austin_data$season, levels = c("Winter", "Spring", "Summer", "Fall"))
```

Create the external_features column by summing up the specified columns

```
austin_data <- austin_data %>%
  mutate(external_features = numOfParkingFeatures +
                             numOfPatioAndPorchFeatures +
                             numOfSecurityFeatures +
                             numOfWaterfrontFeatures +
                             numOfWindowFeatures +
                             numOfCommunityFeatures)
```

after making external_features as one column by summing up all the features columns i am dropping all these columns

```
austin_data <- austin_data %>%
select(-numOfParkingFeatures,
         -numOfPatioAndPorchFeatures,
         -numOfSecurityFeatures,
         -numOfWaterfrontFeatures,
         -numOfWindowFeatures,
         -numOfCommunityFeatures
            )
```

```
austin_data <- austin_data %>%
  mutate(external_features = external_features + numOfAccessibilityFeatures)
```

```
austin_data <- austin_data %>%
select(-numOfAccessibilityFeatures)
```

Create the total_amneties column by summing up the specified columns

```
austin_data <- austin_data %>%
  mutate(total_amneties = numOfBathrooms +
                              numOfBedrooms +
                              numOfStories +
                              numOfAppliances )
```

after making total amneties as one column by summing up all the numof columns, i am dropping all these columns

```
austin_data <- austin_data %>%
select( -numOfBathrooms,
        -numOfBedrooms,
        -numOfStories,
        -numOfAppliances )
```

```
austin_data <- austin_data %>%
  mutate(total_amneties = total_amneties + garageSpaces)
```

```
austin_data <- austin_data %>%
select( -garageSpaces )
```

Create ratio features

#
`{r} #austin_data <- austin_data %>% # mutate( #  size_to_students = avgSchoolSize / MedianStudentsPerTeacher # ) #`

(a) Split the data set into a training set and a test set.

```
# Hold out 20% of the data as a final validation set
train_ix = createDataPartition(austin_data$logLatestPrice, p = 0.8)
austin_train = austin_data[train_ix$Resample1,]
austin_test  = austin_data[-train_ix$Resample1,]
```

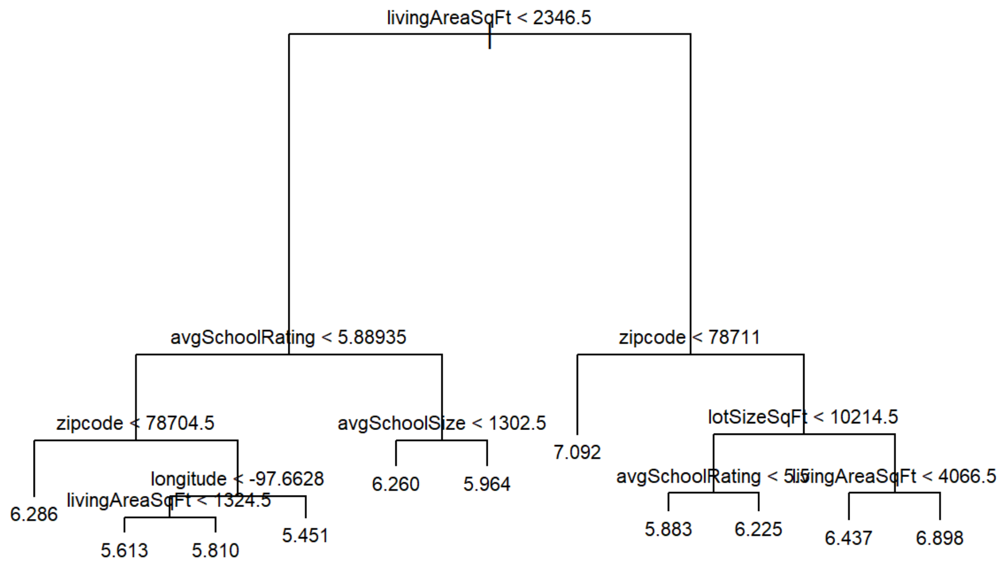
#——————————TREE——AND———PRUNEDTREE——————————

```
tree.austin_data <-tree(logLatestPrice ~ zipcode + latitude + longitude + latest_salemonth + lotSizeSqFt +
livingAreaSqFt + avgSchoolDistance + avgSchoolRating + avgSchoolSize + MedianStudentsPerTeacher + time_sin
ce_last_sale + combined_features + location_cluster + season + property_age + external_features + total_am
neties , austin_train)
```

```
summary(tree.austin_data)
```

```
##
## Regression tree:
## tree(formula = logLatestPrice ~ zipcode + latitude + longitude +
##     latest_salemonth + lotSizeSqFt + livingAreaSqFt + avgSchoolDistance +
##     avgSchoolRating + avgSchoolSize + MedianStudentsPerTeacher +
##     time_since_last_sale + combined_features + location_cluster +
##     season + property_age + external_features + total_amneties,
##     data = austin_train)
## Variables actually used in tree construction:
## [1] "livingAreaSqFt"  "avgSchoolRating" "zipcode"         "longitude"
## [5] "avgSchoolSize"   "lotSizeSqFt"
## Number of terminal nodes:  11
## Residual mean deviance:  0.1245 = 674.4 / 5418
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -5.251000 -0.180600  0.001712  0.000000  0.183000  1.973000
```

```
plot(tree.austin_data)
text(tree.austin_data, pretty = 0,  cex = 0.7)
```

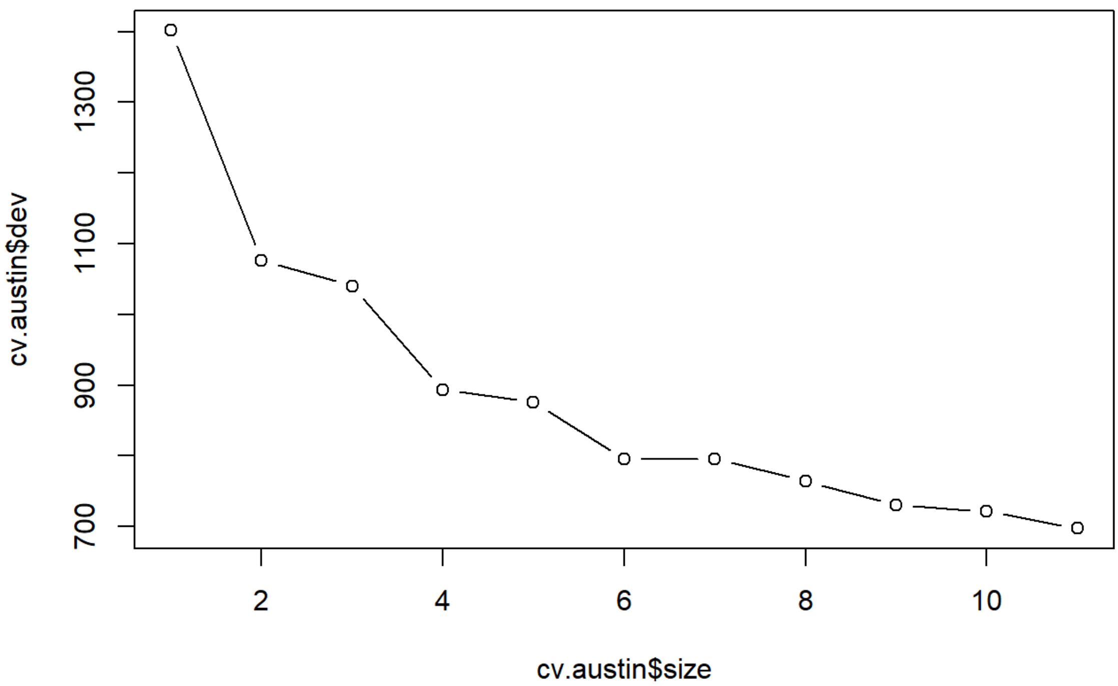



```
logpredictions <- predict(tree.austin_data, newdata = austin_test)
predictions <- exp(logpredictions)
actual_values <- austin_test$latestPrice
mse <- mean((actual_values - predictions)^2)
print(paste("Mean Squared Error: ", mse))
```

```
## [1] "Mean Squared Error:  104917.496891454"
```

```
cv.austin <-cv.tree(tree.austin_data)
```

```
plot(cv.austin$size, cv.austin$dev, type = "b")
```



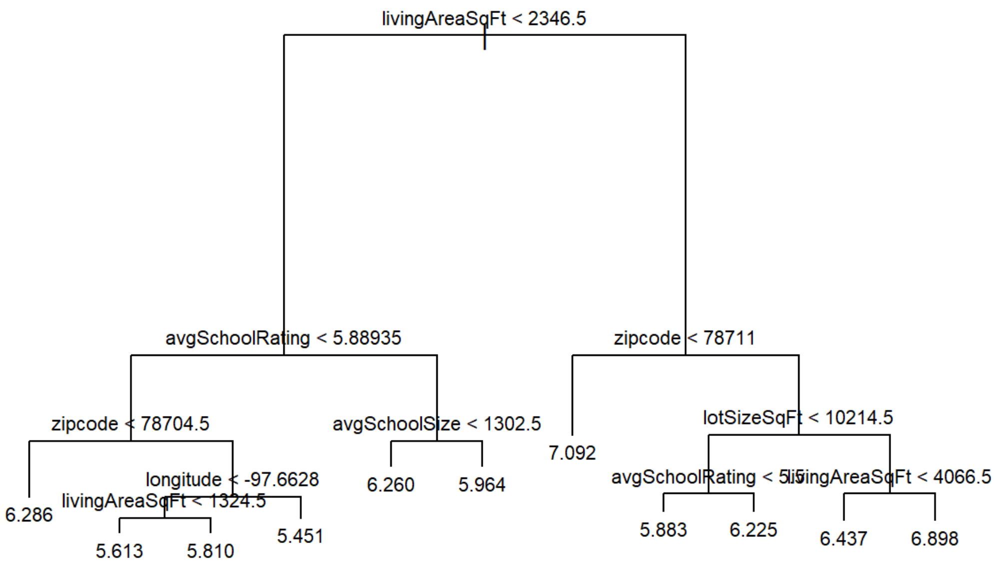

```
sizes <- cv.austin$size
deviances <- cv.austin$dev
min_deviance <- min(deviances)
min_deviance_index <- which.min(deviances)
min_size <- sizes[min_deviance_index]

se <- sd(deviances) / sqrt(length(deviances))
deviance_1se <- min_deviance + se

bestsize <- sizes[which(deviances <= deviance_1se)][1]

prune.austin <- prune.tree(tree.austin_data, best = bestsize)
print(paste("The minimum tree size after cross validation: ",bestsize))
```

```
## [1] "The minimum tree size after cross validation:  11"
```

```
print(paste("The optimal tree size(after pruning based on 1SE over best crossvalidation): ",min_size))
```

```
## [1] "The optimal tree size(after pruning based on 1SE over best crossvalidation):  11"
```

```
plot(prune.austin)
text(prune.austin, pretty = 0, cex = 0.7)
```




```
log_predictions <- predict(prune.austin, newdata = austin_test)
predictions <- exp(log_predictions)
actual_values <- austin_test$latestPrice
mse <- mean((actual_values - predictions)^2)
print(paste("Mean Squared Error after 1SE rule: ", mse))
```

```
## [1] "Mean Squared Error after 1SE rule:  104917.496891454"
```

#——————BAGGING————————————————————-

```
library(randomForest)

bag.austin_data <- randomForest(logLatestPrice ~ zipcode + latitude + longitude + latest_salemonth + lotSi
zeSqFt + livingAreaSqFt + avgSchoolDistance + avgSchoolRating + avgSchoolSize + MedianStudentsPerTeacher +
time_since_last_sale + combined_features + location_cluster + season + property_age + external_features +
total_amneties , austin_train,mtry =17, importance = TRUE)
bag.austin_data
```

```
##
## Call:
##  randomForest(formula = logLatestPrice ~ zipcode + latitude +      longitude + latest_salemonth + lotSi
zeSqFt + livingAreaSqFt +      avgSchoolDistance + avgSchoolRating + avgSchoolSize + MedianStudentsPerTeac
her +      time_since_last_sale + combined_features + location_cluster +      season + property_age + exte
rnal_features + total_amneties,      data = austin_train, mtry = 17, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 17
##
##           Mean of squared residuals: 0.0818269
##                     % Var explained: 68.29
```

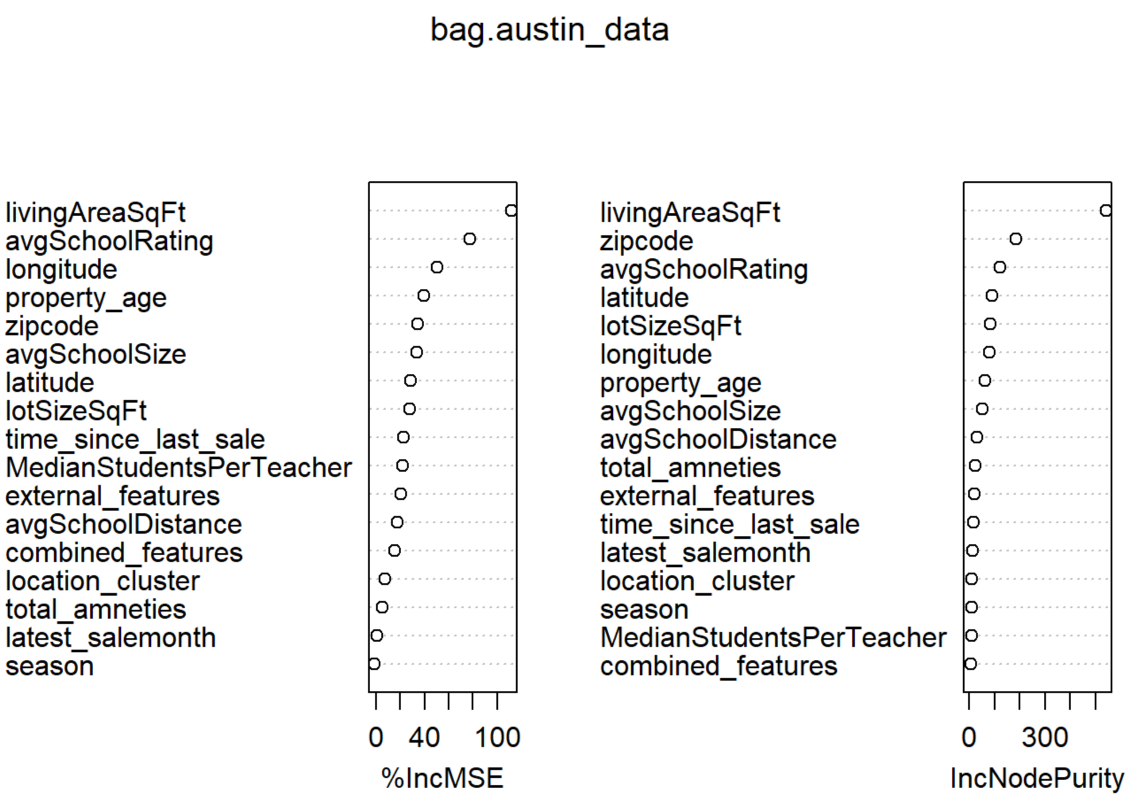
```
logpredictions <- predict(bag.austin_data, newdata = austin_test)
predictions <- exp(logpredictions)
actual_values <- austin_test$latestPrice
mse <- mean((actual_values - predictions)^2)
print(paste("Mean Squared Error for Bagging: ", mse))
```

```
## [1] "Mean Squared Error for Bagging:  66664.2998034025"
```

```
# VARIABLE IMPORTANCE
important_variables <- importance(bag.austin_data)
print(important_variables)
```

```
##                              %IncMSE IncNodePurity
## zipcode                   34.5850594    185.225163
## latitude                  28.2902187     92.331662
## longitude                 50.5969800     79.777923
## latest_salemonth           0.8349735     14.558850
## lotSizeSqFt               27.8241057     84.016955
## livingAreaSqFt           111.9704765    542.627984
## avgSchoolDistance         17.8082299     32.586985
## avgSchoolRating           77.8149985    122.480468
## avgSchoolSize             33.7929037     51.176408
## MedianStudentsPerTeacher  22.3138598      8.748236
## time_since_last_sale      22.6328484     17.761142
## combined_features         15.1561385      6.167718
## location_cluster           7.7834641     10.846431
## season                    -1.2802299     10.550595
## property_age              39.4687126     62.053731
## external_features         20.5744721     18.992116
## total_amneties             5.1915658     23.163251
```

```
varImpPlot(bag.austin_data)
```



#

—————————————-RANDOMFOREST—————————————-

```r
num_features_values = c(4, 5, 6, 7 , 8 , 9 ,10)


evaluation_results <- matrix(NA, nrow = length(num_features_values), ncol = 2)
colnames(evaluation_results) <- c("num_features", "MSE")


for (index in 1:length(num_features_values)) {
  num_features <- num_features_values[index]


rfmodel <- randomForest(logLatestPrice ~  zipcode + latitude + longitude + latest_salemonth + lotSizeSqFt
+ livingAreaSqFt + avgSchoolDistance + avgSchoolRating + avgSchoolSize + MedianStudentsPerTeacher + time_s
ince_last_sale + combined_features + location_cluster + season + property_age + external_features + total_
amneties , austin_train,mtry = num_features,importance = TRUE)

  logpredictions <- predict(rfmodel, newdata = austin_test)
  predictions <- exp(logpredictions)
  actual_values <- austin_test$latestPrice
  mse <- mean((actual_values - predictions)^2)
  print(mse)
  evaluation_results[index, ] <- c(num_features, mse)
}
```

```
## [1] 73718.74
## [1] 72525.8
## [1] 71650.38
## [1] 69983.79
## [1] 69196.71
## [1] 68338.38
## [1] 68386.21
```

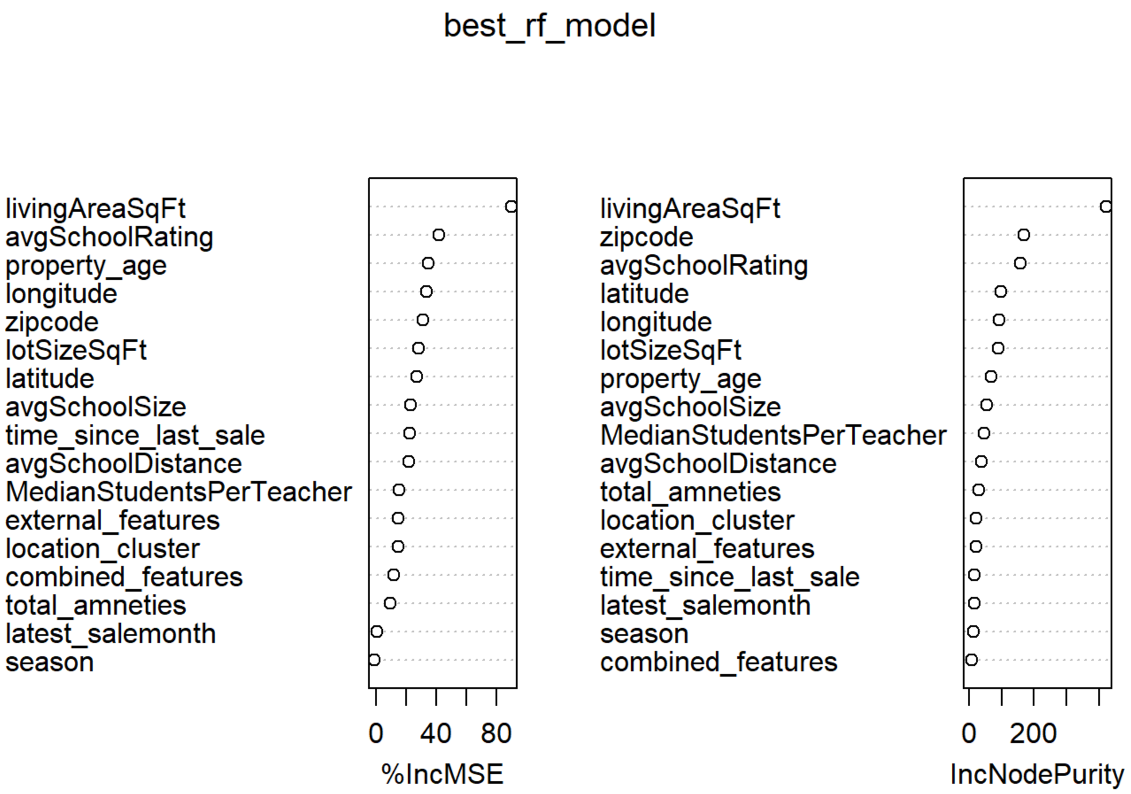
```r
print(evaluation_results)
```

```
##      num_features      MSE
## [1,]            4 73718.74
## [2,]            5 72525.80
## [3,]            6 71650.38
## [4,]            7 69983.79
## [5,]            8 69196.71
## [6,]            9 68338.38
## [7,]           10 68386.21
```

```r
results_df <- as.data.frame(evaluation_results)
colnames(results_df) <- c("num_features", "MSE")
print(results_df)
```

```
##   num_features      MSE
## 1            4 73718.74
## 2            5 72525.80
## 3            6 71650.38
## 4            7 69983.79
## 5            8 69196.71
## 6            9 68338.38
## 7           10 68386.21
```

```
# Train the final random forest model with the best mtry value
best_num_features <- results_df$num_features[which.min(results_df$MSE)]
best_rf_model <- randomForest(logLatestPrice ~ zipcode + latitude + longitude + latest_salemonth + lotSize
SqFt + livingAreaSqFt + avgSchoolDistance + avgSchoolRating + avgSchoolSize + MedianStudentsPerTeacher + t
ime_since_last_sale + combined_features + location_cluster + season + property_age + external_features + t
otal_amneties, austin_train,
  mtry = best_num_features,
  importance = TRUE
)

importance_values <- importance(best_rf_model)
varImpPlot(best_rf_model)
```




```
library(BART)

x_train <- as.data.frame(austin_train[, -which(names(austin_train) %in% c("loglatestPrice", "latestPric
e"))])
y_train <- austin_train$logLatestPrice

x_test <- as.data.frame(austin_test[, -which(names(austin_test) %in% c("loglatestPrice", "latestPrice"))])
y_test <- austin_test$logLatestPrice

# Fit the BART model
bartfit <- gbart(x_train, y_train, x.test = x_test)
```

```
## *****Calling gbart: type=1
## *****Data:
## data:n,p,np: 5429, 21, 1355
## y1,yn: -0.060002, 0.722758
## x1,x[n*p]: 78717.000000, 13.000000
## xp1,xp[np*p]: 78725.000000, 15.000000
## *****Number of Trees: 200
## *****Number of Cut Points: 40 ... 37
## *****burn,nd,thin: 100,1000,1
## *****Prior:beta,alpha,tau,nu,lambda,offset: 2,0.95,0.121778,3,2.12753e-29,6.05147
## *****sigma: 0.000000
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,21,0
## *****printevery: 100
##
## MCMC
## done 0 (out of 1100)
## done 100 (out of 1100)
## done 200 (out of 1100)
## done 300 (out of 1100)
## done 400 (out of 1100)
## done 500 (out of 1100)
## done 600 (out of 1100)
## done 700 (out of 1100)
## done 800 (out of 1100)
## done 900 (out of 1100)
## done 1000 (out of 1100)
## time: 35s
## trcnt,tecnt: 1000,1000
```

```
yhat_bart_log <- bartfit$yhat.test.mean
yhat_bart <- exp(yhat_bart_log)
y_test_exp <- exp(y_test)
mse <- mean((y_test_exp - yhat_bart)^2)
```

```
print(paste("The MSE for BART: ",mse))
```

```
## [1] "The MSE for BART:  12766.1489802102"
```

```
ord <- order(bartfit$varcount.mean , decreasing = T)
bartfit$varcount.mean[ord]
```

```
##          logLatestPrice               longitude          livingAreaSqFt
##                  83.569                  17.519                  13.362
##         avgSchoolRating    time_since_last_sale          total_amneties
##                  12.991                  12.683                  11.237
##                 zipcode                 season4       combined_features
##                  10.128                   9.783                   9.767
##                 season3       avgSchoolDistance           avgSchoolSize
##                   9.389                   8.794                   7.771
##                 season2                latitude MedianStudentsPerTeacher
##                   7.362                   6.895                   6.638
##        location_cluster                 season1       external_features
##                   6.298                   4.599                   3.463
##             lotSizeSqFt            property_age        latest_salemonth
##                   2.224                   1.933                   0.863
```

- The MSE for BART: 12766.1489802102
- The MSE for RANDOMFOREST: mtry(9) 68338.38

- The MSE for Bagging: 66664.2998034025
- The MSE for Tree and Pruned Tree: 104917.496891454

#———————————————————————————-

```
austin_data2 <- read.csv("austinhouses_holdout.csv")

# Include all predictors except 'streetAddress' and 'description'
austin_data2 <- austin_data2 %>%
  select(-streetAddress, -description)

# Add a new column 'logLatestPrice' based on 'latestPrice'
austin_data2 <- austin_data2 %>%
  mutate(logLatestPrice = log(latestPrice))

# Display the first few rows of the modified dataset
head(austin_data2)
```

```
##   zipcode latitude longitude garageSpaces hasAssociation hasGarage hasSpa
## 1   78749 30.20016 -97.85626            0           TRUE     FALSE  FALSE
## 2   78757 30.33993 -97.74895            0          FALSE     FALSE  FALSE
## 3   78747 30.13613 -97.76612            1           TRUE      TRUE  FALSE
## 4   78748 30.15642 -97.81450            2          FALSE      TRUE  FALSE
## 5   78729 30.46093 -97.77524            0          FALSE     FALSE  FALSE
## 6   78729 30.44455 -97.76396            2          FALSE      TRUE  FALSE
##   hasView      homeType yearBuilt latestPrice latest_saledate latest_salemonth
## 1   FALSE Single Family      1999          NA      2018-08-24                8
## 2   FALSE Single Family      1962          NA      2019-02-15                2
## 3   FALSE Single Family      2015          NA      2020-03-10                3
## 4   FALSE Single Family      1987          NA      2020-07-08                7
## 5   FALSE Single Family      1989          NA      2019-02-22                2
## 6   FALSE Single Family      1990          NA      2018-10-03               10
##   latest_saleyear numOfPhotos numOfAccessibilityFeatures numOfAppliances
## 1            2018          37                          0               5
## 2            2019          35                          0               0
## 3            2020          28                          0               3
## 4            2020          63                          0               5
## 5            2019          40                          0               2
## 6            2018          58                          0               3
##   numOfParkingFeatures numOfPatioAndPorchFeatures numOfSecurityFeatures
## 1                    1                          0                     0
## 2                    1                          0                     0
## 3                    2                          0                     0
## 4                    2                          4                     3
## 5                    2                          0                     0
## 6                    3                          0                     0
##   numOfWaterfrontFeatures numOfWindowFeatures numOfCommunityFeatures
## 1                       0                   0                      0
## 2                       0                   0                      0
## 3                       0                   0                      0
## 4                       0                   1                      0
## 5                       0                   0                      0
## 6                       0                   0                      0
##   lotSizeSqFt livingAreaSqFt avgSchoolDistance avgSchoolRating avgSchoolSize
## 1        9888           2023         1.3333333        6.666667          1460
## 2       10715           1806         0.7333333        6.666667          1153
## 3        6359           2314         2.4333333        5.333333          1506
## 4        5009           1891         1.0000000        3.333333          1424
## 5        7230           2311         1.2333333        5.333333          1369
## 6        6838           2593         0.9333333        5.666667          1402
##   MedianStudentsPerTeacher numOfBathrooms numOfBedrooms numOfStories
## 1                       16              3             4            2
## 2                       16              2             3            1
## 3                       15              3             5            2
## 4                       14              3             4            2
## 5                       12              2             3            2
## 6                       12              3             4            2
##   logLatestPrice
## 1             NA
## 2             NA
## 3             NA
## 4             NA
## 5             NA
## 6             NA
```

Handling missing values

```
austin_data2[is.na(austin_data2)] <- -1
austin_data2 <- austin_data2 %>%
  drop_na()
```

```
print(colnames(austin_data2))
```

```
##  [1] "zipcode"                    "latitude"
##  [3] "longitude"                  "garageSpaces"
##  [5] "hasAssociation"             "hasGarage"
##  [7] "hasSpa"                     "hasView"
##  [9] "homeType"                   "yearBuilt"
## [11] "latestPrice"                "latest_saledate"
## [13] "latest_salemonth"           "latest_saleyear"
## [15] "numOfPhotos"                "numOfAccessibilityFeatures"
## [17] "numOfAppliances"            "numOfParkingFeatures"
## [19] "numOfPatioAndPorchFeatures" "numOfSecurityFeatures"
## [21] "numOfWaterfrontFeatures"    "numOfWindowFeatures"
## [23] "numOfCommunityFeatures"     "lotSizeSqFt"
## [25] "livingAreaSqFt"             "avgSchoolDistance"
## [27] "avgSchoolRating"            "avgSchoolSize"
## [29] "MedianStudentsPerTeacher"   "numOfBathrooms"
## [31] "numOfBedrooms"              "numOfStories"
## [33] "logLatestPrice"
```

```
austin_data2 <- austin_data2 %>%
select(-numOfPhotos)
```

```
austin_data2 <- austin_data2 %>%
select(-homeType)
```

```
austin_data2 <- austin_data2 %>%
select(-hasGarage)
```

Calculate the age of the property

```
current_year <- as.numeric(format(Sys.Date(), "%Y"))
austin_data2 <- austin_data2 %>%
  mutate(property_age = current_year - yearBuilt)
```

Calculate the time since the last sale

```
current_year <- as.numeric(format(Sys.Date(), "%Y"))
austin_data2 <- austin_data2 %>%
  mutate(time_since_last_sale = current_year - latest_saleyear)
```

removing yearbuilt after calculating property_age column

```
austin_data2 <- austin_data2 %>%
select(-yearBuilt)
```

removing latest_salesdate and latest_salesyear

```
austin_data2 <- austin_data2 %>%
select(-latest_saledate)
```

```
austin_data2 <- austin_data2 %>%
select(-latest_saleyear)
```

printing the num rows in my dataset

```
num_rows <- nrow(austin_data2)
print(num_rows)
```

```
## [1] 6785
```

Convert binary columns to numeric

```
austin_data2 <- austin_data2 %>%
  mutate(
    hasAssociation = as.numeric(hasAssociation),
    hasSpa = as.numeric(hasSpa),
    hasView = as.numeric(hasView)

  )
```

```
austin_data2 <- austin_data2 %>%
  mutate(combined_features = hasAssociation + hasSpa + hasView)
```

after making the groupmeanprice column with has-columns, i am dropping these 3 columns

```
austin_data2 <- austin_data2 %>%
select(-hasAssociation, -hasSpa, -hasView)
```

Perform k-means clustering

```
set.seed(123)
k <- 7  # Number of clusters
kmeans_result <- kmeans(austin_data2[, c("latitude", "longitude")], centers = k)
```

Add the cluster assignments to the dataset

```
austin_data2$location_cluster <- kmeans_result$cluster
```

Create a season feature

```
austin_data2 <- austin_data2 %>%
  mutate(season = case_when(
    latest_salemonth %in% c(12, 1, 2) ~ "Winter",
    latest_salemonth %in% c(3, 4, 5) ~ "Spring",
    latest_salemonth %in% c(6, 7, 8) ~ "Summer",
    latest_salemonth %in% c(9, 10, 11) ~ "Fall"
  ))
```

Convert season to a factor

```
austin_data2$season <- factor(austin_data2$season, levels = c("Winter", "Spring", "Summer", "Fall"))
```

Create the external_features column by summing up the specified columns

```
austin_data2 <- austin_data2 %>%
  mutate(external_features = numOfParkingFeatures +
                             numOfPatioAndPorchFeatures +
                             numOfSecurityFeatures +
                             numOfWaterfrontFeatures +
                             numOfWindowFeatures +
                             numOfCommunityFeatures)
```

after making external_features as one column by summing up all the features columns i am dropping all these columns

```
austin_data2 <- austin_data2 %>%
select(-numOfParkingFeatures,
         -numOfPatioAndPorchFeatures,
         -numOfSecurityFeatures,
         -numOfWaterfrontFeatures,
         -numOfWindowFeatures,
         -numOfCommunityFeatures
            )
```

```
austin_data2 <- austin_data2 %>%
  mutate(external_features = external_features + numOfAccessibilityFeatures)
```

```
austin_data2 <- austin_data2 %>%
select(-numOfAccessibilityFeatures)
```

Create the total_amneties column by summing up the specified columns

```
austin_data2 <- austin_data2 %>%
  mutate(total_amneties = numOfBathrooms +
                             numOfBedrooms +
                             numOfStories +
                             numOfAppliances )
```

after making total amneties as one column by summing up all the numof columns, i am dropping all these columns

```
austin_data2 <- austin_data2 %>%
select( -numOfBathrooms,
        -numOfBedrooms,
        -numOfStories,
        -numOfAppliances )
```

```
austin_data2 <- austin_data2 %>%
  mutate(total_amneties = total_amneties + garageSpaces)
```

```
austin_data2 <- austin_data2 %>%
select( -garageSpaces )
```

```
# Ensure consistent columns between austin_data and austin_data2
common_columns <- intersect(colnames(austin_data), colnames(austin_data2))
austin_data <- austin_data[, common_columns]
austin_data2 <- austin_data2[, common_columns]

# Print dimensions of the data to check for consistency
print(paste("Dimensions of austin_data:", dim(austin_data)))
```

```
## [1] "Dimensions of austin_data: 6784" "Dimensions of austin_data: 19"
```

```
print(paste("Dimensions of austin_data2:", dim(austin_data2)))
```

```
## [1] "Dimensions of austin_data2: 6785" "Dimensions of austin_data2: 19"
```

```
# Prepare training and testing data
x_train <- as.data.frame(austin_data[, -which(names(austin_data) %in% c("logLatestPrice", "latestPric
e"))])
y_train <- austin_data$logLatestPrice

x_test <- as.data.frame(austin_data2[, -which(names(austin_data2) %in% c("logLatestPrice", "latestPric
e"))])
y_test <- austin_data2$logLatestPrice

# Print dimensions of the training and test sets
print(paste("Dimensions of x_train:", dim(x_train)))
```

```
## [1] "Dimensions of x_train: 6784" "Dimensions of x_train: 17"
```

```
print(paste("Length of y_train:", length(y_train)))
```

```
## [1] "Length of y_train: 6784"
```

```
print(paste("Dimensions of x_test:", dim(x_test)))
```

```
## [1] "Dimensions of x_test: 6785" "Dimensions of x_test: 17"
```

```r
print(paste("Length of y_test:", length(y_test)))
```

```
## [1] "Length of y_test: 6785"
```

```r
# Check for missing values
print(paste("Number of missing values in x_train:", sum(is.na(x_train))))
```

```
## [1] "Number of missing values in x_train: 0"
```

```r
print(paste("Number of missing values in y_train:", sum(is.na(y_train))))
```

```
## [1] "Number of missing values in y_train: 0"
```

```r
print(paste("Number of missing values in x_test:", sum(is.na(x_test))))
```

```
## [1] "Number of missing values in x_test: 0"
```

```r
print(paste("Number of missing values in y_test:", sum(is.na(y_test))))
```

```
## [1] "Number of missing values in y_test: 0"
```

```r
# Check for missing values in logLatestPrice in austin_data2
sum(is.na(austin_data2$logLatestPrice))
```

```
## [1] 0
```

After comparing the MSE values of the models, the BART model had the lowest MSE. However, due to overfitting, I opted to use the second-best model, Bagging, instead.

```r
library(randomForest)

bag.austin_data_bg <- randomForest(logLatestPrice ~ zipcode + latitude + longitude + latest_salemonth + lo
tSizeSqFt + livingAreaSqFt + avgSchoolDistance + avgSchoolRating + avgSchoolSize + MedianStudentsPerTeache
r + time_since_last_sale + combined_features + location_cluster + season + property_age + external_feature
s + total_amneties , austin_data,mtry =17, importance = TRUE)
bag.austin_data_bg
```

```
##
## Call:
##  randomForest(formula = logLatestPrice ~ zipcode + latitude +      longitude + latest_salemonth + lotSizeSqFt + livingAreaSqFt +      avgSchoolDistance + avgSchoolRating + avgSchoolSize + MedianStudentsPerTeacher +      time_since_last_sale + combined_features + location_cluster +      season + property_age + external_features + total_amneties,      data = austin_data, mtry = 17, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 17
##
##           Mean of squared residuals: 0.07435401
##                     % Var explained: 71.27
```

```
logpredictions <- predict(bag.austin_data_bg, newdata = austin_data2)
predictions <- exp(logpredictions)
austin_data2$latestPrice <-predictions
```

{r} #write.csv(austin_data2, "test_bag.csv",row.names = FALSE)