HARVARD EXTENSION SCHOOL

CSCI E-106 - Data Modeling - Final Project

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

03 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

House Sales in King County, USA data to be used in the Final Project

Variable	Description
id	**Unique ID for each home sold (it is not a predictor)**
date	*Date of the home sale*
price	*Price of each home sold*
bedrooms	*Number of bedrooms*
bathrooms	*Number of bathrooms, where ".5" accounts for a bathroom with a toilet but no shower*
sqft_living	*Square footage of the apartment interior living space*
sqft_lot	*Square footage of the land space*
floors	*Number of floors*
waterfront	*A dummy variable for whether the apartment was overlooking the waterfront or not*
view	*An index from 0 to 4 of how good the view of the property was*
condition	*An index from 1 to 5 on the condition of the apartment,*
grade	*An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 has a high-quality level of construction and design.*
sqft_above	*The square footage of the interior housing space that is above ground level*
sqft_basement	*The square footage of the interior housing space that is below ground level*
yr_built	*The year the house was initially built*
yr_renovated	*The year of the house's last renovation*
zipcode	*What zipcode area the house is in*
lat	*Latitude*
long	*Longitude*
sqft_living15	*The square footage of interior housing living space for the nearest 15 neighbors*
sqft_lot15	*The square footage of the land lots of the nearest 15 neighbors*

Instructions:

0. Join a team with your fellow students with appropriate size (Four Students total)
1. Load and Review the dataset named "KC_House_Sales'csv
2. 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.
3. Investigate the data and combine the level of categorical variables if needed and drop variables as needed. For example, you can drop id, Latitude, Longitude, etc.
4. Build a regression model to predict price.
5. Create scatter plots and a correlation matrix for the train data set. Interpret the possible relationship between the response.
6. Build the best multiple linear models by using the stepwise selection method. Compare the performance of the best two linear models.
7. Make sure that model assumption(s) are checked for the final model. Apply remedy measures (transformation, etc.) that helps satisfy the assumptions.
8. Investigate unequal variances and multicollinearity. If necessary, apply remedial methods (WLS, Ridge, Elastic Net, Lasso, etc.).
9. Build an alternative model based on one of the following approaches to predict price: regression tree, NN, or SVM. Check the applicable model assumptions. Explore using a logistic regression.
10. Use the test data set to assess the model performances from above.
11. 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.
12. Create a model development document that describes the model following this template, input the name of the authors, Harvard IDs, the name of the Group, all of your code and calculations, etc..:

Due Date: December 18th, 2023 at 11:59 pm EST

Notes No typographical errors, grammar mistakes, or misspelled words, use English language All tables need to be numbered and describe their content in the body of the document All figures/graphs need to be numbered and describe their content All results must be accurate and clearly explained for a casual reviewer to fully understand their purpose and impact Submit both the RMD markdown file and PDF with the sections with appropriate explanations. A more formal document in Word can be used in place of the pdf file but must include all appropriate explanations.

Executive Summary

This section will describe the model usage, your conclusions and any regulatory and internal requirements. In a real world scneario, 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-progress —

This project aims to develop a predictive model for house prices in King County, USA. The model's purpose is to provide an analytical tool for understanding the key factors influencing property values in this area, which is significant for various stakeholders in the real estate sector.

Our primary data source is the 'KC_House_Sales' dataset, which includes detailed information on various house attributes. **(Add here the hypothesis of initial exploration on house prices)**.

The methodology adopted for this project combines conventional statistical techniques with modern data analytics methods. Initial models will be based on linear regression, with further exploration into more complex approaches like neural networks and decision trees, depending on their performance in preliminary analyses.

To ensure robust model training and validation, the dataset has been divided into a training set (70%) and a test set (30%). This division is crucial for evaluating the model's effectiveness in making predictions on new, unseen data.

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

[Add here initial observation analysis]

```
df_house = read.csv("KC_House_Sales.csv")
cat("Number of NA values in dataframe:",sum(is.na(df_house))) #no NA values
```

```
## 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

[Add here info about the data and what was performed]

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

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
```

```
# [Kaleo]
#converting all to numeric...ignores date column
ndf_house = df_house[sapply(df_house, is.numeric)]
# use df_house instead, unless you have a specific reason!
head(ndf_house)
```

```
##     price bedrooms bathrooms sqft_living sqft_lot floors waterfront view
## 1  221900        3      1.00        1180     5650      1          0    0
## 2  538000        3      2.25        2570     7242      2          0    0
## 3  180000        2      1.00         770    10000      1          0    0
## 4  604000        4      3.00        1960     5000      1          0    0
## 5  510000        3      2.00        1680     8080      1          0    0
## 6 1225000        4      4.50        5420   101930      1          0    0
##   condition grade sqft_above sqft_basement yr_built yr_renovated zipcode
## 1         3     7       1180             0     1955            0   98178
## 2         3     7       2170           400     1951         1991   98125
## 3         3     6        770             0     1933            0   98028
## 4         5     7       1050           910     1965            0   98136
## 5         3     8       1680             0     1987            0   98074
## 6         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
```

Stat Summary - in progress

[Add here the initial analysis of the summary, if applicable]

```
str(ndf_house)
```

```
## 'data.frame':    21613 obs. of  22 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_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
##  $ 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 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 98038 ...
##  $ 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 ...
##  $ month        : num  10 12 2 12 2 5 6 1 4 3 ...
##  $ day          : num  13 9 25 9 18 12 27 15 15 12 ...
```

```
summary(ndf_house)
```

```
##      price            bedrooms        bathrooms      sqft_living
##  Min.   :  75000   Min.   : 0.000   Min.   :0.000   Min.   :  290
##  1st Qu.: 321950   1st Qu.: 3.000   1st Qu.:1.750   1st Qu.: 1427
##  Median : 450000   Median : 3.000   Median :2.250   Median : 1910
##  Mean   : 540088   Mean   : 3.371   Mean   :2.115   Mean   : 2080
##  3rd Qu.: 645000   3rd Qu.: 4.000   3rd Qu.:2.500   3rd Qu.: 2550
##  Max.   :7700000   Max.   :33.000   Max.   :8.000   Max.   :13540
##     sqft_lot           floors        waterfront           view
##  Min.   :    520   Min.   :1.000   Min.   :0.000000   Min.   :0.0000
##  1st Qu.:   5040   1st Qu.:1.000   1st Qu.:0.000000   1st Qu.:0.0000
##  Median :   7618   Median :1.500   Median :0.000000   Median :0.0000
##  Mean   :  15107   Mean   :1.494   Mean   :0.007542   Mean   :0.2343
##  3rd Qu.:  10688   3rd Qu.:2.000   3rd Qu.:0.000000   3rd Qu.:0.0000
##  Max.   :1651359   Max.   :3.500   Max.   :1.000000   Max.   :4.0000
##    condition         grade           sqft_above   sqft_basement
##  Min.   :1.000   Min.   : 1.000   Min.   : 290   Min.   :   0.0
##  1st Qu.:3.000   1st Qu.: 7.000   1st Qu.:1190   1st Qu.:   0.0
##  Median :3.000   Median : 7.000   Median :1560   Median :   0.0
##  Mean   :3.409   Mean   : 7.657   Mean   :1788   Mean   : 291.5
##  3rd Qu.:4.000   3rd Qu.: 8.000   3rd Qu.:2210   3rd Qu.: 560.0
##  Max.   :5.000   Max.   :13.000   Max.   :9410   Max.   :4820.0
##     yr_built     yr_renovated       zipcode           lat
##  Min.   :1900   Min.   :   0.0   Min.   :98001   Min.   :47.16
##  1st Qu.:1951   1st Qu.:   0.0   1st Qu.:98033   1st Qu.:47.47
##  Median :1975   Median :   0.0   Median :98065   Median :47.57
##  Mean   :1971   Mean   :  84.4   Mean   :98078   Mean   :47.56
##  3rd Qu.:1997   3rd Qu.:   0.0   3rd Qu.:98118   3rd Qu.:47.68
##  Max.   :2015   Max.   :2015.0   Max.   :98199   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
##      month             day
##  Min.   : 1.000   Min.   : 1.00
##  1st Qu.: 4.000   1st Qu.: 8.00
##  Median : 6.000   Median :16.00
##  Mean   : 6.574   Mean   :15.69
##  3rd Qu.: 9.000   3rd Qu.:23.00
##  Max.   :12.000   Max.   :31.00
```

```
summary(ndf_house$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##    75000   321950   450000   540088   645000 7700000
```

Graphical Analysis - in progress - Seymur

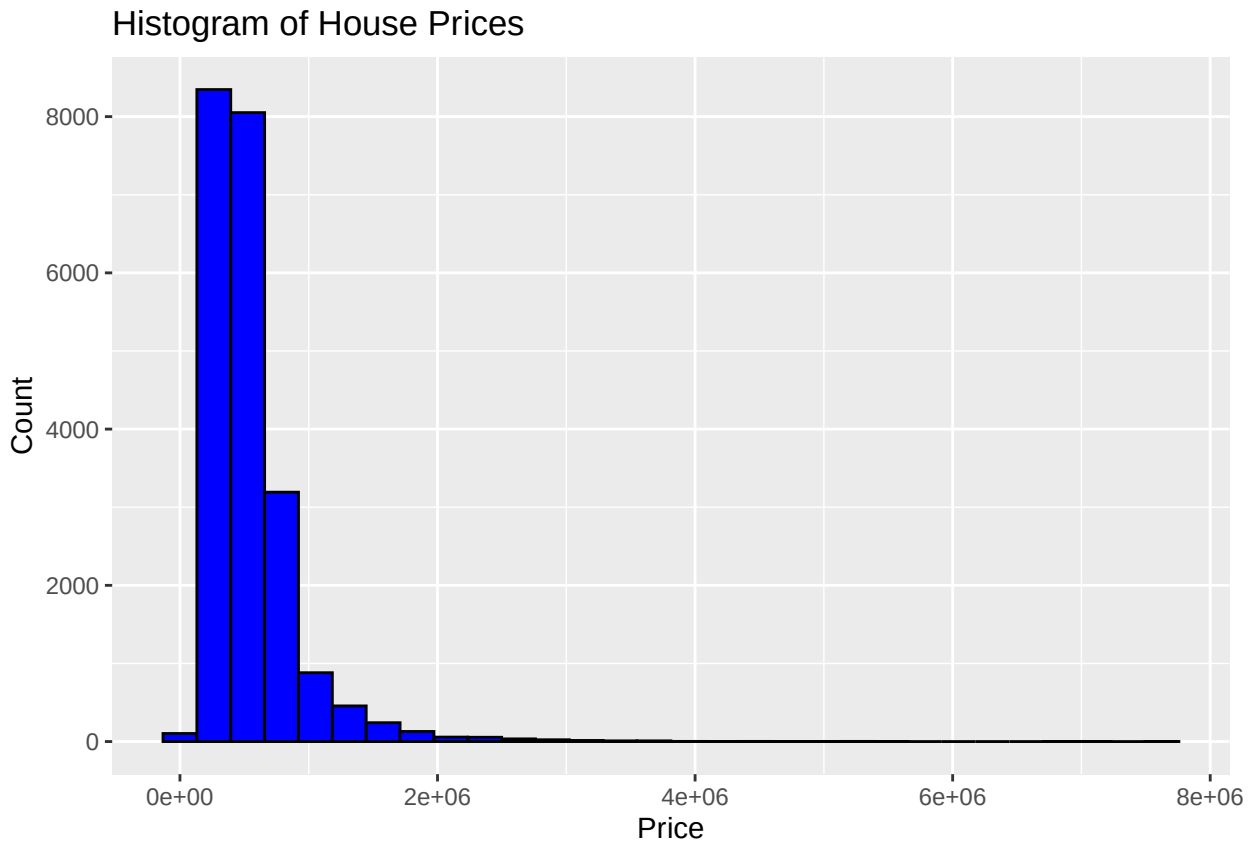
The pairwise scatter plot shows the correlation between highly correlated variables extracted from heatmap correlation matrix.

- 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.

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

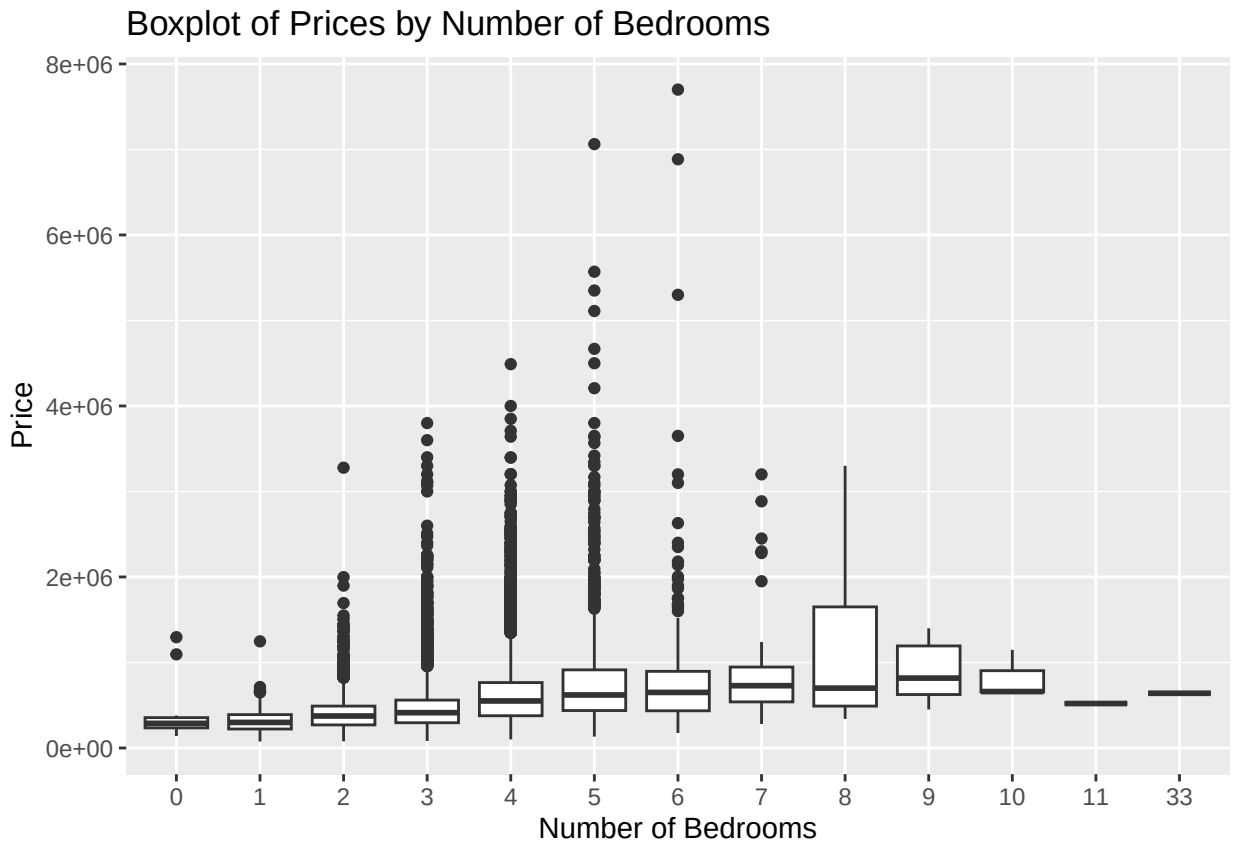
Analysis: The histogram below shows the distribution of house prices, revealing that most houses are in the lower price range with a significant decrease in the number of houses as the price increases, indicating a right-skewed distribution with relatively few high-priced houses.

```
# 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")
```

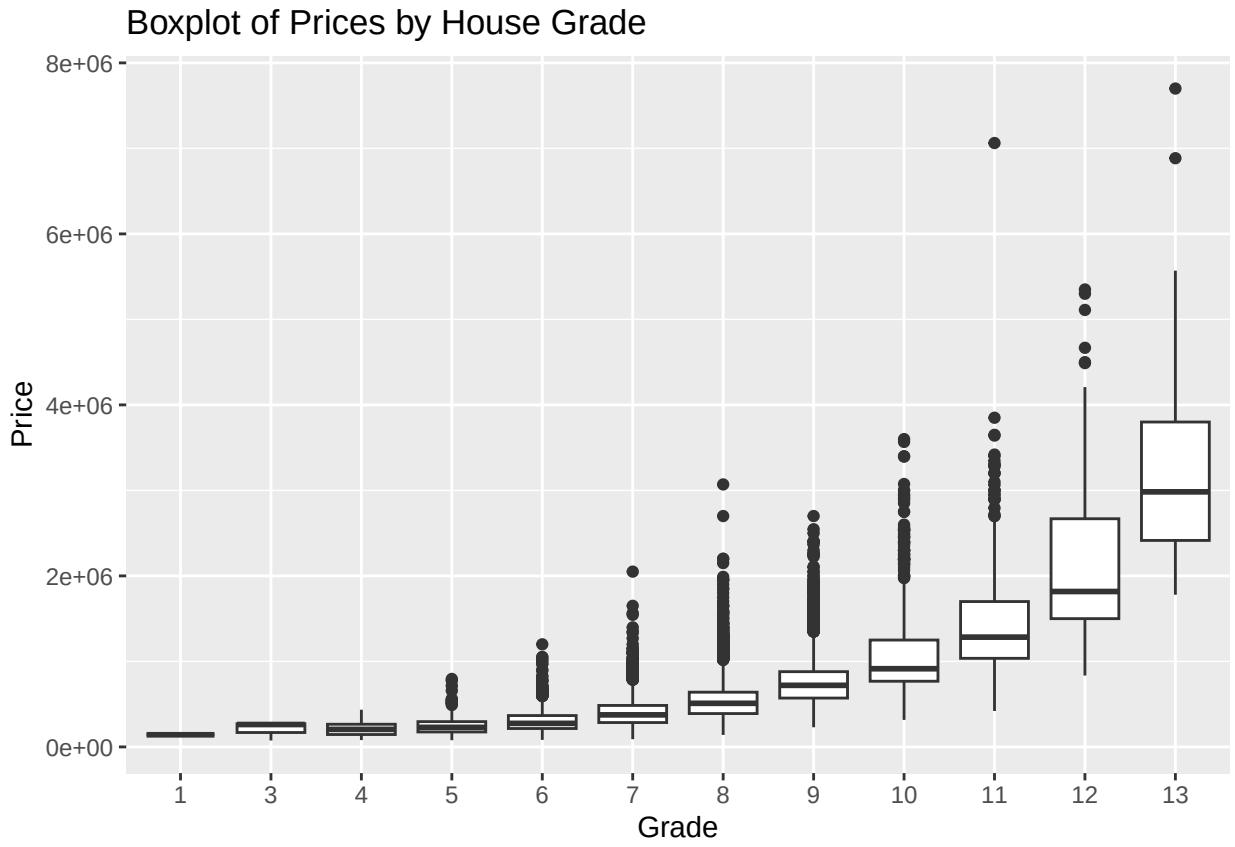


Analysis: The first boxplot shows that house prices generally increase with the number of bedrooms, but vary widely, especially for homes with more bedrooms. The second boxplot indicates that house prices rise with better house grades, with greater price variability at higher grades.Both plots demonstrate 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")
```



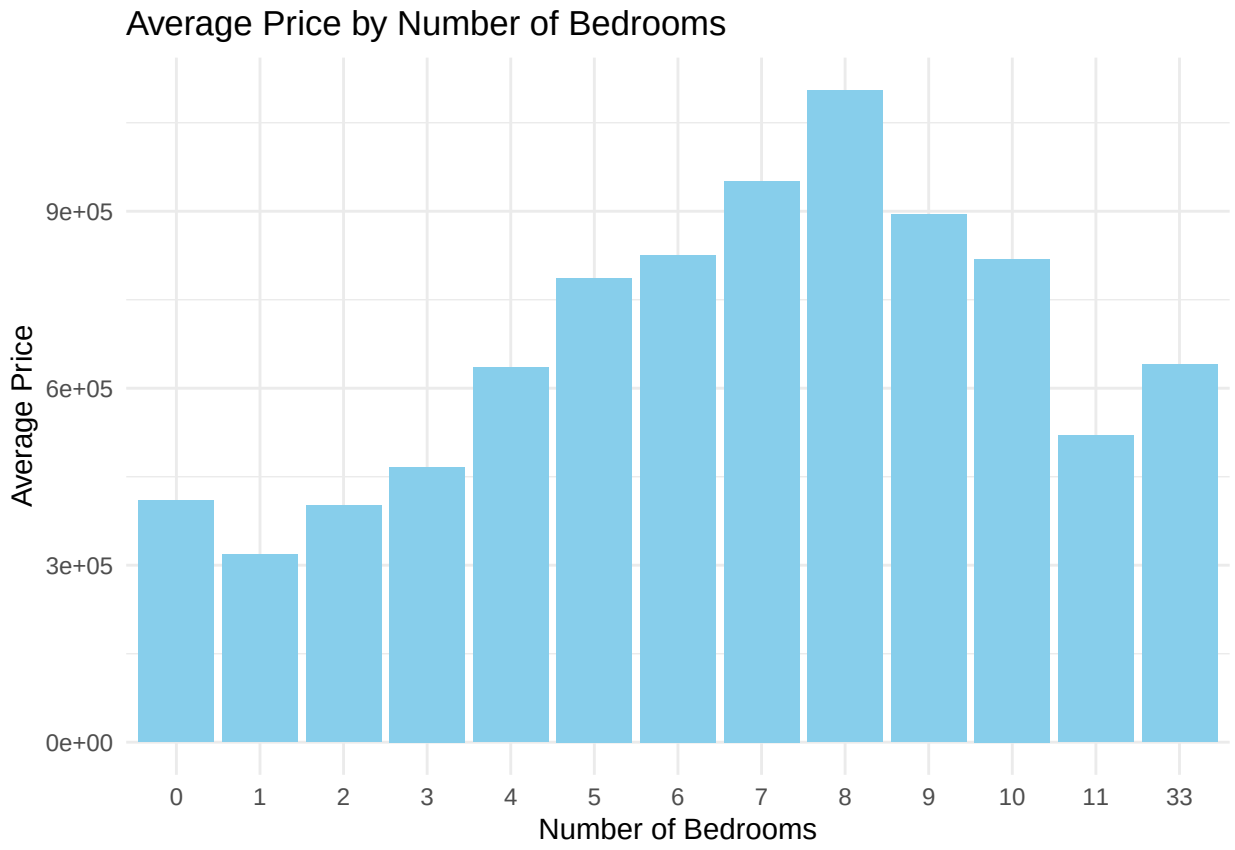

```r
# Boxplot for price by house grade
ggplot(df_house, aes(x = factor(grade), y = price)) +
  geom_boxplot() +
  labs(title = "Boxplot of Prices by House Grade", x = "Grade", y = "Price")
```



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.

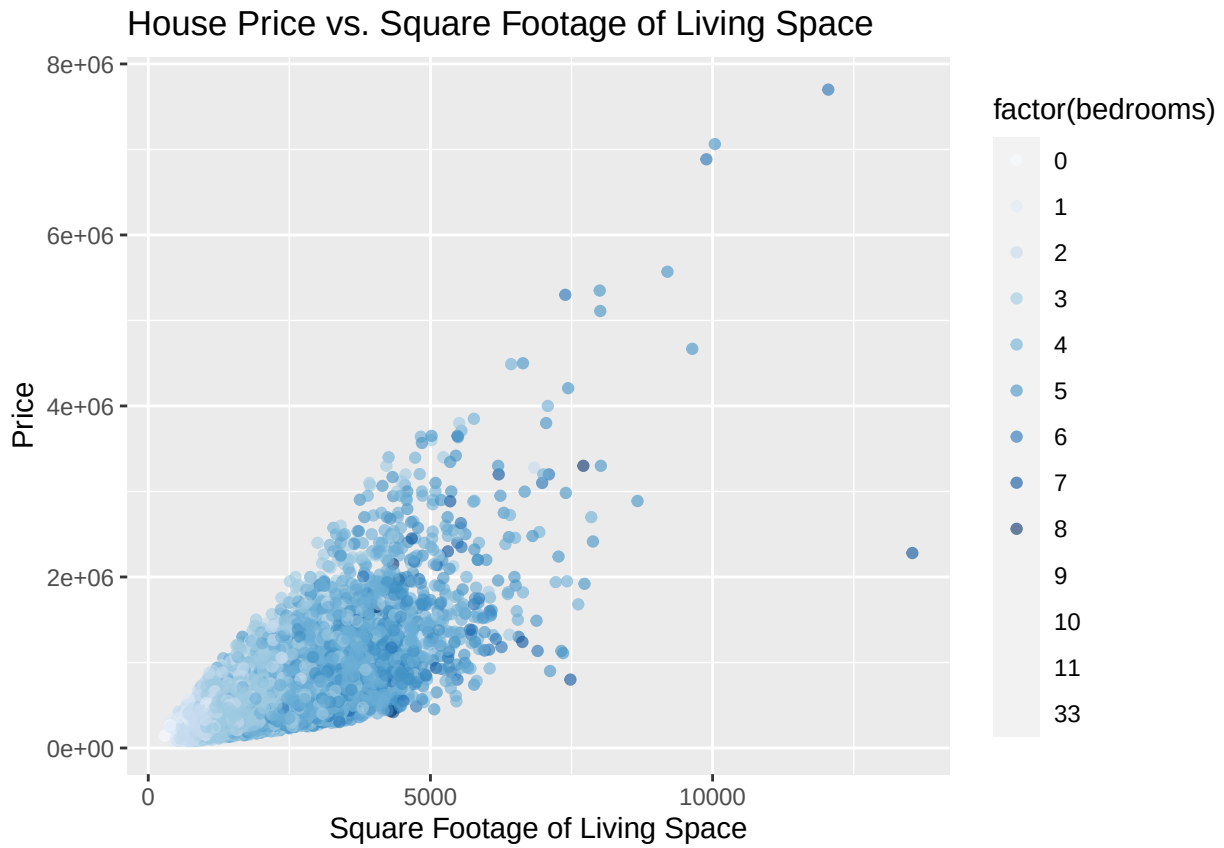
```
average_prices <- aggregate(price ~ bedrooms, data = ndf_house, FUN = mean)
ggplot(average_prices, aes(x = factor(bedrooms), y = price)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Average Price by Number of Bedrooms",
       x = "Number of Bedrooms",
       y = "Average Price")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



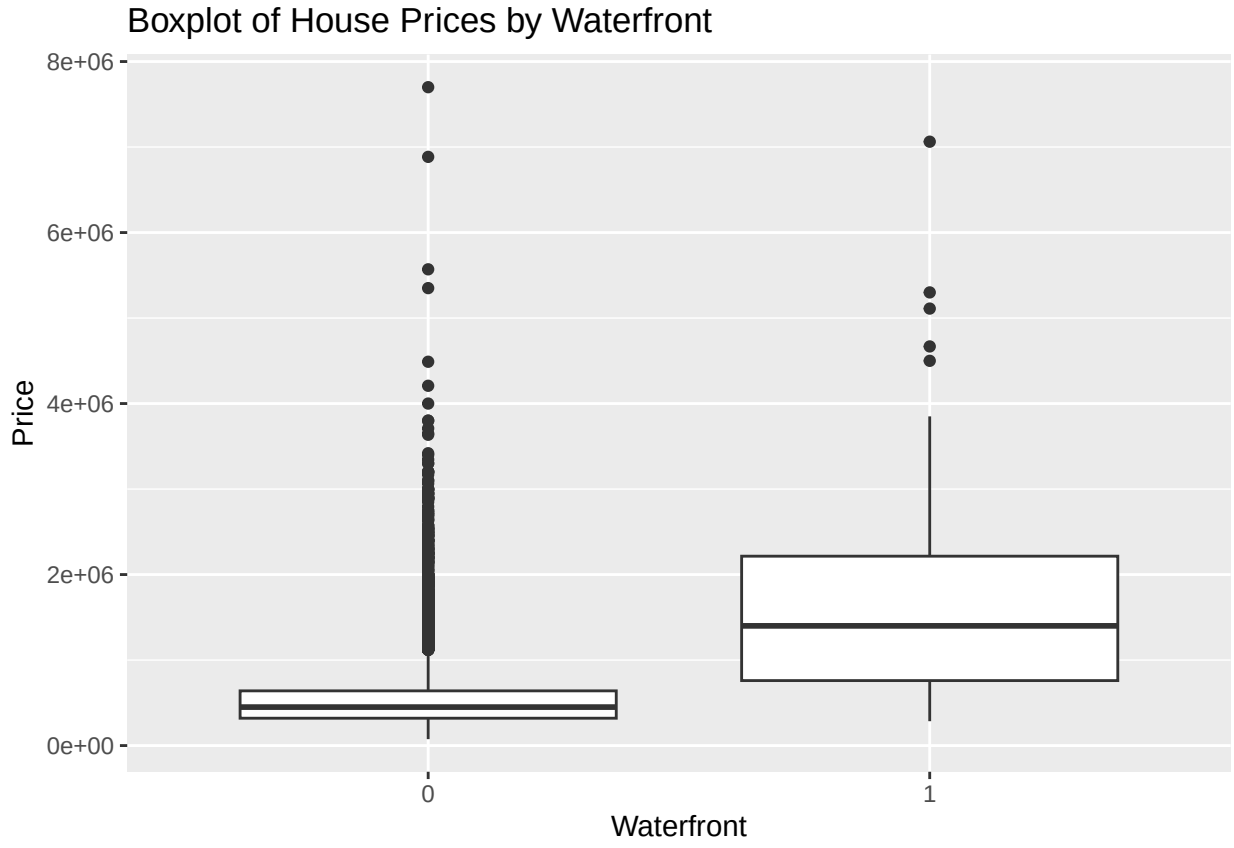

```
ggplot(df_house, aes(x = sqft_living, y = price)) +
  geom_point(aes(color = factor(bedrooms)), alpha = 0.6) +
  scale_color_brewer(type = 'seq', palette = 'Blues') +
  labs(title = "House Price vs. Square Footage of Living Space", x = "Square Footage of Living Space", y = "Pr
```

```
## Warning in RColorBrewer::brewer.pal(n, pal): n too large, allowed maximum for palette Blues is 9
## Returning the palette you asked for with that many colors
```

```
## Warning: Removed 11 rows containing missing values (`geom_point()`).
```




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

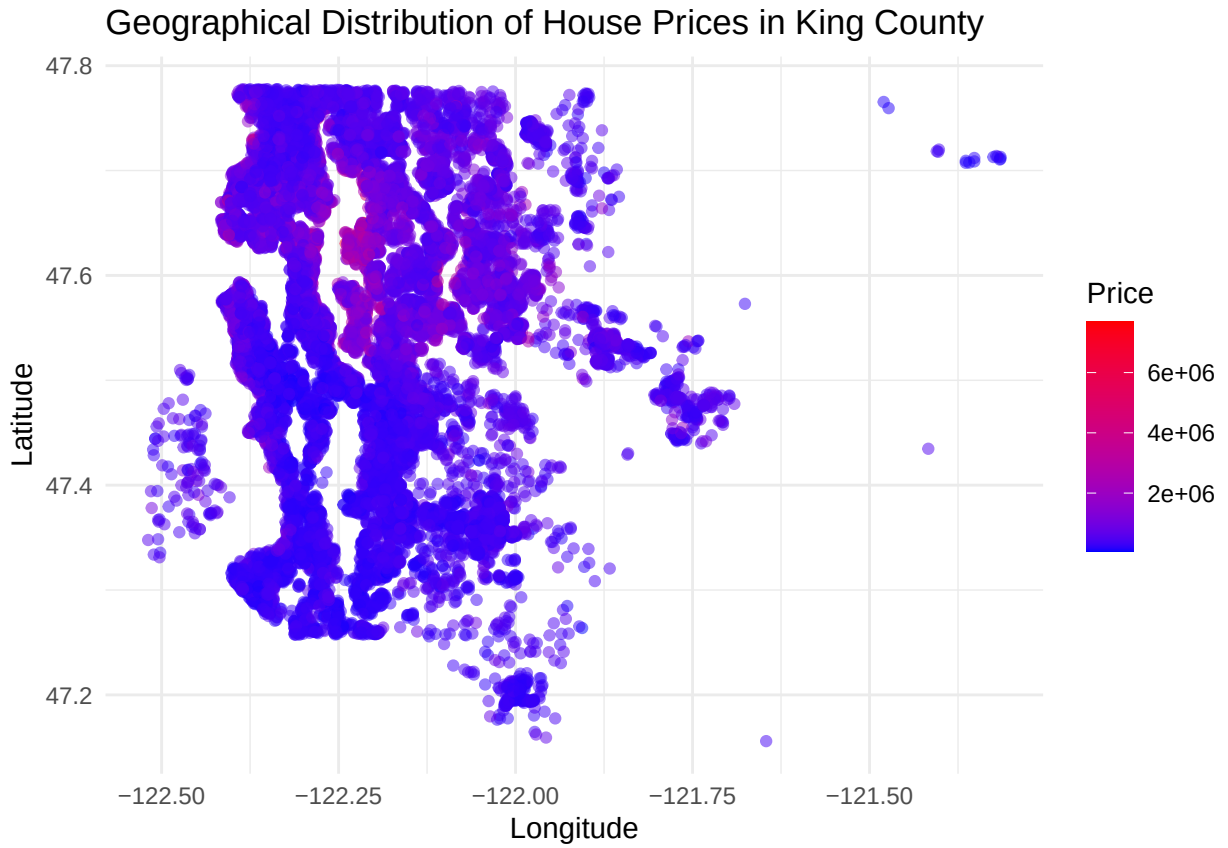


Analysis: From geographical map, it is evident that properties situated around the latitude line of 47.6 and longitude

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()
```



[Analysis section]

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.

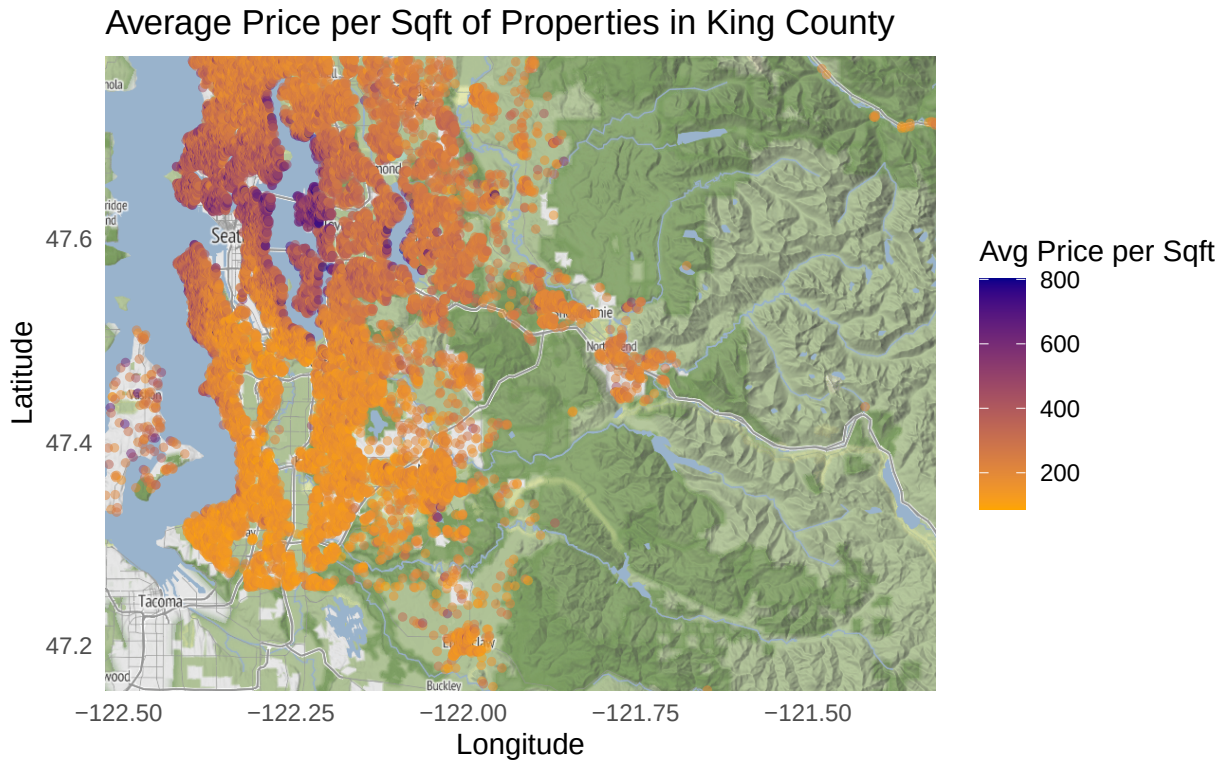
The map clearly shows a correlation between location and property value per square foot, with central urban areas exhibiting the highest values. This pattern is typical for urban centers where proximity to amenities, employment opportunities, and other socioeconomic factors drive up real estate prices.

```
register_stadiamaps(key = '765cbcd1-2ec2-40e3-8a81-98624e616208')
df_summary <- ndf_house %>%
  mutate(price_per_sqft = price / sqft_living) %>%
```

```
  group_by(long, lat) %>%
  summarize(avg_price_per_sqft = mean(price_per_sqft, na.rm = TRUE), .groups = 'drop')

king_county_map <- get_stadiamap(bbox = c(left = min(df_house$long),
                                          bottom = min(df_house$lat),
                                          right = max(df_house$long),
                                          top = max(df_house$lat)),
                                 maptype = "stamen_terrain",
                                 zoom = 10)
```

```
## i © Stadia Maps © Stamen Design © OpenMapTiles © OpenStreetMap contributors.
```

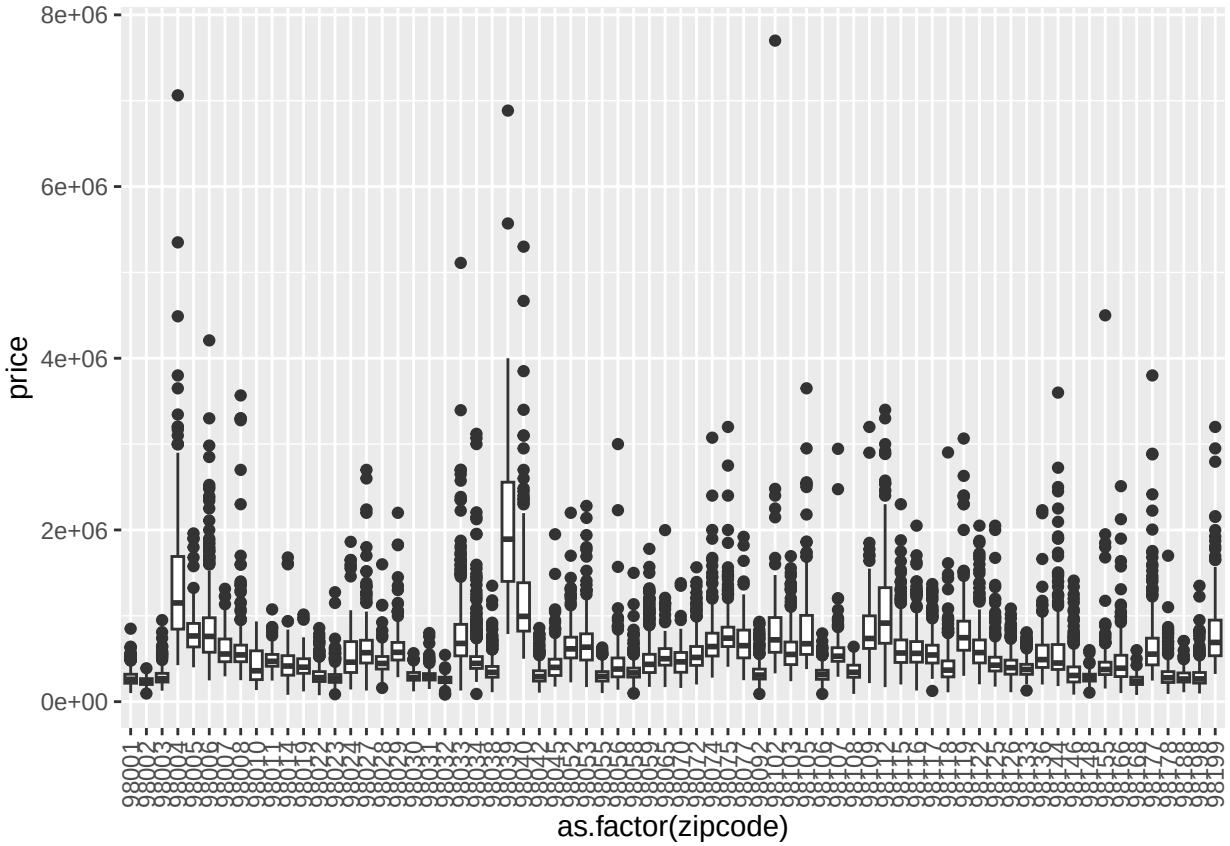
```
ggmap(king_county_map) +
  geom_point(data = df_summary, aes(x = long, y = lat, color = avg_price_per_sqft),
             alpha = 0.5, size = 1) +
  scale_color_gradient(low = "orange", high = "darkblue", name = "Avg Price per Sqft") +
  labs(title = "Average Price per Sqft of Properties in King County",
       x = "Longitude", y = "Latitude") +
  theme_minimal()
```



Zipcode

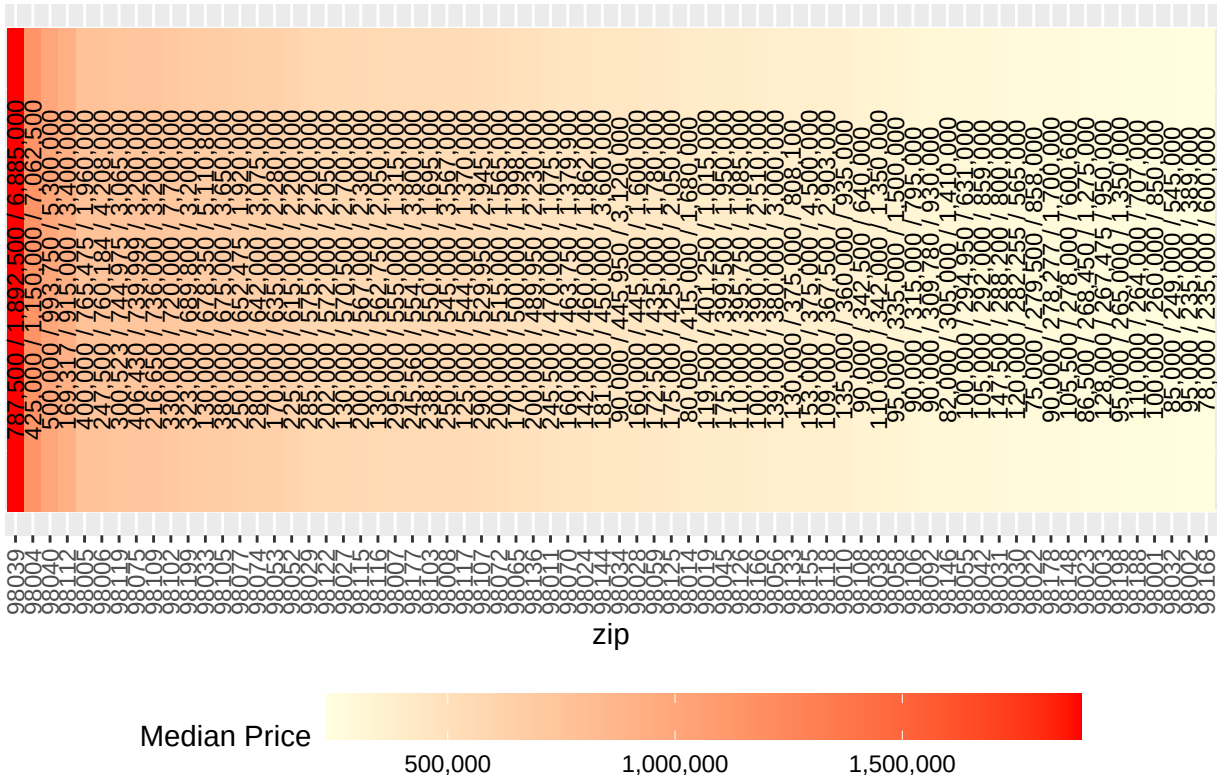
visualizations (can discard) ***Zipcode is a valid predictor***

```
# [Kaleo]
ggplot( df_house, aes(x = as.factor(zipcode), y = price)) +
  geom_boxplot()+theme(axis.text.x = element_text(angle = 90,vjust=0.5))
```




```
# [Kaleo]
# Prices by Zipcode heatmap
heatmap_data <- df_house %>%
  group_by(zipcode) %>%
  summarize(med_price = median(price),
            min_price = min(price),
            max_price = max(price))
heatmap_data <- heatmap_data %>%
  arrange(desc(med_price))

ggplot(heatmap_data, aes(x = factor(zipcode, levels = zipcode), y = 1, fill = med_price)) +
  geom_tile() +
  geom_text(aes(label = paste(scales::comma(min_price),"/",
                              scales::comma(med_price),"/",
                              scales::comma(max_price)),
                vjust = 0.5,angle=90),
            color = "black", size = 3)+
  scale_fill_gradient(low = "lightyellow", high = "red", name = "Median Price",
                      labels = scales::comma_format()) +
  labs(title = "Heatmap of median prices by zipcode (min/med/max)",
       x = "zip",
       y = "") +
  theme(axis.text.x = element_text(angle = 90, hjust = 0,vjust=0.5),
        axis.title.y = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        legend.position = "bottom",
        legend.key.width = unit(2, "cm"))
```

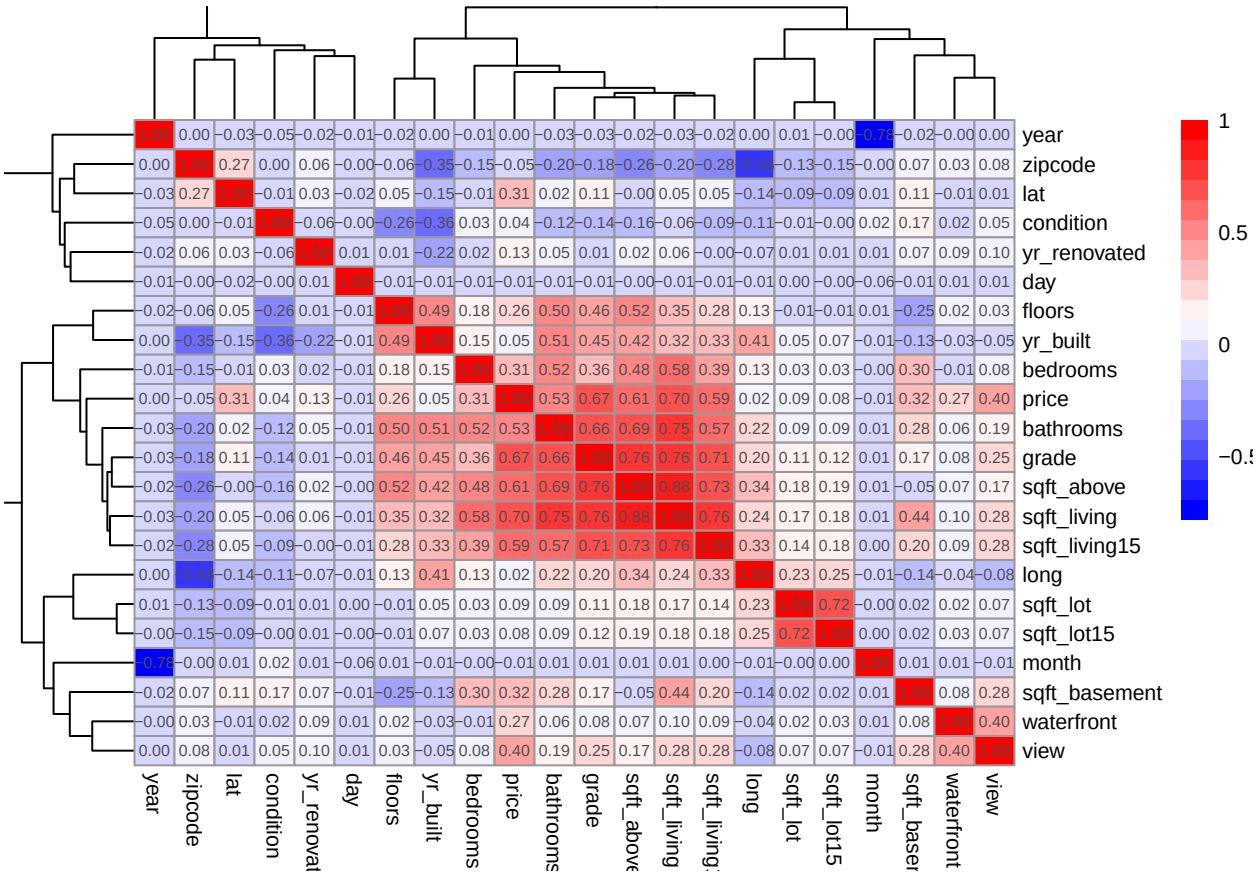
Heatmap of median prices by zipcode (min/med/max)



Correlation Analysis - in progress

[Add here the initial analysis of the correlation matrix]

```
# [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
)
```



Creating a new dataframe with variables 'highly' (>0.2) correlated with price

```
# [Kaleo]
head(x_high)
```

```
##   view waterfront sqft_basement sqft_living sqft_living15 sqft_above grade
## 1    0          0             0        1180          1340        1180     7
## 2    0          0           400        2570          1690        2170     7
## 3    0          0             0         770          2720         770     6
## 4    0          0           910        1960          1360        1050     7
## 5    0          0             0        1680          1800        1680     8
## 6    0          0          1530        5420          4760        3890    11
##   bathrooms bedrooms floors     lat
## 1      1.00        3      1 47.5112
## 2      2.25        3      2 47.7210
## 3      1.00        2      1 47.7379
## 4      3.00        4      1 47.5208
## 5      2.00        3      1 47.6168
## 6      4.50        4      1 47.6561
```

Data Transformation

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.

```
# 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.

```
# 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 = df_house[-c(1, 13)]

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    23
```

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

```
## [1] 6484   23
```

Summary of the section II

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

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.

```
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    23
```

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

```
## [1] 6484   23
```

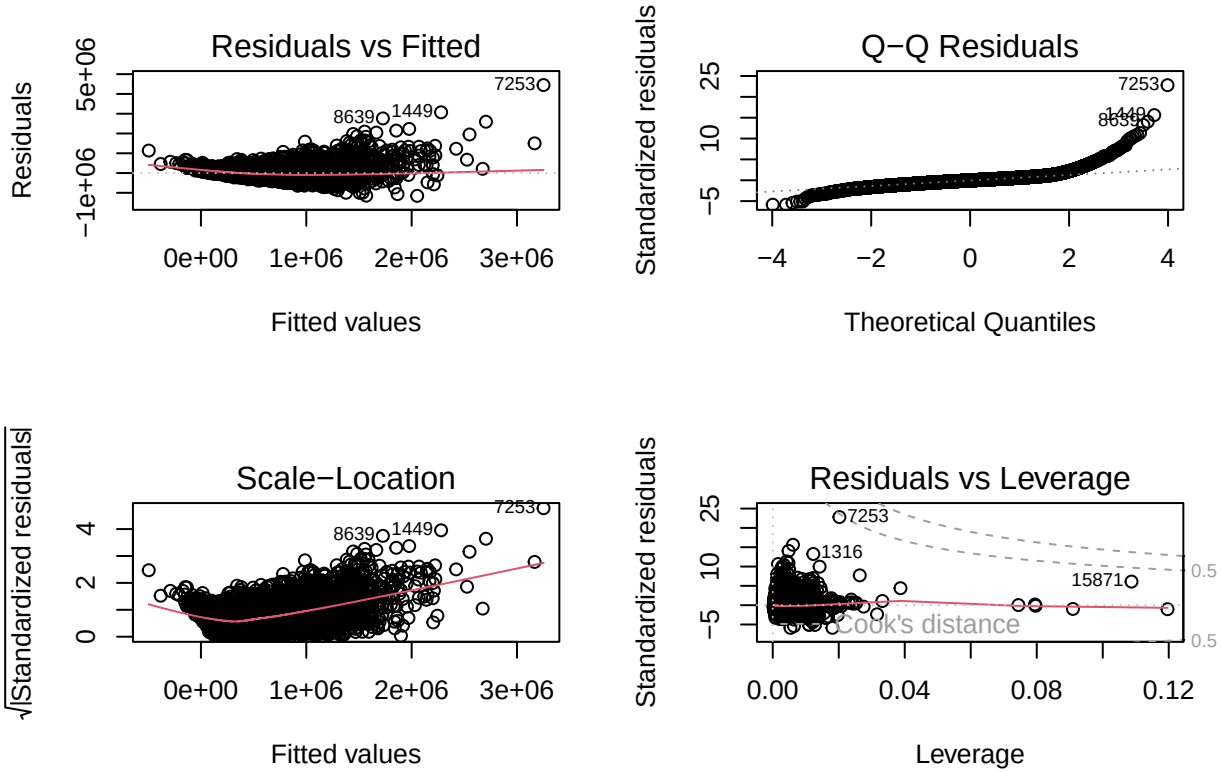
Analysis:

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

```
##
## Call:
## lm(formula = price ~ ., data = train.dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1152429   -98384    -7862    76610  4448174
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -7.996e+07  1.169e+07  -6.842 8.09e-12 ***
## bedrooms      -3.404e+04  2.176e+03 -15.643  < 2e-16 ***
## bathrooms      3.734e+04  3.782e+03   9.874  < 2e-16 ***
## sqft_living    1.536e+02  5.122e+00  29.986  < 2e-16 ***
## sqft_lot       1.437e-01  5.916e-02   2.429  0.01514 *
## floors         1.901e+04  4.243e+03   4.480 7.52e-06 ***
## waterfront     5.360e+05  2.125e+04  25.225  < 2e-16 ***
## view           5.484e+04  2.526e+03  21.715  < 2e-16 ***
## condition      2.946e+04  2.757e+03  10.683  < 2e-16 ***
## grade          9.463e+04  2.522e+03  37.522  < 2e-16 ***
## sqft_above     1.661e+01  5.103e+00   3.255  0.00114 **
## yr_built      -1.120e+03  3.567e+02  -3.139  0.00170 **
## yr_renovated   4.527e+03  5.297e+02   8.547  < 2e-16 ***
## zipcode       -5.720e+02  3.885e+01 -14.724  < 2e-16 ***
## lat            6.063e+05  1.263e+04  48.014  < 2e-16 ***
## long          -2.041e+05  1.540e+04 -13.252  < 2e-16 ***
## sqft_living15  3.156e+01  4.070e+00   7.753 9.56e-15 ***
## sqft_lot15    -3.268e-01  8.541e-02  -3.827  0.00013 ***
## year           4.155e+04  5.539e+03   7.503 6.60e-14 ***
## month          1.181e+03  8.315e+02   1.420  0.15550
## day           -4.000e+02  1.866e+02  -2.144  0.03206 *
## renovated     -8.896e+06  1.049e+06  -8.478  < 2e-16 ***
## age            1.544e+03  3.645e+02   4.237 2.27e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 197100 on 15106 degrees of freedom
## Multiple R-squared:  0.7036, Adjusted R-squared:  0.7032
```

```
## F-statistic:  1630 on 22 and 15106 DF,  p-value: < 2.2e-16
```

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



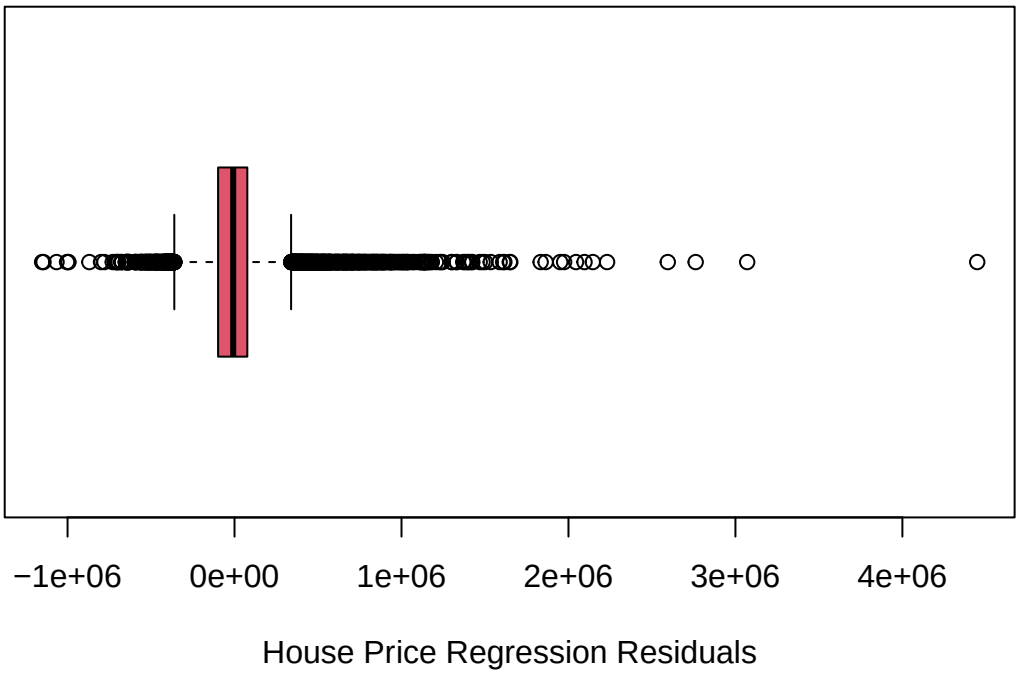

```
# Results dataframe to append model scores

# Calculate scores (mse,rmse,r-squared)

# Test data Predictions
house_lm_test_pred <- predict(house_lm, newdata = test.dat)
# Test MSE
house_lm_test_mse <- mean((house_lm_test_pred - test.dat$price)^2)
# Test RMSE
house_lm_test_rmse <- sqrt(house_lm_test_mse)
# Test Residuals
house_lm_test_residuals <- test.dat$price - house_lm_test_pred
# Test R-squared
house_lm_test_rsq <- 1 - var(house_lm_test_residuals) / var(test.dat$price)
# SSE
house_lm_test_sse <- sum((test.dat$price - house_lm_test_pred)^2)

results.df <- data.frame(model = "Linear Regression All (Baseline)",
                         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)
```

```
predictors_to_drop <- c("<=1900", "1901-1925", "1926-1950", "1951-1975", "1976-2000", "2001+", "zipcode_98198"

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

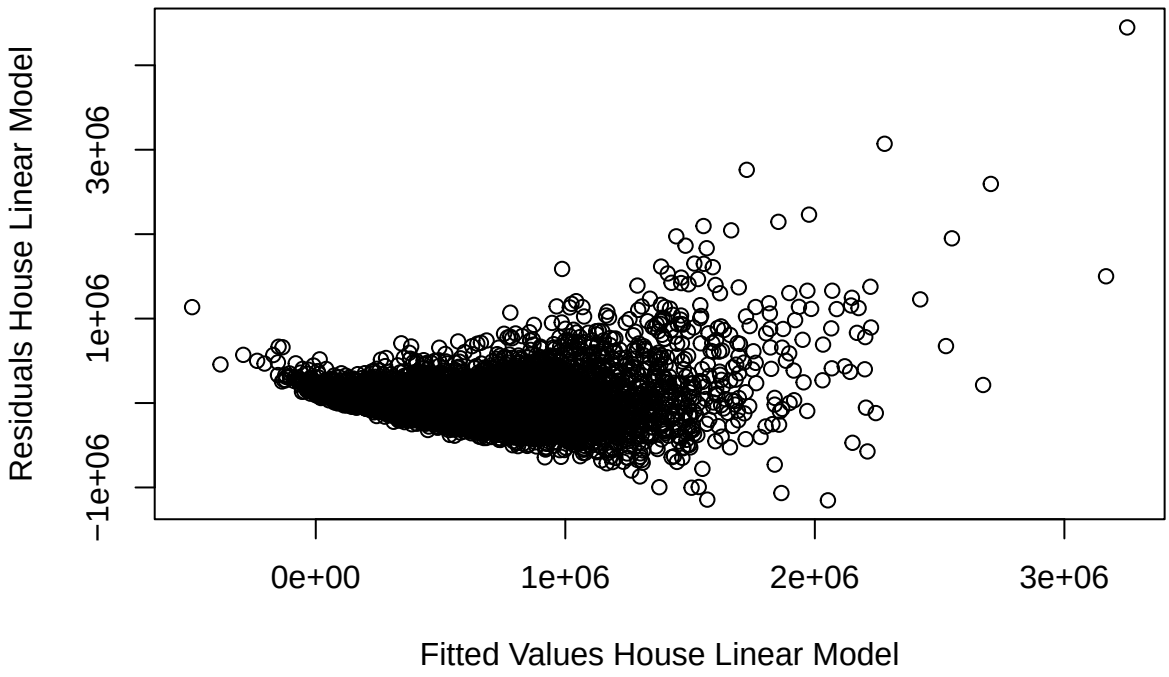
```r
summary(house_lm)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + sqft_lot +
##     floors + waterfront + view + condition + grade + sqft_above +
##     yr_built + yr_renovated + zipcode + lat + long + sqft_living15 +
##     sqft_lot15 + year + month + day + renovated + age, data = train.dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1152429   -98384    -7862    76610  4448174
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -7.996e+07  1.169e+07  -6.842 8.09e-12 ***
## bedrooms      -3.404e+04  2.176e+03 -15.643  < 2e-16 ***
## bathrooms      3.734e+04  3.782e+03   9.874  < 2e-16 ***
## sqft_living    1.536e+02  5.122e+00  29.986  < 2e-16 ***
## sqft_lot       1.437e-01  5.916e-02   2.429  0.01514 *
## floors         1.901e+04  4.243e+03   4.480 7.52e-06 ***
## waterfront     5.360e+05  2.125e+04  25.225  < 2e-16 ***
## view           5.484e+04  2.526e+03  21.715  < 2e-16 ***
## condition      2.946e+04  2.757e+03  10.683  < 2e-16 ***
## grade          9.463e+04  2.522e+03  37.522  < 2e-16 ***
## sqft_above     1.661e+01  5.103e+00   3.255  0.00114 **
## yr_built      -1.120e+03  3.567e+02  -3.139  0.00170 **
## yr_renovated   4.527e+03  5.297e+02   8.547  < 2e-16 ***
## zipcode       -5.720e+02  3.885e+01 -14.724  < 2e-16 ***
## lat            6.063e+05  1.263e+04  48.014  < 2e-16 ***
## long          -2.041e+05  1.540e+04 -13.252  < 2e-16 ***
## sqft_living15  3.156e+01  4.070e+00   7.753 9.56e-15 ***
## sqft_lot15    -3.268e-01  8.541e-02  -3.827  0.00013 ***
## year           4.155e+04  5.539e+03   7.503 6.60e-14 ***
## month          1.181e+03  8.315e+02   1.420  0.15550
## day           -4.000e+02  1.866e+02  -2.144  0.03206 *
## renovated     -8.896e+06  1.049e+06  -8.478  < 2e-16 ***
## age            1.544e+03  3.645e+02   4.237 2.27e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 197100 on 15106 degrees of freedom
## Multiple R-squared:  0.7036, Adjusted R-squared:  0.7032
## F-statistic:  1630 on 22 and 15106 DF,  p-value: < 2.2e-16
```

```r
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")
```



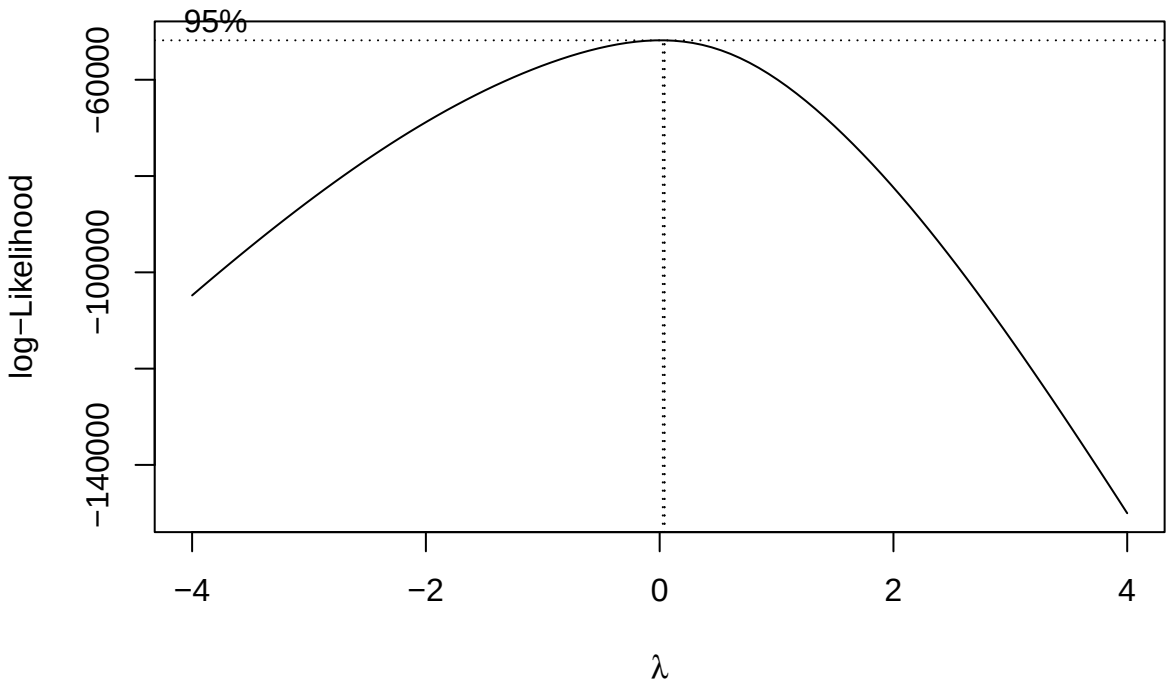
H_o: Error variances are constant H_a: Error variances are not constant Decision Rule is if statistic> critical reject the null or if p-value < alpha (0.01) reject the hull

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

```
bptest(house_lm, studentize = FALSE)
```

```
##
##  Breusch-Pagan test
##
## data:  house_lm
## BP = 39547, df = 22, p-value < 2.2e-16
```

```
par(mfrow=c(1,1))
bc<-boxcox(house_lm,lambda=seq(-4,4,by=0.1))
```




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

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

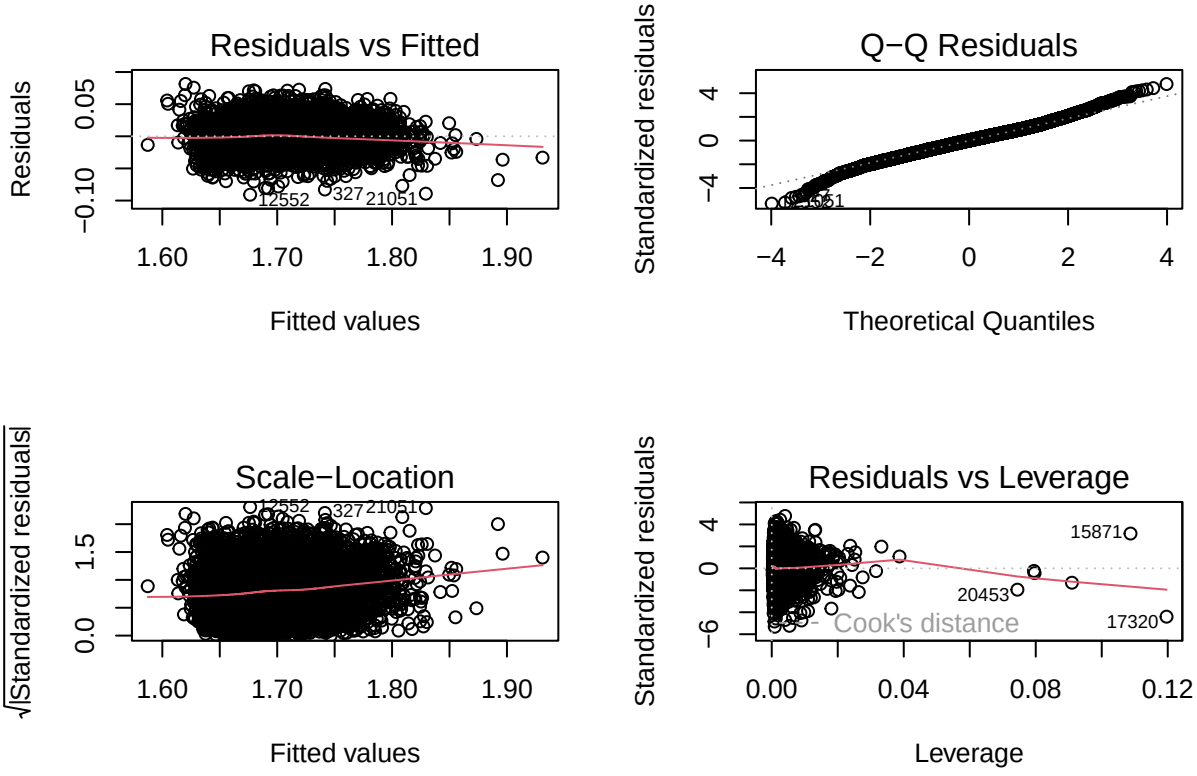
Analysis:

```
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
## -0.090925 -0.010921  0.000203  0.010707  0.081425
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -8.543e+00  1.014e+00  -8.424  < 2e-16 ***
## bedrooms      -1.052e-03  1.888e-04  -5.570 2.59e-08 ***
```

```
## bathrooms      4.434e-03  3.282e-04  13.511  < 2e-16 ***
## sqft_living    1.118e-05  4.445e-07  25.157  < 2e-16 ***
## sqft_lot       3.484e-08  5.134e-09   6.786 1.20e-11 ***
## floors         5.578e-03  3.682e-04  15.149  < 2e-16 ***
## waterfront     2.555e-02  1.844e-03  13.858  < 2e-16 ***
## view           4.319e-03  2.192e-04  19.704  < 2e-16 ***
## condition      4.560e-03  2.393e-04  19.057  < 2e-16 ***
## grade          1.053e-02  2.189e-04  48.095  < 2e-16 ***
## sqft_above    -1.390e-06  4.429e-07  -3.139   0.0017 **
## yr_built      -1.441e-04  3.095e-05  -4.656 3.25e-06 ***
## yr_renovated   4.019e-04  4.597e-05   8.744  < 2e-16 ***
## zipcode       -4.535e-05  3.371e-06 -13.450  < 2e-16 ***
## lat            9.639e-02  1.096e-03  87.951  < 2e-16 ***
## long          -1.115e-02  1.336e-03  -8.340  < 2e-16 ***
## sqft_living15  7.098e-06  3.532e-07  20.094  < 2e-16 ***
## sqft_lot15    -1.240e-08  7.412e-09  -1.673   0.0943 .
## year           4.404e-03  4.807e-04   9.163  < 2e-16 ***
## month          9.533e-05  7.216e-05   1.321   0.1865
## day           -4.162e-05  1.619e-05  -2.570   0.0102 *
## renovated     -7.912e-01  9.106e-02  -8.689  < 2e-16 ***
## age            8.965e-05  3.163e-05   2.834   0.0046 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0171 on 15106 degrees of freedom
## Multiple R-squared:  0.7755, Adjusted R-squared:  0.7752
## F-statistic:  2372 on 22 and 15106 DF,  p-value: < 2.2e-16
```

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




```
pred<- predict(house_lm1,test.dat)^(1/lambda)
act<-test.dat$price
```

```r
SST<-var(act)*(length(act)-1)
SSE<-sum((act-pred)^2)
R.Square.Test <- 1- SSE/SST
R.Square.Test
```

```
## [1] 0.6669345
```

```r
R.Square.Train=0.887

knitr::kable(cbind(R.Square.Train,R.Square.Test), align = "c",
      caption = "Pseudo $R^2$ values for Train and Test")
```

Table 2: Pseudo R^2 values for Train and Test

R.Square.Train	R.Square.Test
0.887	0.6669345

III. Model Development Process (15 points) - SVM Mohanish (in progress)

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.

```
# find optimal cost of misclassification
# tune.out <- tune(svm, price~., data = train.dat, kernel = "linear",
#                  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
# extract the best model
# (bestmod <- tune.out$best.model)
```

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.).

##Best Subset Regression

```
house_lm2<-lm(price^lambda~.,data=test.dat)
# k1<-ols_step_best_subset(house_lm2)
# k1
# plot(k1,guide="none")
```

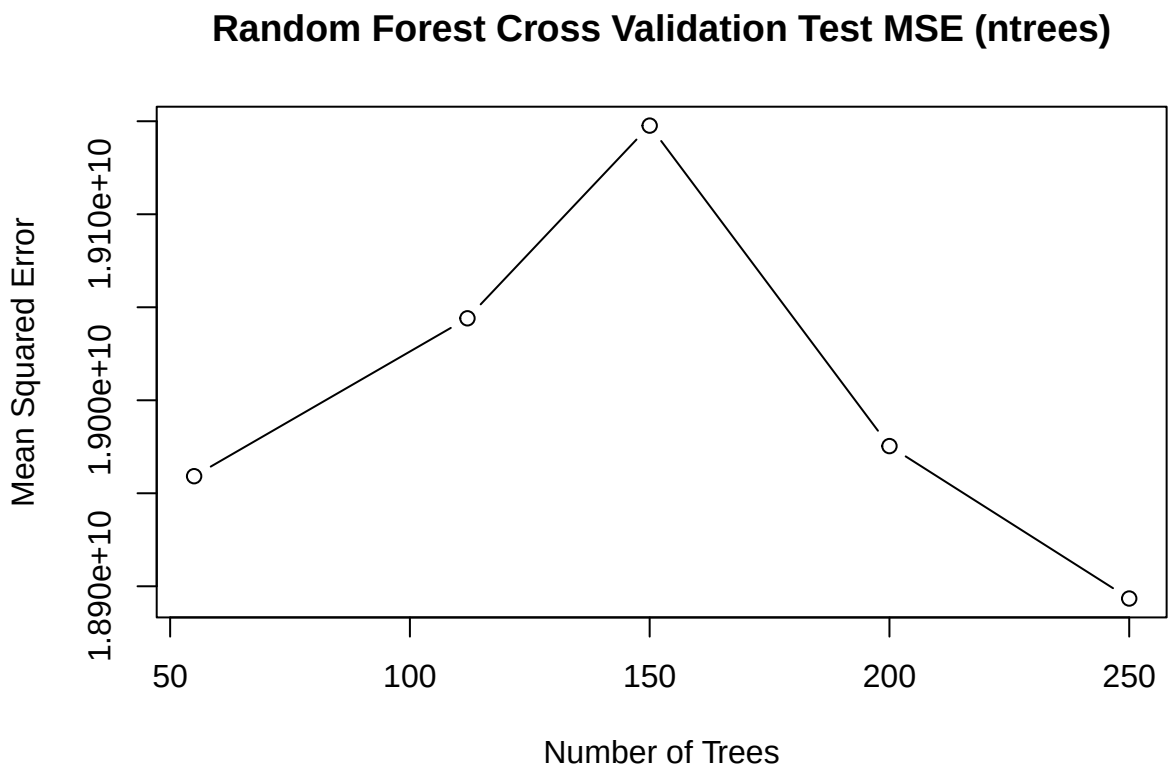
V. Challenger Models (15 points)

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.

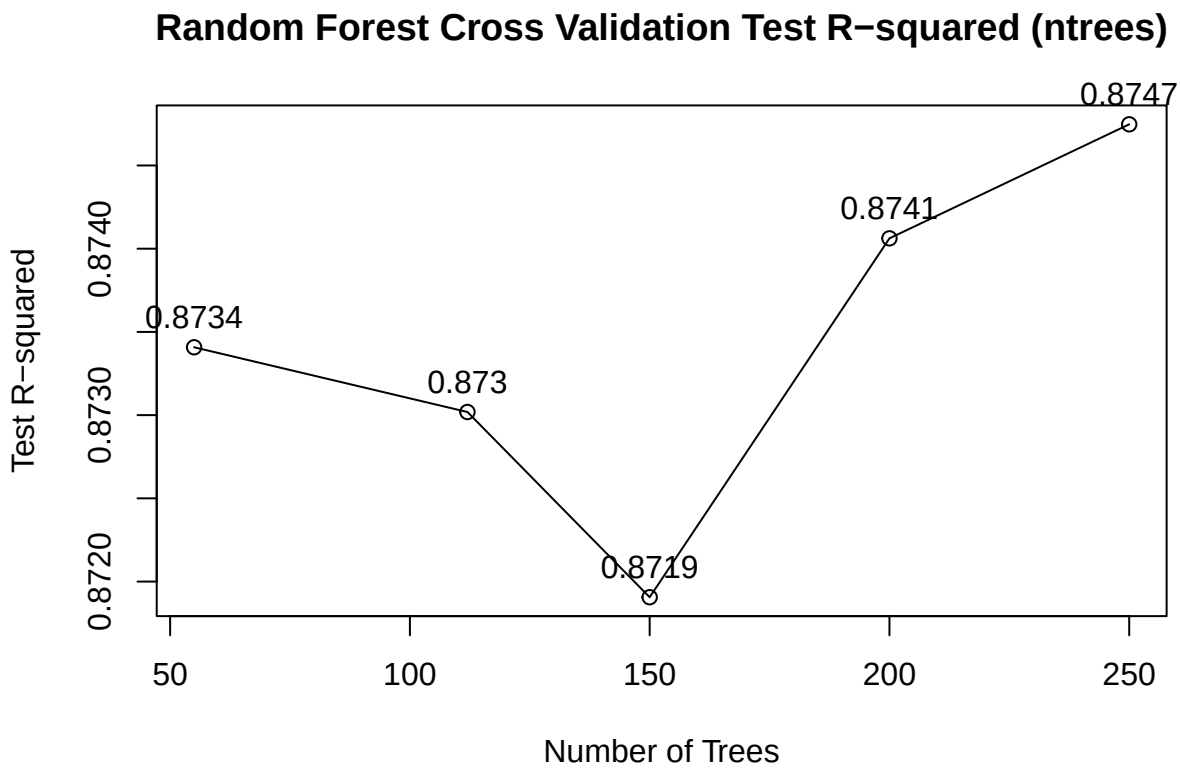
Random Forest

```
#[Kaleo]

ntree_values <- c(55, 112, 150, 200, 250)

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

# commented out cross validation to save computation time!!

# for (i in seq_along(ntree_values)) {
#   ntree = ntree_values[i]
#
#   model = randomForest(price ~ ., data = train.dat, ntree = ntree)
#   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]
mse_hard.coded = c(18959180557,19044061189,19147650224,18975362291,18893426330)
test_rsq_values_hard.coded = c(0.8734077,0.8730189,0.8719060,0.8740622,0.8747472)

# Plot the results
plot(ntree_values, mse_hard.coded, type = "b", xlab = "Number of Trees", ylab = "Mean Squared Error", main = "
```




```
plot(ntree_values, test_rsq_values_hard.coded,type="o", xlab = "Number of Trees", ylab = "Test R-squared", mai
text(ntree_values,test_rsq_values_hard.coded,labels=round(test_rsq_values_hard.coded,4),pos=3,xpd=TRUE)
```

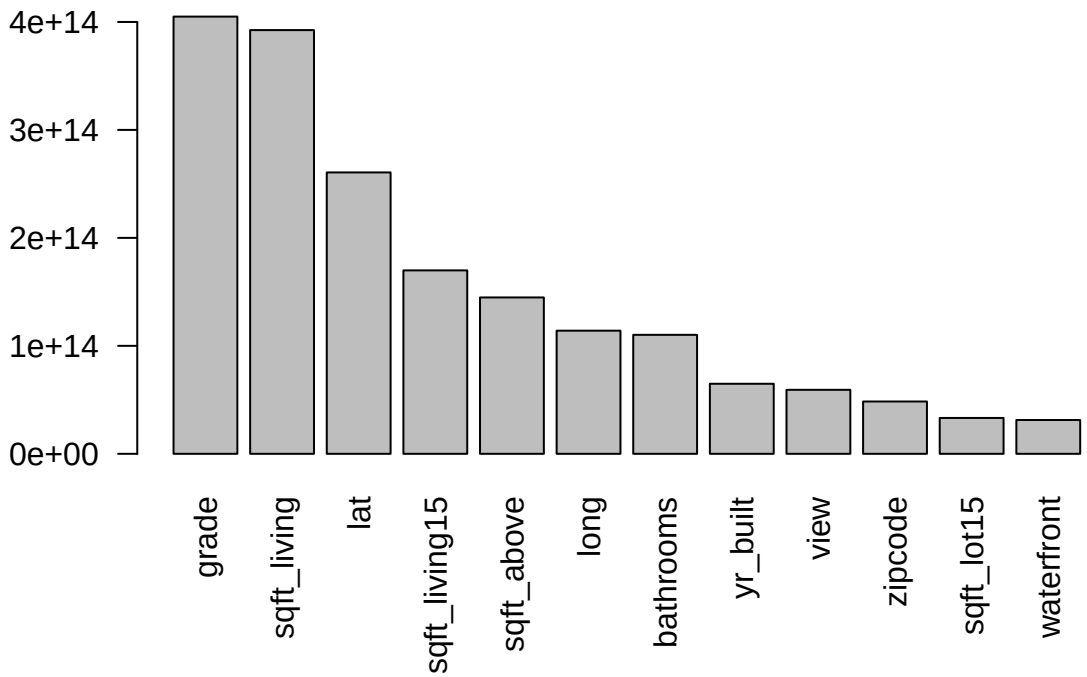



```
# [Kaleo]
```

```
#commented out for computation time..

#using efficient model (50 trees...for now)
house_rf = randomForest(price ~ ., data = train.dat, ntree = 50)
rf_test_pred = predict(house_rf,newdata=test.dat)
rf_train_pred = predict(house_rf,newdata=train.dat)

rf_test_mse = mean((rf_test_pred - test.dat$price)^2)
rf_test_rsq = cor(rf_test_pred,test.dat$price)^2
rf_train_rsq = cor(rf_train_pred,train.dat$price)^2

cat("test mse:",rf_test_mse,"train rsq:", rf_train_rsq, "test rsq:",rf_test_rsq)
```

```
## test mse: 19783510526 train rsq: 0.9781213 test rsq: 0.8664648
```

```
#[Kaleo]

# relative variable importance
par(mar = c(7, 5, 4, 2))

var_importance = importance(house_rf)
top_features = var_importance[order(var_importance[,1], decreasing = TRUE), ][1:12]

barplot(top_features,names.arg = rownames(top_features),las=2)
```



Regression Tree Model

```
# [Luciano]

# 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 1.0000923 0.04576240
## 2  0.11465966      1 0.6800989 0.6924361 0.03581501
## 3  0.06599331      2 0.5654392 0.5761635 0.02746974
## 4  0.04114284      3 0.4994459 0.5058884 0.02695583
## 5  0.02978008      4 0.4583031 0.4733424 0.02351208
## 6  0.02946001      5 0.4285230 0.4658304 0.02378910
## 7  0.02890907      6 0.3990630 0.4493149 0.02347807
## 8  0.01997761      7 0.3701539 0.4086909 0.02206601
## 9  0.01178433      8 0.3501763 0.3713263 0.01706559
## 10 0.01134065      9 0.3383919 0.3635276 0.01698824
## 11 0.01043418     10 0.3270513 0.3578200 0.01690015
## 12 0.01012668     11 0.3166171 0.3503020 0.01667408
## 13 0.01000000     12 0.3064904 0.3482220 0.01665636
##
## Variable importance
##         grade   sqft_living    sqft_above sqft_living15     bathrooms
##            25            20            15            12             7
##           lat          long      yr_built       zipcode           age
##             7             3             2             2             2
##    sqft_lot15          view      bedrooms      sqft_lot
##             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_living   < 2847.5    to the left,  improve=0.3162891, (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)
##   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)
##       sqft_living   < 2902.5    to the left,  agree=0.880, adj=0.392, (0 split)
##       bathrooms     < 3.125     to the left,  agree=0.838, adj=0.182, (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)
##       sqft_living   < 2039      to the left,  improve=0.16493960, (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)
##   Surrogate splits:
##       zipcode    < 98071     to the left,  agree=0.671, adj=0.222, (0 split)
##       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)
```

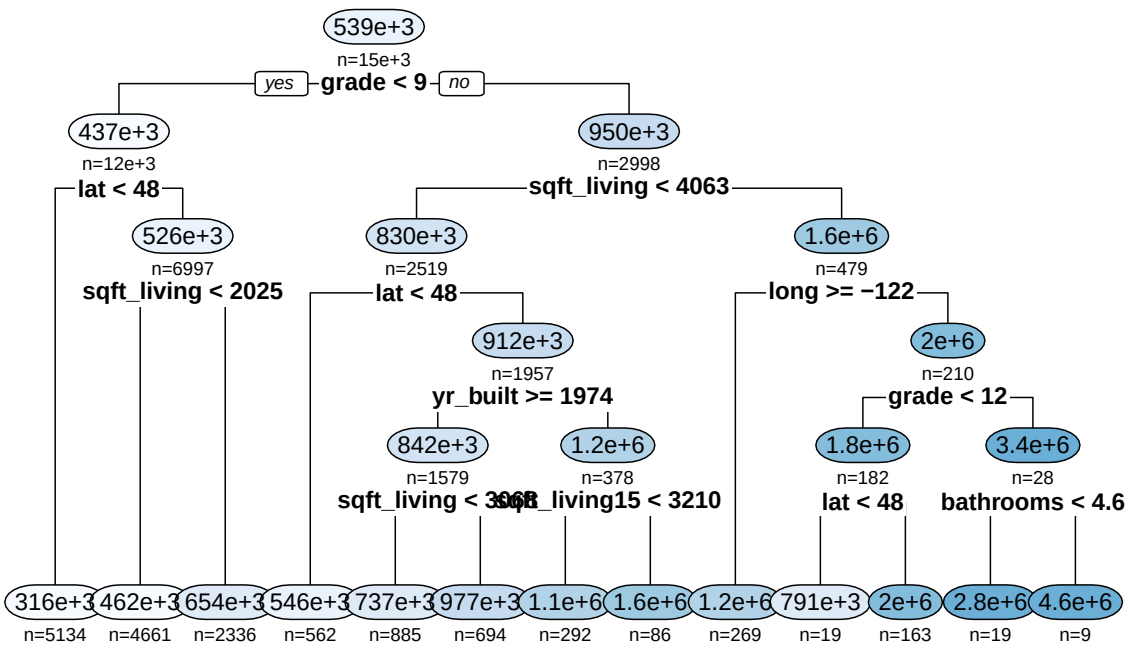
```
##       grade      < 6.5       to the left,  agree=0.601, adj=0.057, (0 split)
##
## Node number 3: 2998 observations,    complexity param=0.1146597
##   mean=950192.7, MSE=2.954463e+11
##   left son=6 (2519 obs) right son=7 (479 obs)
##   Primary splits:
##       sqft_living   < 4062.5    to the left,  improve=0.2561684, (0 missing)
##       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)
##   Surrogate splits:
##       sqft_above    < 4062.5    to the left,  agree=0.924, adj=0.522, (0 split)
##       grade         < 10.5      to the left,  agree=0.880, adj=0.246, (0 split)
##       bathrooms     < 3.875     to the left,  agree=0.879, adj=0.242, (0 split)
##       sqft_living15 < 3838      to the left,  agree=0.874, adj=0.213, (0 split)
##       bedrooms      < 5.5       to the left,  agree=0.846, adj=0.033, (0 split)
##
## Node number 4: 5134 observations
##   mean=315895.1, MSE=1.401515e+10
##
## Node number 5: 6997 observations,    complexity param=0.02890907
##   mean=525898.8, MSE=3.680939e+10
##   left son=10 (4661 obs) right son=11 (2336 obs)
##   Primary splits:
##       sqft_living   < 2025      to the left,  improve=0.22212060, (0 missing)
##       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)
##   Surrogate splits:
##       sqft_above    < 2025      to the left,  agree=0.815, adj=0.446, (0 split)
##       sqft_living15 < 1995      to the left,  agree=0.782, adj=0.346, (0 split)
##       bedrooms      < 3.5       to the left,  agree=0.780, adj=0.340, (0 split)
##       bathrooms     < 2.375     to the left,  agree=0.746, adj=0.239, (0 split)
##       grade         < 7.5       to the left,  agree=0.696, adj=0.090, (0 split)
##
## Node number 6: 2519 observations,    complexity param=0.02946001
##   mean=830227.4, MSE=1.25407e+11
##   left son=12 (562 obs) right son=13 (1957 obs)
##   Primary splits:
##       lat         < 47.5231   to the left,  improve=0.1845475, (0 missing)
##       yr_built    < 1972.5    to the right, improve=0.1763381, (0 missing)
##       sqft_living < 3155      to the left,  improve=0.1300291, (0 missing)
##       age         < 52.5      to the left,  improve=0.1180599, (0 missing)
##       grade       < 9.5       to the left,  improve=0.1093538, (0 missing)
##   Surrogate splits:
##       zipcode    < 98003.5   to the left,  agree=0.793, adj=0.071, (0 split)
##       long       < -121.8495 to the right, agree=0.786, adj=0.041, (0 split)
##       sqft_lot   < 75092     to the right, agree=0.784, adj=0.034, (0 split)
##       sqft_lot15 < 89951     to the right, agree=0.782, adj=0.021, (0 split)
##       sqft_above < 750       to the left,  agree=0.778, adj=0.004, (0 split)
##
## Node number 7: 479 observations,    complexity param=0.04114284
##   mean=1581075, MSE=7.159647e+11
##   left son=14 (269 obs) right son=15 (210 obs)
##   Primary splits:
##       long        < -122.1875 to the right, improve=0.2374064, (0 missing)
##       sqft_living < 6205      to the left,  improve=0.1839034, (0 missing)
##       grade       < 11.5      to the left,  improve=0.1668863, (0 missing)
```

```
##       waterfront  < 0.5       to the left,  improve=0.1474759, (0 missing)
##       yr_built    < 1943.5    to the right, improve=0.1323719, (0 missing)
##   Surrogate splits:
##       zipcode    < 98097     to the left,  agree=0.729, adj=0.381, (0 split)
##       yr_built   < 1980.5    to the right, agree=0.727, adj=0.376, (0 split)
##       age        < 42.5      to the left,  agree=0.685, adj=0.281, (0 split)
##       view       < 0.5       to the left,  agree=0.683, adj=0.276, (0 split)
##       sqft_lot15 < 22325.5   to the right, agree=0.670, adj=0.248, (0 split)
##
## Node number 10: 4661 observations
##   mean=461885.5, MSE=1.954151e+10
##
## Node number 11: 2336 observations
##   mean=653624.1, MSE=4.677397e+10
##
## Node number 12: 562 observations
##   mean=546342.5, MSE=3.597194e+10
##
## Node number 13: 1957 observations,    complexity param=0.01997761
##   mean=911751.8, MSE=1.213007e+11
##   left son=26 (1579 obs) right son=27 (378 obs)
##   Primary splits:
##       yr_built    < 1973.5    to the right, improve=0.1665385, (0 missing)
##       long        < -122.1875 to the right, improve=0.1485563, (0 missing)
##       sqft_living < 3067.5    to the left,  improve=0.1477582, (0 missing)
##       view        < 1.5       to the left,  improve=0.1164698, (0 missing)
##       age         < 53.5      to the left,  improve=0.1102756, (0 missing)
##   Surrogate splits:
##       age          < 49.5      to the left,  agree=0.957, adj=0.775, (0 split)
##       yr_renovated < 978       to the left,  agree=0.847, adj=0.209, (0 split)
##       renovated    < 0.5       to the left,  agree=0.847, adj=0.209, (0 split)
##       condition    < 4.5       to the left,  agree=0.836, adj=0.151, (0 split)
##       floors       < 1.75      to the right, agree=0.818, adj=0.056, (0 split)
##
## Node number 14: 269 observations
##   mean=1216803, MSE=2.077558e+11
##
## Node number 15: 210 observations,    complexity param=0.02978008
##   mean=2047690, MSE=9.792521e+11
##   left son=30 (182 obs) right son=31 (28 obs)
##   Primary splits:
##       grade       < 11.5      to the left,  improve=0.2865743, (0 missing)
##       sqft_living < 6575      to the left,  improve=0.2559396, (0 missing)
##       sqft_above  < 4735      to the left,  improve=0.2074284, (0 missing)
##       bathrooms   < 4.625     to the left,  improve=0.2036505, (0 missing)
##       lat         < 47.534    to the left,  improve=0.1518220, (0 missing)
##   Surrogate splits:
##       sqft_living < 6255      to the left,  agree=0.924, adj=0.429, (0 split)
##       sqft_above  < 5175      to the left,  agree=0.905, adj=0.286, (0 split)
##       bathrooms   < 5.25      to the left,  agree=0.886, adj=0.143, (0 split)
##       sqft_lot15  < 50338     to the left,  agree=0.886, adj=0.143, (0 split)
##       sqft_lot    < 52239.5   to the left,  agree=0.876, adj=0.071, (0 split)
##
## Node number 26: 1579 observations,    complexity param=0.01134065
##   mean=842210.3, MSE=7.853571e+10
##   left son=52 (885 obs) right son=53 (694 obs)
##   Primary splits:
##       sqft_living < 3067.5    to the left,  improve=0.18097300, (0 missing)
##       grade       < 9.5       to the left,  improve=0.16964900, (0 missing)
##       bathrooms   < 3.125     to the left,  improve=0.11626180, (0 missing)
```

```
##       view        < 2.5       to the left,  improve=0.08560643, (0 missing)
##       zipcode     < 98004.5   to the right, improve=0.08504714, (0 missing)
##   Surrogate splits:
##       sqft_above    < 3067.5    to the left,  agree=0.866, adj=0.696, (0 split)
##       sqft_living15 < 3035      to the left,  agree=0.760, adj=0.454, (0 split)
##       bathrooms     < 2.625     to the left,  agree=0.699, adj=0.316, (0 split)
##       grade         < 9.5       to the left,  agree=0.680, adj=0.271, (0 split)
##       bedrooms      < 3.5       to the left,  agree=0.646, adj=0.195, (0 split)
##
## Node number 27: 378 observations,    complexity param=0.01012668
##   mean=1202244, MSE=1.953539e+11
##   left son=54 (292 obs) right son=55 (86 obs)
##   Primary splits:
##       sqft_living15 < 3210      to the left,  improve=0.2713804, (0 missing)
##       sqft_living   < 3225      to the left,  improve=0.2451904, (0 missing)
##       sqft_above    < 2995      to the left,  improve=0.1517278, (0 missing)
##       grade         < 9.5       to the left,  improve=0.1360036, (0 missing)
##       waterfront    < 0.5       to the left,  improve=0.1297860, (0 missing)
##   Surrogate splits:
##       sqft_above  < 3095      to the left,  agree=0.810, adj=0.163, (0 split)
##       sqft_living < 3790      to the left,  agree=0.802, adj=0.128, (0 split)
##       sqft_lot15  < 15623.5   to the left,  agree=0.788, adj=0.070, (0 split)
##       waterfront  < 0.5       to the left,  agree=0.783, adj=0.047, (0 split)
##       sqft_lot    < 62072.5   to the left,  agree=0.780, adj=0.035, (0 split)
##
## Node number 30: 182 observations,    complexity param=0.01178433
##   mean=1839907, MSE=5.520121e+11
##   left son=60 (19 obs) right son=61 (163 obs)
##   Primary splits:
##       lat           < 47.52355  to the left,  improve=0.2321187, (0 missing)
##       yr_built      < 1943.5    to the right, improve=0.1455867, (0 missing)
##       sqft_living15 < 2910      to the left,  improve=0.1255228, (0 missing)
##       sqft_living   < 5005      to the left,  improve=0.1220854, (0 missing)
##       grade         < 9.5       to the left,  improve=0.1174251, (0 missing)
##   Surrogate splits:
##       zipcode    < 98003.5   to the left,  agree=0.912, adj=0.158, (0 split)
##       sqft_lot   < 40109.5   to the right, agree=0.907, adj=0.105, (0 split)
##       sqft_above < 5070      to the right, agree=0.901, adj=0.053, (0 split)
##
## Node number 31: 28 observations,    complexity param=0.01043418
##   mean=3398277, MSE=1.651598e+12
##   left son=62 (19 obs) right son=63 (9 obs)
##   Primary splits:
##       bathrooms   < 4.625     to the left,  improve=0.4464995, (0 missing)
##       sqft_living < 7245      to the left,  improve=0.3816591, (0 missing)
##       zipcode     < 98107     to the right, improve=0.2206444, (0 missing)
##       sqft_above  < 5685      to the left,  improve=0.2139866, (0 missing)
##       day         < 17.5      to the right, improve=0.1636134, (0 missing)
##   Surrogate splits:
##       sqft_living  < 7030      to the left,  agree=0.929, adj=0.778, (0 split)
##       sqft_above   < 6115      to the left,  agree=0.786, adj=0.333, (0 split)
##       yr_renovated < 1986      to the left,  agree=0.786, adj=0.333, (0 split)
##       age          < 21.5      to the right, agree=0.786, adj=0.333, (0 split)
##       lat          < 47.72915  to the left,  agree=0.750, adj=0.222, (0 split)
##
## Node number 52: 885 observations
##   mean=736638.4, MSE=4.350047e+10
##
## Node number 53: 694 observations
##   mean=976837.4, MSE=9.087592e+10
```

```
##
## Node number 54: 292 observations
##   mean=1077288, MSE=1.044727e+11
##
## Node number 55: 86 observations
##   mean=1626514, MSE=2.709072e+11
##
## Node number 60: 19 observations
##   mean=791460.5, MSE=2.221173e+11
##
## Node number 61: 163 observations
##   mean=1962119, MSE=4.473981e+11
##
## Node number 62: 19 observations
##   mean=2807250, MSE=5.017792e+11
##
## Node number 63: 9 observations
##   mean=4646000, MSE=1.784743e+12
##
## 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_living< 2025 4661 9.108298e+13  461885.5 *
##       11) sqft_living>=2025 2336 1.092640e+14  653624.1 *
##    3) grade>=8.5 2998 8.857481e+14  950192.7
##      6) sqft_living< 4062.5 2519 3.159003e+14  830227.4
##       12) lat< 47.5231 562 2.021623e+13  546342.5 *
##       13) lat>=47.5231 1957 2.373855e+14  911751.8
##         26) yr_built>=1973.5 1579 1.240079e+14  842210.3
##           52) sqft_living< 3067.5 885 3.849791e+13  736638.4 *
##           53) sqft_living>=3067.5 694 6.306789e+13  976837.4 *
##         27) yr_built< 1973.5 378 7.384378e+13 1202244.0
##           54) sqft_living15< 3210 292 3.050601e+13 1077288.0 *
##           55) sqft_living15>=3210 86 2.329802e+13 1626514.0 *
##      7) sqft_living>=4062.5 479 3.429471e+14 1581075.0
##       14) long>=-122.1875 269 5.588632e+13 1216803.0 *
##       15) long< -122.1875 210 2.056429e+14 2047690.0
##         30) grade< 11.5 182 1.004662e+14 1839907.0
##           60) lat< 47.52355 19 4.220228e+12  791460.5 *
##           61) lat>=47.52355 163 7.292589e+13 1962119.0 *
##         31) grade>=11.5 28 4.624475e+13 3398277.0
##           62) bathrooms< 4.625 19 9.533805e+12 2807250.0 *
##           63) bathrooms>=4.625 9 1.606269e+13 4646000.0 *
```

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




```
# 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: 222404.107685843"
```

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 22-5-1 network with 121 weights
## inputs: bedrooms bathrooms sqft_living sqft_lot floors waterfront view condition grade sqft_above yr_built
## 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: 377755.844677147"
```

Support Vector Machine (SVM) Model

```
# [Luciano]

# Build the SVM model
svm_model <- svm(price ~ ., data = train.dat)
```

```r
print(summary(svm_model))
```

```
##
## Call:
## svm(formula = price ~ ., data = train.dat)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.04545455
##     epsilon:  0.1
##
##
## Number of Support Vectors:  8983
```

```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: 191567.001160528"
```

Logistic Regression Model

```r
# [Luciano]

# Calculate the median price
median_price <- median(train.dat$price)

# Create a binary variable based on the median price
train.dat$price_binary <- ifelse(train.dat$price > median_price, 1, 0)
test.dat$price_binary <- ifelse(test.dat$price > median_price, 1, 0)
```

```r
# [Luciano]

log_model <- glm(price_binary ~ ., data = train.dat, family = binomial())
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
print(summary(log_model))
```

```
##
## Call:
## glm(formula = price_binary ~ ., family = binomial(), data = train.dat)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.425e+04  2.855e+05  -0.050    0.960
## price          2.855e-02  4.822e-02   0.592    0.554
## bedrooms       2.426e+00  8.210e+01   0.030    0.976
## bathrooms     -2.561e+00  8.396e+01  -0.031    0.976
## sqft_living   -1.823e-02  1.597e-01  -0.114    0.909
## sqft_lot      -1.091e-04  3.074e-03  -0.035    0.972
## floors         5.457e-01  9.466e+01   0.006    0.995
## waterfront    -3.882e+02  1.797e+04  -0.022    0.983
## view          -1.085e+00  1.056e+02  -0.010    0.992
## condition      1.775e+00  7.339e+01   0.024    0.981
## grade          1.791e+00  5.970e+01   0.030    0.976
```

```
## sqft_above     1.503e-02  1.839e-01   0.082    0.935
## yr_built      -2.419e-02  2.658e+01  -0.001    0.999
## yr_renovated  -9.940e-02  1.980e+01  -0.005    0.996
## zipcode        4.170e-02  8.045e-01   0.052    0.959
## lat            1.837e+00  3.396e+02   0.005    0.996
## long          -5.804e+00  3.231e+02  -0.018    0.986
## sqft_living15  4.903e-03  1.132e-01   0.043    0.965
## sqft_lot15     8.516e-05  3.210e-03   0.027    0.979
## year          -1.719e+00  1.541e+02  -0.011    0.991
## month         -9.957e-01  3.709e+01  -0.027    0.979
## day           -2.658e-02  2.906e+00  -0.009    0.993
## renovated      1.962e+02  3.809e+04   0.005    0.996
## age           -3.870e-02  2.659e+01  -0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.0973e+04  on 15128  degrees of freedom
## Residual deviance: 3.0630e-03  on 15105  degrees of freedom
## AIC: 48.003
##
## Number of Fisher Scoring iterations: 25
```

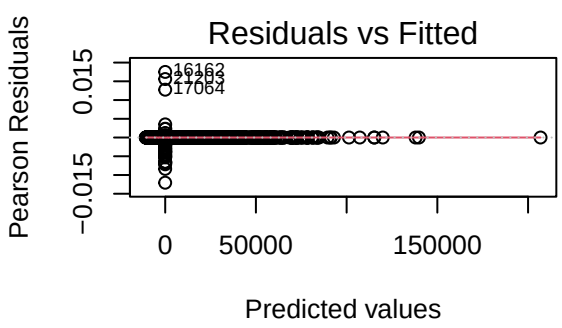
```r
# Predicting probabilities
log_predictions_prob <- predict(log_model, test.dat, type = "response")

# Binning probabilities into two categories: 0 and 1
log_predictions <- ifelse(log_predictions_prob > 0.5, 1, 0)

# Ensuring factors have the same levels
log_predictions_factor <- factor(log_predictions, levels = c(0, 1))
test_price_binary_factor <- factor(test.dat$price_binary, levels = c(0, 1))

# Confusion Matrix
log_conf_mat <- caret::confusionMatrix(log_predictions_factor, test_price_binary_factor)
print(log_conf_mat)
```

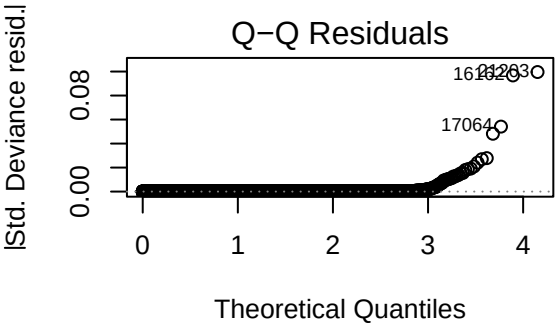
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3281    0
##          1    0 3203
##
##                Accuracy : 1
##                  95% CI : (0.9994, 1)
##     No Information Rate : 0.506
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.506
##          Detection Rate : 0.506
##    Detection Prevalence : 0.506
##       Balanced Accuracy : 1.000
##
```

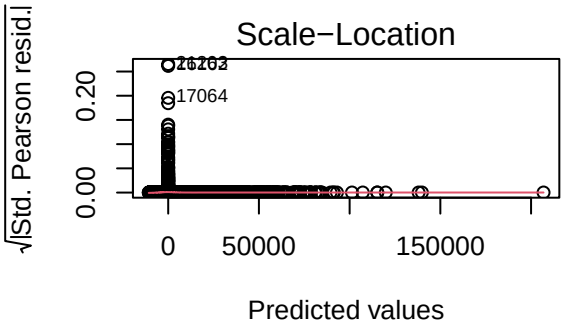
```
##         'Positive' Class : 0
##
```

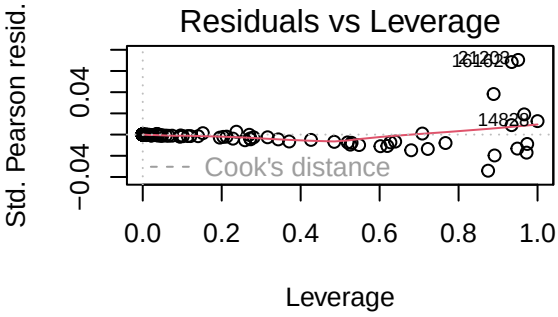
```
# Diagnostic Plots for the Logistic Regression
par(mfrow = c(2, 2))
plot(log_model)
```

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

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









4 UPs

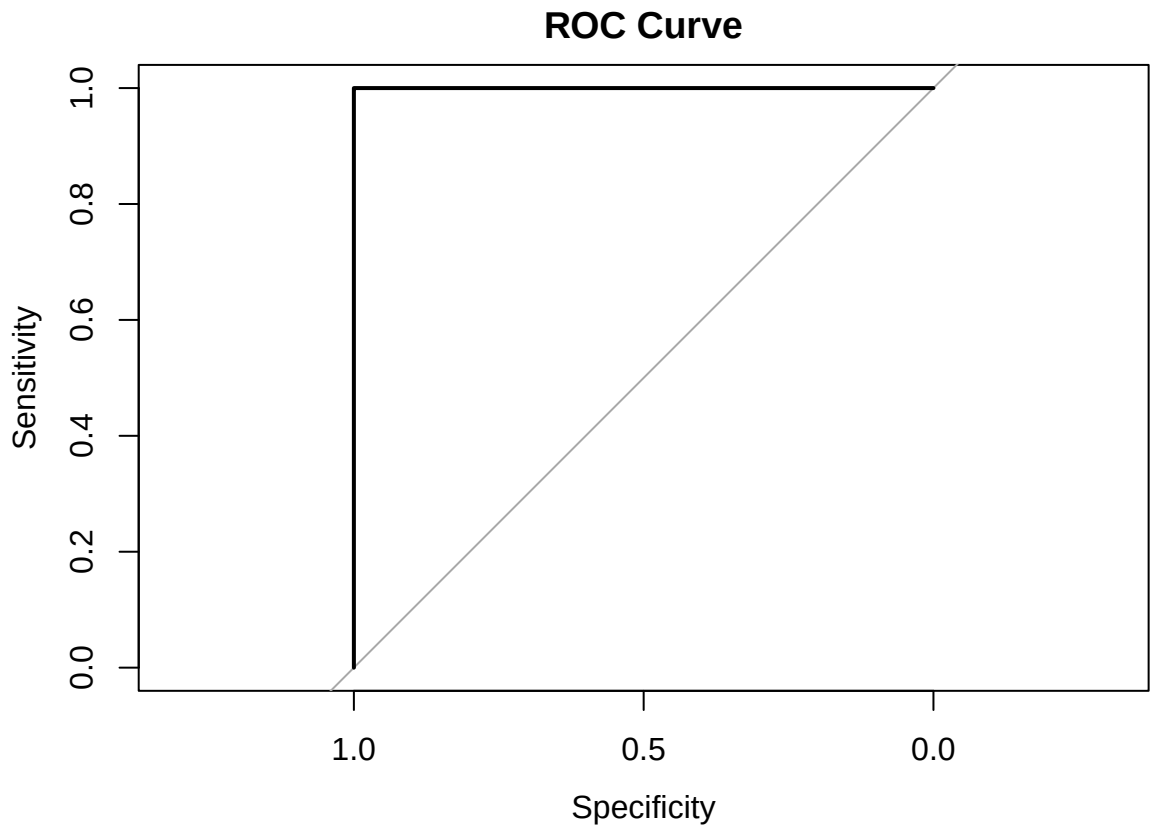
```
# Assuming binary classification
roc_curve <- pROC::roc(test.dat$price_binary, log_predictions)
```

***Analysis*: The diagnostic plots suggest that while the regression model captures the overall trend, there are indications of non-normality in residuals, potential heteroscedasticity, and the presence of influential outliers, warranting further investigation into data transformation or alternative modeling approaches.**

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

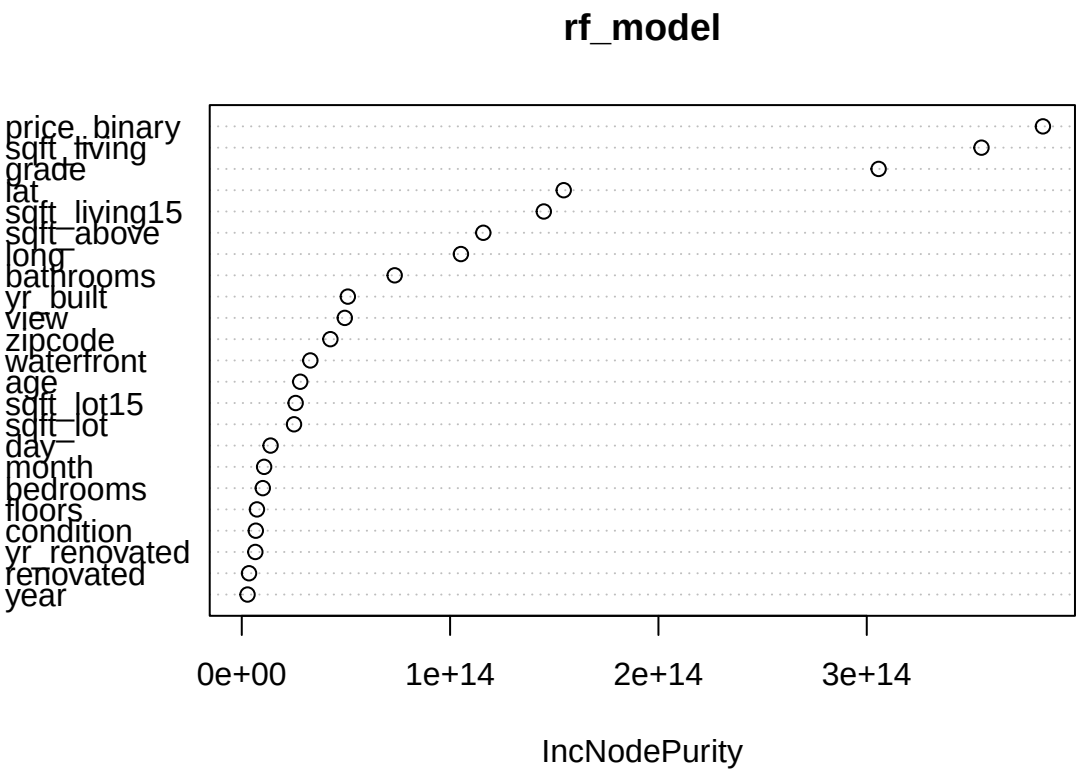
```
plot(roc_curve, main = "ROC Curve")
```



ROC Curve

```
# randomForest model for variable importance
rf_model <- randomForest(price ~ ., data = train.dat)
randomForest::varImpPlot(rf_model)
```

***Analysis*: The ROC curve demonstrates perfect classification performance with maximal sensitivity and**



specificity
Random Forest plot

***Analysis*: The Variable Importance plot from a Random Forest model highlights 'price.binary' as the most influential predictor, with 'sqft_living', 'grade', and 'sqft_living15' also contributing significantly to model accuracy, suggesting their strong predictive power in the Random Forest model.**

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 RMSE.test
## 1 Linear Regression All (Baseline)        0.703594      0.6987682  208302.5
##       SSE.test
## 1 2.813404e+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.