PRANAV GARG (PG26855) PROJECT

2024-07-24

Approach

Approach Followed in the Code

1. **Data Import and Initial Exploration**:
   - Load the dataset and display its structure.
   - Perform initial data exploration to understand the data characteristics.
2. **Data Manipulation**:
   - Add a new column `logLatestPrice` by taking the logarithm of `latestPrice`.
   - Derive the age of the house and categorize it into different age groups.
3. **Binary Columns and Amenities**:
   - Convert binary columns (`hasAssociation` , `hasGarage` , `hasSpa` , `hasView`) to numeric.
   - Create a new feature `total_amenities` by summing up the binary columns.
4. **Sale Date Features**:
   - Extract and convert sale date components into year, month, and day.
   - Categorize the sale date into seasons (`Winter` , `Spring` , `Summer` , `Fall`).
5. **Geospatial Features**:
   - Perform clustering based on latitude and longitude to create a new feature `crossCluster` .
6. **Calculating Correlations and Aggregating Amenities**:
   - Calculate correlations of various amenity features with the target variable `logLatestPrice` .
   - Aggregate amenities with specified weights to create a new `total_amenities` feature.
7. **Handling Missing Values**:
   - Handle missing values by replacing them with -1 and dropping any remaining NA values.
   - Convert categorical variables to factors.
8. **Data Visualization**:
   - Plot a correlation matrix to understand the relationships between different variables.
9. **Feature Engineering**:
   - Define various formulas for the regression models, selecting different sets of features based on their correlation with the target variable.
10. **Modeling**:
    - Split the data into training and test sets.
    - Train and evaluate multiple models including Regression Tree, Pruned Tree, Bagging with Random Forest, Random Forest with different `mtry` values, XGBoost, BART, Ridge, and Lasso Regression.
    - Calculate and compare the MSE and RMSE for each model to identify the best performer.
11. **Model Comparison and Conclusion**:
    - Summarize the performance of different models.
    - Provide insights and recommendations based on the model performance.
12. **Applying the Best Model on the Holdout Set**:

- Load and preprocess the holdout dataset similarly to the training dataset.
- Train the BART model using the training data.
- Make predictions on the holdout dataset using the trained BART model.
- Export the updated holdout dataset with the predicted `latestPrice`.

Summary:

The approach involves comprehensive data preprocessing, feature engineering, model training, and evaluation. It concludes with applying the best model (BART) on a holdout dataset to make predictions, ensuring that all preprocessing steps are consistently applied to both the training and holdout datasets.

Importing Libraries

Data Import and Initial Exploration

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

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

```
glimpse((austin_data))
```

```
## Rows: 6,784
## Columns: 34
## $ streetAddress                <chr> "14004 Chisos Trl", "14405 Laurinburg Dr", …
## $ zipcode                      <int> 78717, 78717, 78725, 78725, 78726, 78726, 7…
## $ description                  <chr> "COVETED, SPACIOUS 3 bed + OFFICE, 1story M…
## $ latitude                     <dbl> 30.49564, 30.48878, 30.23315, 30.23824, 30.…
## $ longitude                    <dbl> -97.79787, -97.79490, -97.58732, -97.57833,…
## $ garageSpaces                 <int> 0, 2, 2, 2, 2, 0, 0, 0, 2, 2, 0, 2, 0, 2, 0…
## $ hasAssociation               <lgl> TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, …
## $ hasGarage                    <lgl> FALSE, TRUE, TRUE, TRUE, TRUE, FALSE, FALSE…
## $ hasSpa                       <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F…
## $ hasView                      <lgl> FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FA…
## $ homeType                     <chr> "Single Family", "Single Family", "Single F…
## $ yearBuilt                    <int> 2008, 2013, 1999, 2012, 2004, 2005, 2000, 2…
## $ latestPrice                  <dbl> 400.000, 549.900, 240.000, 200.000, 875.000…
## $ latest_saledate              <chr> "2020-01-10", "2018-03-13", "2020-12-31", "…
## $ latest_salemonth             <int> 1, 3, 12, 1, 11, 9, 6, 3, 3, 3, 1, 6, 2, 4,…
## $ latest_saleyear              <int> 2020, 2018, 2020, 2018, 2020, 2019, 2019, 2…
## $ numOfPhotos                  <int> 20, 69, 10, 33, 38, 37, 24, 26, 31, 8, 24, …
## $ numOfAccessibilityFeatures <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ numOfAppliances              <int> 3, 4, 4, 5, 8, 4, 4, 4, 7, 2, 6, 4, 3, 5, 3…
## $ numOfParkingFeatures         <int> 2, 3, 2, 2, 2, 1, 1, 1, 2, 2, 1, 2, 1, 2, 2…
## $ numOfPatioAndPorchFeatures <int> 0, 0, 2, 0, 4, 1, 0, 0, 0, 2, 0, 1, 0, 1, 2…
## $ numOfSecurityFeatures        <int> 0, 0, 0, 0, 1, 3, 1, 0, 0, 2, 0, 4, 0, 1, 1…
## $ numOfWaterfrontFeatures      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ numOfWindowFeatures          <int> 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1…
## $ numOfCommunityFeatures       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ lotSizeSqFt                  <dbl> 7666.0, 8494.0, 5183.0, 8145.0, 30056.4, 19…
## $ livingAreaSqFt               <int> 2228, 3494, 1534, 1652, 3402, 3573, 2035, 1…
## $ avgSchoolDistance            <dbl> 1.900000, 3.300000, 1.800000, 1.966667, 2.0…
## $ avgSchoolRating              <dbl> 8.333333, 7.666667, 3.000000, 3.000000, 7.0…
## $ avgSchoolSize                <int> 1481, 1259, 1457, 1457, 1277, 1277, 1457, 1…
## $ MedianStudentsPerTeacher     <int> 16, 14, 13, 13, 16, 16, 13, 13, 12, 15, 13,…
## $ numOfBathrooms               <dbl> 2, 5, 3, 2, 4, 5, 3, 2, 3, 3, 3, 3, 2, 2, 2…
## $ numOfBedrooms                <int> 3, 4, 3, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3…
## $ numOfStories                 <int> 1, 2, 1, 1, 2, 2, 2, 1, 2, 2, 1, 1, 1, 1, 1…
```

Data Manipulation

```
# Adding the logLatestPrice to the original df
austin_data <- austin_data %>%
  select(-streetAddress, -description) %>%
  mutate(logLatestPrice = log(latestPrice))

# Derive the age of the house
current_year <- year(Sys.Date())
austin_data <- austin_data %>%
  mutate(houseAge = current_year - yearBuilt,
         property_age_category = cut(houseAge,
                                     breaks = c(-Inf, 10, 30, 50, Inf),
                                     labels = c("New", "Moderate", "Old", "Very Ol
d")),
         property_age_category = as.factor(property_age_category)) %>%
  select(-yearBuilt)
```

Binary Columns and Amenities

```
# Convert binary columns to numeric and aggregate amenities
binary_columns <- c('hasAssociation', 'hasGarage', 'hasSpa', 'hasView')
austin_data[binary_columns] <- lapply(austin_data[binary_columns], as.numeric)

# Create a total_amenities feature
austin_data <- austin_data %>%
  mutate(total_amenities = rowSums(select(., all_of(binary_columns))))

# Ensure we retain the individual binary features
#austin_data <- austin_data %>%
#  select(-hasAssociation, -hasSpa, -hasView, -hasGarage)
```

Sale Date Features

```
# Extract and convert sale date components
austin_data <- austin_data %>%
  mutate(saleDate = as.Date(latest_saledate, format = "%Y-%m-%d"),
         saleYear = year(latest_saledate),
         saleMonth = month(latest_saledate),
         saleDay = day(latest_saledate)) %>%
  mutate(season = case_when(
    saleMonth %in% c(12, 1, 2) ~ "Winter",
    saleMonth %in% c(3, 4, 5) ~ "Spring",
    saleMonth %in% c(6, 7, 8) ~ "Summer",
    saleMonth %in% c(9, 10, 11) ~ "Fall"
  )) %>%
  mutate(season = factor(season, levels = c("Winter", "Spring", "Summer", "Fall")))
%>%
  select(-latest_saledate, -saleDay, -saleDate)
```

Geospatial Features

```
# Clustering based on latitude and longitude
set.seed(123)
coords <- austin_data %>% select(latitude, longitude)
clusters <- kmeans(coords, centers = 5)$cluster
austin_data <- austin_data %>% mutate(crossCluster = as.factor(clusters))
```

Calculating Correlations and Aggregating Amenities

```
# Calculate correlations and aggregate amenities with weights
amenity_features <- c("numOfAccessibilityFeatures", "numOfAppliances", "numOfParkin
gFeatures",
                      "numOfPatioAndPorchFeatures", "numOfSecurityFeatures", "numOf
WaterfrontFeatures",
                      "numOfWindowFeatures", "numOfCommunityFeatures")

correlations <- sapply(austin_data[amenity_features], function(x) cor(x, austin_dat
a$logLatestPrice, use = "complete.obs"))
print(correlations)
```

```
## numOfAccessibilityFeatures            numOfAppliances
##                0.026152950                0.046333333
##       numOfParkingFeatures numOfPatioAndPorchFeatures
##                0.123709340                0.147231535
##      numOfSecurityFeatures    numOfWaterfrontFeatures
##                0.110830708                0.052710182
##        numOfWindowFeatures     numOfCommunityFeatures
##                0.116152259               -0.002181723
```

```
weights <- c(numOfAccessibilityFeatures = 1, numOfAppliances = 2, numOfParkingFeatu
res = 1.5,
             numOfPatioAndPorchFeatures = 1, numOfSecurityFeatures = 1, numOfWaterf
rontFeatures = 2.5,
             numOfWindowFeatures = 1, numOfCommunityFeatures = 1.5)

austin_data <- austin_data %>%
  mutate(total_amenities = numOfAccessibilityFeatures * weights["numOfAccessibility
Features"] +
                          numOfAppliances * weights["numOfAppliances"] +
                          numOfParkingFeatures * weights["numOfParkingFeatures"] +
                          numOfPatioAndPorchFeatures * weights["numOfPatioAndPorchF
eatures"]+
                          numOfSecurityFeatures * weights["numOfSecurityFeatures"]
+
                          numOfWaterfrontFeatures * weights["numOfWaterfrontFeature
s"] +
                          numOfWindowFeatures * weights["numOfWindowFeatures"] +
                          numOfCommunityFeatures * weights["numOfCommunityFeature
s"]) %>%
  select(-all_of(amenity_features))
```

Handling Missing Values

```
# Handling missing values
austin_data[is.na(austin_data)] <- -1
austin_data <- austin_data %>%
  drop_na()

# Convert categorical variables to factors
austin_data <- austin_data %>%
  mutate(
    zipcode = as.factor(zipcode),
    garageSpaces = factor(garageSpaces),
    homeType = factor(homeType)
  ) %>%
  select(-homeType) #removing Hometype becuase
# glimpse(austin_data)
head(austin_data)
```

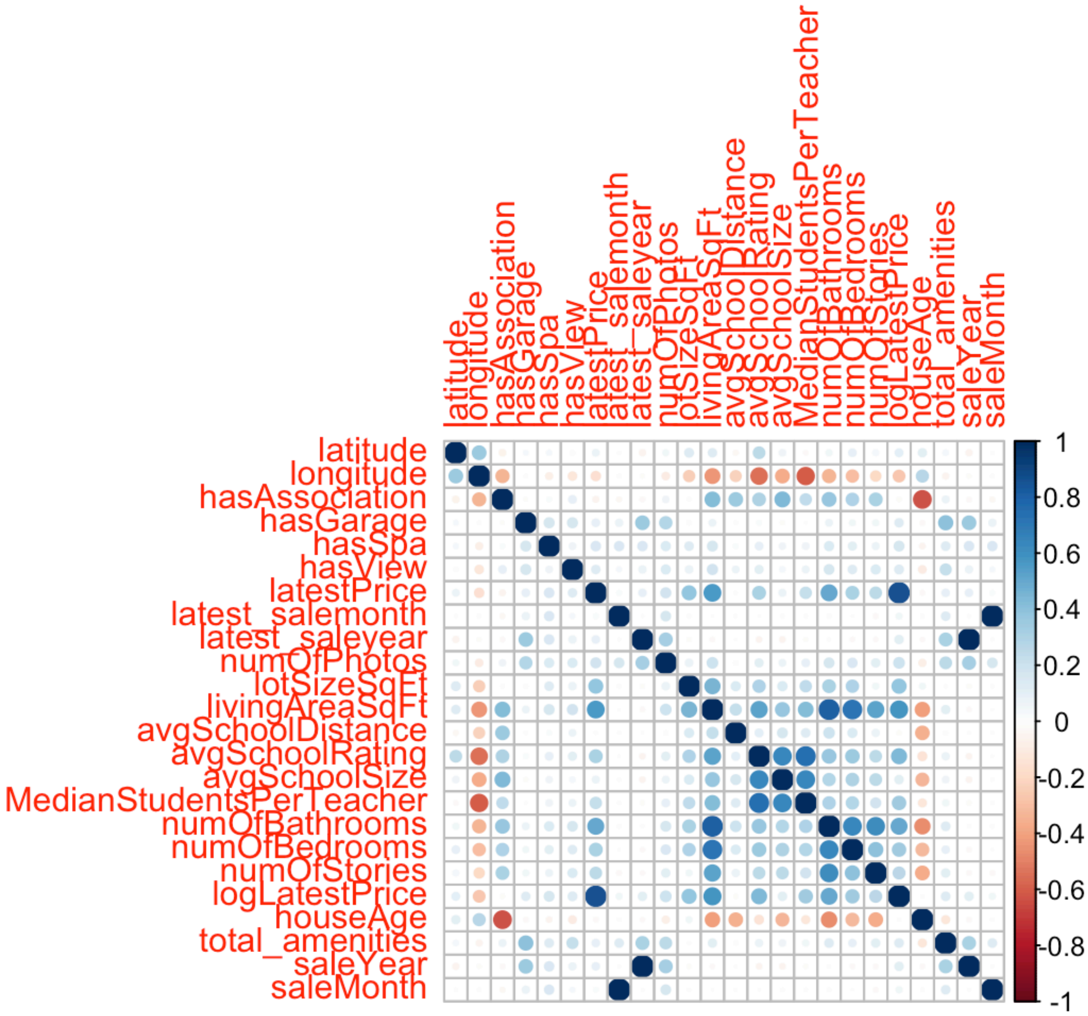
```
##   zipcode latitude longitude garageSpaces hasAssociation hasGarage hasSpa
## 1   78717 30.49564 -97.79787            0              1         0      0
## 2   78717 30.48878 -97.79490            2              1         1      0
## 3   78725 30.23315 -97.58732            2              0         1      0
## 4   78725 30.23824 -97.57833            2              1         1      0
## 5   78726 30.42646 -97.85929            2              1         1      0
## 6   78726 30.42596 -97.85841            0              1         0      0
##   hasView latestPrice latest_salemonth latest_saleyear numOfPhotos lotSizeSqFt
## 1       0       400.0                1            2020          20      7666.0
## 2       0       549.9                3            2018          69      8494.0
## 3       0       240.0               12            2020          10      5183.0
## 4       0       200.0                1            2018          33      8145.0
## 5       1       875.0               11            2020          38     30056.4
## 6       0       830.0                9            2019          37     19166.4
##   livingAreaSqFt avgSchoolDistance avgSchoolRating avgSchoolSize
## 1           2228          1.900000        8.333333          1481
## 2           3494          3.300000        7.666667          1259
## 3           1534          1.800000        3.000000          1457
## 4           1652          1.966667        3.000000          1457
## 5           3402          2.066667        7.000000          1277
## 6           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 houseAge property_age_category total_amenities saleYear
## 1       5.991465       16              Moderate             9.0     2020
## 2       6.309736       11              Moderate            12.5     2018
## 3       5.480639       25              Moderate            13.0     2020
## 4       5.298317       12              Moderate            13.0     2018
## 5       6.774224       20              Moderate            24.0     2020
## 6       6.721426       19              Moderate            14.5     2019
##   saleMonth season crossCluster
## 1         1 Winter            2
## 2         3 Spring            2
## 3        12 Winter            4
## 4         1 Winter            4
## 5        11   Fall            1
## 6         9   Fall            1
```

Data Visualization

Plot Correlation Matrix

```
# Plot correlation matrix
columns <- c(
  "zipcode" ,"latitude" , "longitude","garageSpaces","latest_salemonth" ,"latest_sa
leyear" ,"numOfPhotos","lotSizeSqFt","livingAreaSqFt" ,
  "numOfBathrooms" ,"numOfBedrooms","numOfStories",
  "houseAge","property_age_category","total_amenities","saleYear" ,"saleMonth","sea
son","crossCluster"
)

numeric_data <- austin_data %>% select_if(is.numeric)
cor_matrix <- cor(numeric_data, use = "complete.obs")
corrplot(cor_matrix, method = "circle")
```



```
'
melted_cor_matrix <- melt(cor_matrix)
ggplot(data = melted_cor_matrix, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 15, hjust = 1)) +
  coord_fixed()
'
```

```
## [1] "\nmelted_cor_matrix <- melt(cor_matrix)\nggplot(data = melted_cor_matrix, a
es(x = Var1, y = Var2, fill = value)) +\n  geom_tile() +\n  scale_fill_gradient2(lo
w = \"blue\", high = \"red\", mid = \"white\", \n                      midpoint = 
0, limit = c(-1,1), space = \"Lab\", \n                      name=\"Correlation\")
+\n  theme_minimal() +\n  theme(axis.text.x = element_text(angle = 45, vjust = 1, \
n                                   size = 15, hjust = 1)) +\n  coord_fixed()\n"
```

```
formulaa = logLatestPrice ~  zipcode + latitude + longitude + garageSpaces + latest
_salemonth + latest_saleyear + numOfPhotos + lotSizeSqFt + livingAreaSqFt + avgScho
olDistance + avgSchoolRating + avgSchoolSize + MedianStudentsPerTeacher + numOfBath
rooms + numOfBedrooms + numOfStories + houseAge + property_age_category + total_ame
nities + saleYear + saleMonth + season + crossCluster

# removing few features " avgSchoolDistance + avgSchoolRating + avgSchoolSize + Med
ianStudentsPerTeacher " as they have low correlation

formulaa2 = logLatestPrice ~  zipcode + latitude + longitude + garageSpaces + lates
t_salemonth + latest_saleyear + numOfPhotos + lotSizeSqFt + livingAreaSqFt + numOfB
athrooms + numOfBedrooms + numOfStories + houseAge + property_age_category + total_
amenities + saleYear + saleMonth + season + crossCluster

formulaa3 = logLatestPrice ~  zipcode + latitude + longitude + garageSpaces + lates
t_salemonth + latest_saleyear + numOfPhotos + lotSizeSqFt + livingAreaSqFt + avgSch
oolDistance + avgSchoolRating + avgSchoolSize + MedianStudentsPerTeacher + numOfBat
hrooms + numOfBedrooms + numOfStories + houseAge + property_age_category + total_am
enities + saleYear + saleMonth + season + crossCluster + hasAssociation + hasGarage
+ hasSpa + hasView
```

Modeling

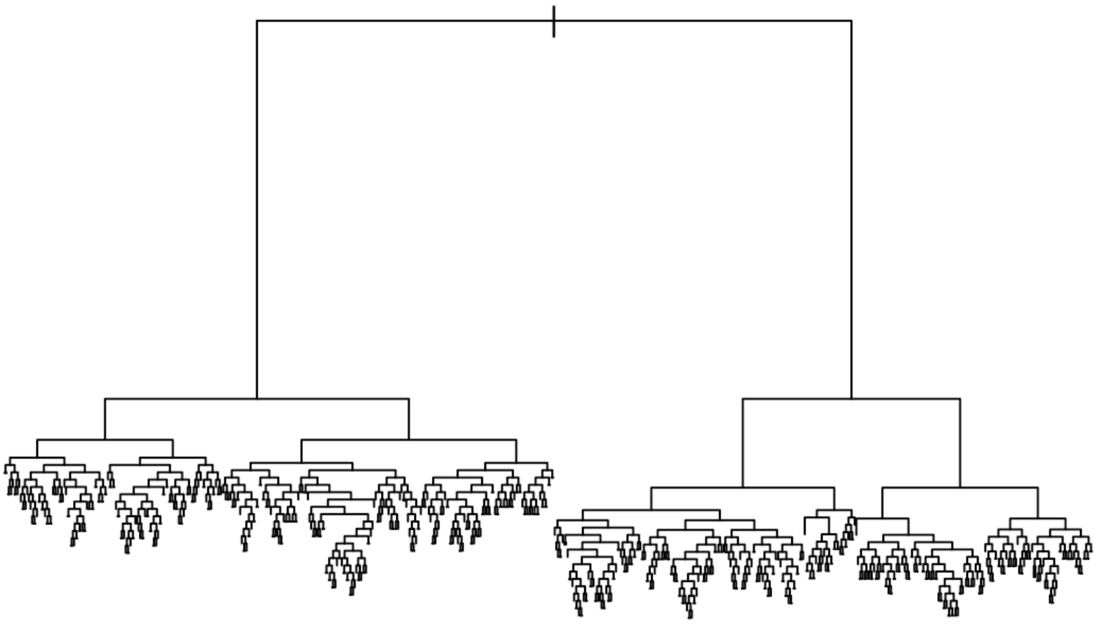
Split the Data into Training and Test Sets

```
# Split the data into training and test sets
train_ix = createDataPartition(austin_data$logLatestPrice, p = 0.8)
austin_train = austin_data[train_ix$Resample1,]
austin_test  = austin_data[-train_ix$Resample1,]
```

Regression Tree

```
# Fit a regression tree
single_big_tree_model <- rpart(
  formulaa,
  data = austin_train,
  method = 'anova',
  control = rpart.control(minsplit = 2, cp = .0001)
)

# Plot the tree
plot(single_big_tree_model)
```



```r
# Size of the big tree
nbig <- length(unique(single_big_tree_model$where))
cat('Size of Single Big Tree:', nbig, '\n')
```

```
## Size of Single Big Tree: 675
```

```r
# Predict on test data
predictions <- predict(single_big_tree_model, newdata = austin_test)

# Calculate MSE
big_mse <- mean((austin_test$latestPrice - exp(predictions))^2)
cat('MSE for Single Big Tree:', big_mse, '\n')
```

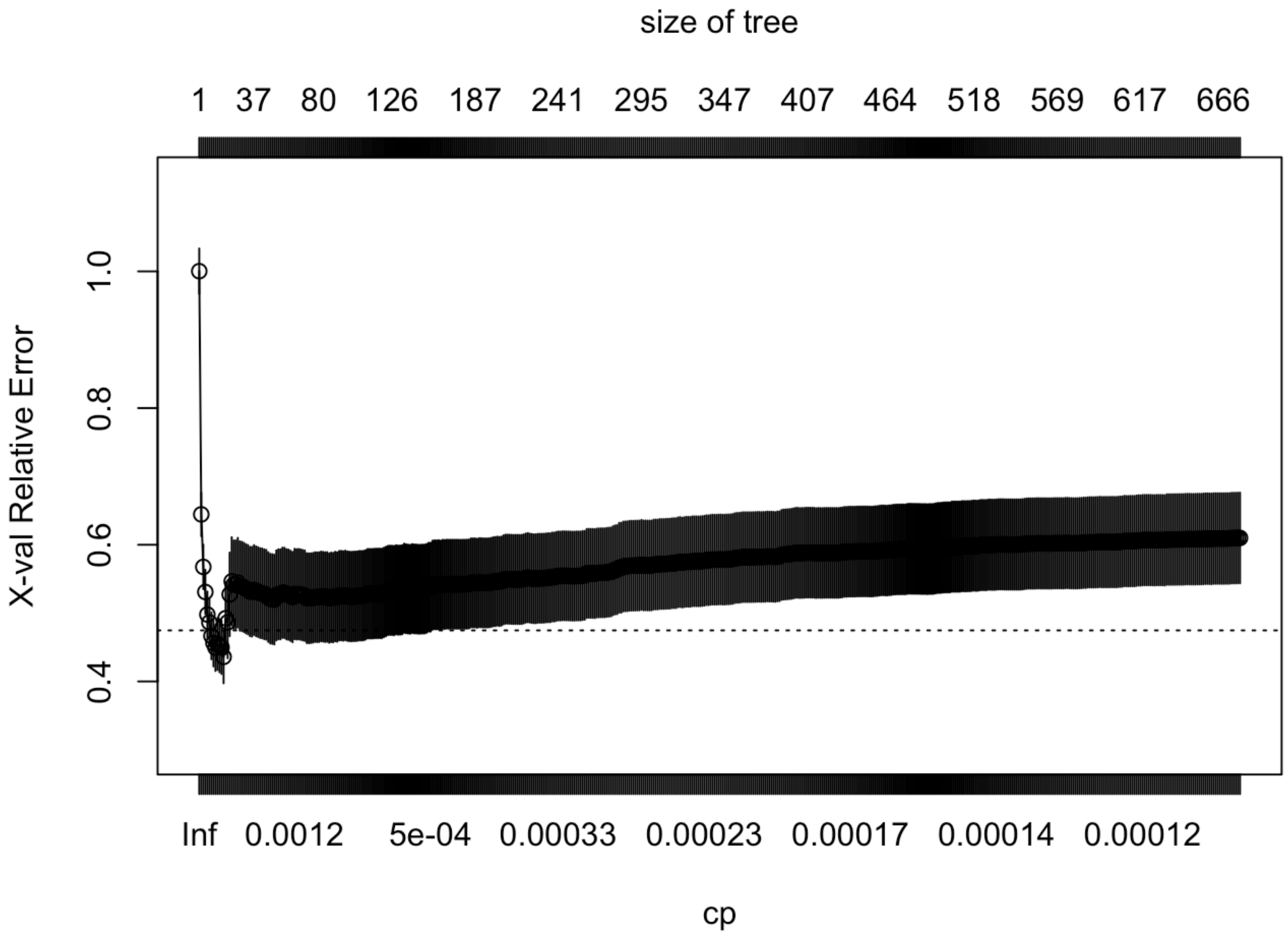
```
## MSE for Single Big Tree: 107407.4
```

```r
big_rmse <- sqrt(big_mse)
cat('RMSE for Single Big Tree:', big_rmse, '\n')
```

```
## RMSE for Single Big Tree: 327.7307
```

Pruning the Tree

```r
# Cross-validation and pruning
plotcp(single_big_tree_model) # Cross validating on cp values
```



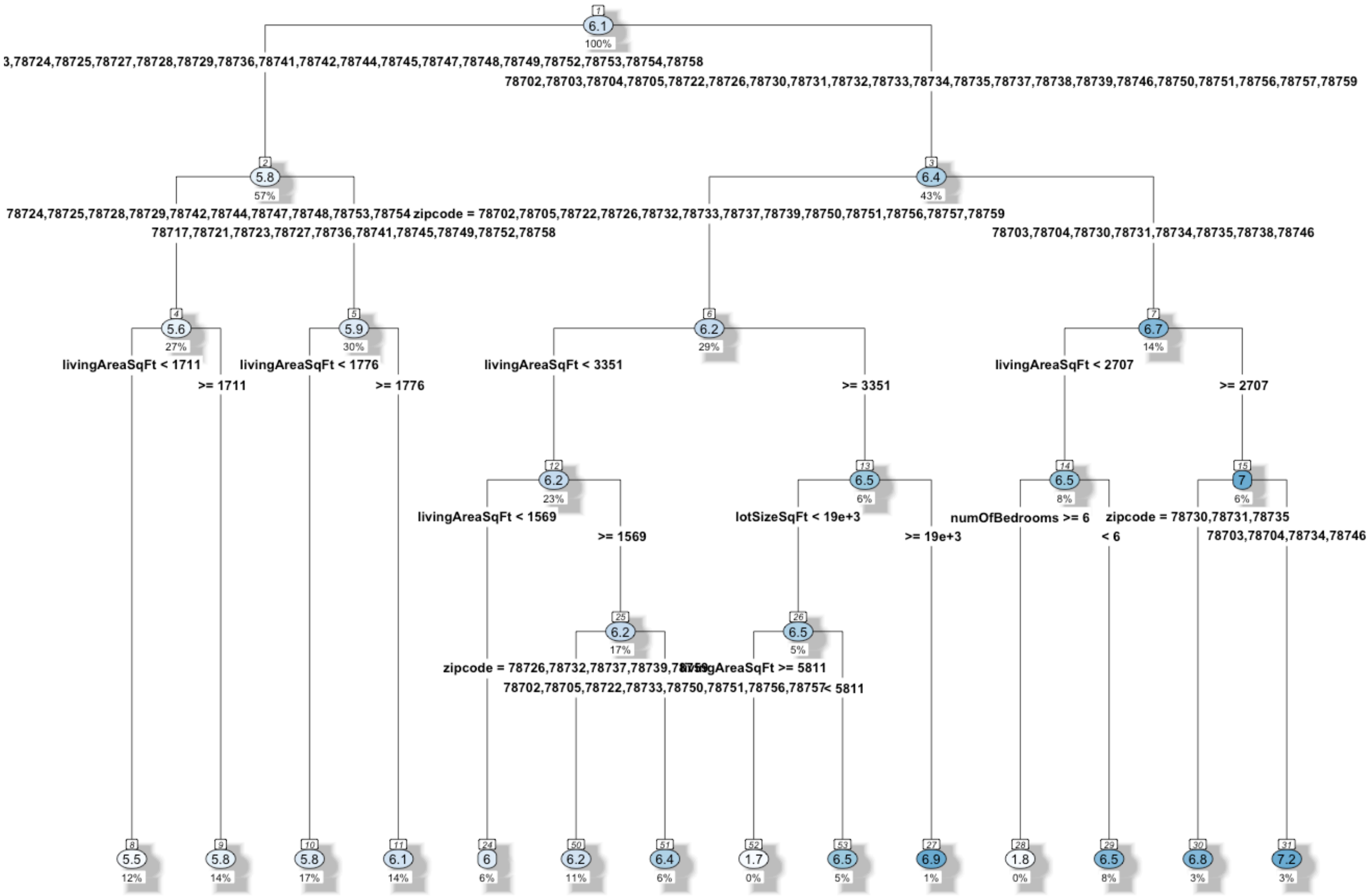

```
bestcp <- single_big_tree_model$cptable[which.min(single_big_tree_model$cptable[, '
xerror']), 'CP']
cat('Best CP:', bestcp, '\n')
```

```
## Best CP: 0.006316799
```

```
best_tree <- prune(single_big_tree_model, cp = bestcp)
rpart.plot(best_tree, type = 4, under = TRUE, faclen = 0, cex = 0.8, tweak = 0.5, b
ox.palette = "Blues", shadow.col = "gray", nn = TRUE)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

```
## Warning: cex and tweak both specified, applying both
```



```
# Size of the pruned tree
nbest <- length(unique(best_tree$where))
cat('Size of Pruned Tree:', nbest, '\n')
```

```
## Size of Pruned Tree: 14
```

```
# Predict on test data
pruned_predictions <- predict(best_tree, newdata = austin_test)

# Calculate MSE
pruned_mse <- mean((austin_test$latestPrice - exp(pruned_predictions))^2)
cat('MSE for Pruned Tree:', pruned_mse, '\n')
```

```
## MSE for Pruned Tree: 91419.22
```

```
pruned_rmse <- sqrt(pruned_mse)
cat('RMSE for Pruned Tree:', pruned_rmse, '\n')
```

```
## RMSE for Pruned Tree: 302.3561
```

Bagging with Random Forest

```
# Bagging using random forest
bagging_rf <- randomForest(
  formulaa,
  data = austin_train, mtry = 21, importance = TRUE
)

# Predictions and MSE
log_predictions_bagging <- predict(bagging_rf, newdata = austin_test)
bagging_mse <- mean((austin_test$latestPrice - exp(log_predictions_bagging))^2)
cat('MSE for Bagging:', bagging_mse, '\n')
```

```
## MSE for Bagging: 67584.75
```

```
bagging_rmse <- sqrt(bagging_mse)
cat('RMSE for Bagging:', bagging_rmse, '\n')
```

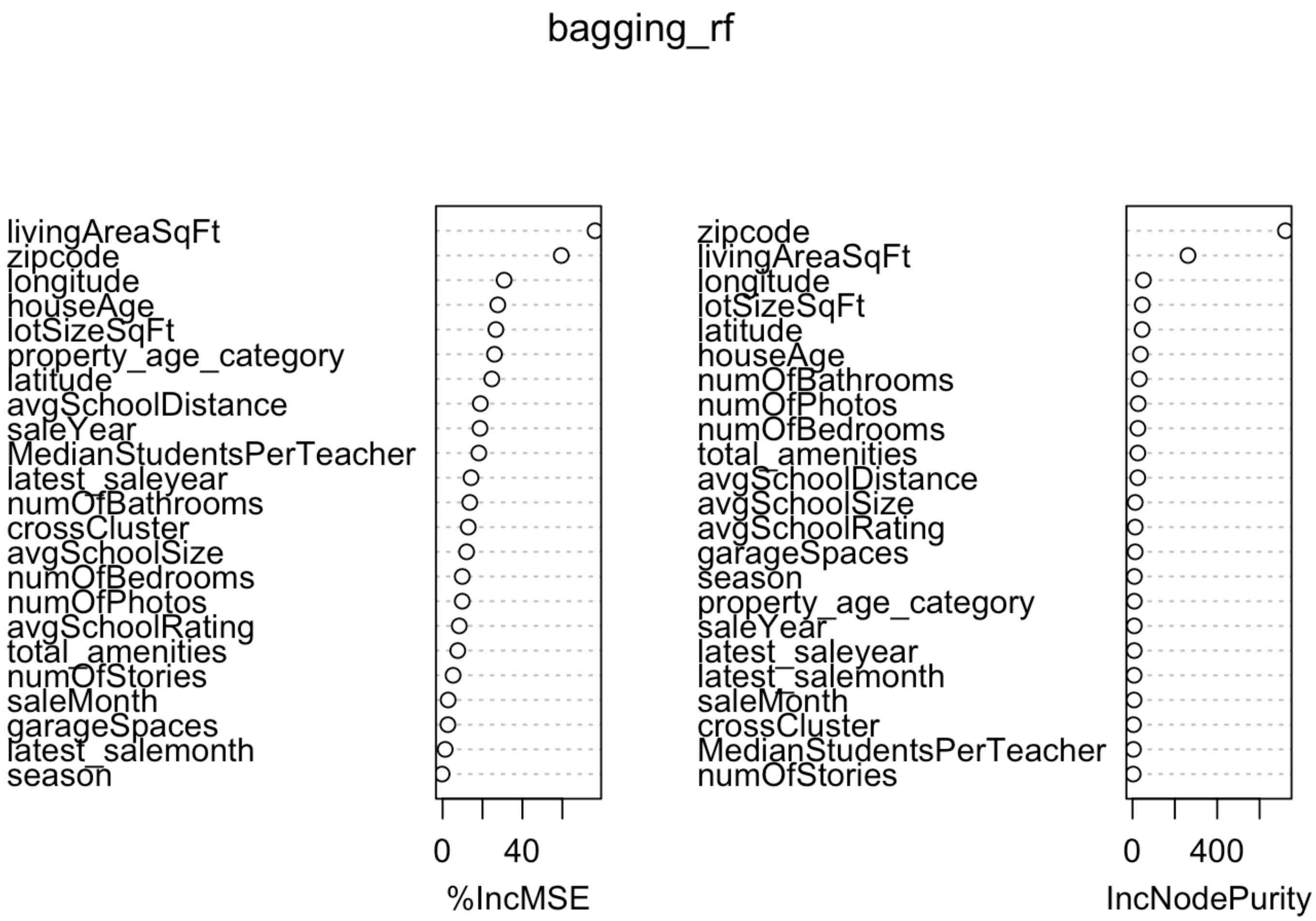
```
## RMSE for Bagging: 259.9707
```

Variable Importance

```
# Importance of variables
importance(bagging_rf)
```

```
##                           %IncMSE IncNodePurity
## zipcode                 59.4579425    721.863622
## latitude                24.6539572     45.224360
## longitude               30.8209534     51.406468
## garageSpaces             2.6446217     12.528283
## latest_salemonth         1.2854079      7.623377
## latest_saleyear         14.2354807      8.278027
## numOfPhotos              9.8484208     27.000095
## lotSizeSqFt             26.6865223     45.894889
## livingAreaSqFt          76.3805539    261.977618
## avgSchoolDistance       18.8968927     25.031676
## avgSchoolRating          8.3445745     12.642343
## avgSchoolSize           12.0213910     12.754717
## MedianStudentsPerTeacher 18.1496097     4.322041
## numOfBathrooms          13.5975283     32.255386
## numOfBedrooms            9.8494148     25.569572
## numOfStories             5.2474289      3.277383
## houseAge                27.6553167     37.505235
## property_age_category   26.0046760      8.599090
## total_amenities          7.5903165     25.168316
## saleYear                18.7278942      8.354474
## saleMonth                2.7773619      7.302067
## season                  -0.2600851      9.006943
## crossCluster            12.8480675      4.590523
```

```
varImpPlot(bagging_rf)
```



Random Forest

```
# Random forest with different mtry values
num_mtry_values <- c(4, 5, 6, 7, 8, 9)
rf_results <- matrix(NA, nrow = length(num_mtry_values), ncol = 3)
colnames(rf_results) <- c("num_mtry", "MSE", "RMSE")

for (index in 1:length(num_mtry_values)) {
  num_mtry <- num_mtry_values[index]
  rf_model <- randomForest(
    formulaa,
    data = austin_train, mtry = num_mtry, importance = TRUE
  )

  log_predictions_rf <- predict(rf_model, newdata = austin_test)
  mse_rf <- mean((austin_test$latestPrice - exp(log_predictions_rf))^2)
  rf_results[index, ] <- c(num_mtry, mse_rf, sqrt(mse_rf))
  print("---")
}
```

```
## [1] "---"
## [1] "---"
## [1] "---"
## [1] "---"
## [1] "---"
## [1] "---"
```

```
results_df <- as.data.frame(rf_results)
colnames(results_df) <- c("num_mtry", "MSE", "RMSE")
print(results_df)
```

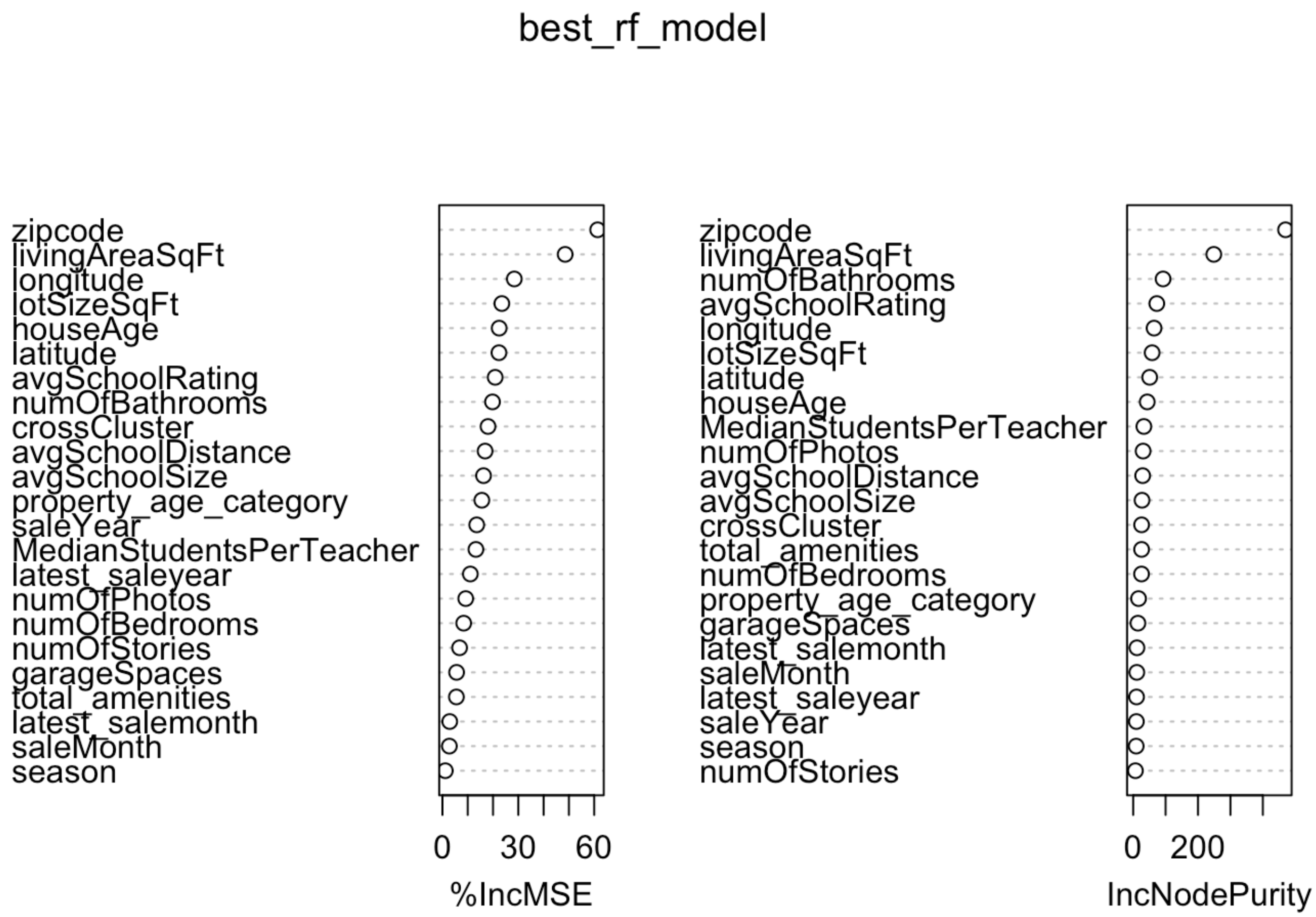
```
##   num_mtry      MSE     RMSE
## 1        4 69843.56 264.2793
## 2        5 68717.72 262.1406
## 3        6 68337.80 261.4150
## 4        7 69061.30 262.7952
## 5        8 67067.45 258.9738
## 6        9 67952.68 260.6774
```

```
# Train the final random forest model with the best mtry value
best_num_mtry <- results_df$num_mtry[which.min(results_df$MSE)]
best_rf_model <- randomForest(
  formulaa,
  data = austin_train, mtry = best_num_mtry, importance = TRUE
)

# Importance of variables
importance(best_rf_model)
```

```
##                           %IncMSE IncNodePurity
## zipcode                 61.363689    469.965024
## latitude                22.343177     50.921862
## longitude               28.406663     64.443371
## garageSpaces             5.592409     14.625774
## latest_salemonth         2.878822     11.688902
## latest_saleyear         11.028780     10.412021
## numOfPhotos              9.206919     30.621041
## lotSizeSqFt             23.481677     58.458308
## livingAreaSqFt          48.519019    248.742067
## avgSchoolDistance       16.864988     29.226117
## avgSchoolRating         20.850824     72.899016
## avgSchoolSize           16.247135     27.286574
## MedianStudentsPerTeacher 13.236078     32.987764
## numOfBathrooms          19.794244     92.233671
## numOfBedrooms            8.398072     25.748472
## numOfStories             6.776281      6.937687
## houseAge                22.431319     43.409426
## property_age_category   15.583225     16.554140
## total_amenities          5.519264     25.954282
## saleYear                13.552263      9.914701
## saleMonth                2.736114     11.264690
## season                   1.130483      9.346028
## crossCluster            18.020215     25.974648
```

```
varImpPlot(best_rf_model)
```



XGBoost Model

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
f <- c(
  "zipcode" ,"latitude" , "longitude","garageSpaces","latest_salemonth" ,"latest_sa
leyear" ,"numOfPhotos","lotSizeSqFt","livingAreaSqFt" ,
  "numOfBathrooms" ,"numOfBedrooms","numOfStories",
  "houseAge","property_age_category","total_amenities","saleYear" ,"saleMonth","sea
son","crossCluster"
)

# Prepare training and test data
#x_train <- as.data.frame(austin_train[f])
x_train <- as.data.frame(austin_train[, -which(names(austin_train) %in% c("loglates
tPrice", "latestPrice"))])
y_train <- austin_train$logLatestPrice
x_test <- as.data.frame(austin_test[, -which(names(austin_test) %in% c("loglatestPr
ice", "latestPrice"))])
y_test <- austin_test$logLatestPrice

# Convert the data to matrix format for XGBoost
X_train_matrix <- model.matrix(~., data = x_train)[, -1]
X_test_matrix <- model.matrix(~., data = x_test)[, -1]

dtrain <- xgb.DMatrix(data = X_train_matrix, label = y_train)
dtest <- xgb.DMatrix(data = X_test_matrix, label = y_test)

# Set parameters for XGBoost
params <- list(
  objective = "reg:squarederror",
  eta = 0.1,
  max_depth = 6,
  subsample = 0.7,
  colsample_bytree = 0.7
)

# Train the model
set.seed(42)
xgb_model <- xgb.train(params, dtrain, nrounds = 100)

# Make predictions on the test set
xgb_pred <- predict(xgb_model, dtest)
xgb_pred_exp <- exp(xgb_pred)

# Calculate MSE and RMSE for XGBoost
xgb_mse <- mean((exp(y_test) - xgb_pred_exp)^2)
xgb_rmse <- sqrt(xgb_mse)
cat('MSE for XGBoost:', xgb_mse, '\n')
```

```
## MSE for XGBoost: 26151.38
```

```
cat('RMSE for XGBoost:', xgb_rmse, '\n')
```

```
## RMSE for XGBoost: 161.7139
```

Bayesian Additive Regression Trees (BART)

```
# Data Preparation for BART
X_train_matrix <- model.matrix(~., data = x_train)[, -1]
X_test_matrix <- model.matrix(~., data = x_test)[, -1]

# Fit the BART model
set.seed(42)
bart_model <- wbart(x.train = X_train_matrix, y.train = y_train, x.test = X_test_ma
trix)
```

```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 5429, 85, 1355
## y1,yn: -0.061233, 0.721527
## x1,x[n*p]: 0.000000, 1.000000
## xp1,xp[np*p]: 0.000000, 0.000000
## *****Number of Trees: 200
## *****Number of Cut Points: 1 ... 1
## *****burn and ndpost: 100, 1000
## *****Prior:beta,alpha,tau,nu,lambda: 2.000000,0.950000,0.121778,3.000000,0.00000
0
## *****sigma: 0.000000
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,85,0
## *****nkeeptrain,nkeeptest,nkeeptestme,nkeeptreedraws: 1000,1000,1000,1000
## *****printevery: 100
## *****skiptr,skipte,skipteme,skiptreedraws: 1,1,1,1
##
## 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: 19s
## check counts
## trcnt,tecnt,temecnt,treedrawscnt: 1000,1000,1000,1000
```

```
# Predictions
pred <- bart_model$yhat.test.mean
yhat_bart <- exp(pred)

# Calculate MSE and RMSE for BART
y_test_exp <- exp(y_test)
bart_mse <- mean((y_test_exp - yhat_bart)^2)
bart_rmse <- sqrt(bart_mse)
cat('MSE for BART:', bart_mse, '\n')
```

```
## MSE for BART: 19995.7
```

```
cat('RMSE for BART:', bart_rmse, '\n')
```

```
## RMSE for BART: 141.4061
```

Ridge and Lasso Regression

```
library(glmnet)
```

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

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

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

```
# Preparing data for Ridge and Lasso Regression
X_train <- model.matrix(formulaa3, data = austin_train)
y_train <- austin_train$logLatestPrice
X_test <- model.matrix(formulaa3, data = austin_test)
y_test <- austin_test$logLatestPrice

# Ridge Regression
ridge_model <- cv.glmnet(X_train, y_train, alpha = 0)
ridge_pred <- predict(ridge_model, s = ridge_model$lambda.min, newx = X_test)

# Transform predictions back to original scale
ridge_pred_exp <- exp(ridge_pred)

# Calculate MSE and RMSE
ridge_mse <- mean((exp(y_test) - ridge_pred_exp)^2)
ridge_rmse <- sqrt(ridge_mse)
cat('MSE for Ridge Regression:', ridge_mse, '\n')
```

```
## MSE for Ridge Regression: 60583.55
```

```
cat('RMSE for Ridge Regression:', ridge_rmse, '\n')
```

```
## RMSE for Ridge Regression: 246.1373
```

```
# Lasso Regression
lasso_model <- cv.glmnet(X_train, y_train, alpha = 1)
lasso_pred <- predict(lasso_model, s = lasso_model$lambda.min, newx = X_test)

# Transform predictions back to original scale
lasso_pred_exp <- exp(lasso_pred)

# Calculate MSE and RMSE
lasso_mse <- mean((exp(y_test) - lasso_pred_exp)^2)
lasso_rmse <- sqrt(lasso_mse)
cat('MSE for Lasso Regression:', lasso_mse, '\n')
```

```
## MSE for Lasso Regression: 57363.87
```

```
cat('RMSE for Lasso Regression:', lasso_rmse, '\n')
```

```
## RMSE for Lasso Regression: 239.5075
```

Model Comparison and Conclusion

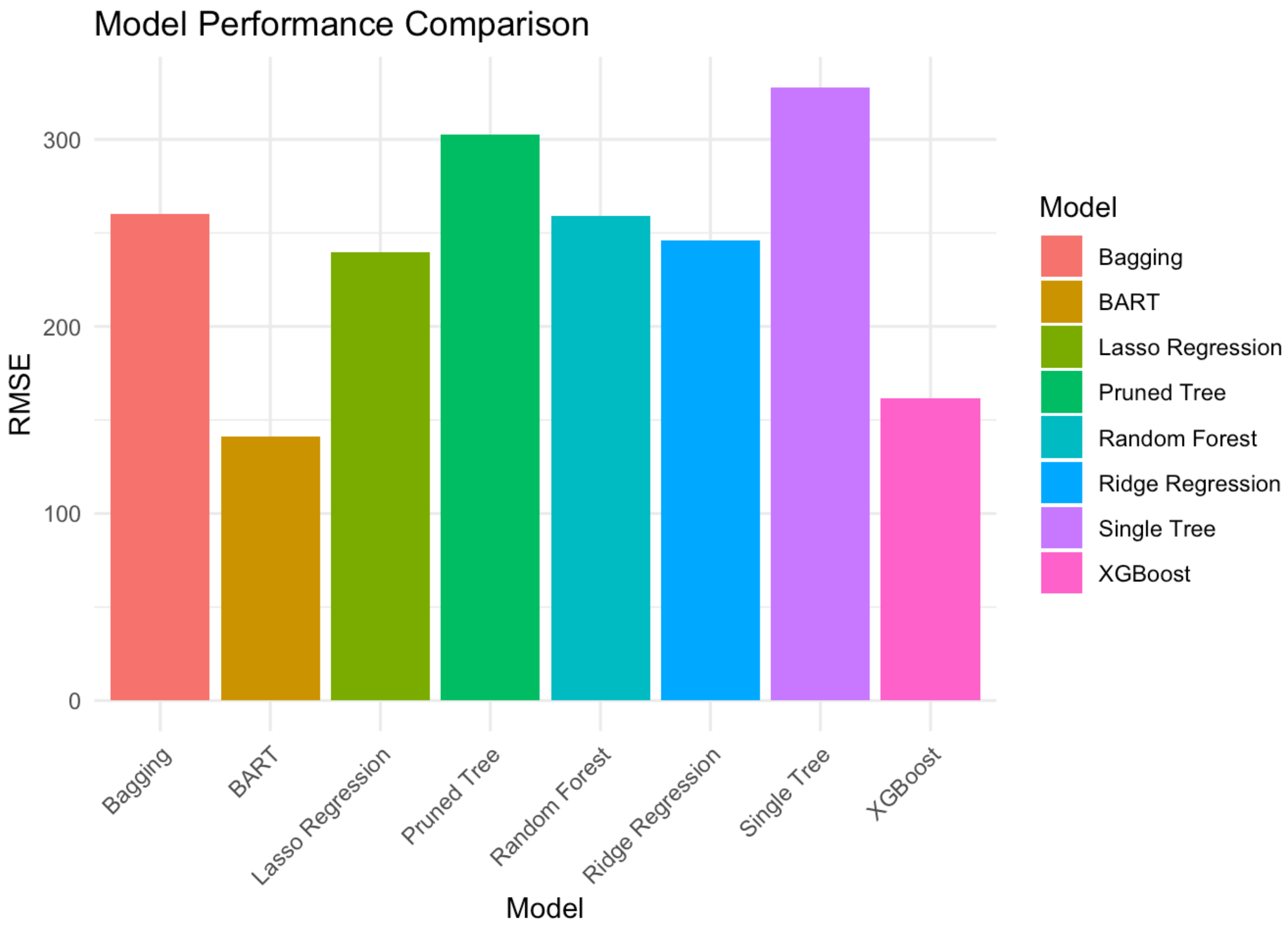
Model Performance Summary

```
# Create a summary table of the model performances
model_performance <- data.frame(
  Model = c("Single Tree", "Pruned Tree", "Bagging", "Random Forest", "XGBoost", "B
ART", "Ridge Regression", "Lasso Regression"),
  MSE = c(big_mse, pruned_mse, bagging_mse, min(results_df$MSE), xgb_mse, bart_mse,
ridge_mse, lasso_mse),
  RMSE = c(big_rmse, pruned_rmse, bagging_rmse, min(results_df$RMSE), xgb_rmse, bar
t_rmse, ridge_rmse, lasso_rmse)
)

# Print the summary table
print(model_performance)
```

```
##              Model       MSE     RMSE
## 1      Single Tree 107407.38 327.7307
## 2      Pruned Tree  91419.22 302.3561
## 3          Bagging  67584.75 259.9707
## 4    Random Forest  67067.45 258.9738
## 5          XGBoost  26151.38 161.7139
## 6             BART  19995.70 141.4061
## 7 Ridge Regression  60583.55 246.1373
## 8 Lasso Regression  57363.87 239.5075
```

```
# Plotting the model performances for visual comparison
library(ggplot2)
ggplot(model_performance, aes(x = Model, y = RMSE, fill = Model)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(title = "Model Performance Comparison", y = "RMSE", x = "Model") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Conclusion

The various models applied to the Austin housing data have shown different levels of effectiveness. Below are the key insights from each model:

Inference:

- **Single Tree**: While simple and interpretable, this model had the highest RMSE, indicating the lowest predictive accuracy among the models tested.
- **Pruned Tree**: Slightly better than the single tree, but still resulted in high RMSE, showing that even after pruning, the decision tree struggled to capture the complexity of the data.
- **Bagging**: Improved performance by reducing variance, showcasing the benefits of ensemble methods. However, it still did not outperform the more sophisticated models.
- **Random Forest**: Demonstrated strong performance by effectively reducing overfitting and capturing

more complex patterns in the data. Fine-tuning `mtry` further enhanced its accuracy.
- **XGBoost**: Achieved a significant reduction in RMSE compared to Random Forest, highlighting the effectiveness of gradient boosting in handling various data complexities.
- **BART**: Outperformed all other models, achieving the lowest RMSE, indicating its superior ability to capture complex relationships in the data.
- **Ridge Regression**: While it managed multicollinearity, its performance was moderate, indicating that linear models with regularization might not fully capture the non-linear patterns in the data.
- **Lasso Regression**: Performed better than Ridge Regression by also performing feature selection, reducing RMSE compared to Ridge, but still not matching the performance of tree-based methods.

Overall, **BART** and **XGBoost** emerged as the top performers in this analysis, with **BART** having a slight edge in terms of RMSE. These models are well-suited for capturing complex, non-linear relationships in the data. Future efforts should focus on further tuning these models and exploring additional advanced techniques to enhance prediction accuracy. **Future Work**: - Further hyperparameter tuning for models like XGBoost and Random Forest. - Exploring more advanced feature engineering techniques. - Considering additional ensemble methods like stacking to further improve predictions.

Overall, Random Forest and XGBoost are the top performers in this analysis, with Random Forest having a slight edge in terms of MSE and RMSE.

Doing the teating on Austin Holdout set

```
library(BART)
library(dplyr)

# Load the holdout dataset
holdout_data <- read.csv("austinhouses_holdout.csv")

# Adding the logLatestPrice to the original df
holdout_data <- holdout_data %>%
  select(-streetAddress, -description) %>%
  mutate(logLatestPrice = log(latestPrice))

# Derive the age of the house
current_year <- year(Sys.Date())
holdout_data <- holdout_data %>%
  mutate(houseAge = current_year - yearBuilt,
         property_age_category = cut(houseAge,
                                     breaks = c(-Inf, 10, 30, 50, Inf),
                                     labels = c("New", "Moderate", "Old", "Very Ol
d")),
         property_age_category = as.factor(property_age_category)) %>%
  select(-yearBuilt)

### Binary Columns and Amenities
# Convert binary columns to numeric and aggregate amenities
# Convert binary columns to numeric
binary_columns <- c('hasAssociation', 'hasGarage', 'hasSpa', 'hasView')
```

```
holdout_data[binary_columns] <- lapply(holdout_data[binary_columns], as.numeric)


# Create a total_amenities feature
holdout_data <- holdout_data %>%
  mutate(total_amenities = rowSums(select(., all_of(binary_columns))))
# Ensure we retain the individual binary features

### Sale Date Features
# Extract and convert sale date components
holdout_data <- holdout_data %>%
  mutate(saleDate = as.Date(latest_saledate, format = "%Y-%m-%d"),
         saleYear = year(latest_saledate),
         saleMonth = month(latest_saledate),
         saleDay = day(latest_saledate)) %>%
  mutate(season = case_when(
    saleMonth %in% c(12, 1, 2) ~ "Winter",
    saleMonth %in% c(3, 4, 5) ~ "Spring",
    saleMonth %in% c(6, 7, 8) ~ "Summer",
    saleMonth %in% c(9, 10, 11) ~ "Fall"
  )) %>%
  mutate(season = factor(season, levels = c("Winter", "Spring", "Summer", "Fall")))
%>%
  select(-latest_saledate, -saleDay, -saleDate)

### Geospatial Features
# Clustering based on latitude and longitude
set.seed(123)
coords <- holdout_data %>% select(latitude, longitude)
clusters <- kmeans(coords, centers = 5)$cluster
holdout_data <- holdout_data %>% mutate(crossCluster = as.factor(clusters))

### Calculating Correlations and Aggregating Amenities
# Calculate correlations and aggregate amenities with weights
amenity_features <- c("numOfAccessibilityFeatures", "numOfAppliances", "numOfParkin
gFeatures",
                      "numOfPatioAndPorchFeatures", "numOfSecurityFeatures", "numOf
WaterfrontFeatures",
                      "numOfWindowFeatures", "numOfCommunityFeatures")


weights <- c(numOfAccessibilityFeatures = 1, numOfAppliances = 2, numOfParkingFeatu
res = 1.5,
             numOfPatioAndPorchFeatures = 1, numOfSecurityFeatures = 1, numOfWaterf
rontFeatures = 2.5,
             numOfWindowFeatures = 1, numOfCommunityFeatures = 1.5)

holdout_data <- holdout_data %>%
  mutate(total_amenities = numOfAccessibilityFeatures * weights["numOfAccessibility
Features"] +
                          numOfAppliances * weights["numOfAppliances"] +
```

```
                              numOfParkingFeatures * weights["numOfParkingFeatures"] +
                              numOfPatioAndPorchFeatures * weights["numOfPatioAndPorchF
eatures"]+
                              numOfSecurityFeatures * weights["numOfSecurityFeatures"]
+
                              numOfWaterfrontFeatures * weights["numOfWaterfrontFeature
s"] +
                              numOfWindowFeatures * weights["numOfWindowFeatures"] +
                              numOfCommunityFeatures * weights["numOfCommunityFeature
s"]) %>%
  select(-all_of(amenity_features))

### Handling Missing Values
# Handling missing values
holdout_data[is.na(holdout_data)] <- -1
holdout_data <- holdout_data %>%
  drop_na()

# Convert categorical variables to factors
holdout_data <- holdout_data %>%
  mutate(
    zipcode = as.factor(zipcode),
    garageSpaces = factor(garageSpaces),
    homeType = factor(homeType)
  ) %>%
  select(-homeType) #removing Hometype becuase
# glimpse(austin_data)
head(holdout_data)
```

```
##   zipcode latitude longitude garageSpaces hasAssociation hasGarage hasSpa
## 1   78749 30.20016 -97.85626            0              1         0      0
## 2   78757 30.33993 -97.74895            0              0         0      0
## 3   78747 30.13613 -97.76612            1              1         1      0
## 4   78748 30.15642 -97.81450            2              0         1      0
## 5   78729 30.46093 -97.77524            0              0         0      0
## 6   78729 30.44455 -97.76396            2              0         1      0
##   hasView latestPrice latest_salemonth latest_saleyear numOfPhotos lotSizeSqFt
## 1       0          -1                8            2018          37        9888
## 2       0          -1                2            2019          35       10715
## 3       0          -1                3            2020          28        6359
## 4       0          -1                7            2020          63        5009
## 5       0          -1                2            2019          40        7230
## 6       0          -1               10            2018          58        6838
##   livingAreaSqFt avgSchoolDistance avgSchoolRating avgSchoolSize
## 1           2023         1.3333333        6.666667          1460
## 2           1806         0.7333333        6.666667          1153
## 3           2314         2.4333333        5.333333          1506
## 4           1891         1.0000000        3.333333          1424
## 5           2311         1.2333333        5.333333          1369
## 6           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 houseAge property_age_category total_amenities saleYear
## 1             -1       25              Moderate            11.5     2018
## 2             -1       62              Very Old             1.5     2019
## 3             -1        9                   New             9.0     2020
## 4             -1       37                   Old            21.0     2020
## 5             -1       35                   Old             7.0     2019
## 6             -1       34                   Old            10.5     2018
##   saleMonth season crossCluster
## 1         8 Summer            4
## 2         2 Winter            1
## 3         3 Spring            5
## 4         7 Summer            5
## 5         2 Winter            2
## 6        10   Fall            2
```

```
# Prepare training and test data
x_train <- as.data.frame(austin_data[f])
#x_train <- as.data.frame(austin_data[, -which(names(austin_data) %in% c("loglatest
Price", "latestPrice"))])
y_train <- austin_data$logLatestPrice
x_test <- as.data.frame(holdout_data[f])
#x_test <- as.data.frame(holdout_data[, -which(names(holdout_data) %in% c("loglates
tPrice", "latestPrice"))])
y_test <- holdout_data$logLatestPrice

# Convert the data to matrix format for XGBoost
X_train_matrix <- model.matrix(~., data = x_train)[, -1]
X_test_matrix <- model.matrix(~., data = x_test)[, -1]

dtrain <- xgb.DMatrix(data = X_train_matrix, label = y_train)
dtest <- xgb.DMatrix(data = X_test_matrix, label = y_test)

# Data Preparation for BART
X_train_matrix <- model.matrix(~., data = x_train)[, -1]
X_test_matrix <- model.matrix(~., data = x_test)[, -1]

# Fit the BART model
set.seed(42)
bart_model <- wbart(x.train = X_train_matrix, y.train = y_train, x.test = X_test_ma
trix)
```

```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 6784, 76, 6785
## y1,yn: -0.062580, 0.720179
## x1,x[n*p]: 0.000000, 1.000000
## xp1,xp[np*p]: 0.000000, 0.000000
## *****Number of Trees: 200
## *****Number of Cut Points: 1 ... 1
## *****burn and ndpost: 100, 1000
## *****Prior:beta,alpha,tau,nu,lambda: 2.000000,0.950000,0.130101,3.000000,0.01428
8
## *****sigma: 0.270834
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,76,0
## *****nkeeptrain,nkeeptest,nkeeptestme,nkeeptreedraws: 1000,1000,1000,1000
## *****printevery: 100
## *****skiptr,skipte,skipteme,skiptreedraws: 1,1,1,1
##
## 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: 27s
## check counts
## trcnt,tecnt,temecnt,treedrawscnt: 1000,1000,1000,1000
```

```
# Predictions
pred <- bart_model$yhat.test.mean
holdout_data$latestPrice <- exp(pred)
```

```
# Export the updated holdout dataset
write.csv(holdout_data, "austinhouses_holdout_predictions.csv", row.names = FALSE)
```