HARVARD EXTENSION SCHOOL

CSCI E-106 - Data Modeling - Final Project

Kaleo Mungin, Luciano Carvalho, Ibrahim Hashim,
Bethun Bhowmik, Mohanish Kashiwar, Seymur Hasanov

15 December 2023

Abstract

This project, undertaken by a team of students at Harvard Extension School, focuses on developing a comprehensive predictive model for house prices in King County, USA, using the 'KC_House_Sales' dataset. The dataset provides a rich variety of house attributes, allowing for an in-depth analysis of factors influencing property values. The initial phase of the project involves data cleaning and transformation, including the removal of irrelevant variables and conversion of data types. The primary analytical approach combines traditional linear regression models with more complex methods such as neural networks, decision trees, and support vector machines (SVM). The models are trained on a 70% split of the dataset and validated on the remaining 30%, ensuring robustness and accuracy in predictions.In the subsequent stages, the project delves into graphical analysis, revealing key correlations and trends in the data. Techniques such as box plots, scatter plots, and heatmaps provide insights into the relationships between house prices and attributes like square footage, number of bedrooms, and location. The project also explores the impact of categorical variables and property age on pricing. Model performance is thoroughly tested using various metrics, including MSE and R-squared values. The final selection of the primary model, referred to as the "champion" model, is based on its performance on both the training and testing datasets. The project concludes with a discussion on model limitations, assumptions, and an ongoing monitoring plan, ensuring the model's relevance and accuracy over time.

Contents

Executive Summary

This section will describe the model usage, your conclusions and any regulatory and internal requirements. In a real world scenario, this section is for senior management who do not need to know the details. They need to know high level (the purpose of the model, limitations of the model and any issues).

I. Introduction (5 points)

This section needs to introduce the reader to the problem to be resolved, the purpose, and the scope of the statistical testing applied. What you are doing with your prediction? What is the purpose of the model? What methods were trained on the data, how large is the test sample, and how did you build the model?

In the ever-evolving landscape of real estate, the ability to accurately predict house prices is invaluable. This project, undertaken by a dedicated team from Harvard Extension School, delves into this realm, focusing on King County, USA. The motivation behind our work is twofold: firstly, to provide a robust predictive tool for potential investors and market analysts, and secondly, to contribute to the academic understanding of real estate market dynamics.

Our approach is grounded in the analysis of the 'KC_House_Sales' dataset, a comprehensive collection of house attributes within King County. The dataset, rich in detail, includes features such as square footage, the number of bedrooms, and geographical location, etc., all of which are pivotal in determining house prices. The initial phase of our project involved a thorough data cleaning process. This step was crucial in ensuring the integrity of our analysis, involving the removal of irrelevant variables, handling missing values, and converting data types for consistency.

Once the data was prepared, we embarked on a methodological journey, exploring various statistical and machine learning techniques. Our primary method, linear regression, served as a foundation model, offering insights into the direct relationships between house features and their prices. However, recognizing the complexity of the real estate market, we expanded our toolkit to include more advanced models like neural networks, decision trees, and support vector machines (SVM). Each of these methods brought a unique perspective and depth to our analysis, enabling us to capture non-linear relationships and complex interactions between variables.

The structure of our project involved splitting the dataset into two parts: 70% for training and 30% for validation. This split was strategically chosen to ensure the robustness of our models against unseen data, a critical aspect of predictive modeling. The training phase was an iterative process, where each model was refined through a series of evaluations and adjustments. In doing so, we aimed to strike a balance between model complexity and predictive accuracy.

Our exploratory data analysis revealed several key insights. We observed that certain features, such as square footage and location, had a significant impact on house prices. This initial observation guided our feature selection and engineering process, where we developed new variables that could potentially enhance the model's performance.

As we progressed, the need for a rigorous evaluation framework became apparent. To this end, we employed various metrics such as MSE, R-squared and Pseudo-R squared values. These metrics were instrumental in assessing the performance of our models, both on the training and validation sets. The final selection of our primary model, which we refer to as the "champion" model, was based on a comprehensive evaluation of these metrics.

In conclusion, this project represents a significant endeavor in the field of predictive analytics for real estate. Through a blend of traditional statistical methods and advanced machine learning techniques, we have developed a model that not only predicts house prices in King County with a high degree of accuracy but also offers insights into the factors that drive these prices. As we move forward, our focus will be on refining the model, exploring new data sources, and adapting to the changing dynamics of the real estate market.

II. Description of the data and quality (15 points)

Here you need to review your data, the statistical test applied to understand the predictors and the response and how are they correlated. Extensive graph analysis is recommended. Is the data continuous, or categorical, do any transformation needed? Do you need dummies?

Data Overview

The dataset presented for analysis encompasses a range of housing attributes for properties sold in King County, including the sale price, number of bedrooms and bathrooms, square footage of living and lot space, and other characteristics like the presence of a waterfront, views, and the condition and grade of the house. Each entry is timestamped, providing a date of sale, which can be instrumental in understanding market trends over time.

```
df_house = read.csv("KC_House_Sales.csv")
cat("Number of NA values in dataframe:",sum(is.na(df_house)))
```

```
## Number of NA values in dataframe: 0
```

```
head(df_house)
```

```
##           id            date           price bedrooms bathrooms sqft_living
## 1 7129300520 20141013T000000    $221,900.00         3      1.00        1180
## 2 6414100192 20141209T000000    $538,000.00         3      2.25        2570
## 3 5631500400 20150225T000000    $180,000.00         2      1.00         770
## 4 2487200875 20141209T000000    $604,000.00         4      3.00        1960
## 5 1954400510 20150218T000000    $510,000.00         3      2.00        1680
## 6 7237550310 20140512T000000  $1,225,000.00         4      4.50        5420
##   sqft_lot floors waterfront view condition grade sqft_above sqft_basement
## 1     5650      1          0    0         3     7       1180             0
## 2     7242      2          0    0         3     7       2170           400
## 3    10000      1          0    0         3     6        770             0
## 4     5000      1          0    0         5     7       1050           910
## 5     8080      1          0    0         3     8       1680             0
## 6   101930      1          0    0         3    11       3890          1530
##   yr_built yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 1     1955            0   98178 47.5112 -122.257          1340       5650
## 2     1951         1991   98125 47.7210 -122.319          1690       7639
## 3     1933            0   98028 47.7379 -122.233          2720       8062
## 4     1965            0   98136 47.5208 -122.393          1360       5000
## 5     1987            0   98074 47.6168 -122.045          1800       7503
## 6     2001            0   98053 47.6561 -122.005          4760     101930
```

Data Types, Categories and Cleaning

Our dataset includes a blend of continuous and categorical data types. Notably, the 'zipcode' and 'waterfront' variables are categorical despite their numeric appearance. 'Zipcode' represents different regions, and 'waterfront' is a binary indicator, thus requiring special attention during preprocessing. These variables, along with others like 'view' and 'condition', will be transformed into dummy variables to facilitate their use in our regression models.

The data cleaning process has involved removing non-informative variables such as IDs, correcting data types (e.g., transforming sale price to a numeric format and parsing dates into a usable format), and creating new variables that could reveal temporal trends (e.g., year, month, day of sale).

```
# [Kaleo]
#removing `id` column
df_house = subset(df_house, select = -id)

#converting `price` to numeric
df_house$price <- as.numeric(gsub("[\\$,]", "", df_house$price))

##cleaning `date` and adding `year`,`month`,`day` columns
df_house$date <- as.POSIXct(df_house$date, format = "%Y%m%d")
df_house$year <- as.numeric(format(df_house$date, "%Y"))
```

```r
df_house$month <- as.numeric(format(df_house$date, "%m"))
df_house$day <- as.numeric(format(df_house$date, "%d"))

ndf_house = df_house[sapply(df_house, is.numeric)]

head(df_house)
```

```
##         date   price bedrooms bathrooms sqft_living sqft_lot floors waterfront
## 1 2014-10-13  221900        3      1.00        1180     5650      1          0
## 2 2014-12-09  538000        3      2.25        2570     7242      2          0
## 3 2015-02-25  180000        2      1.00         770    10000      1          0
## 4 2014-12-09  604000        4      3.00        1960     5000      1          0
## 5 2015-02-18  510000        3      2.00        1680     8080      1          0
## 6 2014-05-12 1225000        4      4.50        5420   101930      1          0
##   view condition grade sqft_above sqft_basement yr_built yr_renovated zipcode
## 1    0         3     7       1180             0     1955            0   98178
## 2    0         3     7       2170           400     1951         1991   98125
## 3    0         3     6        770             0     1933            0   98028
## 4    0         5     7       1050           910     1965            0   98136
## 5    0         3     8       1680             0     1987            0   98074
## 6    0         3    11       3890          1530     2001            0   98053
##       lat     long sqft_living15 sqft_lot15 year month day
## 1 47.5112 -122.257          1340       5650 2014    10  13
## 2 47.7210 -122.319          1690       7639 2014    12   9
## 3 47.7379 -122.233          2720       8062 2015     2  25
## 4 47.5208 -122.393          1360       5000 2014    12   9
## 5 47.6168 -122.045          1800       7503 2015     2  18
## 6 47.6561 -122.005          4760     101930 2014     5  12
```

Categorical Variables and Age Analysis

Renovation Indicator Variable

A binary variable named 'renovated' was introduced to indicate whether a property has undergone renovation. This variable is set to 1 if the 'yr_renovated' field is not zero, signifying that the property has been renovated at least once. Otherwise, it is set to 0, indicating no renovation. This distinction provides a straightforward way to assess the impact of renovations on property values.

```r
# Creating a new variable 'renovated'
# 1 if the property has been renovated (yr_renovated != 0), 0 otherwise
df_house$renovated = ifelse(df_house$yr_renovated != 0, 1, 0)
```

Property Age Calculation

This is a tentative way to consider "age since las renovation" and see if there's a correlation between the time a house was last built/renovated on its selling price. A new variable called 'age' was calculated to represent the current age of each property. If a property was renovated, its age is the difference between 2023 and the renovation year ('yr_renovated'). If not renovated, the age is the difference between 2023 and the year the house was built ('yr_built'). This variable helps in understanding the effect of property age and recent renovations on house prices.

```r
# Creating 'age' column
df_house$age = ifelse(df_house$renovated == 1, 2023 - df_house$yr_renovated, 2023 - df_house$yr_built)

head(df_house)
```

```
##         date   price bedrooms bathrooms sqft_living sqft_lot floors waterfront
## 1 2014-10-13  221900        3      1.00        1180     5650      1          0
## 2 2014-12-09  538000        3      2.25        2570     7242      2          0
## 3 2015-02-25  180000        2      1.00         770    10000      1          0
## 4 2014-12-09  604000        4      3.00        1960     5000      1          0
## 5 2015-02-18  510000        3      2.00        1680     8080      1          0
## 6 2014-05-12 1225000        4      4.50        5420   101930      1          0
```

```
##   view condition grade sqft_above sqft_basement yr_built yr_renovated zipcode
## 1    0         3     7       1180             0     1955            0   98178
## 2    0         3     7       2170           400     1951         1991   98125
## 3    0         3     6        770             0     1933            0   98028
## 4    0         5     7       1050           910     1965            0   98136
## 5    0         3     8       1680             0     1987            0   98074
## 6    0         3    11       3890          1530     2001            0   98053
##       lat     long sqft_living15 sqft_lot15 year month day renovated age
## 1 47.5112 -122.257          1340       5650 2014    10  13         0  68
## 2 47.7210 -122.319          1690       7639 2014    12   9         1  32
## 3 47.7379 -122.233          2720       8062 2015     2  25         0  90
## 4 47.5208 -122.393          1360       5000 2014    12   9         0  58
## 5 47.6168 -122.045          1800       7503 2015     2  18         0  36
## 6 47.6561 -122.005          4760     101930 2014     5  12         0  22
```

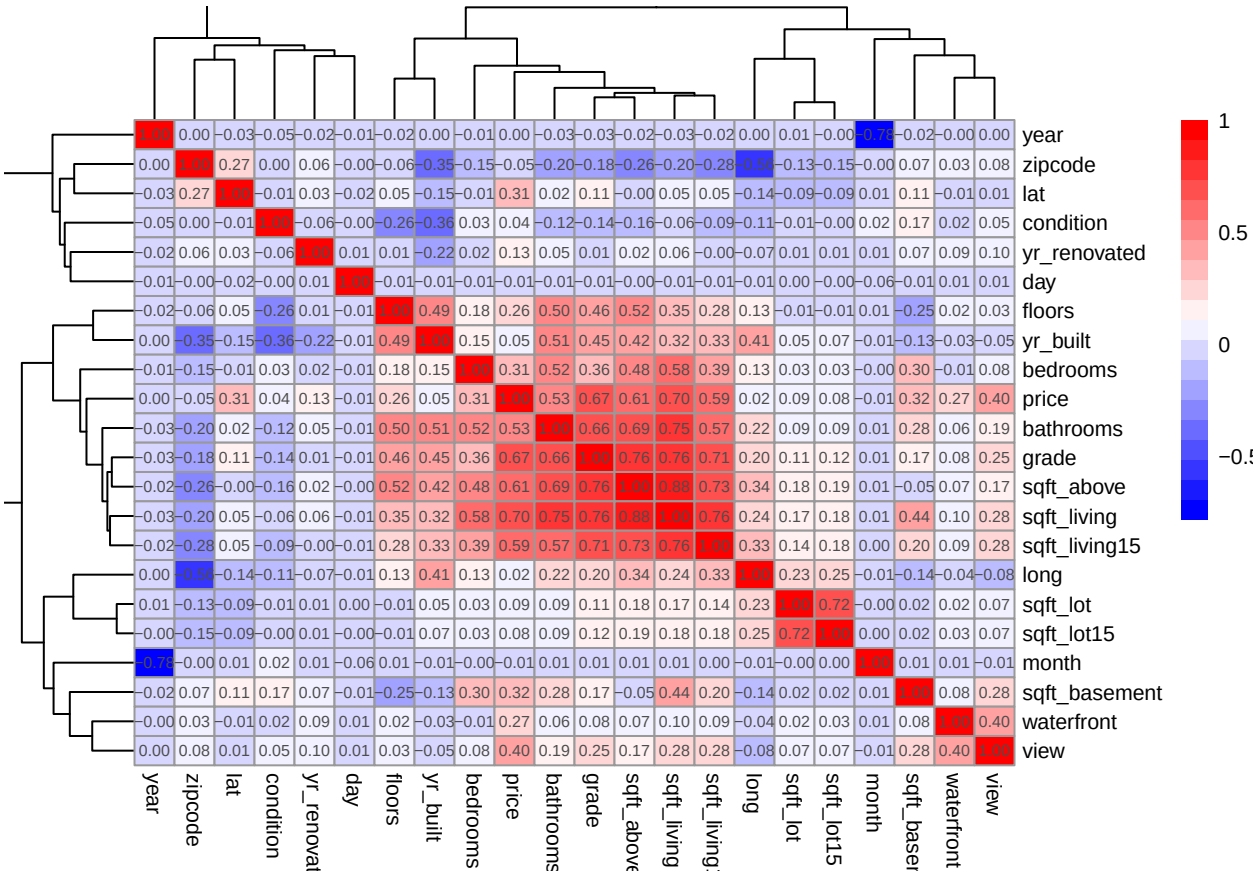
```
df_house$zipcode = as.character(df_house$zipcode)
```

Data Cleaning - handling unnecessary and highly correlated variables

Correlation Analysis

Analysis: the heatmap shows the correlation between different variables. The heatmap indicates that 'sqft_living', 'grade', and 'sqft_above' are highly correlated with 'price'. This suggests that larger living spaces, higher house grades, and more above-ground square footage are associated with higher house prices. These variables could serve as key predictors in a pricing model. We will explore them further int he next model development process.

```
# [Kaleo]
#correlation matrix
cor_matrix = cor(ndf_house)
correlation_df = as.data.frame(cor_matrix)

#new dataframe with variables 'highly' (>0.2) correlated with price
x_high = subset(ndf_house, select = c(view,waterfront,sqft_basement,sqft_living,
                                      sqft_living15,sqft_above,grade,bathrooms,bedrooms,floors,lat))

pheatmap(cor_matrix,
         color = colorRampPalette(c("blue", "white", "red"))(20),
         main = "correlation matrix heatmap",
         fontsize = 8,
         cellwidth = 15,
         cellheight = 11,
         display_numbers = TRUE
)
```




```
# [Seymur]

df_house <- df_house[sapply(df_house, is.numeric)]

corr_matrix <- cor(df_house, use = "complete.obs")  # use 'complete.obs' to handle missing values

threshold <- 0.8
high_corr <- which(abs(corr_matrix) > threshold, arr.ind = TRUE)
high_corr <- high_corr[high_corr[, 1] < high_corr[, 2], ]

for (pair in 1:nrow(high_corr)) {
    row <- high_corr[pair, "row"]
    col <- high_corr[pair, "col"]
    cat(names(df_house)[row], "and", names(df_house)[col], "have a correlation of", corr_matrix[row, col], "\n
}
```

```
## sqft_living and sqft_above have a correlation of 0.8765966
## yr_renovated and renovated have a correlation of 0.9999685
## yr_built and age have a correlation of -0.9099238
```

```
# [Seymur]

# Remove redundant, unnecessary columns from dataset.
df_house[c("sqft_living", "yr_renovated", "yr_built", "month", "day")] <- list(NULL)
```

Initial Statistical Data Summary

The dataset from King County includes 21,613 observations with 22 variables related to house sales. Variables include continuous data like price, square footage, and lat/long coordinates, and categorical data such as bedrooms, floors, and waterfront status. The price ranges from $75,000 to $7,700,000, with a mean of $540,088. Houses range from 0 to 33 bedrooms, reflecting diverse property types. The dataset also contains binary and ordinal variables, such as view and condition, that require dummy coding for analysis.

```
str(df_house)
```

```
## 'data.frame':    21613 obs. of  18 variables:
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
##  $ year         : num  2014 2014 2015 2014 2015 ...
##  $ renovated    : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ age          : num  68 32 90 58 36 22 28 60 63 20 ...
```

```
summary(df_house)
```

```
##      price            bedrooms        bathrooms        sqft_lot
##  Min.   :  75000   Min.   : 0.000   Min.   :0.000   Min.   :    520
##  1st Qu.: 321950   1st Qu.: 3.000   1st Qu.:1.750   1st Qu.:   5040
##  Median : 450000   Median : 3.000   Median :2.250   Median :   7618
##  Mean   : 540088   Mean   : 3.371   Mean   :2.115   Mean   :  15107
##  3rd Qu.: 645000   3rd Qu.: 4.000   3rd Qu.:2.500   3rd Qu.:  10688
##  Max.   :7700000   Max.   :33.000   Max.   :8.000   Max.   :1651359
##      floors        waterfront            view          condition
##  Min.   :1.000   Min.   :0.000000   Min.   :0.0000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:3.000
##  Median :1.500   Median :0.000000   Median :0.0000   Median :3.000
##  Mean   :1.494   Mean   :0.007542   Mean   :0.2343   Mean   :3.409
##  3rd Qu.:2.000   3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:4.000
##  Max.   :3.500   Max.   :1.000000   Max.   :4.0000   Max.   :5.000
##      grade          sqft_above   sqft_basement         lat
##  Min.   : 1.000   Min.   : 290   Min.   :   0.0   Min.   :47.16
##  1st Qu.: 7.000   1st Qu.:1190   1st Qu.:   0.0   1st Qu.:47.47
##  Median : 7.000   Median :1560   Median :   0.0   Median :47.57
##  Mean   : 7.657   Mean   :1788   Mean   : 291.5   Mean   :47.56
##  3rd Qu.: 8.000   3rd Qu.:2210   3rd Qu.: 560.0   3rd Qu.:47.68
##  Max.   :13.000   Max.   :9410   Max.   :4820.0   Max.   :47.78
##       long        sqft_living15    sqft_lot15          year
##  Min.   :-122.5   Min.   : 399   Min.   :   651   Min.   :2014
##  1st Qu.:-122.3   1st Qu.:1490   1st Qu.:  5100   1st Qu.:2014
##  Median :-122.2   Median :1840   Median :  7620   Median :2014
##  Mean   :-122.2   Mean   :1987   Mean   : 12768   Mean   :2014
##  3rd Qu.:-122.1   3rd Qu.:2360   3rd Qu.: 10083   3rd Qu.:2015
##  Max.   :-121.3   Max.   :6210   Max.   :871200   Max.   :2015
##    renovated            age
##  Min.   :0.00000   Min.   :  8.00
##  1st Qu.:0.00000   1st Qu.: 24.00
##  Median :0.00000   Median : 46.00
##  Mean   :0.04229   Mean   : 49.61
##  3rd Qu.:0.00000   3rd Qu.: 69.00
##  Max.   :1.00000   Max.   :123.00
```

```r
summary(df_house$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   75000  321950  450000  540088  645000 7700000
```

Graphical Analysis

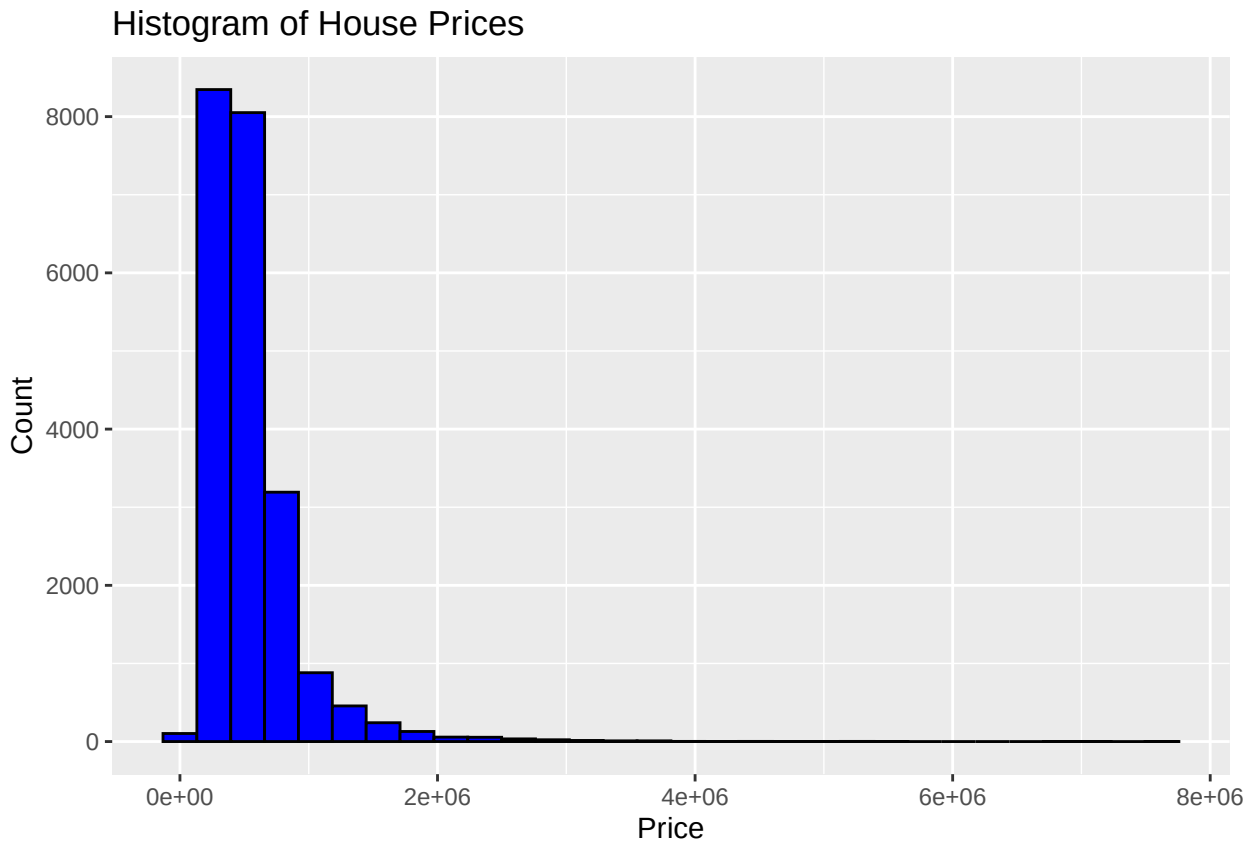
The pairwise scatter plot shows the correlation between highly correlated variables extracted from heatmap correlation matrix.

- there is a strong correlation between sqft_living and price, indicated that as the living area increases, so does the house price increase.
- A strong positive correlation with grade, implying that higher-quality houses tend to be more expensive.
- Each variable's distribution is shown on the diagonal, with price notably skewed towards lower values, indicating most homes are on the more affordable end of the spectrum with fewer high-priced outliers.

```r
# Pairwise scatter plot with correlation coefficients
ggpairs(ndf_house, columns = c("price", "sqft_living", "bedrooms", "bathrooms", "grade"),
        title = "Pairwise Scatter Plots with House Price")
```

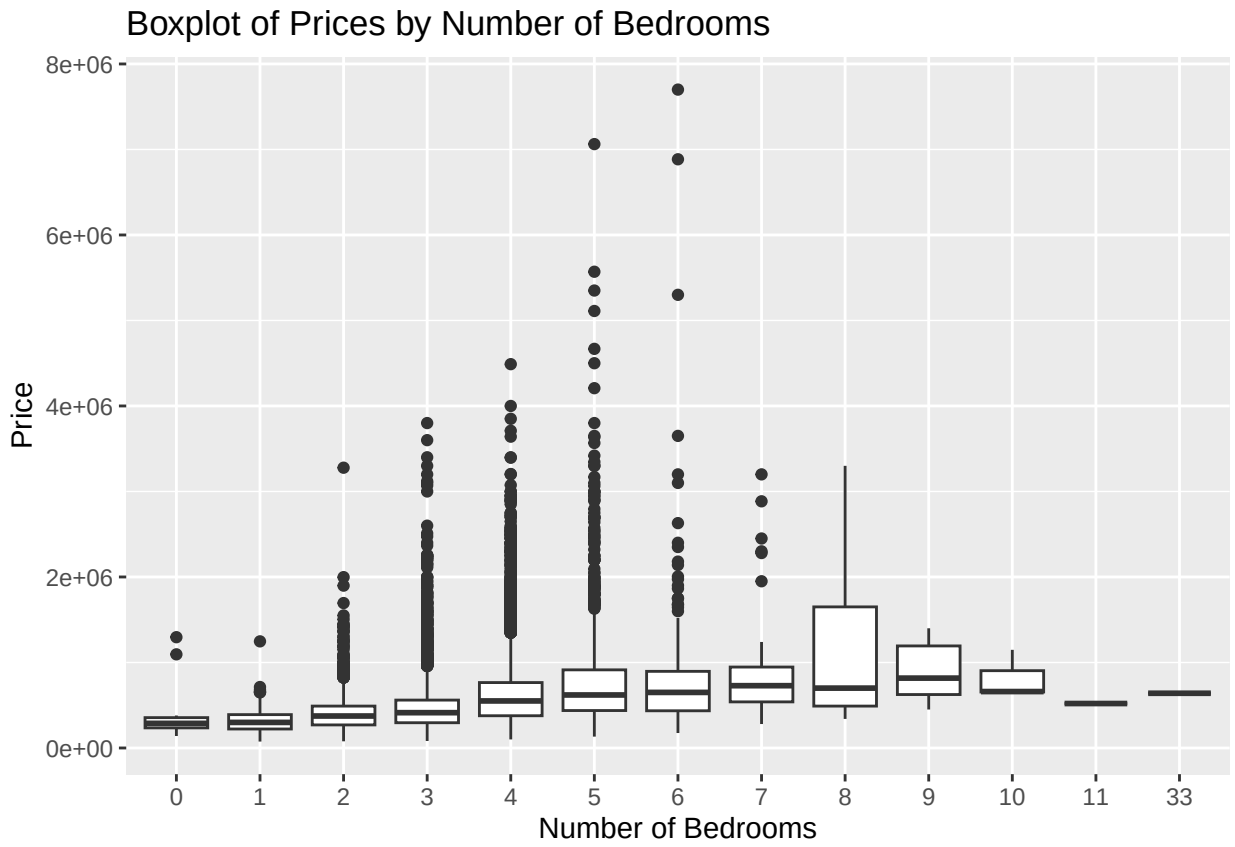
Analysis: The histogram below shows the distribution of house prices, revealing that most houses are in the lower price range with a significant decrease in the number of houses as the price increases, indicating a right-skewed distribution with relatively few high-priced houses.

```r
# Histogram of house prices
ggplot(df_house, aes(x = price)) +
  geom_histogram(bins = 30, fill = "blue", color = "black") +
  labs(title = "Histogram of House Prices", x = "Price", y = "Count")
```

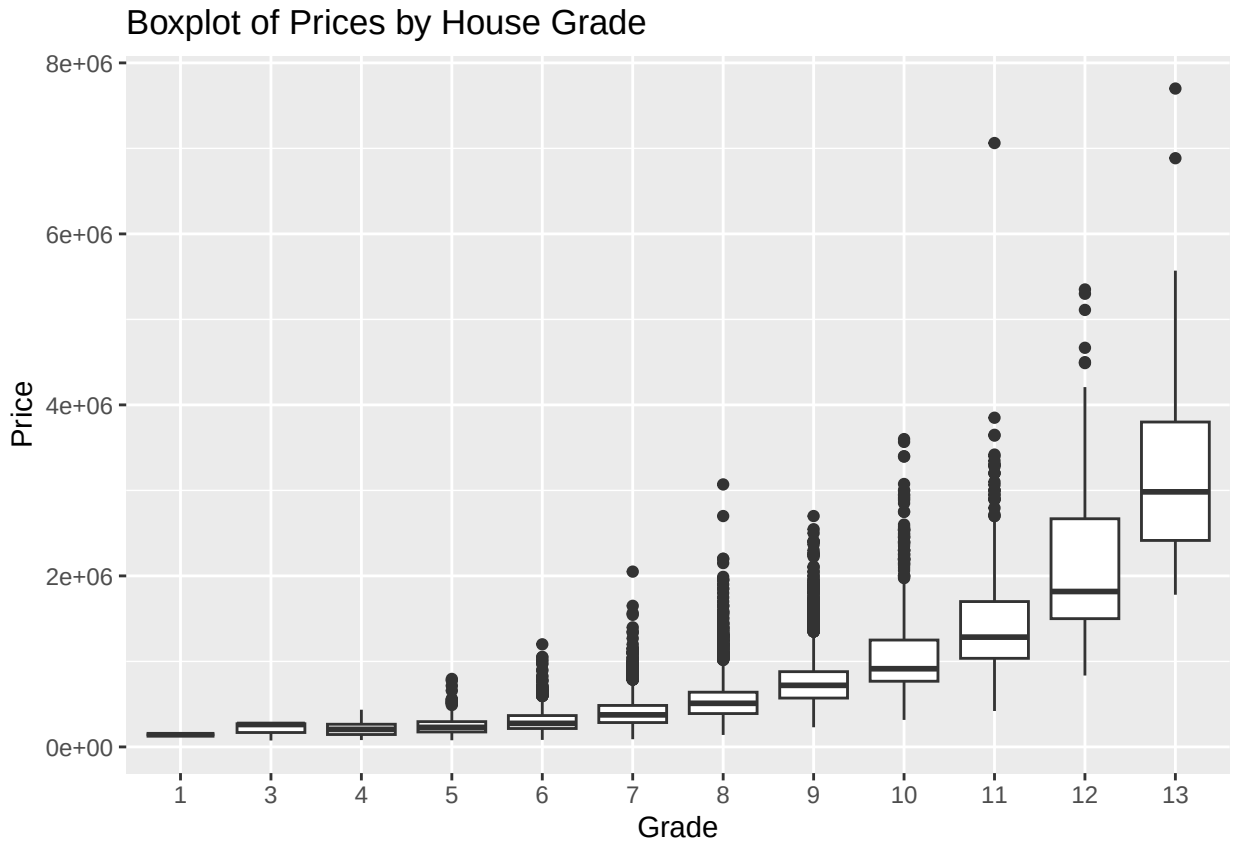


Analysis: The first boxplot shows that house prices generally increase with the number of bedrooms, but vary widely, especially for homes with more bedrooms. The second boxplot indicates that house prices rise with better house grades, with greater price variability at higher grades.Both plots demonstrate that while there is a general trend of increasing price with more bedrooms and higher grades, there is also a considerable variation within these categories. The presence of outliers suggests that factors other than the number of bedrooms and house grade can significantly influence house prices.

```
# Boxplot for price by number of bedrooms
ggplot(df_house, aes(x = factor(bedrooms), y = price)) +
  geom_boxplot() +
  labs(title = "Boxplot of Prices by Number of Bedrooms", x = "Number of Bedrooms", y = "Price")
```



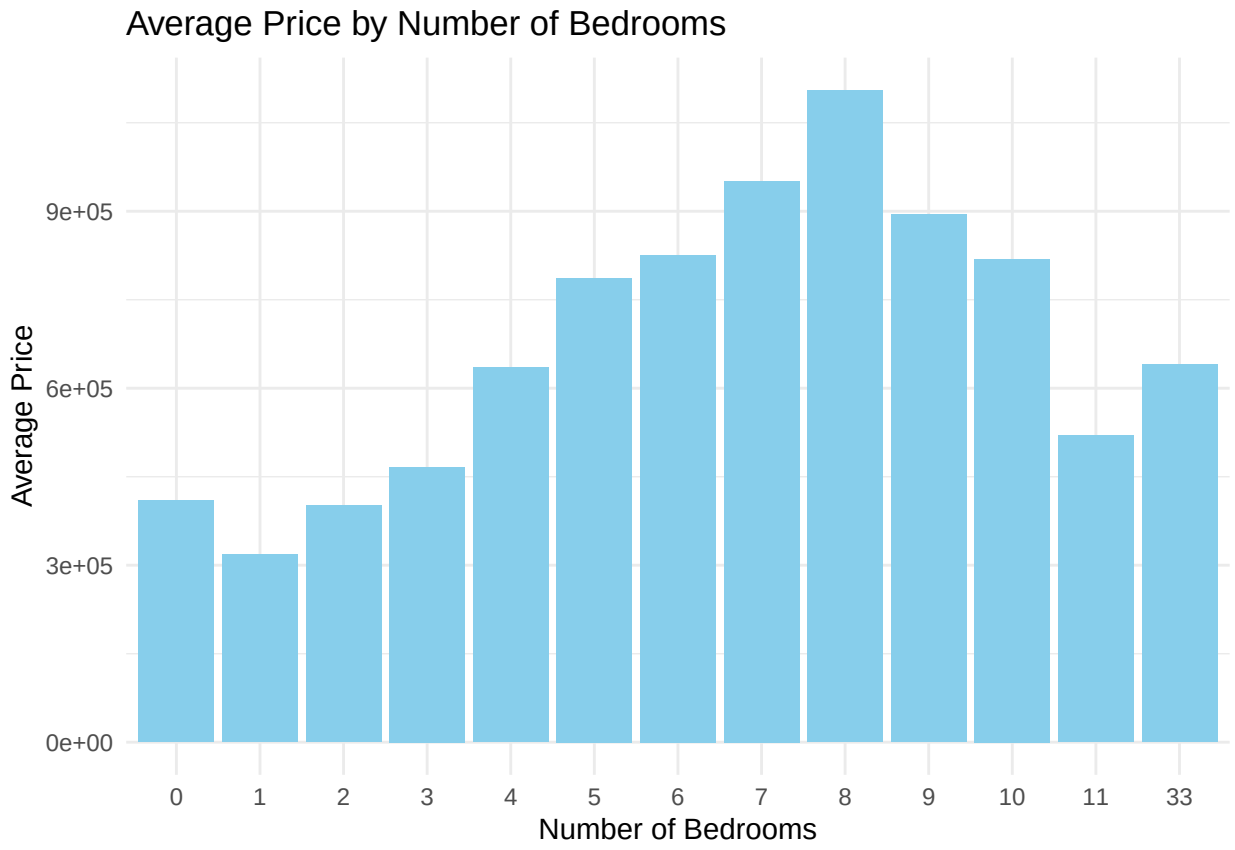

```
# Boxplot for price by house grade
ggplot(df_house, aes(x = factor(grade), y = price)) +
  geom_boxplot() +
  labs(title = "Boxplot of Prices by House Grade", x = "Grade", y = "Price")
```



Analysis: the

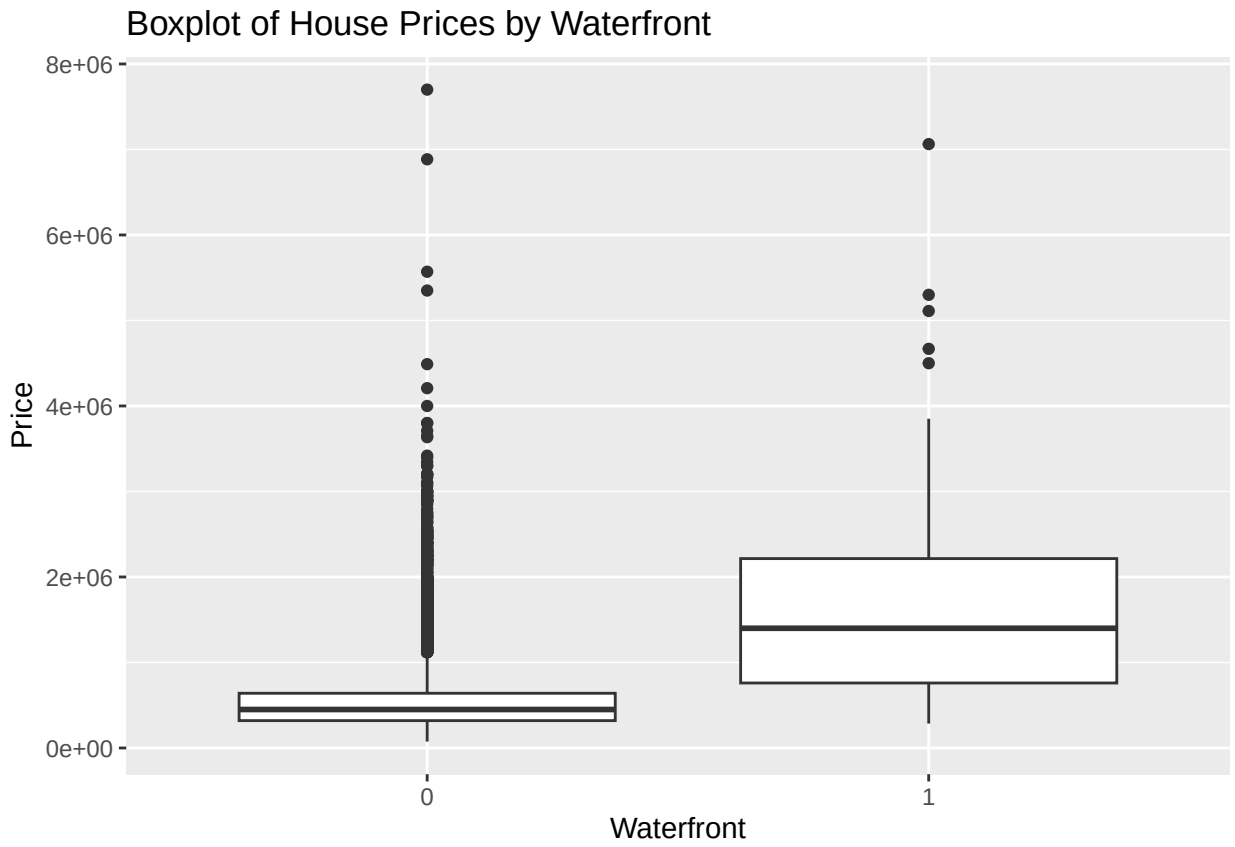
average price tends to increase with the number of bedrooms up to a certain point, but then the trend is less consistent. For instance, homes with 6 bedrooms have a higher average price than those with 7 or 8 bedrooms, and the average price for homes with 11 bedrooms is lower than for those with fewer bedrooms. There is a notable outlier at 33 bedrooms with a relatively low average price, which could indicate an atypical property or data error.

```
average_prices <- aggregate(price ~ bedrooms, data = ndf_house, FUN = mean)
ggplot(average_prices, aes(x = factor(bedrooms), y = price)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Average Price by Number of Bedrooms",
       x = "Number of Bedrooms",
       y = "Average Price") +
  theme_minimal()
```



Analysis: the spread of prices for waterfront homes is wider, indicating more variability in price. Notably, waterfront properties have more high-priced outliers, suggesting that while many waterfront homes are priced higher, a few are exceptionally expensive. Non-waterfront homes have a more compact price distribution but also feature outliers, indicating some are priced significantly above the median.

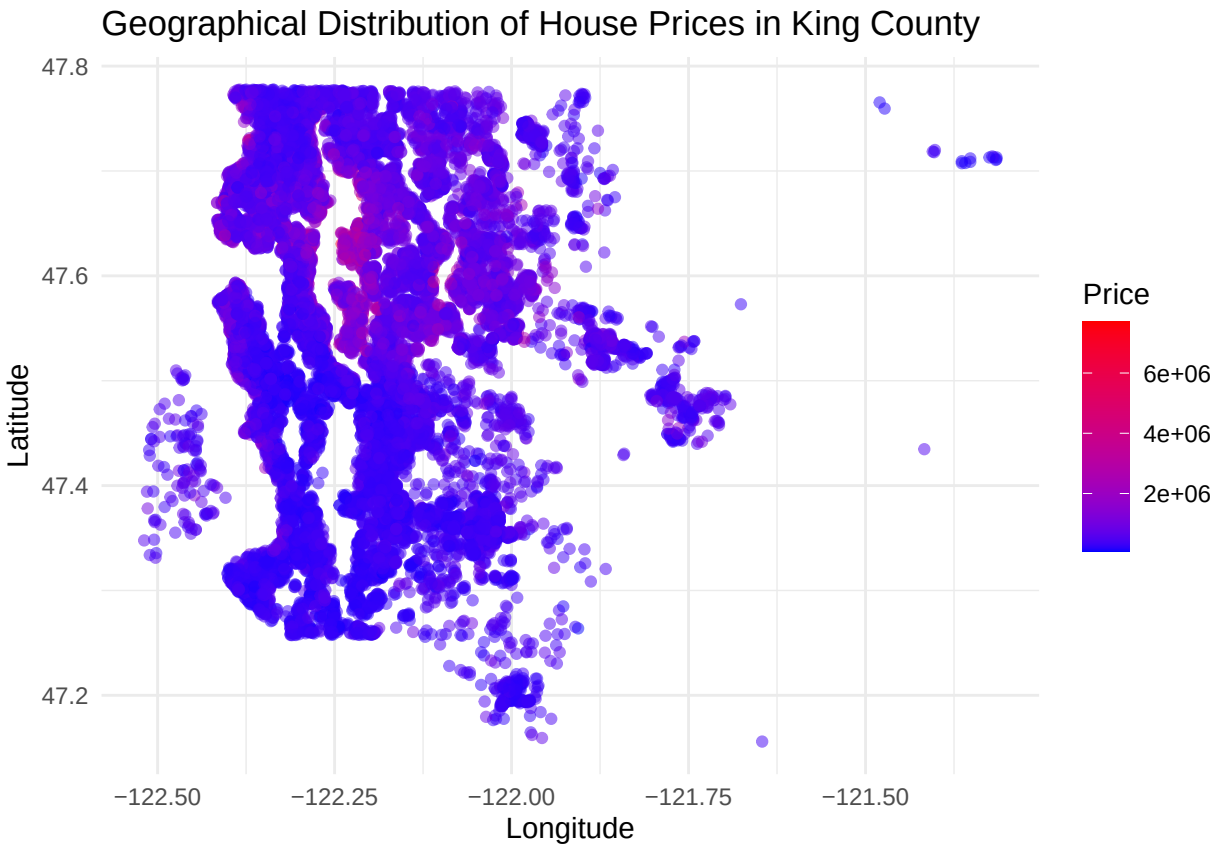
```
ggplot(df_house, aes(x = factor(waterfront), y = price)) +
  geom_boxplot() +
  labs(title = "Boxplot of House Prices by Waterfront", x = "Waterfront", y = "Price")
```



Analysis: From geographical map, it is evident that properties situated around the latitude line of 47.6 and longitude between -122.25 and -122.00, which corresponds to the central and northern parts of Seattle, command higher prices per square foot. The relatively lower-priced properties per square foot, shown in orange, are more dispersed and located primarily south of central Seattle, extending towards Tacoma, as well as in the outlying suburban areas.It is also noticeable that along the latitudinal line around 47.4, there are pockets of high-priced properties per square foot, potentially indicating affluent neighborhoods or areas with high-value real estate.

The map clearly shows a correlation between location and property value per square foot, with central urban areas exhibiting the highest values. This pattern is typical for urban centers where proximity to amenities, employment opportunities, and other socioeconomic factors drive up real estate prices.

```
# Scatter plot of properties
ggplot(data = ndf_house, aes(x = long, y = lat, color = price)) +
  geom_point(alpha = 0.5) +
  scale_color_gradient(low = "blue", high = "red") +
  labs(title = "Geographical Distribution of House Prices in King County",
       x = "Longitude", y = "Latitude", color = "Price") +
  theme_minimal()
```



Summary of the section II

In Section II, we thoroughly explored the King County house sales dataset, examining both continuous and categorical variables through statistical tests and extensive graphical analysis. We highlighted the strong correlations between variables like 'sqft_living', 'grade', 'sqft_above', and 'price'. Transformations and dummy variable creation were crucial in preparing data for modeling. The section concluded with the development of new variables to encapsulate the age and renovation status of properties, offering deeper insights for our predictive models.

III. Model Development Process (15 points) - (Bethun Bhowmik)

Build a regression model to predict price. And of course, create the train data set which contains 70% of the data and use set.seed (1023). The remaining 30% will be your test data set. Investigate the data and combine the level of categorical variables if needed and drop variables. For example, you can drop id, Latitude, Longitude, etc.

The KC House sales dataset is split into train(70%) and test(30%) datasets, with 15129 observations in the train dataset and 6484 observations in the test dataset.

```
#[Bethun]

set.seed(1023)
n<-dim(df_house)[1]
IND<-sample(c(1:n),round(n*0.7))
train.dat<-df_house[IND,]
test.dat<-df_house[-c(IND),]

dim(train.dat)
```

```
## [1] 15129    18
```

```
dim(test.dat)
```

```
## [1] 6484   18
```

Analysis: The house_lm model to predict price has: - Residual Standard Error of 193,600 - Multiple R-square of 71.38% - p-value of < 2.2e-16 - MSE of 37,494,980,807

Overall, the model appears to be statistically significant overall given the low p-value for the F-statistic. The multiple R-square suggests that 71.38% of the variability in price is explained by the model. Moreover, all most predictors (except sqft_lot) have a statistically significant impact on the house price. Predictions on the test dataset should be done to further validate model usefulness.

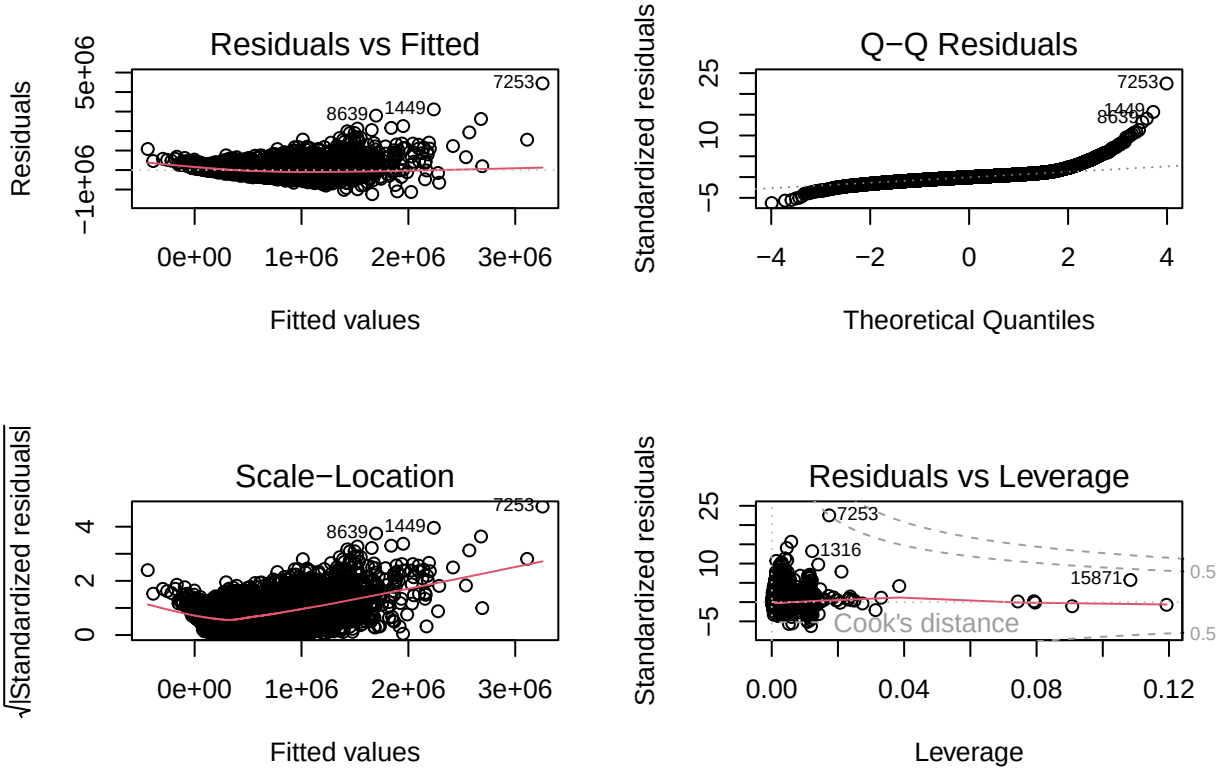
From the model plot we observe the following: - LINEARITY ASSUMPTION: From the Residuals vs Fitted plot, as the residuals seem to be randomly dispersed around the horizontal axis and the line is horizontal, a linear regression model may be appropriate, however we may need to transforming the response variable. - NORMALITY ASSUMPTION: From the Q-Q Residuals plot, as the points are out of the diagonal line on either ends, the residuals do not meet the normality assumption. - CONSTANT VARIANCE ASSUMPTION: The scale location plot points to non constant variance. The HOMOSCEDASTICITY ASSUMPTION is violated. Moreover the line is curvilinear. - The impact of bad data like influential and outlier observations seems to be impactful when inspecting the Residual vs. Leverage graph

```
#[Bethun]
house_lm<-lm(price ~.,data=train.dat)
summary(house_lm)
```

```
##
## Call:
## lm(formula = price ~ ., data = train.dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1236951   -98912    -9119    76684  4444553
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.118e+08  7.226e+06 -15.475  < 2e-16 ***
## bedrooms      -3.234e+04  2.197e+03 -14.720  < 2e-16 ***
## bathrooms      3.793e+04  3.821e+03   9.926  < 2e-16 ***
## sqft_lot       1.302e-01  5.979e-02   2.177 0.029494 *
## floors         1.109e+04  4.261e+03   2.603 0.009242 **
## waterfront     5.261e+05  2.143e+04  24.553  < 2e-16 ***
## view           5.158e+04  2.543e+03  20.285  < 2e-16 ***
## condition      3.320e+04  2.768e+03  11.995  < 2e-16 ***
## grade          9.651e+04  2.546e+03  37.902  < 2e-16 ***
```

```
## sqft_above     1.685e+02  4.374e+00  38.523  < 2e-16 ***
## sqft_basement  1.529e+02  5.167e+00  29.583  < 2e-16 ***
## lat            5.673e+05  1.243e+04  45.649  < 2e-16 ***
## long          -1.151e+05  1.410e+04  -8.163 3.52e-16 ***
## sqft_living15  3.578e+01  4.090e+00   8.748  < 2e-16 ***
## sqft_lot15    -3.239e-01  8.632e-02  -3.752 0.000176 ***
## year           3.471e+04  3.473e+03   9.994  < 2e-16 ***
## renovated      1.895e+05  8.381e+03  22.613  < 2e-16 ***
## age            2.397e+03  8.630e+01  27.772  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 199200 on 15111 degrees of freedom
## Multiple R-squared:  0.697,  Adjusted R-squared:  0.6967
## F-statistic:  2045 on 17 and 15111 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(house_lm)
```




```
MSE <- summary(house_lm)$sigma^2
print(MSE)
```

```
## [1] 39678071311
```

```
vif(house_lm)
```

```
##      bedrooms     bathrooms      sqft_lot        floors    waterfront
##      1.625094      3.301059      2.076177      2.015287      1.183776
##          view     condition         grade    sqft_above sqft_basement
##      1.409452      1.248831      3.399313      4.947021      2.001923
##           lat          long sqft_living15    sqft_lot15          year
##      1.124557      1.512058      3.022717      2.089128      1.006528
##     renovated           age
##      1.093130      2.366014
```

Analysis: The R-square values on both training and test data are quite close - 71.38% and 70.86% respectively. This suggests that the model generalizes well to new data, maintaining a similar level of explanatory power. The RMSE of 204,868.3 indicates the average magnitude of errors in predicting house prices on the test data. The SSE of 2.721402e+14 represents the sum of squared differences between predicted and observed values on the test data. A lower RMSE and SSE indicates better model fit, and we will explore furher on how these values can be lowered.

```
#[Bethun]
# Test data Predictions
house_lm_test_pred <- predict(house_lm, newdata = test.dat)

house_lm_test_mse <- mean((house_lm_test_pred - test.dat$price)^2)
house_lm_test_rmse <- sqrt(house_lm_test_mse)
house_lm_test_residuals <- test.dat$price - house_lm_test_pred
house_lm_test_rsq <- 1 - var(house_lm_test_residuals) / var(test.dat$price)
house_lm_test_sse <- sum((test.dat$price - house_lm_test_pred)^2)
results.df <- data.frame(model = "Linear Regression Test Data Predictions",
                         R.Squared.Train = summary(house_lm)$r.square,
                         R.Squared.Test = house_lm_test_rsq,
                         RMSE.test = house_lm_test_rmse,
                         SSE.test = house_lm_test_sse)
print(results.df)
```

```
##                                   model R.Squared.Train R.Squared.Test
## 1 Linear Regression Test Data Predictions       0.6970168      0.6921461
##   RMSE.test     SSE.test
## 1  210589.7 2.875525e+14
```

Analysis: The insignificant predictor sqft_lot is dropped and the model is refitted. We observe that the Multiple R-square value remains unchanged at 71.38%. The model is as follows:

Price = -74344803.92767 + -34633.405255bedrooms + 32821.338262bathrooms + 143.671881sqft_living + 8767.015515floors + 531351.549043waterfront + 52065.87946view + 24875.912771condition + 82737.693562grade + 22.301049sqft_above + -896.223982yr_built + 4065.130653yr_renovated + -507.797776zipcode + 473638.095898lat + -206204.537141long + 21.65297sqft_living15 + -0.25171sqft_lot15 + 38482.930296year + 1380.790546month + -331.741298day + -7982223.272107renovated + 1440.530732age + 101303.883852price_binary

```
#[Bethun]
predictors_to_drop <- c("sqft_lot")

# Update the model by excluding the specified predictors
updated <- as.formula(paste("price ~ .", paste0("- `", predictors_to_drop, "`", collapse = "")))
house_lm <- update(house_lm, updated)
summary(house_lm)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + floors + waterfront +
##     view + condition + grade + sqft_above + sqft_basement + lat +
##     long + sqft_living15 + sqft_lot15 + year + renovated + age,
##     data = train.dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1229780   -99024    -9212    76414  4442353
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.116e+08  7.226e+06 -15.442  < 2e-16 ***
## bedrooms      -3.252e+04  2.196e+03 -14.814  < 2e-16 ***
## bathrooms      3.804e+04  3.822e+03   9.953  < 2e-16 ***
## floors         1.079e+04  4.259e+03   2.533  0.01133 *
## waterfront     5.259e+05  2.143e+04  24.542  < 2e-16 ***
```

```
## view           5.172e+04  2.543e+03  20.342  < 2e-16 ***
## condition      3.307e+04  2.768e+03  11.947  < 2e-16 ***
## grade          9.661e+04  2.546e+03  37.942  < 2e-16 ***
## sqft_above     1.691e+02  4.365e+00  38.744  < 2e-16 ***
## sqft_basement  1.531e+02  5.167e+00  29.639  < 2e-16 ***
## lat            5.664e+05  1.242e+04  45.597  < 2e-16 ***
## long          -1.123e+05  1.404e+04  -7.996 1.38e-15 ***
## sqft_living15  3.522e+01  4.083e+00   8.627  < 2e-16 ***
## sqft_lot15    -1.960e-01  6.323e-02  -3.099  0.00194 **
## year           3.479e+04  3.474e+03  10.014  < 2e-16 ***
## renovated      1.898e+05  8.381e+03  22.643  < 2e-16 ***
## age            2.405e+03  8.623e+01  27.884  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 199200 on 15112 degrees of freedom
## Multiple R-squared:  0.6969, Adjusted R-squared:  0.6966
## F-statistic:  2172 on 16 and 15112 DF,  p-value: < 2.2e-16
```

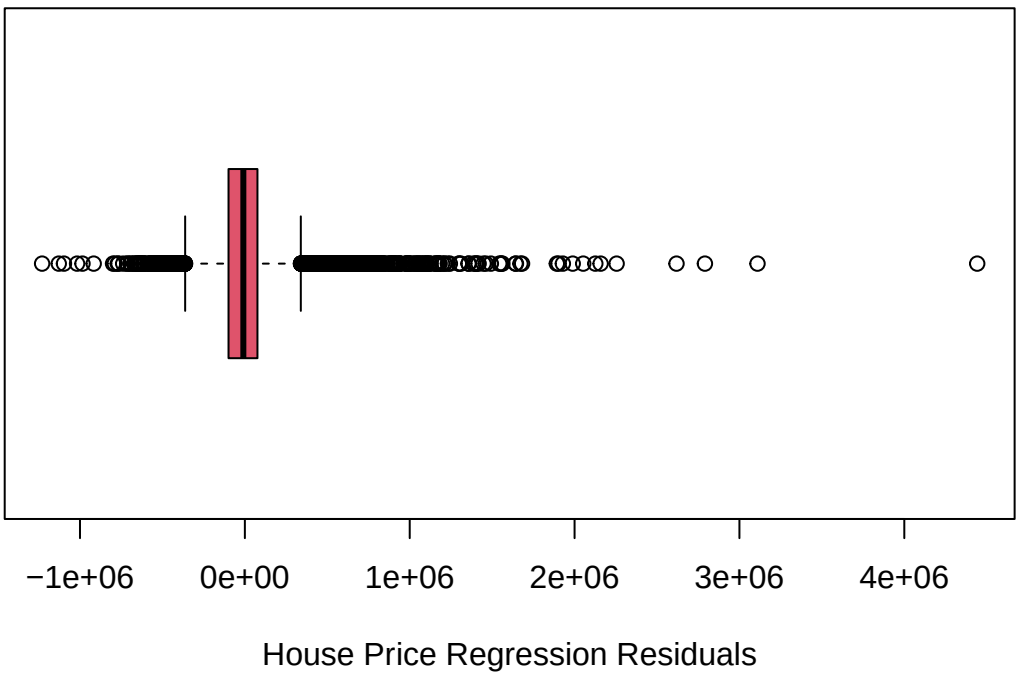
```
dispRegFunc(house_lm)
```

```
## [1] "Y = -111590227.725111 + -32524.766156bedrooms + 38035.229035bathrooms + 10787.175028floors + 525929.27
```

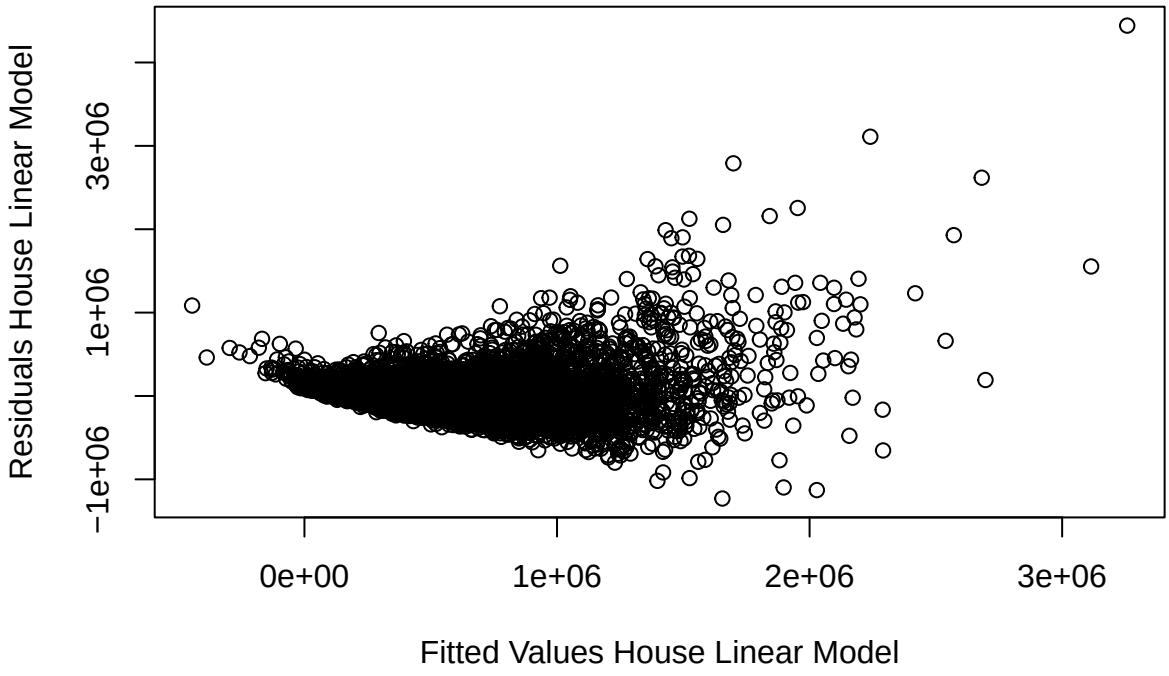
Analysis:

On plotting the boxplot of residuals, we observe multiple outliers on both ends of the whiskers. Moreover, when we plot the residuals against the fitted values, we observe that the residuals spread out with increase in fitted values along x-axis.

```
#[Bethun]
ei<-house_lm$residuals
boxplot(ei,horizontal=TRUE,staplewex=0.5,col=2,xlab="House Price Regression Residuals")
```




```
plot(house_lm$fitted.values,ei,xlab="Fitted Values House Linear Model",ylab="Residuals House Linear Model")
```



Analysis: Predictors with low p-values of < 2.2e-16 (e.g. bedrooms, bathrooms, sqft_living, etc.) are statistically significant and are important in predicting price. Some predictors that have a have higher p-value (e.g. floors, zipcode, month, day) may not be statistically significant in predicting price.

```
#[Bethun]
anova(house_lm)
```

```
## Analysis of Variance Table
##
## Response: price
##                  Df     Sum Sq    Mean Sq   F value  Pr(>F)
## bedrooms          1 1.8429e+14 1.8429e+14 4643.4916 < 2e-16 ***
## bathrooms         1 3.5576e+14 3.5576e+14 8963.8807 < 2e-16 ***
## floors            1 3.0574e+09 3.0574e+09    0.0770 0.78136
## waterfront        1 9.2404e+13 9.2404e+13 2328.2653 < 2e-16 ***
## view              1 1.1533e+14 1.1533e+14 2905.9410 < 2e-16 ***
## condition         1 1.6207e+13 1.6207e+13  408.3704 < 2e-16 ***
## grade             1 3.0945e+14 3.0945e+14 7797.1050 < 2e-16 ***
## sqft_above        1 4.0555e+13 4.0555e+13 1021.8607 < 2e-16 ***
## sqft_basement     1 7.8844e+13 7.8844e+13 1986.6033 < 2e-16 ***
## lat               1 1.2513e+14 1.2513e+14 3152.7916 < 2e-16 ***
## long              1 1.2985e+13 1.2985e+13  327.1806 < 2e-16 ***
## sqft_living15     1 2.8011e+12 2.8011e+12   70.5792 < 2e-16 ***
## sqft_lot15        1 2.0893e+11 2.0893e+11    5.2644 0.02178 *
## year              1 3.5173e+12 3.5173e+12   88.6236 < 2e-16 ***
## renovated         1 1.0800e+13 1.0800e+13  272.1317 < 2e-16 ***
## age               1 3.0859e+13 3.0859e+13  777.5446 < 2e-16 ***
## Residuals     15112 5.9976e+14 3.9688e+10
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Analysis:

A two sides t statistical test is used for testing individual coefficients and their significance. The critical t -value applicable

given alpha=0.01 and n-2 degrees of freedom is 2.576. H0: intercept =0, HA: intercept is not 0 H0: slope= 0 , HA: slope is not 0 Decision rule: when t > t critical reject null

The model, as indicated by the significant coefficients, suggests that features such as the number of bedrooms, bathrooms, square footage of living space, waterfront status, view, grade, etc, play a statistically significant role in predicting the house price.

```
#[Bethun]

conf= 0.01/2     #Note we divide by 2 because this is a two tail test,if alpha=0.05 then use .05/2
df<-21613-2            #manually calculating the degrees of freedom
value<-formatC(qt(conf,df,lower.tail=FALSE))
print(paste("Critical T values: ",value))
```

```
## [1] "Critical T values:  2.576"
```

```
#coefficients
matrix_coef <- summary(house_lm)$coefficients  # Extract coefficients in matrix
matrix_coef
```

```
##                    Estimate   Std. Error    t value      Pr(>|t|)
## (Intercept)   -1.115902e+08 7.226416e+06 -15.441987  2.184066e-53
## bedrooms      -3.252477e+04 2.195592e+03 -14.813668  2.650259e-49
## bathrooms      3.803523e+04 3.821611e+03   9.952668  2.895154e-23
## floors         1.078718e+04 4.259203e+03   2.532675  1.132963e-02
## waterfront     5.259293e+05 2.143014e+04  24.541568 1.877088e-130
## view           5.171977e+04 2.542536e+03  20.341800  8.982785e-91
## condition      3.306570e+04 2.767776e+03  11.946667  9.490708e-33
## grade          9.660723e+04 2.546148e+03  37.942497 5.462908e-301
## sqft_above     1.691345e+02 4.365482e+00  38.743612 3.730091e-313
## sqft_basement  1.531308e+02 5.166582e+00  29.638698 1.063234e-187
## lat            5.664104e+05 1.242203e+04  45.597256  0.000000e+00
## long          -1.122994e+05 1.404470e+04  -7.995852  1.379457e-15
## sqft_living15  3.522177e+01 4.082684e+00   8.627111  6.910724e-18
## sqft_lot15    -1.959742e-01 6.323252e-02  -3.099262  1.943595e-03
## year           3.478544e+04 3.473546e+03  10.014389  1.560267e-23
## renovated      1.897771e+05 8.381128e+03  22.643383 1.159083e-111
## age            2.404611e+03 8.623474e+01  27.884487 6.689618e-167
```

```
my_estimates <- matrix_coef[ , 1]  # Matrix manipulation to extract estimates

#Step 6: Pr(>|t|) < 0.01 there is sufficient statistical evidence to reject null for both parameters. Using a
```

H_o: Error variances are constant H_a: Error variances are not constant Decision Rule is if statistic> critical reject the null or if p-value < alpha (0.01) reject the hull

P value is < 2.2e-16. So we reject H_o, Error variances are not constant.

Analysis:

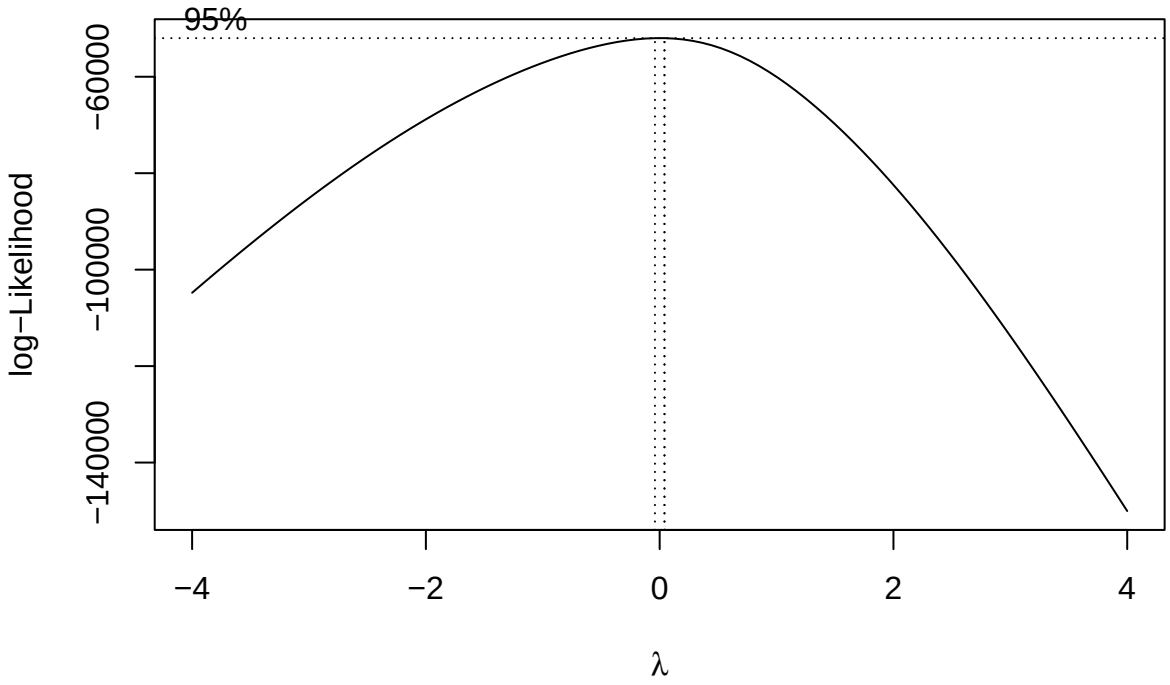
```
#[Bethun]
bptest(house_lm, studentize = FALSE)
```

```
##
##  Breusch-Pagan test
##
## data:  house_lm
## BP = 38445, df = 16, p-value < 2.2e-16
```

Analysis: For Boxcox transormation, the optimal lambda is -0.04040404 with 95% confidence interval range which is close to zero.Although, we can estimate lambda to be zero and apply logarithmic transformation of the data, we have done the boxcox transformation with exact optimal lambda value for better accuracy.

```
#[Bethun]
par(mfrow=c(1,1))
```

```r
bc<-boxcox(house_lm,lambda=seq(-4,4,by=0.1))
```




```r
lambda <- bc$x[which.max(bc$y)]
lambda  #This is the optimal lambda
```

```
## [1] 0.04040404
```

Analysis: After boxcox transformation, we observe that - Residual Standard Error has reduced significantly to 0.004981 from 193,600 - Multiple R-square has increased to 84.03% from 71.38% - p-value remains same at $< 2.2e\text{-}16$

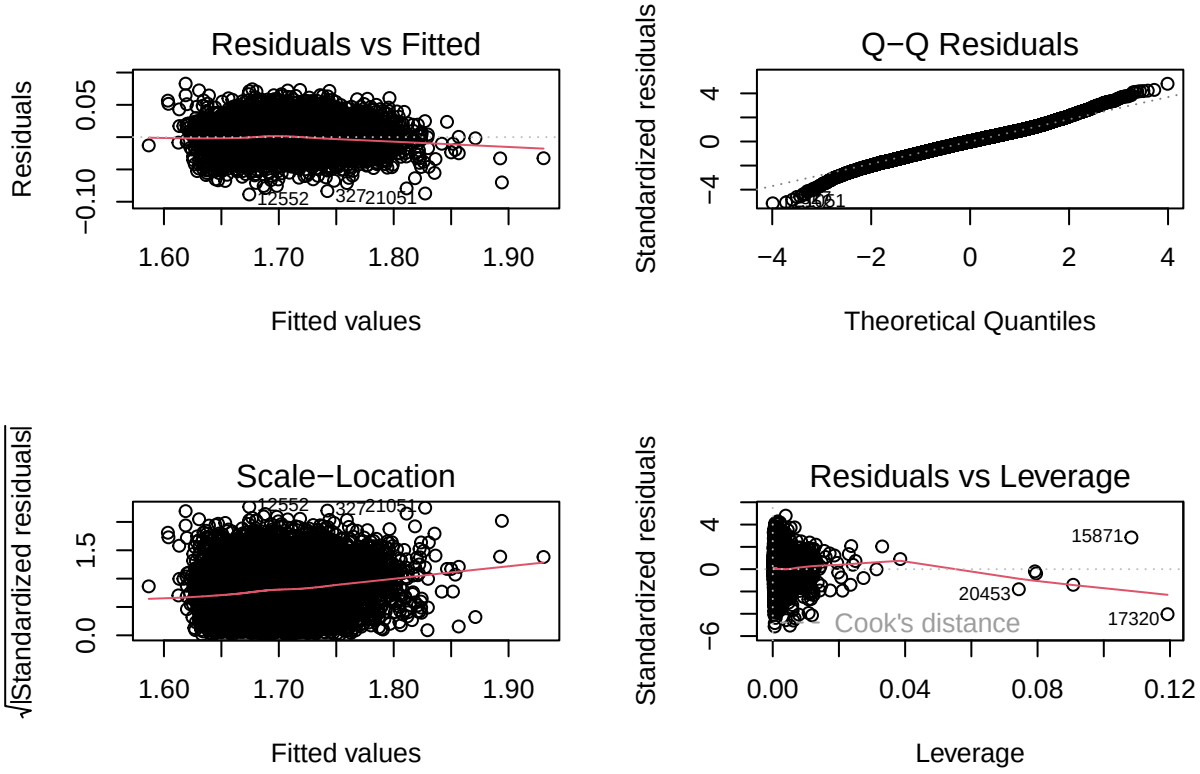
Overall, the model appears to be a much better fit that before. From the model plot we observe the following: - LINEARITY ASSUMPTION: The Residuals vs Fitted plot looks better than before and confirms that a linear regression model is appropriate - NORMALITY ASSUMPTION: The points are much better aligned along the diagonal in the Q-Q Residuals plot, however some tails remain - CONSTANT VARIANCE ASSUMPTION: The Scale Location plot points to constant variance - There are still some outlier observations which may be to be impactful as seen from the the Residual vs. Leverage graph

```r
#[Bethun]
house_lm1<-lm(price^lambda~.,data=train.dat)
summary(house_lm1)
```

```
##
## Call:
## lm(formula = price^lambda ~ ., data = train.dat)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.088758 -0.010861  0.000204  0.010680  0.082753
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.118e+01  6.270e-01 -17.824  < 2e-16 ***
## bedrooms      -9.122e-04  1.906e-04  -4.785 1.72e-06 ***
## bathrooms      4.468e-03  3.316e-04  13.475  < 2e-16 ***
```

```
## sqft_lot       3.371e-08  5.188e-09   6.497 8.48e-11 ***
## floors         4.938e-03  3.697e-04  13.355  < 2e-16 ***
## waterfront     2.446e-02  1.859e-03  13.153  < 2e-16 ***
## view           4.060e-03  2.207e-04  18.402  < 2e-16 ***
## condition      4.852e-03  2.402e-04  20.199  < 2e-16 ***
## grade          1.068e-02  2.209e-04  48.356  < 2e-16 ***
## sqft_above     9.662e-06  3.796e-07  25.455  < 2e-16 ***
## sqft_basement  1.118e-05  4.484e-07  24.926  < 2e-16 ***
## lat            9.337e-02  1.078e-03  86.593  < 2e-16 ***
## long          -4.247e-03  1.224e-03  -3.471  0.00052 ***
## sqft_living15  7.396e-06  3.549e-07  20.838  < 2e-16 ***
## sqft_lot15    -1.222e-08  7.490e-09  -1.631  0.10290
## year           3.848e-03  3.014e-04  12.768  < 2e-16 ***
## renovated      1.789e-02  7.272e-04  24.600  < 2e-16 ***
## age            2.108e-04  7.488e-06  28.148  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01728 on 15111 degrees of freedom
## Multiple R-squared:  0.7706, Adjusted R-squared:  0.7704
## F-statistic:  2986 on 17 and 15111 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(house_lm1)
```




```
dispRegFunc(house_lm1)
```

```
## [1] "Y = -11.176596 + -0.000912bedrooms + 0.004468bathrooms + 0sqft_lot + 0.004938floors + 0.024455waterfro
```

Analysis: On predicting the values of the Boxcox transformed model on the test dataset, we observe a R-square test of 75.79% compared to R-square train of 84.03%, from which we conclude that the transformed model generalizes well to new data.

```r
# [Bethun]
# Test data Predictions
pred<- predict(house_lm1,test.dat)^(1/lambda)
act<-test.dat$price

house_lm1_test_mse <- mean((pred - act)^2)
house_lm1_test_rmse <- sqrt(house_lm1_test_mse)
house_lm1_test_residuals <- act - pred
house_lm1_test_rsq <- 1 - var(house_lm1_test_residuals) / var(act)
house_lm1_test_sse <- sum((act - pred)^2)
results.df1 <- data.frame(model = "Linear Regression Test Data Predictions after Boxcox",
                          R.Squared.Train = summary(house_lm1)$r.square,
                          R.Squared.Test = house_lm1_test_rsq,
                          RMSE.test = house_lm1_test_rmse,
                          SSE.test = house_lm1_test_sse)
print(results.df1)
```

```
##                                                 model R.Squared.Train
## 1 Linear Regression Test Data Predictions after Boxcox       0.7706297
##   R.Squared.Test RMSE.test     SSE.test
## 1      0.6609158    221573 3.183293e+14
```

IV. Model Performance Testing (15 points)

Use the test data set to assess the model performances. Here, build the best multiple linear models by using the stepwise both ways selection method. Compare the performance of the best two linear models. Make sure that model assumption(s) are checked for the final linear model. Apply remedy measures (transformation, etc.) that helps satisfy the assumptions. In particular you must deeply investigate unequal variances and multicollinearity. If necessary, apply remedial methods (WLS, Ridge, Elastic Net, Lasso, etc.).

Analysis: - add here the analysis of the stepwise model summary outcome.

Graph 1 (Residuals vs Fitted): (linearity assumption) A linear relationship seems appropriate. (The average mean error equal to zero assumption) The average mean seems to be equal to zero.

Graph 2 (Q-Q Residuals): (normality assumption) There is minor departure from a normal distribution at the tails.

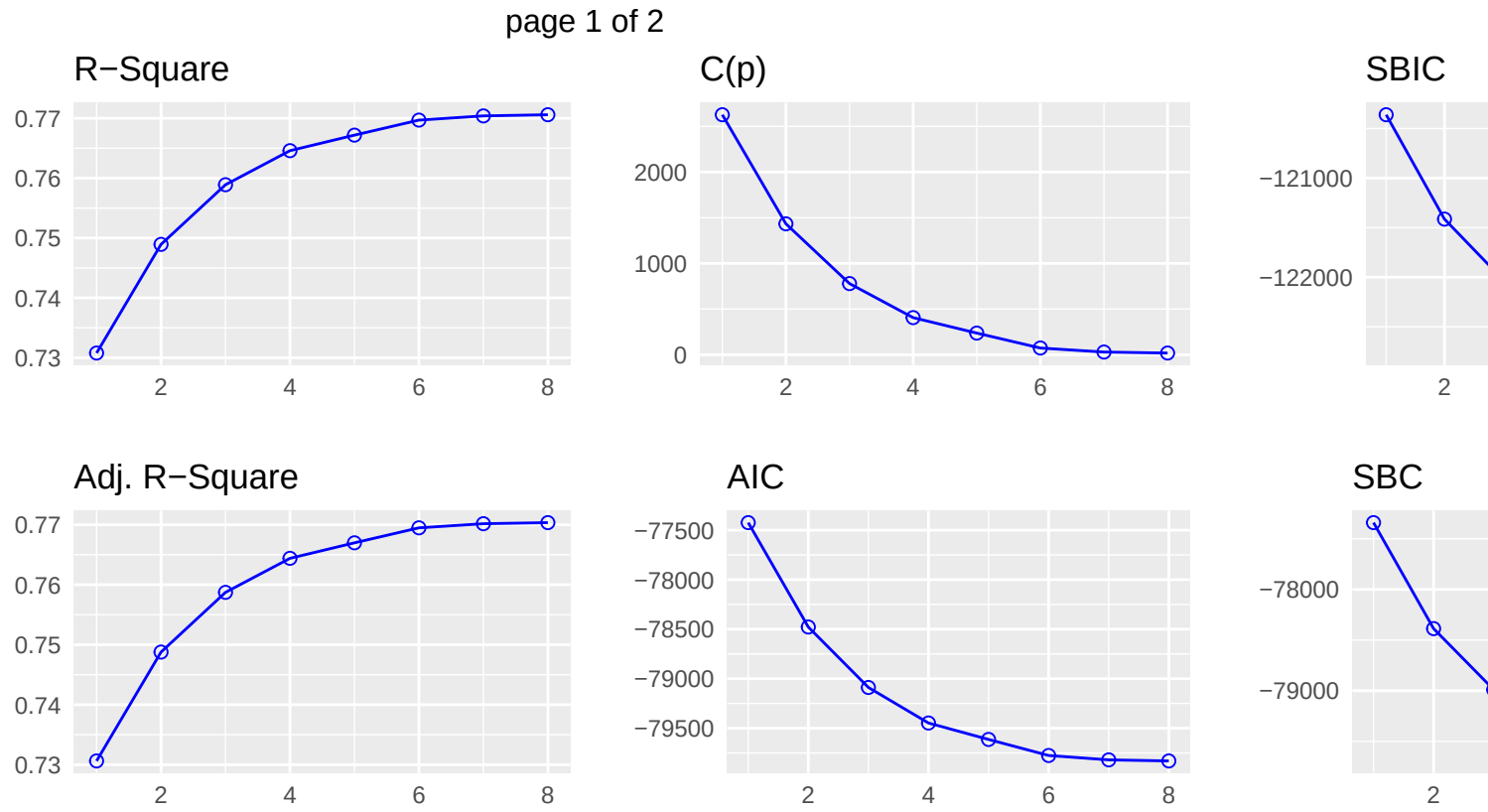
Graph 3 (Scale-Location): (constant variance assumption) The residuals do not seem to be equally distributed throughout. (homoscedasticity assumption) The variances do not appear constant and the Breush-Pagan test indicates this with the p-value being lower than alpha.

Graph 4 (Residuals vs Leverage): There are no influential outliers.

```
library(olsrr)

initial_model <- house_lm1

stepwise_model <- ols_step_both_p(initial_model, penter = 0.05, premove = 0.05)

final_model <- lm(formula = stepwise_model$model, data = train.dat)
summary(final_model)
```

```
##
## Call:
## lm(formula = stepwise_model$model, data = train.dat)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.088710 -0.010884  0.000223  0.010686  0.081879
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.120e+01  6.269e-01 -17.865  < 2e-16 ***
## bedrooms      -9.000e-04  1.905e-04  -4.724 2.33e-06 ***
## bathrooms      4.483e-03  3.315e-04  13.525  < 2e-16 ***
## floors         4.958e-03  3.695e-04  13.416  < 2e-16 ***
## view           4.057e-03  2.207e-04  18.385  < 2e-16 ***
## grade          1.069e-02  2.209e-04  48.406  < 2e-16 ***
## sqft_above     9.627e-06  3.790e-07  25.403  < 2e-16 ***
## sqft_basement  1.116e-05  4.483e-07  24.893  < 2e-16 ***
## lat            9.341e-02  1.078e-03  86.639  < 2e-16 ***
## sqft_living15  7.372e-06  3.546e-07  20.787  < 2e-16 ***
## age            2.110e-04  7.488e-06  28.180  < 2e-16 ***
## renovated      1.788e-02  7.272e-04  24.585  < 2e-16 ***
## condition      4.844e-03  2.402e-04  20.171  < 2e-16 ***
## waterfront     2.444e-02  1.859e-03  13.143  < 2e-16 ***
## year           3.850e-03  3.014e-04  12.774  < 2e-16 ***
## sqft_lot       2.794e-08  3.800e-09   7.353 2.03e-13 ***
## long          -4.389e-03  1.221e-03  -3.596 0.000324 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01729 on 15112 degrees of freedom
## Multiple R-squared:  0.7706, Adjusted R-squared:  0.7703
## F-statistic:  3173 on 16 and 15112 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(stepwise_model)
```



Analysis: - Please add here your analysis.

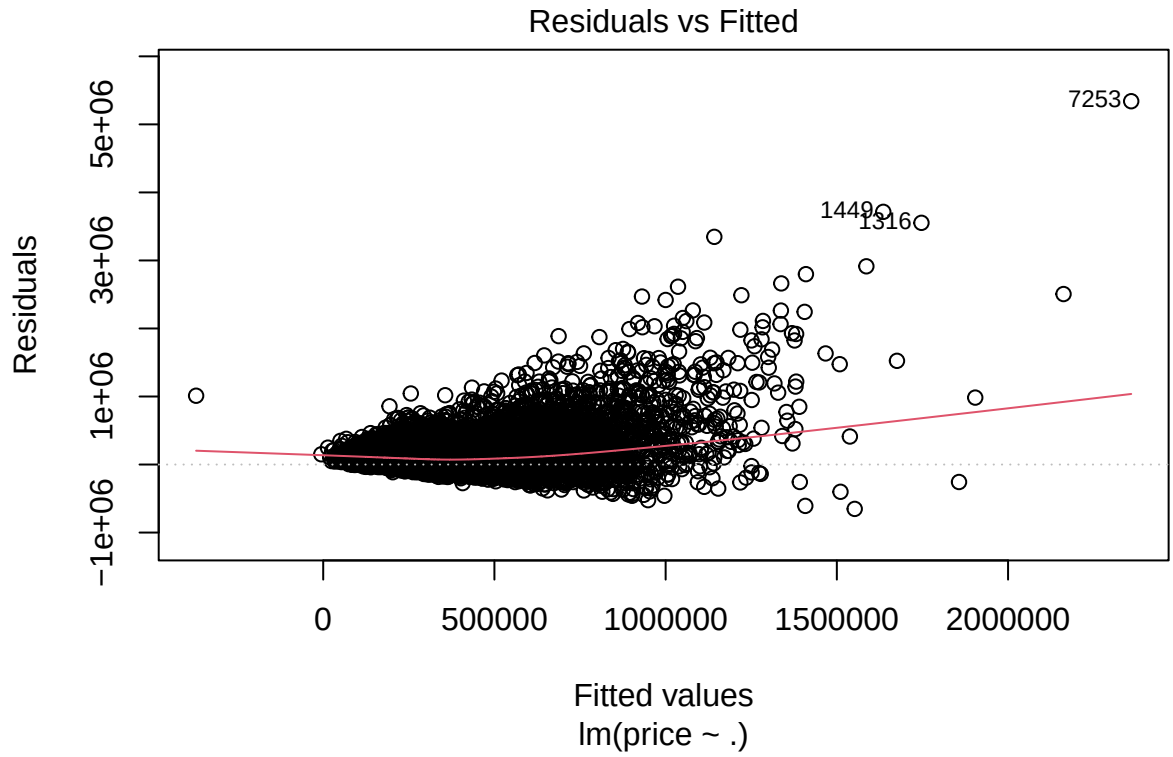
```
# Weighted Least Squares Regression
# We notice from the scale-location graph of the main model that the error variances are unequal.

# Calculating the weights for the model
ei <- house_lm$residuals
abs.ei <- abs(ei)
g1 <- lm(abs.ei ~ train.dat$price)
summary(g1)
```

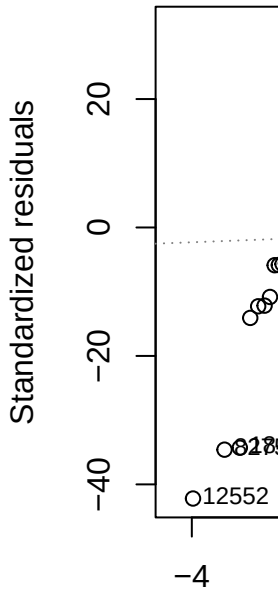
```
##
## Call:
## lm(formula = abs.ei ~ train.dat$price)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -580405  -66049  -12242   52720 2346203
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -2.282e+04  1.726e+03  -13.22   <2e-16 ***
## train.dat$price  2.752e-01  2.660e-03  103.44   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 118300 on 15127 degrees of freedom
## Multiple R-squared:  0.4143, Adjusted R-squared:  0.4143
## F-statistic: 1.07e+04 on 1 and 15127 DF,  p-value: < 2.2e-16
```

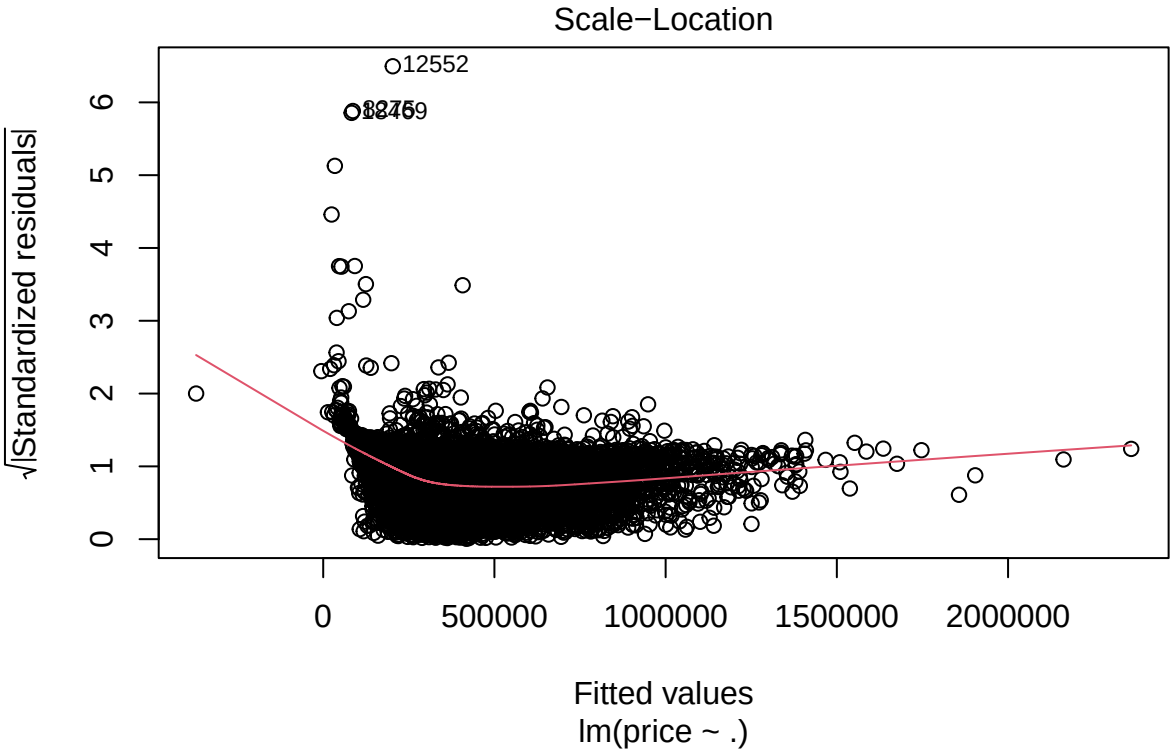
```r
s <- g1$fitted.values
wi = 1/(s^2)

# Weighted-least squares regression
house_lm_wls <- lm(price ~ ., weights = wi, data = train.dat)

summary(house_lm_wls)
```

```
##
## Call:
## lm(formula = price ~ ., data = train.dat, weights = wi)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -57.982   0.043   0.873   1.607  32.558
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -4.453e+07  2.325e+06 -19.154  < 2e-16 ***
## bedrooms      -2.377e+04  9.076e+02 -26.187  < 2e-16 ***
## bathrooms      2.443e+04  1.962e+03  12.449  < 2e-16 ***
## sqft_lot       1.772e-01  3.348e-02   5.295 1.21e-07 ***
## floors         2.923e+04  2.063e+03  14.168  < 2e-16 ***
## waterfront     1.532e+05  3.316e+04   4.621 3.85e-06 ***
## view           4.510e+04  2.634e+03  17.121  < 2e-16 ***
## condition      1.774e+04  7.723e+02  22.968  < 2e-16 ***
## grade          2.919e+04  8.608e+02  33.917  < 2e-16 ***
## sqft_above     1.403e+02  2.918e+00  48.076  < 2e-16 ***
## sqft_basement  1.747e+02  3.414e+00  51.156  < 2e-16 ***
## lat            2.576e+05  4.289e+03  60.049  < 2e-16 ***
## long           1.099e+05  3.951e+03  27.817  < 2e-16 ***
## sqft_living15 -2.035e+00  1.145e+00  -1.778   0.0755 .
## sqft_lot15     4.182e-01  4.989e-02   8.383  < 2e-16 ***
## year           2.259e+04  1.155e+03  19.554  < 2e-16 ***
## renovated      5.035e+04  5.495e+03   9.163  < 2e-16 ***
## age           -1.613e+01  3.198e+01  -0.504   0.6141
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.657 on 15111 degrees of freedom
## Multiple R-squared:  0.7918, Adjusted R-squared:  0.7915
## F-statistic:  3380 on 17 and 15111 DF,  p-value: < 2.2e-16
```

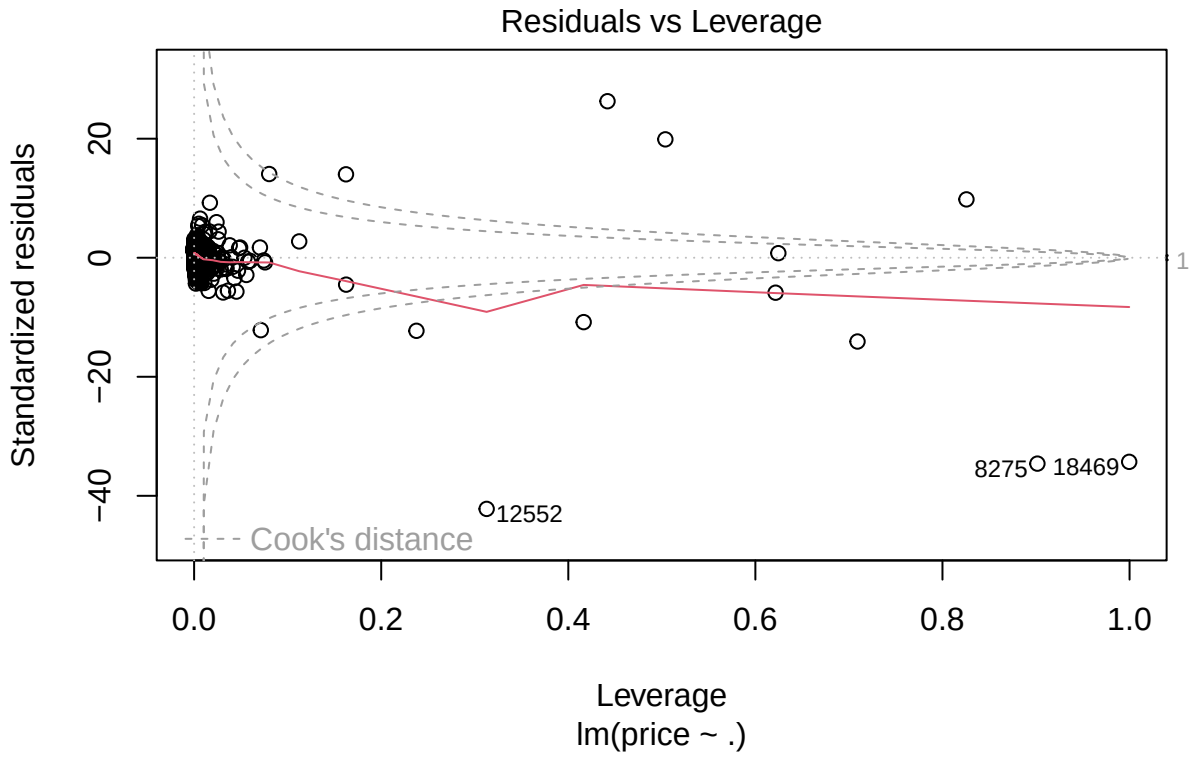
```r
#par(mfrow=c(2,2))
plot(house_lm_wls)
```








```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced

## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```



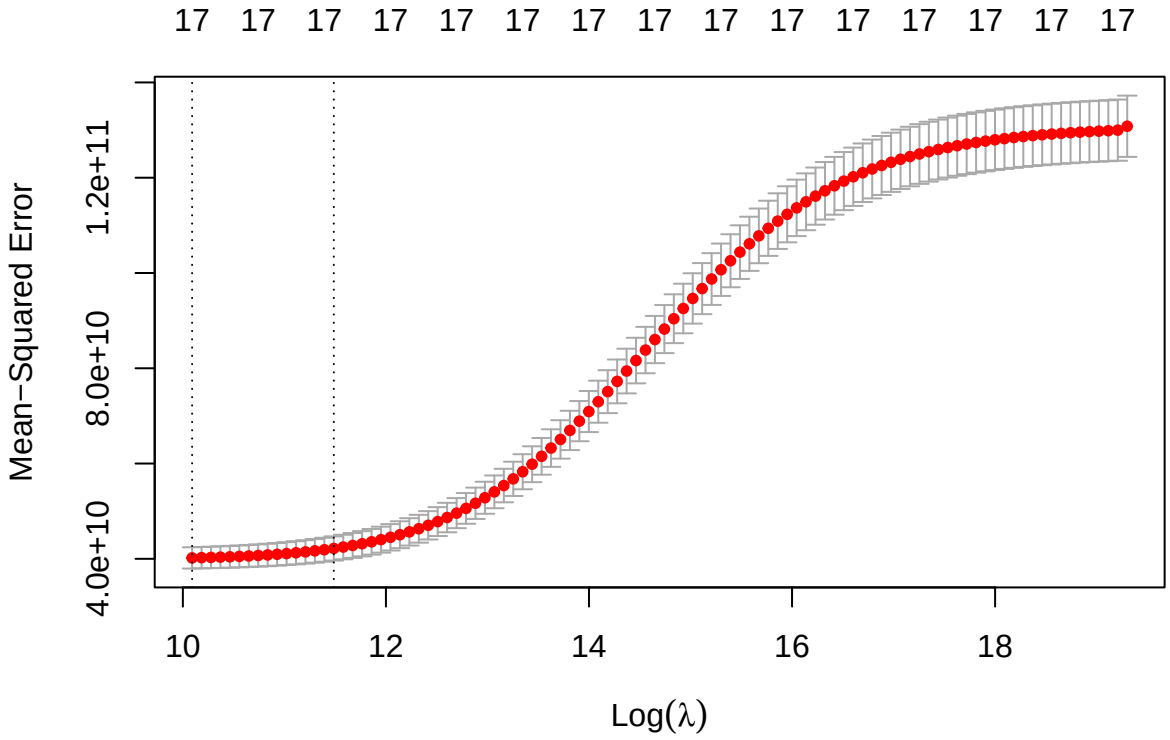
Analysis: As of now, there is no multicollinearity issues in the train data set since we have handled the highly correlated variables at the initial section.

```
# Testing the main (boxcox) model for multicollinearity
vif(house_lm1)
```

```
##      bedrooms     bathrooms      sqft_lot        floors    waterfront
##      1.625094      3.301059      2.076177      2.015287      1.183776
##          view     condition         grade    sqft_above sqft_basement
##      1.409452      1.248831      3.399313      4.947021      2.001923
##           lat          long sqft_living15    sqft_lot15          year
##      1.124557      1.512058      3.022717      2.089128      1.006528
##     renovated           age
##      1.093130      2.366014
```

Analysis: - please add here the analysis of the outcome

```
# Ridge Regression

# Extract 'x' and 'y'
x <- data.matrix(dplyr::select(train.dat, -price))
y <- train.dat$price

# Perform ridge regression
house_lm_ridge <- glmnet::cv.glmnet(x, y, alpha = 0, nlambda = 100, lambda.min.ratio = 0.0001)
best.lambda.ridge <- house_lm_ridge$lambda.min
plot(house_lm_ridge)
```

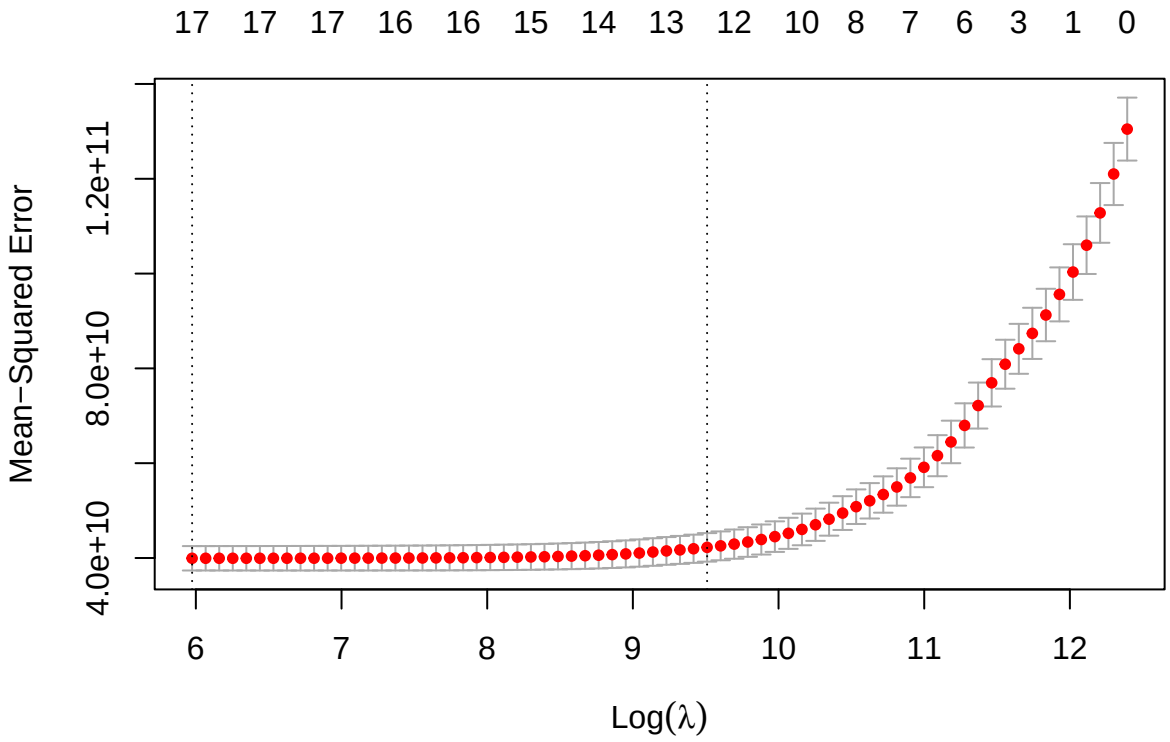



```
print(paste0("Ridge best lambda of ", round(best.lambda.ridge, digits = 3)))
```

```
## [1] "Ridge best lambda of 24125.359"
```

Analysis: - please add here the analysis of the outcome

```
# Lasso Regression
house_lm_lasso <- glmnet::cv.glmnet(x, y, alpha = 1, nlambda = 100, lambda.min.ratio = 0.0001)
best.lambda.lasso <- house_lm_lasso$lambda.min
plot(house_lm_lasso)
```

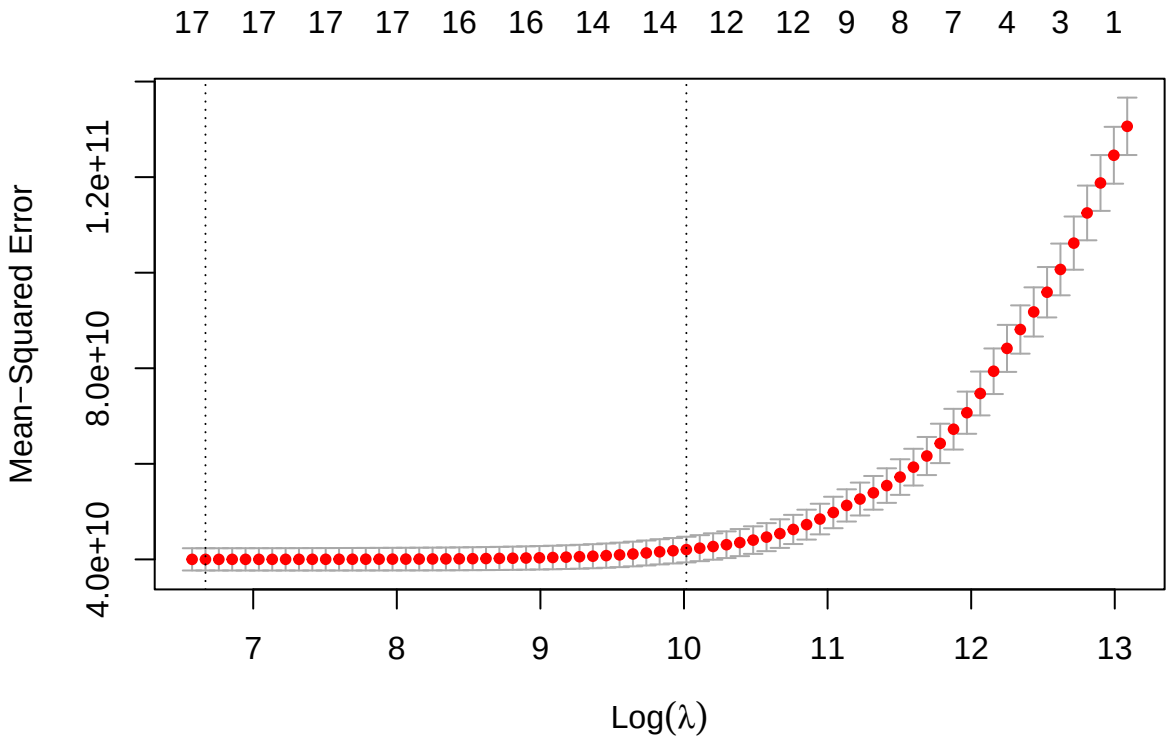



```
print(paste0("Lasso best lambda of ", round(best.lambda.lasso, digits = 3)))
```

```
## [1] "Lasso best lambda of 393.183"
```

Analysis: - please add here the analysis of the outcome

```
# Elastic Net Regression
house_lm_enet <- glmnet::cv.glmnet(x, y, alpha = 0.5, nlambda = 100, lambda.min.ratio = 0.0001)
plot(house_lm_enet)
```

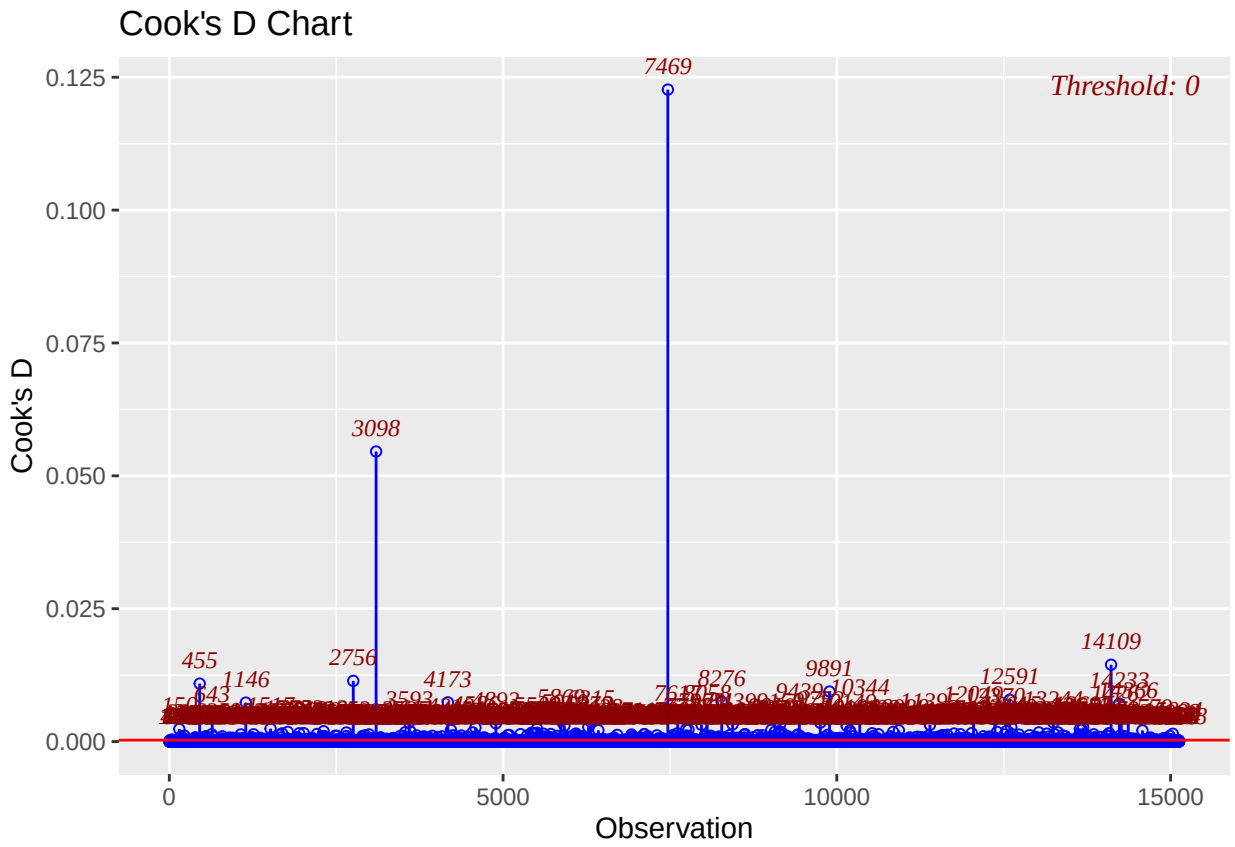



```
best.lambda.enet <- house_lm_enet$lambda.min
print(paste0("ElasticNet best lambda of ", round(best.lambda.enet, digits = 3)))
```

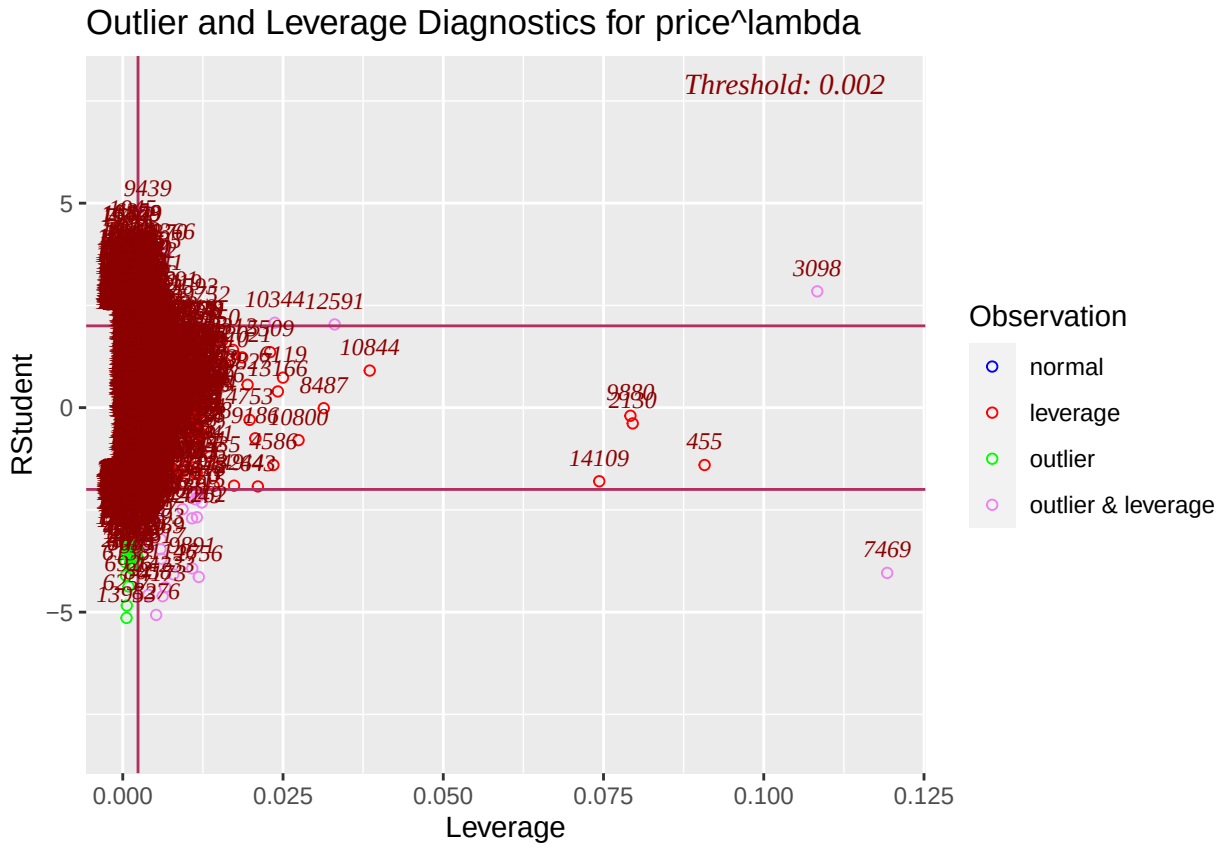
```
## [1] "ElasticNet best lambda of 786.366"
```

Analysis: - please add here the analysis of the outcome

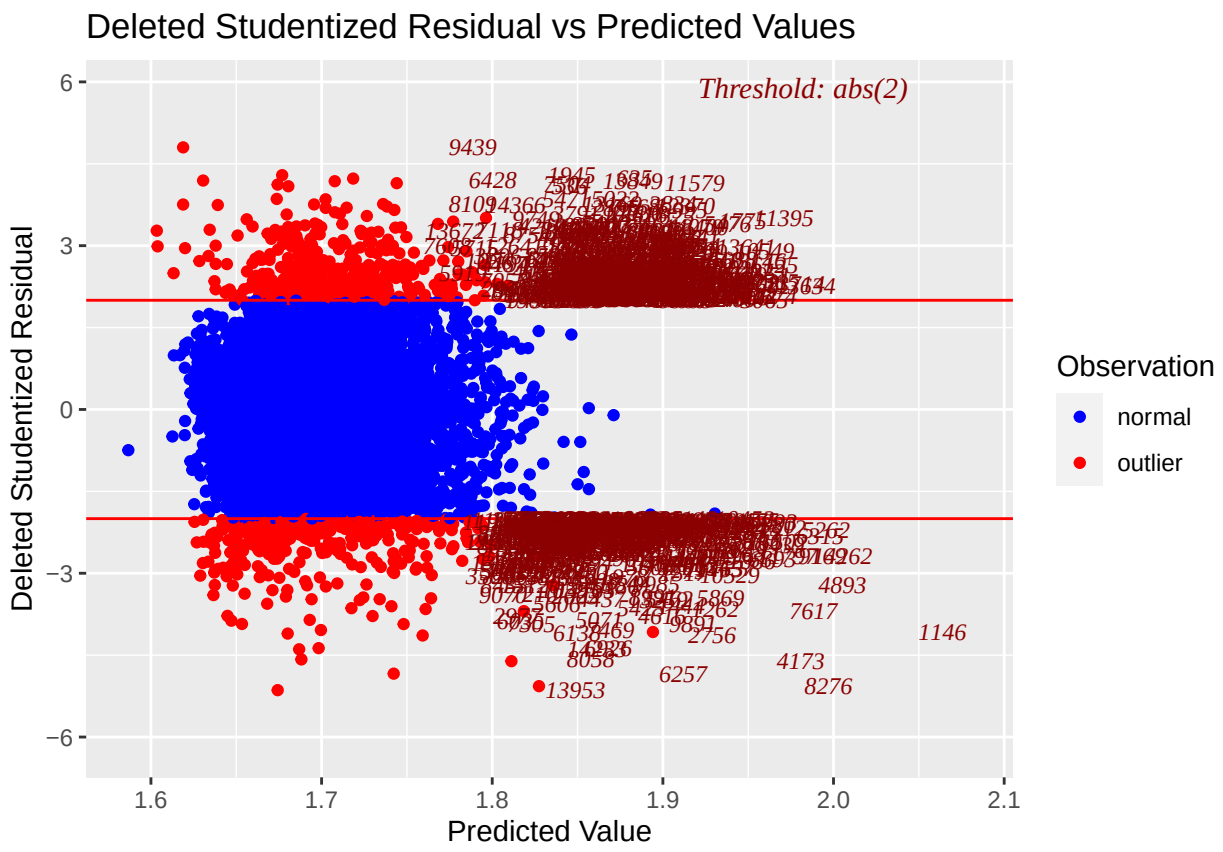
```
# Looking at the residuals using the Cook's distance chart
ols_plot_cooksd_chart(house_lm1)
```




```
# Studentized Residuals vs Leverage Plot
ols_plot_resid_lev(house_lm1)
```




```
# Deleted Studentized Residuals vs Fitted Values Plot
ols_plot_resid_stud_fit(house_lm1)
```



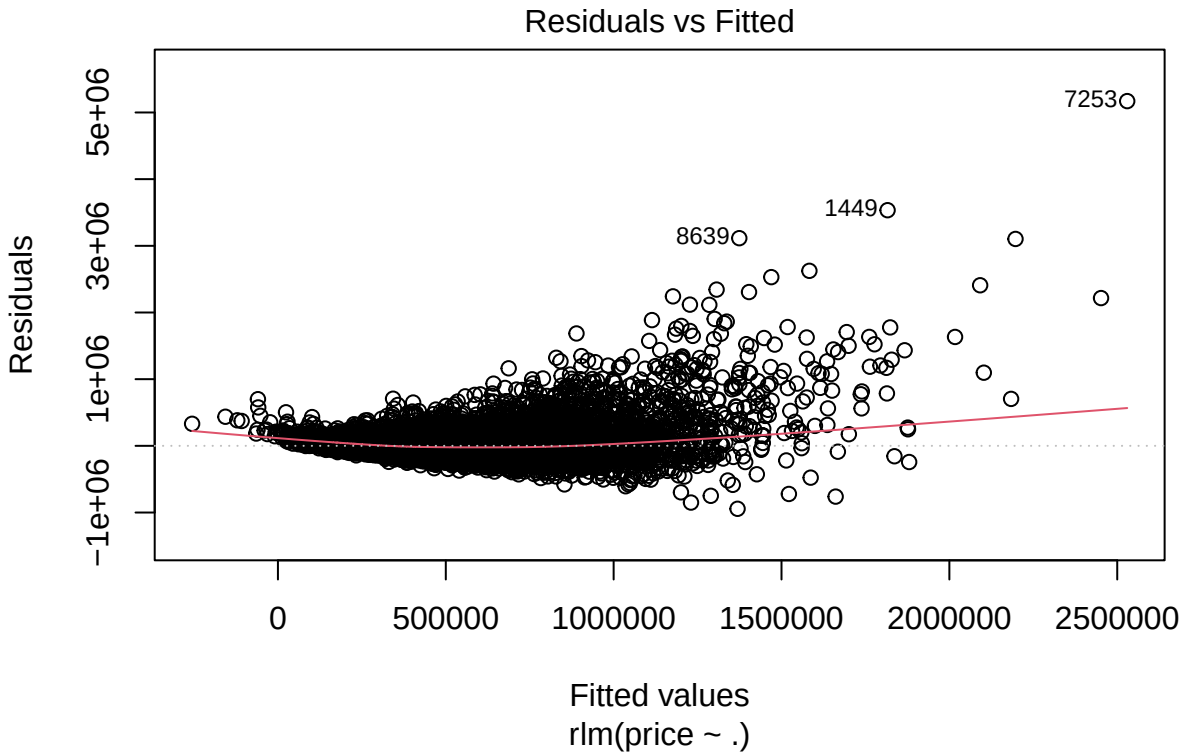
Analysis: - please add here the analysis of the outcome

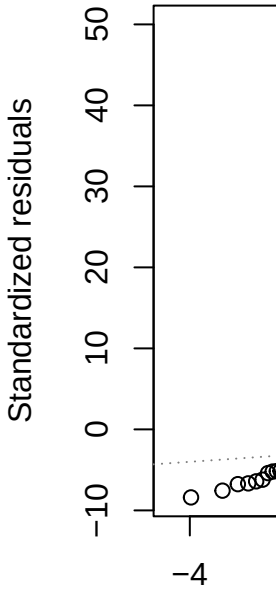
```
# Robust Regression using Huber weights
# There are influential points (see plots above)
house_lm_huber <- MASS::rlm(price ~ ., psi = psi.huber, data = train.dat)
summary(house_lm_huber)
```

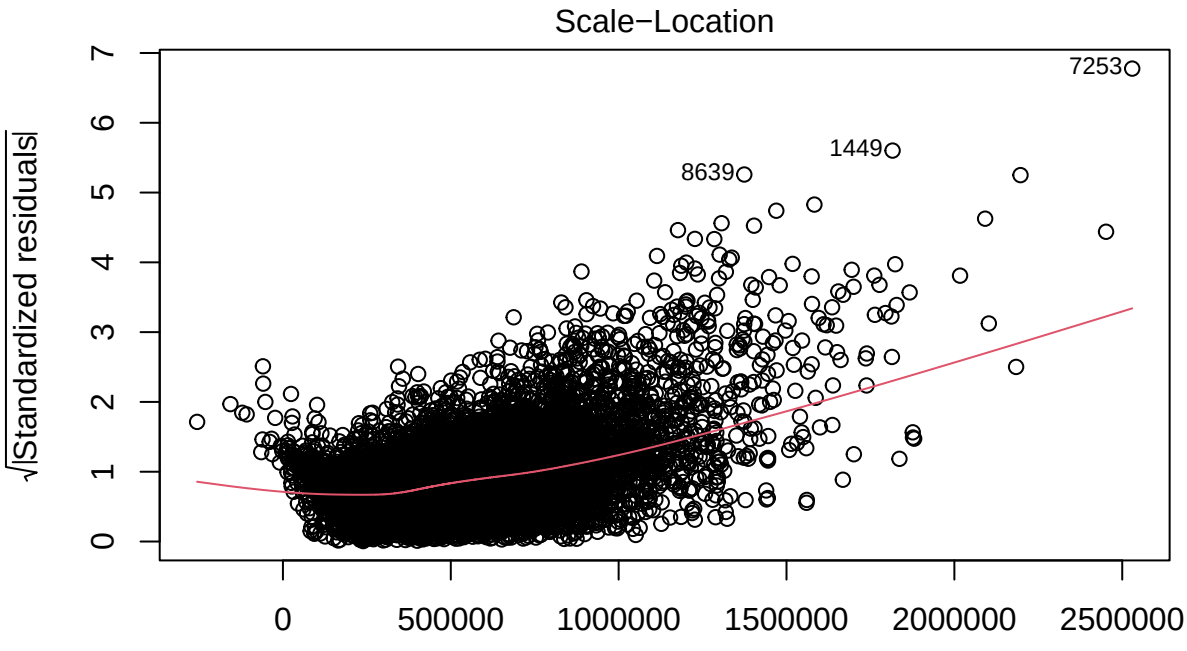
```
##
## Call: rlm(formula = price ~ ., data = train.dat, psi = psi.huber)
## Residuals:
##     Min      1Q  Median      3Q     Max
## -944042  -76206   -1941   75613 5170253
##
## Coefficients:
##               Value         Std. Error    t value
## (Intercept)   -8.156368e+07  4.474412e+06 -1.822890e+01
## bedrooms      -1.901387e+04  1.360324e+03 -1.397750e+01
## bathrooms      2.883477e+04  2.366155e+03  1.218630e+01
## sqft_lot       1.931000e-01  3.700000e-02  5.215800e+00
## floors         2.781035e+04  2.638323e+03  1.054090e+01
## waterfront     4.145041e+05  1.326761e+04  3.124180e+01
## view           4.784910e+04  1.574573e+03  3.038860e+01
## condition      2.986406e+04  1.713990e+03  1.742370e+01
## grade          8.334183e+04  1.576589e+03  5.286210e+01
## sqft_above     1.035569e+02  2.708500e+00  3.823430e+01
## sqft_basement  1.040108e+02  3.199500e+00  3.250810e+01
## lat            5.412197e+05  7.694540e+03  7.033810e+01
## long          -3.764926e+04  8.732129e+03 -4.311600e+00
## sqft_living15  4.455220e+01  2.532600e+00  1.759150e+01
## sqft_lot15    -1.187000e-01  5.340000e-02 -2.221500e+00
## year           2.508902e+04  2.150594e+03  1.166610e+01
```

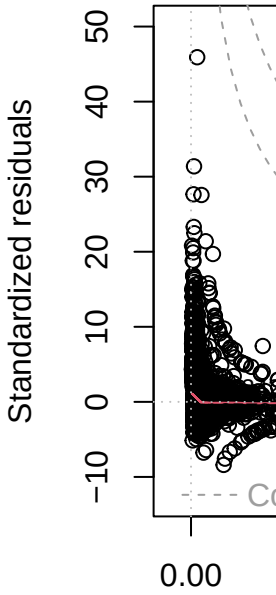
```
## renovated      1.485525e+05  5.189313e+03  2.862660e+01
## age            1.950037e+03  5.343510e+01  3.649360e+01
##
## Residual standard error: 112700 on 15111 degrees of freedom
```

```
plot(house_lm_huber)
```










```
huber_weights <- data.frame(Observation = 1:nrow(train.dat), Residual = house_lm_huber$resid, Weight = house_l

a <- huber_weights[order(house_lm_huber$w), ]

head10 <- head(a$Observation,10)
head10
```

```
##  [1] 13244  9986 11395  5127  7807 13634  8814  6125 13714  1775
```

Analysis: - please add here the analysis of the outcome

```
# Robust Regression using bisquare weights

house_lm_bisquare <- MASS::rlm(price ~ ., psi = psi.bisquare, data = train.dat)
summary(house_lm_bisquare)
```

```
##
## Call: rlm(formula = price ~ ., data = train.dat, psi = psi.bisquare)
## Residuals:
##     Min      1Q  Median      3Q     Max
## -610680  -65634    2985   80109 5515652
##
## Coefficients:
##               Value         Std. Error    t value
## (Intercept)   -6.813489e+07  4.136702e+06 -1.647080e+01
## bedrooms      -1.345810e+04  1.257652e+03 -1.070100e+01
## bathrooms      2.595323e+04  2.187567e+03  1.186400e+01
## sqft_lot       2.397000e-01  3.420000e-02  7.004000e+00
## floors         3.095547e+04  2.439193e+03  1.269090e+01
## waterfront     1.851339e+05  1.226622e+04  1.509300e+01
## view           4.059427e+04  1.455731e+03  2.788580e+01
## condition      2.723312e+04  1.584625e+03  1.718580e+01
## grade          7.391210e+04  1.457595e+03  5.070830e+01
## sqft_above     7.877650e+01  2.504100e+00  3.145950e+01
```

```
## sqft_basement  8.074940e+01  2.958000e+00  2.729820e+01
## lat            5.341132e+05  7.113788e+03  7.508140e+01
## long          -1.967416e+03  8.073064e+03 -2.437000e-01
## sqft_living15  4.757570e+01  2.341400e+00  2.031890e+01
## sqft_lot15    -7.260000e-02  4.940000e-02 -1.468300e+00
## year           2.080893e+04  1.988276e+03  1.046580e+01
## renovated      1.227366e+05  4.797645e+03  2.558270e+01
## age            1.710056e+03  4.940200e+01  3.461510e+01
##
## Residual standard error: 107200 on 15111 degrees of freedom
```

Analysis: - please add here the analysis of the outcome

```r
price.predictors <- colnames(dplyr::select(df_house, -price))

# Predictions for each model using the test dataset
predictions <- data.frame(
  price = test.dat$price,
  price.lm = predict(house_lm1, test.dat),
  price.sw.lm = predict(final_model, test.dat), # Stepwise model name = final_model
  price.ridge = predict(house_lm_ridge, s = best.lambda.ridge, newx = data.matrix(test.dat[price.predictors])),
  price.lasso = predict(house_lm_lasso, s = best.lambda.lasso, newx = data.matrix(test.dat[price.predictors])),
  price.en = predict(house_lm_enet, s = best.lambda.enet, newx = data.matrix(test.dat[price.predictors])),
  price.huber = predict(house_lm_huber, test.dat),
  price.bisquare = predict(house_lm_bisquare, test.dat)
)

# Function to calculate SSE, R2, MSE, and RMSE
calc_metrics <- function(actual, predicted) {
  sse <- sum((actual - predicted) ^ 2)
  mse <- sse / length(actual)
  rmse <- sqrt(mse) # Calculate RMSE
  sst <- sum((actual - mean(actual)) ^ 2)
  r2 <- 1 - sse / sst
  #return(c(RMSE = rmse, R2 = r2))
  return(c(SST = sst, SSE = sse, MSE = mse, RMSE = rmse, R2 = r2))
}

# function to each set of predictions
metrics <- data.frame(
  Model = c("Linear", "Stepwise", "Ridge", "Lasso", "Elastic Net", "Huber", "Bisquare"),
  do.call(rbind, lapply(2:ncol(predictions), function(i) calc_metrics(predictions$price, predictions[,i])))
)

# Display the metrics table with RMSE
metrics %>%
  dplyr::arrange(desc(R2)) %>%
  knitr::kable(caption = "R2, MSE, and RMSE of Different Models")
```

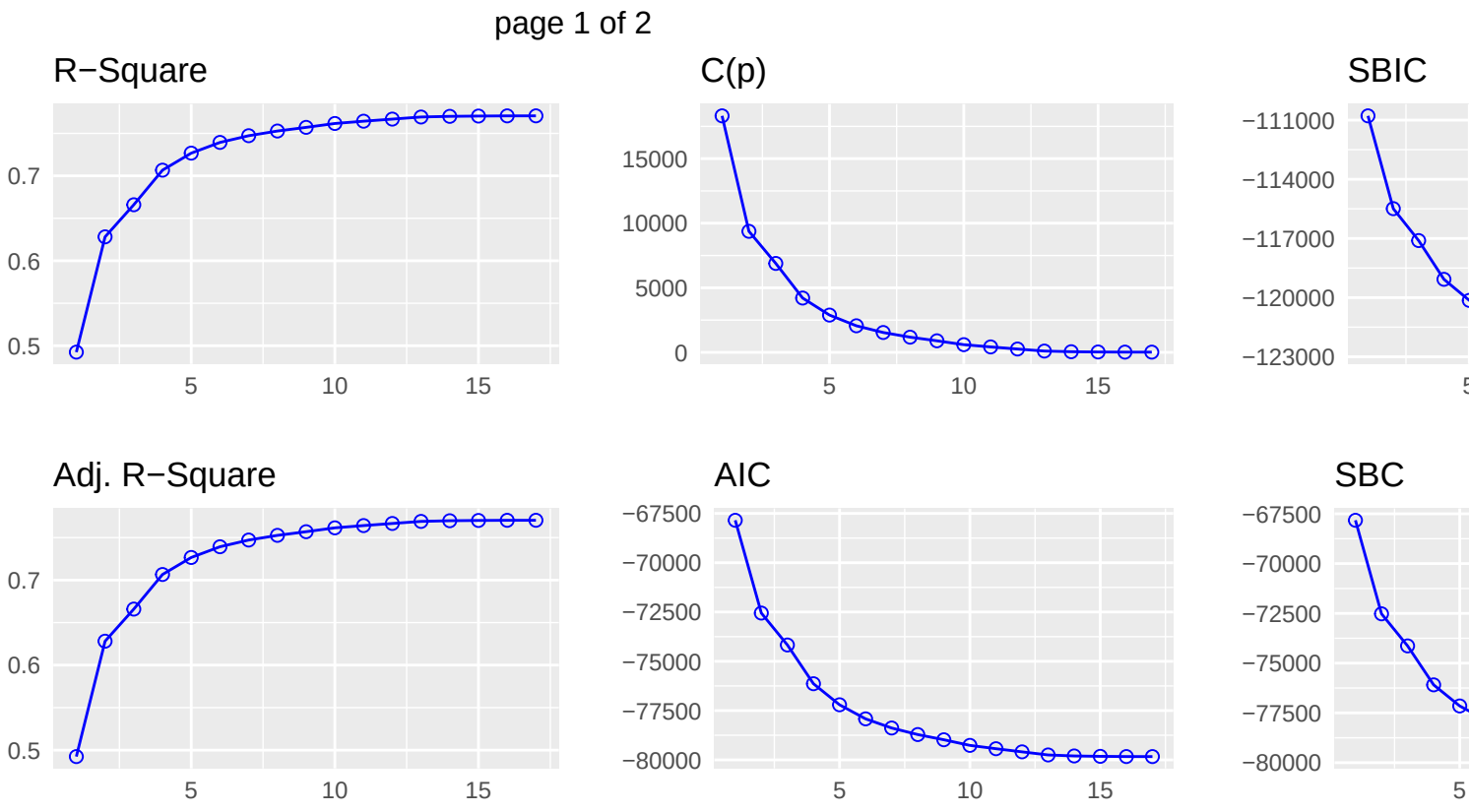
Table 1: R2, MSE, and RMSE of Different Models

Model	SST	SSE	MSE	RMSE	R2
Lasso	9.339154e+14	2.876374e+14	44361103219	210620.8	0.6920092
Elastic Net	9.339154e+14	2.877078e+14	44371968147	210646.5	0.6919337
Ridge	9.339154e+14	2.924757e+14	45107292815	212384.8	0.6868285
Huber	9.339154e+14	3.245680e+14	50056756916	223733.7	0.6524653
Bisquare	9.339154e+14	3.770719e+14	58154206866	241151.8	0.5962462
Linear	9.339154e+14	2.847785e+15	439201932857	662723.1	-2.0492969
Stepwise	9.339154e+14	2.847785e+15	439201932867	662723.1	-2.0492969

```r
# Best Subset Regression

# This is not working
k1<-ols_step_best_subset(house_lm1)
k1
```

```
##                                                                       Best Subsets Regression
## ----------------------------------------------------------------------------------------------------------------------
## Model Index    Predictors
## ----------------------------------------------------------------------------------------------------------------------
##      1         grade
##      2         grade lat
##      3         grade lat sqft_living15
##      4         grade sqft_above sqft_basement lat
##      5         view grade sqft_above sqft_basement lat
##      6         view grade sqft_above sqft_basement lat age
##      7         view grade sqft_above sqft_basement lat renovated age
##      8         view condition grade sqft_above sqft_basement lat renovated age
##      9         view condition grade sqft_above sqft_basement lat sqft_living15 renovated age
##     10         floors view condition grade sqft_above sqft_basement lat sqft_living15 renovated age
##     11         floors waterfront view condition grade sqft_above sqft_basement lat sqft_living15 renovated
##     12         floors waterfront view condition grade sqft_above sqft_basement lat sqft_living15 year renov
##     13         bathrooms floors waterfront view condition grade sqft_above sqft_basement lat sqft_living15
##     14         bathrooms sqft_lot floors waterfront view condition grade sqft_above sqft_basement lat sqft_
##     15         bedrooms bathrooms sqft_lot floors waterfront view condition grade sqft_above sqft_basement
##     16         bedrooms bathrooms sqft_lot floors waterfront view condition grade sqft_above sqft_basement
##     17         bedrooms bathrooms sqft_lot floors waterfront view condition grade sqft_above sqft_basement
## ----------------------------------------------------------------------------------------------------------------------
##
##                                                            Subsets Regression Summary
## ----------------------------------------------------------------------------------------------------------------------
##                        Adj.        Pred
## Model    R-Square    R-Square    R-Square        C(p)           AIC            SBIC            SBC           M
## ----------------------------------------------------------------------------------------------------------------------
##   1        0.4925      0.4924      0.4923    18312.3497    -67847.1985    -110784.2304    -67824.3254    9.
##   2        0.6282      0.6281       0.628     9372.2850    -72553.2356    -115490.0644    -72522.7381    7.
##   3        0.6660      0.6659      0.6657     6885.7347    -74172.0375    -117108.9752    -74133.9156    6.
##   4        0.7067      0.7066      0.7065     4204.6944    -76137.0570    -119073.5641    -76091.3108    5.
##   5        0.7267      0.7266      0.7265     2886.7820    -77205.4366    -120141.6458    -77152.0660    5.
##   6        0.7393      0.7392      0.7391     2057.2682    -77918.8277    -120854.7639    -77857.8327    5.
##   7        0.7473      0.7471      0.7469     1538.1080    -78383.0884    -121318.8098    -78314.4691    4.
##   8        0.7528      0.7526      0.7524     1176.5712    -78715.0528    -121650.5854    -78638.8091    4.
##   9        0.7571      0.7569      0.7567      894.7902    -78978.9711    -121914.3124    -78895.1030    4.
##  10        0.7616      0.7615      0.7612      596.5435    -79263.5022    -122198.5519    -79172.0098    4.
##  11        0.7643      0.7641      0.7639      421.7302    -79432.8126    -122367.6709    -79333.6959    4.
##  12        0.7668      0.7666      0.7663      259.7792    -79591.4134    -122526.0516    -79484.6722    4.
##  13        0.7693      0.7691      0.7687      100.5422    -79749.0359    -122683.4117    -79634.6703    4.
##  14        0.7701      0.7698      0.7695       49.9903    -79799.4276    -122733.7093    -79677.4377    4.
##  15        0.7704      0.7702      0.7697       29.5914    -79819.8135    -122754.0506    -79690.1992    4.
##  16        0.7706      0.7703      0.7699       18.6603    -79830.7522    -122764.9598    -79693.5135    4.
##  17        0.7706      0.7704      0.7699       18.0000    -79831.4154    -122765.6147    -79686.5524    4.
## ----------------------------------------------------------------------------------------------------------------------
## AIC: Akaike Information Criteria
##  SBIC: Sawa's Bayesian Information Criteria
##  SBC: Schwarz Bayesian Criteria
##  MSEP: Estimated error of prediction, assuming multivariate normality
##  FPE: Final Prediction Error
##  HSP: Hocking's Sp
##  APC: Amemiya Prediction Criteria
```

```
plot(k1, guide="none")
```



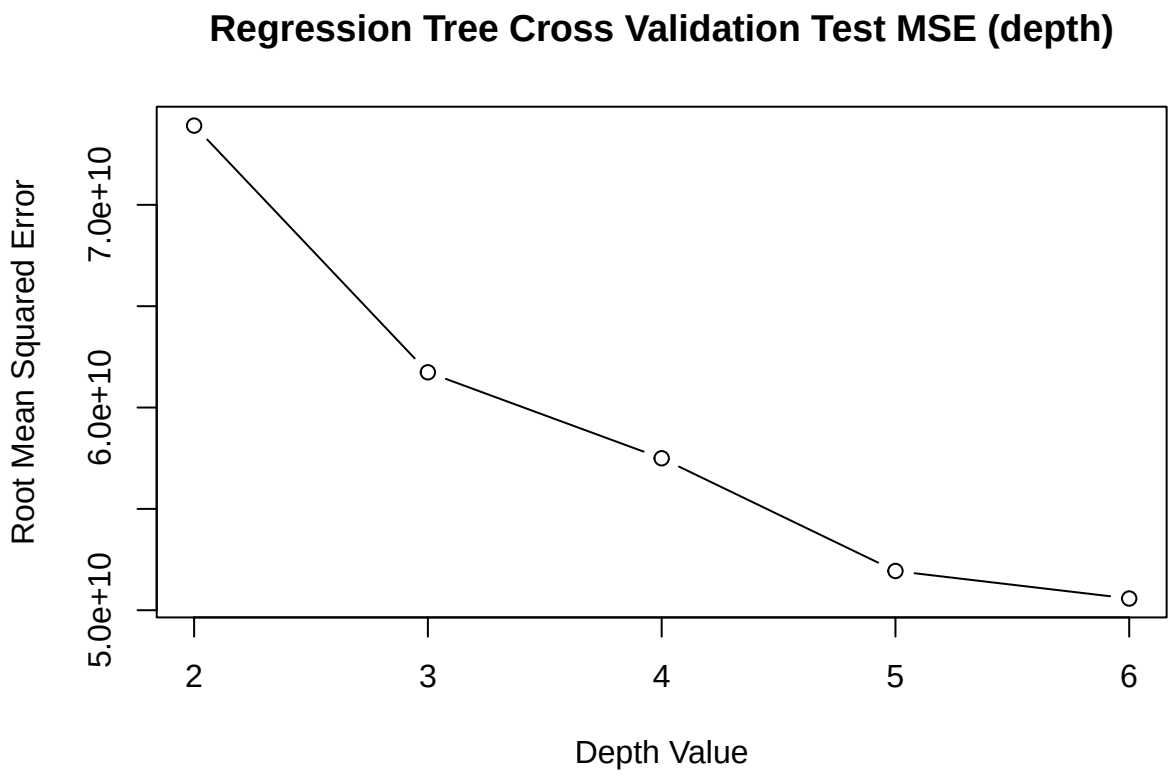
V. Challenger Models (15 points)

Build an alternative model based on one of the following approaches to predict price: regression tree, NN, or SVM. Explore using a logistic regression. Check the applicable model assumptions. Apply in-sample and out-of-sample testing, backtesting and review the comparative goodness of fit of the candidate models. Describe step by step your procedure to get to the best model and why you believe it is fit for purpose.

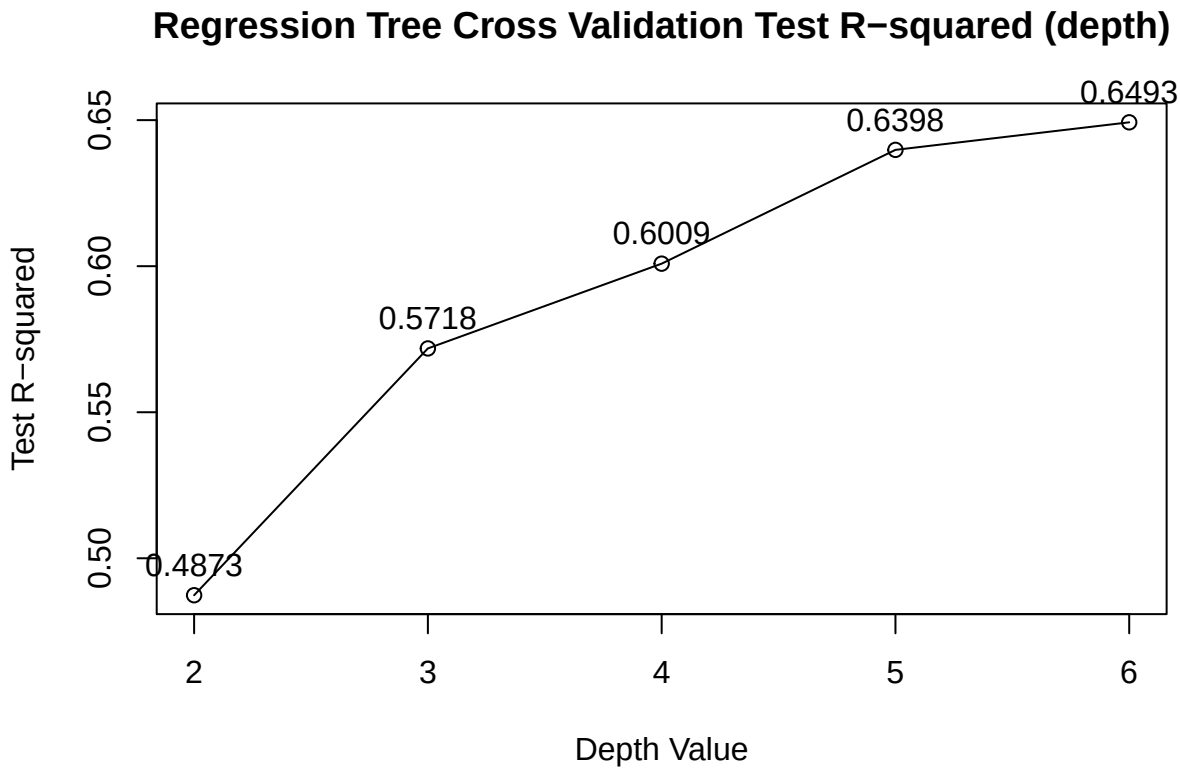
Regression Trees (Kaleo)

```
# [Kaleo]

depth_values <- c(2, 3, 4, 5, 6)

mse_values = numeric(length(depth_values))
test_rsq_values = numeric(length(depth_values))
train_rsq_values = numeric(length(depth_values))

# commented out cross validation to save computation time!!

for (i in seq_along(depth_values)) {
  depth = depth_values[i]

  model = rpart(price ~ .,
                data = train.dat,
                method = "anova",
                control=rpart.control(maxdepth=depth))
  predictions_test <- predict(model, newdata = test.dat)
  predictions_train <- predict(model, newdata = train.dat)

  mse_values[i] <- mean((predictions_test - test.dat$price)^2)
  test_rsq_values[i] = cor(predictions_test,test.dat$price)^2
  train_rsq_values[i] = cor(predictions_train,train.dat$price)^2
}
```

```
#[Kaleo]

# Plot the results
plot(depth_values, mse_values, type = "b", xlab = "Depth Value", ylab = "Root Mean Squared Error", main = "Reg
```

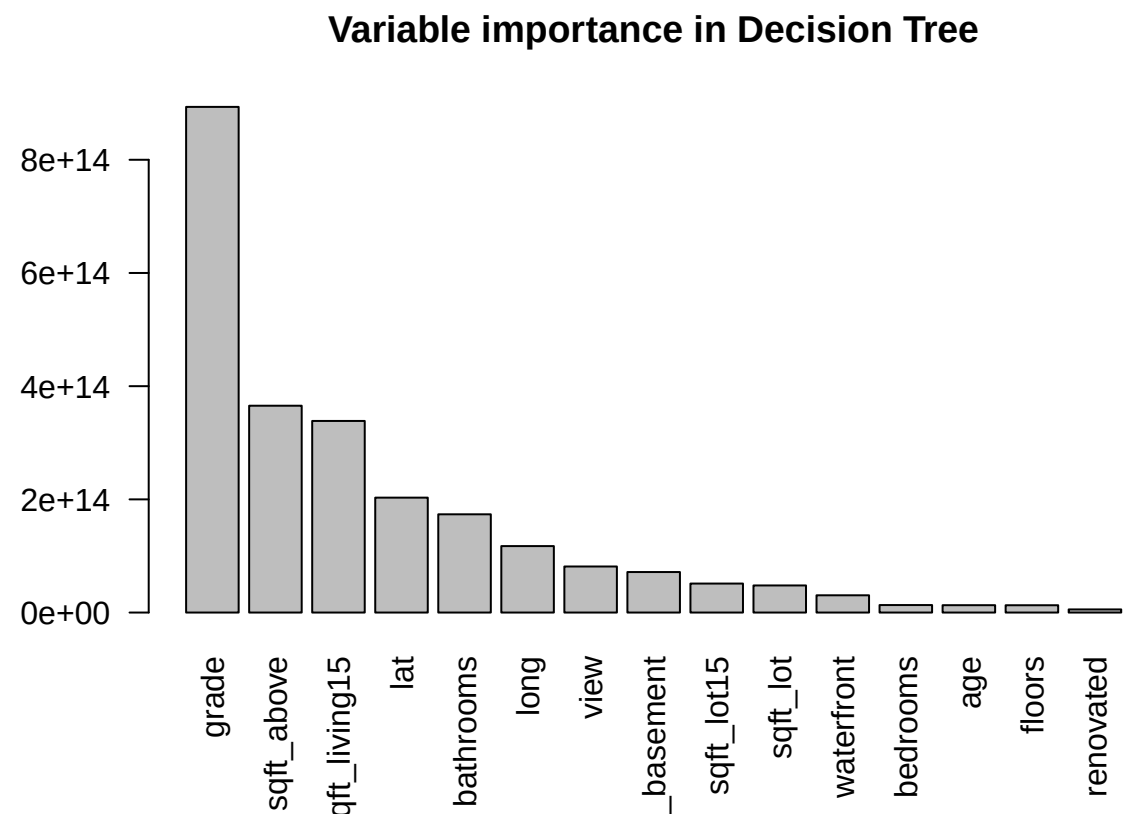



```
plot(depth_values, test_rsq_values, type="o", xlab = "Depth Value", ylab = "Test R-squared", main = "Regressio
text(depth_values,test_rsq_values,labels=round(test_rsq_values,4),pos=3,xpd=TRUE)
```




```
#[Kaleo]
```

```
#fitting decision tree with best depth

dtm = rpart(price ~ .,
              data = train.dat,
              method = "anova",
              control=rpart.control(maxdepth=5))

imp = dtm$variable.importance
dt_test_pred <- predict(dtm, newdata=test.dat)
dt_train_pred <- predict(dtm, newdata=train.dat)
dt_test_results = postResample(pred = dt_test_pred, obs = test.dat$price)
dt_train_results = postResample(pred = dt_train_pred, obs = train.dat$price)
dt_test_sse = sum((dt_test_pred - test.dat$price)^2)

#variable importance may be more reliable if considering other values from cross validation...work in progress
barplot(imp,las=2,main="Variable importance in Decision Tree")
```



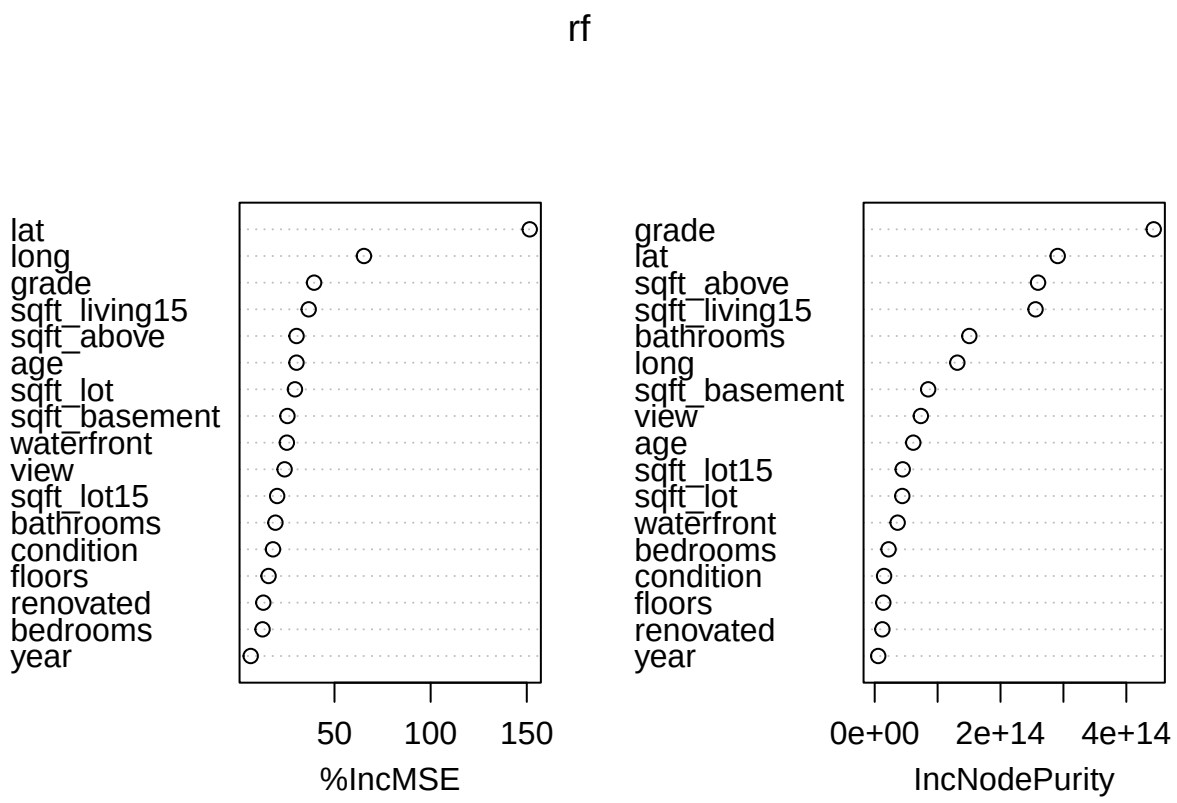

```
# [Kaleo]
#append results
results.df = rbind(results.df,data.frame(model = "Decision Tree Regression",
                           R.Squared.Train = unname(dt_train_results[2]),
                           R.Squared.Test = unname(dt_test_results[2]),
                           RMSE.test = unname(dt_test_results[1]),
                           SSE.test = dt_test_sse))
```

Random Forest Model

```
# [Kaleo]
# Fitting Random Forest model
# Cross validate number of features!!...work in progress
```

```
rf = randomForest(price ~ .,
                  data = train.dat,
                  ntree = 500,
                  importance=TRUE)
```

```
# [Kaleo]
#variable importance
randomForest::varImpPlot(rf)
```




```
#fix permuation importance - better for mixed data (ordinal, categorical, quantitative,etc)
# rfPermute(rf,train.dat,train.dat$price,na.action=na.omit,nrep = 100)
```

```
#[Kaleo]
# Random Forest Results

rf_train_pred = predict(rf, newdata = train.dat)
rf_test_pred = predict(rf, newdata = test.dat)

rf_train_results = postResample(pred = rf_train_pred, obs = train.dat$price)
rf_test_results = postResample(pred = rf_test_pred, obs = test.dat$price)
rf_test_sse = sum((rf_test_pred - test.dat$price)^2)

results.df = rbind(results.df,data.frame(model = "Random Forest",
                            R.Squared.Train = unname(rf_train_results[2]),
                            R.Squared.Test = unname(rf_test_results[2]),
                            RMSE.test = unname(rf_test_results[1]),
                            SSE.test = rf_test_sse))
```

Regression Tree Model

```
# [Luciano]
```

```r
# Build the regression tree model
tree_model <- rpart(price ~ ., data = train.dat, method = "anova")
print(summary(tree_model))
```

```
## Call:
## rpart(formula = price ~ ., data = train.dat, method = "anova")
##   n= 15129
##
##            CP nsplit rel error    xerror       xstd
## 1  0.31990114      0 1.0000000 1.0001327 0.04576254
## 2  0.09879730      1 0.6800989 0.6805090 0.03598741
## 3  0.06599331      2 0.5813016 0.5824339 0.02736495
## 4  0.03730693      3 0.5153083 0.5168828 0.02725540
## 5  0.03659808      4 0.4780013 0.4950891 0.02645514
## 6  0.02491805      5 0.4414033 0.4471264 0.02290194
## 7  0.02163966      6 0.4164852 0.4303342 0.02248832
## 8  0.02078773      7 0.3948455 0.4279233 0.02235469
## 9  0.01700988      8 0.3740578 0.4078955 0.02223293
## 10 0.01257782     10 0.3400380 0.3750467 0.01808530
## 11 0.01065821     11 0.3274602 0.3715796 0.01798923
## 12 0.01000000     12 0.3168020 0.3634293 0.01763457
##
## Variable importance
##         grade sqft_living15    sqft_above           lat     bathrooms
##            36            15            15             8             7
##          long          view sqft_basement    sqft_lot15      sqft_lot
##             5             3             3             2             2
##    waterfront      bedrooms           age        floors
##             1             1             1             1
##
## Node number 1: 15129 observations,    complexity param=0.3199011
##   mean=538713.5, MSE=1.308022e+11
##   left son=2 (12131 obs) right son=3 (2998 obs)
##   Primary splits:
##       grade         < 8.5       to the left,  improve=0.3199011, (0 missing)
##       sqft_living15 < 2835      to the left,  improve=0.2569428, (0 missing)
##       sqft_above    < 2829      to the left,  improve=0.2262678, (0 missing)
##       bathrooms     < 3.125     to the left,  improve=0.2133263, (0 missing)
##       lat           < 47.53465  to the left,  improve=0.1683547, (0 missing)
##   Surrogate splits:
##       sqft_living15 < 2644      to the left,  agree=0.885, adj=0.420, (0 split)
##       sqft_above    < 2486.5    to the left,  agree=0.883, adj=0.410, (0 split)
##       bathrooms     < 3.125     to the left,  agree=0.838, adj=0.182, (0 split)
##       sqft_basement < 1545      to the left,  agree=0.808, adj=0.033, (0 split)
##       view          < 2.5       to the left,  agree=0.806, adj=0.023, (0 split)
##
## Node number 2: 12131 observations,    complexity param=0.06599331
##   mean=437022.5, MSE=3.792793e+10
##   left son=4 (5134 obs) right son=5 (6997 obs)
##   Primary splits:
##       lat           < 47.53435  to the left,  improve=0.28383720, (0 missing)
##       grade         < 7.5       to the left,  improve=0.15573020, (0 missing)
##       sqft_living15 < 2009.5    to the left,  improve=0.11169010, (0 missing)
##       sqft_above    < 1416.5    to the left,  improve=0.08980022, (0 missing)
##       view          < 0.5       to the left,  improve=0.07459978, (0 missing)
##   Surrogate splits:
##       sqft_lot15    < 6593      to the right, agree=0.612, adj=0.083, (0 split)
##       sqft_lot      < 7131.5    to the right, agree=0.610, adj=0.078, (0 split)
##       long          < -122.2755 to the right, agree=0.606, adj=0.068, (0 split)
##       grade         < 6.5       to the left,  agree=0.601, adj=0.057, (0 split)
```

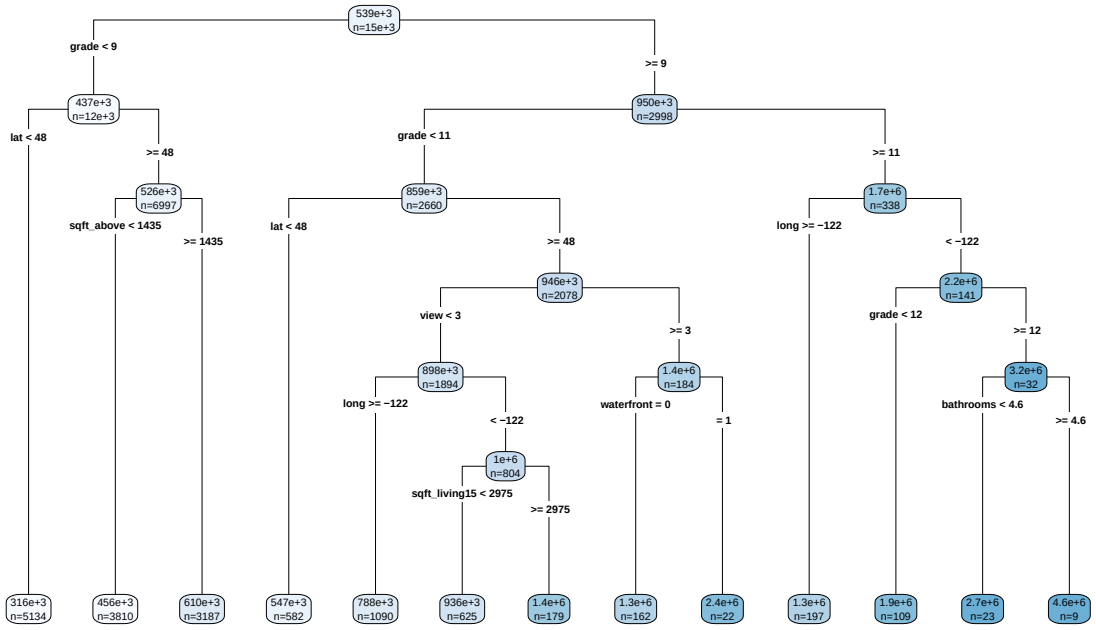
```
##       sqft_living15 < 1074      to the left,  agree=0.585, adj=0.018, (0 split)
##
## Node number 3: 2998 observations,    complexity param=0.0987973
##   mean=950192.7, MSE=2.954463e+11
##   left son=6 (2660 obs) right son=7 (338 obs)
##   Primary splits:
##       grade         < 10.5      to the left,  improve=0.2207293, (0 missing)
##       bathrooms     < 2.875     to the left,  improve=0.1574899, (0 missing)
##       sqft_above    < 3725      to the left,  improve=0.1453303, (0 missing)
##       sqft_living15 < 3665      to the left,  improve=0.1235928, (0 missing)
##       view          < 0.5       to the left,  improve=0.1212877, (0 missing)
##   Surrogate splits:
##       sqft_above    < 4255      to the left,  agree=0.913, adj=0.225, (0 split)
##       sqft_living15 < 4005      to the left,  agree=0.901, adj=0.118, (0 split)
##       bathrooms     < 4.625     to the left,  agree=0.895, adj=0.071, (0 split)
##       sqft_basement < 2230      to the left,  agree=0.891, adj=0.033, (0 split)
##
## Node number 4: 5134 observations
##   mean=315895.1, MSE=1.401515e+10
##
## Node number 5: 6997 observations,    complexity param=0.02078773
##   mean=525898.8, MSE=3.680939e+10
##   left son=10 (3810 obs) right son=11 (3187 obs)
##   Primary splits:
##       sqft_above    < 1435      to the left,  improve=0.15972090, (0 missing)
##       sqft_living15 < 1882      to the left,  improve=0.15397510, (0 missing)
##       grade         < 7.5       to the left,  improve=0.14143180, (0 missing)
##       view          < 0.5       to the left,  improve=0.09286837, (0 missing)
##       lat           < 47.69025  to the right, improve=0.09011159, (0 missing)
##   Surrogate splits:
##       grade         < 7.5       to the left,  agree=0.707, adj=0.356, (0 split)
##       bathrooms     < 2.125     to the left,  agree=0.704, adj=0.349, (0 split)
##       sqft_living15 < 1845      to the left,  agree=0.694, adj=0.327, (0 split)
##       floors        < 1.25      to the left,  agree=0.687, adj=0.314, (0 split)
##       bedrooms      < 3.5       to the left,  agree=0.661, adj=0.255, (0 split)
##
## Node number 6: 2660 observations,    complexity param=0.03659808
##   mean=859162.2, MSE=1.613064e+11
##   left son=12 (582 obs) right son=13 (2078 obs)
##   Primary splits:
##       lat           < 47.5231   to the left,  improve=0.1687914, (0 missing)
##       grade         < 9.5       to the left,  improve=0.1183673, (0 missing)
##       sqft_basement < 655       to the left,  improve=0.1154193, (0 missing)
##       view          < 2.5       to the left,  improve=0.1112810, (0 missing)
##       age           < 53.5      to the left,  improve=0.1100060, (0 missing)
##   Surrogate splits:
##       long       < -121.8495 to the right, agree=0.791, adj=0.043, (0 split)
##       sqft_lot   < 163568    to the right, agree=0.788, adj=0.029, (0 split)
##       sqft_lot15 < 89951     to the right, agree=0.786, adj=0.022, (0 split)
##       sqft_above < 750       to the left,  agree=0.782, adj=0.003, (0 split)
##
## Node number 7: 338 observations,    complexity param=0.03730693
##   mean=1666586, MSE=7.726704e+11
##   left son=14 (197 obs) right son=15 (141 obs)
##   Primary splits:
##       long          < -122.187  to the right, improve=0.2826856, (0 missing)
##       grade         < 11.5      to the left,  improve=0.1839557, (0 missing)
##       sqft_basement < 1135      to the left,  improve=0.1830490, (0 missing)
##       bathrooms     < 5.375     to the left,  improve=0.1519642, (0 missing)
##       sqft_above    < 6087.5    to the left,  improve=0.1236678, (0 missing)
```

```
##   Surrogate splits:
##       sqft_basement < 75        to the left,  agree=0.713, adj=0.312, (0 split)
##       sqft_living15 < 3245      to the right, agree=0.707, adj=0.298, (0 split)
##       sqft_lot      < 10221     to the right, agree=0.689, adj=0.255, (0 split)
##       view          < 0.5       to the left,  agree=0.680, adj=0.234, (0 split)
##       sqft_lot15    < 9290.5    to the right, agree=0.680, adj=0.234, (0 split)
##
## Node number 10: 3810 observations
##   mean=455771.3, MSE=1.935129e+10
##
## Node number 11: 3187 observations
##   mean=609735, MSE=4.477249e+10
##
## Node number 12: 582 observations
##   mean=547372.2, MSE=3.137136e+10
##
## Node number 13: 2078 observations,    complexity param=0.02491805
##   mean=946487.4, MSE=1.628454e+11
##   left son=26 (1894 obs) right son=27 (184 obs)
##   Primary splits:
##       view          < 2.5       to the left,  improve=0.1457197, (0 missing)
##       long          < -122.1865 to the right, improve=0.1414144, (0 missing)
##       waterfront    < 0.5       to the left,  improve=0.1348135, (0 missing)
##       sqft_basement < 655       to the left,  improve=0.1163238, (0 missing)
##       grade         < 9.5       to the left,  improve=0.1098400, (0 missing)
##   Surrogate splits:
##       waterfront    < 0.5       to the left,  agree=0.921, adj=0.109, (0 split)
##       sqft_living15 < 5645      to the left,  agree=0.913, adj=0.022, (0 split)
##       sqft_basement < 2260      to the left,  agree=0.912, adj=0.011, (0 split)
##
## Node number 14: 197 observations
##   mean=1271197, MSE=2.120492e+11
##
## Node number 15: 141 observations,    complexity param=0.02163966
##   mean=2219010, MSE=1.032355e+12
##   left son=30 (109 obs) right son=31 (32 obs)
##   Primary splits:
##       grade         < 11.5      to the left,  improve=0.2941898, (0 missing)
##       sqft_above    < 4755      to the left,  improve=0.2563880, (0 missing)
##       bathrooms     < 4.625     to the left,  improve=0.2518937, (0 missing)
##       bedrooms      < 4.5       to the left,  improve=0.1466578, (0 missing)
##       sqft_basement < 1275      to the left,  improve=0.1418993, (0 missing)
##   Surrogate splits:
##       sqft_above < 4815      to the left,  agree=0.837, adj=0.281, (0 split)
##       sqft_lot15 < 23181     to the left,  agree=0.837, adj=0.281, (0 split)
##       sqft_lot   < 23800.5   to the left,  agree=0.816, adj=0.187, (0 split)
##       bathrooms  < 5.375     to the left,  agree=0.801, adj=0.125, (0 split)
##       waterfront < 0.5       to the left,  agree=0.794, adj=0.094, (0 split)
##
## Node number 26: 1894 observations,    complexity param=0.01700988
##   mean=898473.7, MSE=1.148911e+11
##   left son=52 (1090 obs) right son=53 (804 obs)
##   Primary splits:
##       long          < -122.1865 to the right, improve=0.14508050, (0 missing)
##       bathrooms     < 2.875     to the left,  improve=0.11187630, (0 missing)
##       age           < 53.5      to the left,  improve=0.10538170, (0 missing)
##       grade         < 9.5       to the left,  improve=0.09258197, (0 missing)
##       sqft_basement < 655       to the left,  improve=0.09067083, (0 missing)
##   Surrogate splits:
##       sqft_living15 < 2245      to the right, agree=0.718, adj=0.336, (0 split)
```

```
##       sqft_basement < 30        to the left,  agree=0.708, adj=0.312, (0 split)
##       sqft_lot      < 5167      to the right, agree=0.680, adj=0.245, (0 split)
##       sqft_lot15    < 5164.5    to the right, agree=0.677, adj=0.239, (0 split)
##       age           < 47.5      to the left,  agree=0.676, adj=0.236, (0 split)
##
## Node number 27: 184 observations,    complexity param=0.01065821
##   mean=1440716, MSE=3.884696e+11
##   left son=54 (162 obs) right son=55 (22 obs)
##   Primary splits:
##       waterfront    < 0.5       to the left,  improve=0.2950764, (0 missing)
##       sqft_above    < 2855      to the left,  improve=0.1815718, (0 missing)
##       grade         < 9.5       to the left,  improve=0.1802047, (0 missing)
##       sqft_living15 < 3215      to the left,  improve=0.1208372, (0 missing)
##       bathrooms     < 3.875     to the left,  improve=0.1169908, (0 missing)
##   Surrogate splits:
##       sqft_lot15 < 27499     to the left,  agree=0.891, adj=0.091, (0 split)
##       sqft_lot   < 47675     to the left,  agree=0.886, adj=0.045, (0 split)
##
## Node number 30: 109 observations
##   mean=1920410, MSE=4.611213e+11
##
## Node number 31: 32 observations,    complexity param=0.01257782
##   mean=3236117, MSE=1.639904e+12
##   left son=62 (23 obs) right son=63 (9 obs)
##   Primary splits:
##       bathrooms     < 4.625     to the left,  improve=0.4743101, (0 missing)
##       sqft_above    < 4740      to the left,  improve=0.2475731, (0 missing)
##       sqft_basement < 1875      to the left,  improve=0.2296718, (0 missing)
##       bedrooms      < 4.5       to the left,  improve=0.2137045, (0 missing)
##       long          < -122.3355 to the left,  improve=0.1994171, (0 missing)
##   Surrogate splits:
##       sqft_basement < 2315      to the left,  agree=0.844, adj=0.444, (0 split)
##       sqft_above    < 6115      to the left,  agree=0.812, adj=0.333, (0 split)
##       renovated     < 0.5       to the left,  agree=0.781, adj=0.222, (0 split)
##       age           < 14.5      to the right, agree=0.781, adj=0.222, (0 split)
##       bedrooms      < 5.5       to the left,  agree=0.750, adj=0.111, (0 split)
##
## Node number 52: 1090 observations
##   mean=787591.3, MSE=2.770731e+10
##
## Node number 53: 804 observations,    complexity param=0.01700988
##   mean=1048799, MSE=1.938218e+11
##   left son=106 (625 obs) right son=107 (179 obs)
##   Primary splits:
##       sqft_living15 < 2975      to the left,  improve=0.2294247, (0 missing)
##       sqft_above    < 3205      to the left,  improve=0.2059166, (0 missing)
##       grade         < 9.5       to the left,  improve=0.1618116, (0 missing)
##       sqft_lot      < 3521      to the left,  improve=0.1202496, (0 missing)
##       lat           < 47.7009   to the right, improve=0.1199481, (0 missing)
##   Surrogate splits:
##       sqft_lot15    < 11788.5   to the left,  agree=0.821, adj=0.196, (0 split)
##       sqft_lot      < 16036     to the left,  agree=0.802, adj=0.112, (0 split)
##       sqft_basement < 1245      to the left,  agree=0.796, adj=0.084, (0 split)
##       sqft_above    < 4195      to the left,  agree=0.789, adj=0.050, (0 split)
##       bedrooms      < 6.5       to the left,  agree=0.780, adj=0.011, (0 split)
##
## Node number 54: 162 observations
##   mean=1315949, MSE=2.427098e+11
##
## Node number 55: 22 observations
```

```
##   mean=2359455, MSE=5.03083e+11
##
## Node number 62: 23 observations
##   mean=2684424, MSE=5.010391e+11
##
## Node number 63: 9 observations
##   mean=4646000, MSE=1.784743e+12
##
## Node number 106: 625 observations
##   mean=935947.6, MSE=1.192614e+11
##
## Node number 107: 179 observations
##   mean=1442834, MSE=2.544272e+11
##
## n= 15129
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##   1) root 15129 1.978906e+15  538713.5
##     2) grade< 8.5 12131 4.601037e+14  437022.5
##       4) lat< 47.53435 5134 7.195380e+13  315895.1 *
##       5) lat>=47.53435 6997 2.575553e+14  525898.8
##        10) sqft_above< 1435 3810 7.372842e+13  455771.3 *
##        11) sqft_above>=1435 3187 1.426899e+14  609735.0 *
##     3) grade>=8.5 2998 8.857481e+14  950192.7
##       6) grade< 10.5 2660 4.290749e+14  859162.2
##        12) lat< 47.5231 582 1.825813e+13  547372.2 *
##        13) lat>=47.5231 2078 3.383926e+14  946487.4
##          26) view< 2.5 1894 2.176038e+14  898473.7
##            52) long>=-122.1865 1090 3.020097e+13  787591.3 *
##            53) long< -122.1865 804 1.558327e+14 1048799.0
##             106) sqft_living15< 2975 625 7.453838e+13  935947.6 *
##             107) sqft_living15>=2975 179 4.554247e+13 1442834.0 *
##          27) view>=2.5 184 7.147840e+13 1440716.0
##            54) waterfront< 0.5 162 3.931899e+13 1315949.0 *
##            55) waterfront>=0.5 22 1.106783e+13 2359455.0 *
##       7) grade>=10.5 338 2.611626e+14 1666586.0
##        14) long>=-122.187 197 4.177369e+13 1271197.0 *
##        15) long< -122.187 141 1.455620e+14 2219010.0
##          30) grade< 11.5 109 5.026223e+13 1920410.0 *
##          31) grade>=11.5 32 5.247692e+13 3236117.0
##            62) bathrooms< 4.625 23 1.152390e+13 2684424.0 *
##            63) bathrooms>=4.625 9 1.606269e+13 4646000.0 *
```

```
#rpart.plot(tree_model, main="Regression Tree", type = 4, extra = 1, under=TRUE, faclen=0, cex=0.75)
rpart.plot(tree_model, main="Regression Tree", type = 4, extra = 1)
```




```
# Prediction and Performance
tree_predictions <- predict(tree_model, test.dat)
tree_rmse <- sqrt(mean((tree_predictions - test.dat$price)^2))
print(paste("Tree RMSE:", tree_rmse))
```

```
## [1] "Tree RMSE: 224896.187813564"
```

Neural Network Model

```
# [Luciano]

# Build the neural network model
nn_model <- nnet(price ~ ., data = train.dat, size = 5, linout = TRUE, trace = FALSE, MaxNWts = 5000)
print(nn_model)
```

```
## a 17-5-1 network with 96 weights
## inputs: bedrooms bathrooms sqft_lot floors waterfront view condition grade sqft_above sqft_basement lat lon
## output(s): price
## options were - linear output units
```

```
# Prediction and Performance
nn_predictions <- predict(nn_model, test.dat, type = "raw")
nn_rmse <- sqrt(mean((nn_predictions - test.dat$price)^2))
print(paste("NN RMSE:", nn_rmse))
```

```
## [1] "NN RMSE: 379545.688459742"
```

Support Vector Machine (SVM) Model

```
# [Luciano]

# Build the SVM model
```

```r
svm_model <- svm(price ~ ., data = train.dat)
print(summary(svm_model))
```

```
##
## Call:
## svm(formula = price ~ ., data = train.dat)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.05882353
##     epsilon:  0.1
##
##
## Number of Support Vectors:  9020
```

```r
# Prediction and Performance
svm_predictions <- predict(svm_model, test.dat)
svm_rmse <- sqrt(mean((svm_predictions - test.dat$price)^2))
print(paste("SVM RMSE:", svm_rmse))
```

```
## [1] "SVM RMSE: 195393.382261799"
```

Neural Network - Seymur

```r
normalize <- function(x) { return((x - min(x)) / (max(x) - min(x))) }

train.dat.norm <- as.data.frame(lapply(train.dat, normalize))
test.dat.norm <- as.data.frame(lapply(test.dat, normalize))
```

```r
house_NN <- neuralnet(price ~ ., data = train.dat.norm, hidden = c(2, 2), linear.output = TRUE)
```

```r
model_results <- compute(house_NN, test.dat.norm[1:(ncol(test.dat.norm) - 1)])
predicted_price <- model_results$net.result

rmse_value <- rmse(predicted_price, test.dat.norm$price)
r_squared_value <- r_squared(predicted_price, test.dat.norm$price)
sse_value <- sse(predicted_price, test.dat.norm$price)

cat("RMSE:", rmse_value, "\n")
```

```
## RMSE: 0.04926162
```

```r
cat("R-squared:", r_squared_value, "\n")
```

```
## R-squared: 0.2504242
```

```r
cat("SSE:", sse_value, "\n")
```

```
## SSE: 15.73477
```

VI. Model Limitation and Assumptions (15 points)

Based on the performances on both train and test data sets, determine your primary (champion) model and the other model which would be your benchmark model. Validate your models using the test sample. Do the residuals look normal? Does it matter given your technique? How is the prediction performance using Pseudo R^2, SSE, RMSE? Benchmark the model against alternatives. How good is the relative fit? Are there any serious violations of the model assumptions? Has the model had issues or limitations that the user must know? (Which assumptions are needed to support the Champion model?)

```
# [Kaleo]
# Results dataframe...We're supposed to use Pseudo R-squared, SSE, RMSE, as seen above.
# We'll have to look into 'Pseudo R-squared' most likely

results.df
```

```
##                                   model R.Squared.Train R.Squared.Test
## 1 Linear Regression Test Data Predictions       0.6970168      0.6921461
## 2                Decision Tree Regression       0.6651315      0.6398236
## 3                           Random Forest       0.9774039      0.8685456
##   RMSE.test     SSE.test
## 1  210589.7 2.875525e+14
## 2  227895.6 3.367556e+14
## 3  141084.8 1.290636e+14
```

VII. Ongoing Model Monitoring Plan (5 points)

How would you picture the model needing to be monitored, which quantitative thresholds and triggers would you set to decide when the model needs to be replaced? What are the assumptions that the model must comply with for its continuous use?

VIII. Conclusion (5 points)

Summarize your results here. What is the best model for the data and why?

Bibliography (7 points)

Please include all references, articles and papers in this section.

Appendix (3 points)

Please add any additional supporting graphs, plots and data analysis.