HARVARD EXTENSION SCHOOL

EXT CSCI E-106 Model Data Class Group Project Template

Will Greaves   Flora Lo   Matt Michel   Thaylan Toth   Kaylee Vo   Zhenzhen Yin

05 December 2023

Abstract

This is the location for your abstract. It must consist of two paragraphs.

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 this project, our goal is to build a statistical model that can predict house sales prices in Kings county, USA based on house sales data collected in that area between May 2014 and May 2015. The dataset contains information on 21613 houses, including the sale price, number of rooms, square footage, year built and renovated, view, and condition of the property. The house sale price was used as the outcome variable and all other variables were considered as independent variables. 70% of the dataset was used as training set and 30% was used as testing set.

Using this framework, we built several different models using linear regression, logistic regression, regression tree, and neural network. When building each model, feature selection methods were used and the appropriate diagnostic tests were applied to verify that model assumptions were met.

Finally, the performance of each models was evaluated by its accuracy in predicting house prices in the test set, specifically by examining the MSE of predicted values. Based on that, we propose that the best predictive model is _____. This model indicated that the most important factors that influence house prices in King County, USA are _____.

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?

```r
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
str(HouseSales)
```

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : chr  "20141013T000000" "20141209T000000" "20150225T000000" "20141209T000000" ...
##  $ price        : chr  " $221,900.00 " " $538,000.00 " " $180,000.00 " " $604,000.00 " ...
##  $ 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 ...
```

```r
# Data cleaning (price)
df = read.csv("KC_House_Sales.csv")
df$price = parse_number(df$price)
```

```r
# Transformation (date)
df$year = as.integer(substr(df$date, 1, 4))
df$month = as.integer(substr(df$date, 5, 6))

year = df$year
month = df$month

df = subset(df, select = -c(id, date))
```

```r
# dummy (month, True(1), False(0))
df$month_Jan = ifelse(df$month == 1, 1, 0)
df$month_Feb = ifelse(df$month == 2, 1, 0)
df$month_Mar = ifelse(df$month == 3, 1, 0)
df$month_Apr = ifelse(df$month == 4, 1, 0)
df$month_May = ifelse(df$month == 5, 1, 0)
df$month_Jun = ifelse(df$month == 6, 1, 0)
df$month_Jul = ifelse(df$month == 7, 1, 0)
df$month_Aug = ifelse(df$month == 8, 1, 0)
df$month_Sep = ifelse(df$month == 9, 1, 0)
df$month_Oct = ifelse(df$month == 10, 1, 0)
df$month_Nov = ifelse(df$month == 11, 1, 0)
df$month_Dec = ifelse(df$month == 12, 1, 0)
```

```r
df = subset(df, select = -(month))
```

```r
# dummy (zipcode, Divided into two groups, 980xx and 981xx)
df$zipcode_start = as.integer(substr(df$zipcode, 1, 3))
df = subset(df, select = -(zipcode))
```

```r
summary(df)
```

```
##      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         lat             long
##  Min.   :1900   Min.   :   0.0   Min.   :47.16   Min.   :-122.5
##  1st Qu.:1951   1st Qu.:   0.0   1st Qu.:47.47   1st Qu.:-122.3
##  Median :1975   Median :   0.0   Median :47.57   Median :-122.2
##  Mean   :1971   Mean   :  84.4   Mean   :47.56   Mean   :-122.2
##  3rd Qu.:1997   3rd Qu.:   0.0   3rd Qu.:47.68   3rd Qu.:-122.1
##  Max.   :2015   Max.   :2015.0   Max.   :47.78   Max.   :-121.3
##  sqft_living15    sqft_lot15          year         month_Jan
##  Min.   : 399   Min.   :    651   Min.   :2014   Min.   :0.00000
##  1st Qu.:1490   1st Qu.:   5100   1st Qu.:2014   1st Qu.:0.00000
##  Median :1840   Median :   7620   Median :2014   Median :0.00000
##  Mean   :1987   Mean   :  12768   Mean   :2014   Mean   :0.04525
##  3rd Qu.:2360   3rd Qu.:  10083   3rd Qu.:2015   3rd Qu.:0.00000
##  Max.   :6210   Max.   : 871200   Max.   :2015   Max.   :1.00000
##    month_Feb         month_Mar         month_Apr        month_May
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.00000   Median :0.00000   Median :0.0000   Median :0.0000
##  Mean   :0.05784   Mean   :0.08675   Mean   :0.1032   Mean   :0.1117
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
##    month_Jun         month_Jul        month_Aug         month_Sep
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.0000   Median :0.0000   Median :0.00000   Median :0.00000
##  Mean   :0.1009   Mean   :0.1023   Mean   :0.08976   Mean   :0.08208
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
##    month_Oct         month_Nov         month_Dec       zipcode_start
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :980.0
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:980.0
```

```
##   Median :0.00000   Median :0.00000   Median :0.00000   Median :980.0
##   Mean    :0.08689   Mean    :0.06528   Mean    :0.06806   Mean    :980.4
##   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:981.0
##   Max.    :1.00000   Max.    :1.00000   Max.    :1.00000   Max.    :981.0
```

```r
cor_matrix = round(cor(df), 2)
cor_matrix
```

```
##              price bedrooms bathrooms sqft_living sqft_lot floors waterfront
## price         1.00     0.31      0.53        0.70     0.09   0.26       0.27
## bedrooms      0.31     1.00      0.52        0.58     0.03   0.18      -0.01
## bathrooms     0.53     0.52      1.00        0.75     0.09   0.50       0.06
## sqft_living   0.70     0.58      0.75        1.00     0.17   0.35       0.10
## sqft_lot      0.09     0.03      0.09        0.17     1.00  -0.01       0.02
## floors        0.26     0.18      0.50        0.35    -0.01   1.00       0.02
## waterfront    0.27    -0.01      0.06        0.10     0.02   0.02       1.00
## view          0.40     0.08      0.19        0.28     0.07   0.03       0.40
## condition     0.04     0.03     -0.12       -0.06    -0.01  -0.26       0.02
## grade         0.67     0.36      0.66        0.76     0.11   0.46       0.08
## sqft_above    0.61     0.48      0.69        0.88     0.18   0.52       0.07
## sqft_basement 0.32     0.30      0.28        0.44     0.02  -0.25       0.08
## yr_built      0.05     0.15      0.51        0.32     0.05   0.49      -0.03
## yr_renovated  0.13     0.02      0.05        0.06     0.01   0.01       0.09
## lat           0.31    -0.01      0.02        0.05    -0.09   0.05      -0.01
## long          0.02     0.13      0.22        0.24     0.23   0.13      -0.04
## sqft_living15 0.59     0.39      0.57        0.76     0.14   0.28       0.09
## sqft_lot15    0.08     0.03      0.09        0.18     0.72  -0.01       0.03
## year          0.00    -0.01     -0.03       -0.03     0.01  -0.02       0.00
## month_Jan    -0.01     0.00      0.00        0.00     0.01  -0.01       0.00
## month_Feb    -0.02    -0.01     -0.02       -0.02    -0.01  -0.01      -0.01
## month_Mar     0.00     0.00     -0.02       -0.02     0.00  -0.02       0.00
## month_Apr     0.02     0.00      0.00       -0.01    -0.01   0.00       0.00
## month_May     0.01     0.00      0.00        0.00     0.01  -0.01      -0.01
## month_Jun     0.02     0.02      0.02        0.02    -0.01   0.01       0.00
## month_Jul     0.00     0.01      0.01        0.02    -0.01   0.02       0.00
## month_Aug     0.00    -0.01      0.00        0.00     0.00   0.00       0.00
## month_Sep    -0.01     0.00      0.01        0.00     0.00   0.00       0.00
## month_Oct     0.00     0.00      0.00        0.00     0.00   0.00       0.00
## month_Nov    -0.01    -0.01     -0.01       -0.01     0.00   0.00       0.01
## month_Dec    -0.01     0.01      0.00        0.00     0.00   0.00       0.00
## zipcode_start -0.01    -0.18     -0.24       -0.26    -0.18  -0.05       0.01
##              view condition grade sqft_above sqft_basement yr_built
## price         0.40      0.04  0.67       0.61          0.32     0.05
## bedrooms      0.08      0.03  0.36       0.48          0.30     0.15
## bathrooms     0.19     -0.12  0.66       0.69          0.28     0.51
## sqft_living   0.28     -0.06  0.76       0.88          0.44     0.32
## sqft_lot      0.07     -0.01  0.11       0.18          0.02     0.05
## floors        0.03     -0.26  0.46       0.52         -0.25     0.49
## waterfront    0.40      0.02  0.08       0.07          0.08    -0.03
## view          1.00      0.05  0.25       0.17          0.28    -0.05
## condition     0.05      1.00 -0.14      -0.16          0.17    -0.36
## grade         0.25     -0.14  1.00       0.76          0.17     0.45
## sqft_above    0.17     -0.16  0.76       1.00         -0.05     0.42
## sqft_basement 0.28      0.17  0.17      -0.05          1.00    -0.13
## yr_built     -0.05     -0.36  0.45       0.42         -0.13     1.00
## yr_renovated  0.10     -0.06  0.01       0.02          0.07    -0.22
## lat           0.01     -0.01  0.11       0.00          0.11    -0.15
## long         -0.08     -0.11  0.20       0.34         -0.14     0.41
## sqft_living15 0.28     -0.09  0.71       0.73          0.20     0.33
## sqft_lot15    0.07      0.00  0.12       0.19          0.02     0.07
## year          0.00     -0.05 -0.03      -0.02         -0.02     0.00
```

```
## month_Jan        0.00     -0.02  0.00      0.01          -0.01     0.01
## month_Feb        0.00      0.00 -0.02     -0.02           0.00     0.00
## month_Mar        0.00     -0.03 -0.02     -0.02          -0.01     0.00
## month_Apr        0.00     -0.03  0.00      0.00          -0.01     0.01
## month_May        0.00      0.01  0.01      0.00           0.01    -0.01
## month_Jun        0.00      0.03  0.02      0.01           0.01     0.00
## month_Jul       -0.01      0.02  0.02      0.02           0.00     0.01
## month_Aug        0.00      0.01  0.01      0.00           0.00     0.01
## month_Sep        0.01      0.01  0.00      0.00           0.00     0.00
## month_Oct        0.00      0.00  0.00      0.00           0.00    -0.01
## month_Nov        0.00      0.00 -0.01      0.00          -0.01    -0.01
## month_Dec       -0.01     -0.01  0.00      0.00           0.01     0.00
## zipcode_start    0.08      0.04 -0.23     -0.35           0.12    -0.46
##              yr_renovated    lat  long sqft_living15 sqft_lot15   year month_Jan
## price                0.13   0.31  0.02          0.59       0.08   0.00     -0.01
## bedrooms             0.02  -0.01  0.13          0.39       0.03  -0.01      0.00
## bathrooms            0.05   0.02  0.22          0.57       0.09  -0.03      0.00
## sqft_living          0.06   0.05  0.24          0.76       0.18  -0.03      0.00
## sqft_lot             0.01  -0.09  0.23          0.14       0.72   0.01      0.01
## floors               0.01   0.05  0.13          0.28      -0.01  -0.02     -0.01
## waterfront           0.09  -0.01 -0.04          0.09       0.03   0.00      0.00
## view                 0.10   0.01 -0.08          0.28       0.07   0.00      0.00
## condition           -0.06  -0.01 -0.11         -0.09       0.00  -0.05     -0.02
## grade                0.01   0.11  0.20          0.71       0.12  -0.03      0.00
## sqft_above           0.02   0.00  0.34          0.73       0.19  -0.02      0.01
## sqft_basement        0.07   0.11 -0.14          0.20       0.02  -0.02     -0.01
## yr_built            -0.22  -0.15  0.41          0.33       0.07   0.00      0.01
## yr_renovated         1.00   0.03 -0.07          0.00       0.01  -0.02     -0.01
## lat                  0.03   1.00 -0.14          0.05      -0.09  -0.03     -0.01
## long                -0.07  -0.14  1.00          0.33       0.25   0.00      0.00
## sqft_living15        0.00   0.05  0.33          1.00       0.18  -0.02      0.00
## sqft_lot15           0.01  -0.09  0.25          0.18       1.00   0.00      0.00
## year                -0.02  -0.03  0.00         -0.02       0.00   1.00      0.32
## month_Jan           -0.01  -0.01  0.00          0.00       0.00   0.32      1.00
## month_Feb           -0.02  -0.02  0.00         -0.02      -0.01   0.36     -0.05
## month_Mar            0.00  -0.01  0.00         -0.02       0.00   0.45     -0.07
## month_Apr           -0.01   0.00  0.00          0.00       0.00   0.49     -0.07
## month_May            0.02   0.01  0.00          0.00       0.01  -0.04     -0.08
## month_Jun            0.00   0.01  0.00          0.02       0.00  -0.23     -0.07
## month_Jul            0.01   0.00  0.02          0.02       0.00  -0.23     -0.07
## month_Aug            0.00   0.01  0.01          0.00       0.00  -0.22     -0.07
## month_Sep            0.01   0.00  0.00          0.00      -0.01  -0.21     -0.07
## month_Oct            0.01   0.00 -0.01          0.00       0.00  -0.21     -0.07
## month_Nov            0.00   0.00 -0.01         -0.02       0.00  -0.18     -0.06
## month_Dec            0.00   0.00 -0.01          0.00       0.00  -0.19     -0.06
## zipcode_start        0.08   0.32 -0.71         -0.37      -0.20   0.00      0.00
##              month_Feb month_Mar month_Apr month_May month_Jun month_Jul
## price            -0.02      0.00      0.02      0.01      0.02      0.00
## bedrooms         -0.01      0.00      0.00      0.00      0.02      0.01
## bathrooms        -0.02     -0.02      0.00      0.00      0.02      0.01
## sqft_living      -0.02     -0.02     -0.01      0.00      0.02      0.02
## sqft_lot         -0.01      0.00     -0.01      0.01     -0.01     -0.01
## floors           -0.01     -0.02      0.00     -0.01      0.01      0.02
## waterfront       -0.01      0.00      0.00     -0.01      0.00      0.00
## view              0.00      0.00      0.00      0.00      0.00     -0.01
## condition         0.00     -0.03     -0.03      0.01      0.03      0.02
## grade            -0.02     -0.02      0.00      0.01      0.02      0.02
## sqft_above       -0.02     -0.02      0.00      0.00      0.01      0.02
## sqft_basement     0.00     -0.01     -0.01      0.01      0.01      0.00
## yr_built          0.00      0.00      0.01     -0.01      0.00      0.01
```

```
## yr_renovated       -0.02      0.00     -0.01      0.02      0.00      0.01
## lat                -0.02     -0.01      0.00      0.01      0.01      0.00
## long                0.00      0.00      0.00      0.00      0.00      0.02
## sqft_living15       -0.02     -0.02      0.00      0.00      0.02      0.02
## sqft_lot15          -0.01      0.00      0.00      0.01      0.00      0.00
## year                 0.36      0.45      0.49     -0.04     -0.23     -0.23
## month_Jan           -0.05     -0.07     -0.07     -0.08     -0.07     -0.07
## month_Feb            1.00     -0.08     -0.08     -0.09     -0.08     -0.08
## month_Mar           -0.08      1.00     -0.10     -0.11     -0.10     -0.10
## month_Apr           -0.08     -0.10      1.00     -0.12     -0.11     -0.11
## month_May           -0.09     -0.11     -0.12      1.00     -0.12     -0.12
## month_Jun           -0.08     -0.10     -0.11     -0.12      1.00     -0.11
## month_Jul           -0.08     -0.10     -0.11     -0.12     -0.11      1.00
## month_Aug           -0.08     -0.10     -0.11     -0.11     -0.11     -0.11
## month_Sep           -0.07     -0.09     -0.10     -0.11     -0.10     -0.10
## month_Oct           -0.08     -0.10     -0.10     -0.11     -0.10     -0.10
## month_Nov           -0.07     -0.08     -0.09     -0.09     -0.09     -0.09
## month_Dec           -0.07     -0.08     -0.09     -0.10     -0.09     -0.09
## zipcode_start       -0.01      0.00      0.00      0.01      0.00     -0.02
##               month_Aug month_Sep month_Oct month_Nov month_Dec zipcode_start
## price              0.00     -0.01      0.00     -0.01     -0.01         -0.01
## bedrooms          -0.01      0.00      0.00     -0.01      0.01         -0.18
## bathrooms          0.00      0.01      0.00     -0.01      0.00         -0.24
## sqft_living        0.00      0.00      0.00     -0.01      0.00         -0.26
## sqft_lot           0.00      0.00      0.00      0.00      0.00         -0.18
## floors             0.00      0.00      0.00      0.00      0.00         -0.05
## waterfront         0.00      0.00      0.00      0.01      0.00          0.01
## view               0.00      0.01      0.00      0.00     -0.01          0.08
## condition          0.01      0.01      0.00      0.00     -0.01          0.04
## grade              0.01      0.00      0.00     -0.01      0.00         -0.23
## sqft_above         0.00      0.00      0.00      0.00      0.00         -0.35
## sqft_basement      0.00      0.00      0.00     -0.01      0.01          0.12
## yr_built           0.01      0.00     -0.01     -0.01      0.00         -0.46
## yr_renovated       0.00      0.01      0.01      0.00      0.00          0.08
## lat                0.01      0.00      0.00      0.00      0.00          0.32
## long               0.01      0.00     -0.01     -0.01     -0.01         -0.71
## sqft_living15      0.00      0.00      0.00     -0.02      0.00         -0.37
## sqft_lot15         0.00     -0.01      0.00      0.00      0.00         -0.20
## year              -0.22     -0.21     -0.21     -0.18     -0.19          0.00
## month_Jan         -0.07     -0.07     -0.07     -0.06     -0.06          0.00
## month_Feb         -0.08     -0.07     -0.08     -0.07     -0.07         -0.01
## month_Mar         -0.10     -0.09     -0.10     -0.08     -0.08          0.00
## month_Apr         -0.11     -0.10     -0.10     -0.09     -0.09          0.00
## month_May         -0.11     -0.11     -0.11     -0.09     -0.10          0.01
## month_Jun         -0.11     -0.10     -0.10     -0.09     -0.09          0.00
## month_Jul         -0.11     -0.10     -0.10     -0.09     -0.09         -0.02
## month_Aug          1.00     -0.09     -0.10     -0.08     -0.08          0.00
## month_Sep         -0.09      1.00     -0.09     -0.08     -0.08          0.00
## month_Oct         -0.10     -0.09      1.00     -0.08     -0.08          0.01
## month_Nov         -0.08     -0.08     -0.08      1.00     -0.07          0.00
## month_Dec         -0.08     -0.08     -0.08     -0.07      1.00          0.00
## zipcode_start      0.00      0.00      0.01      0.00      0.00          1.00
```
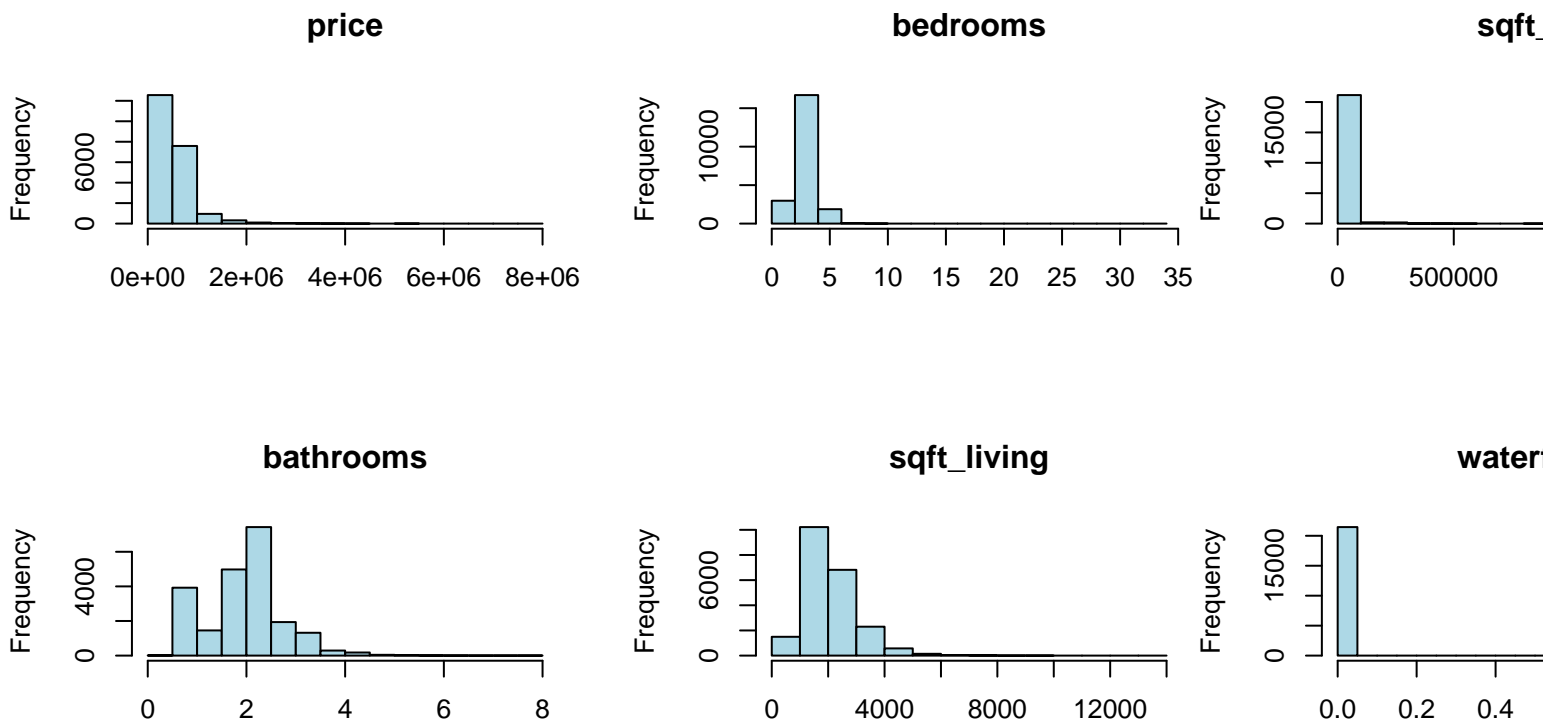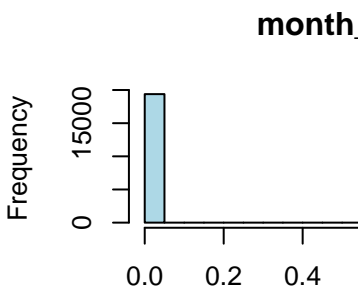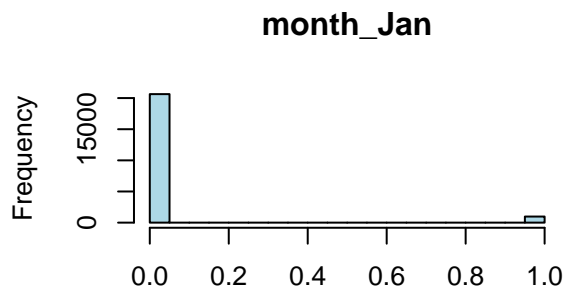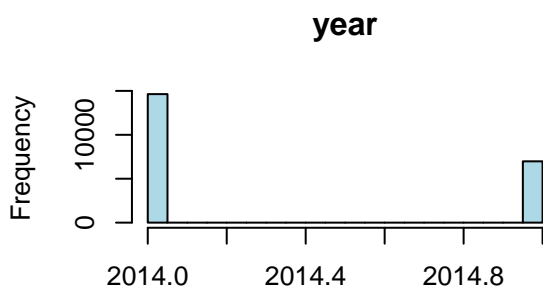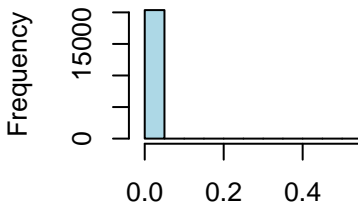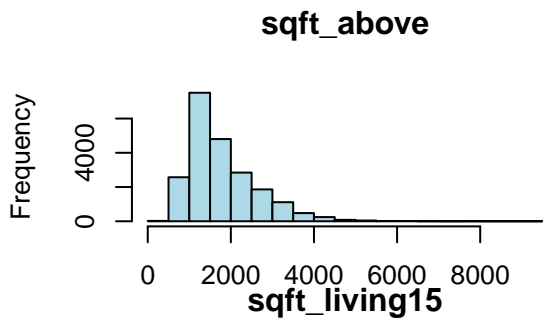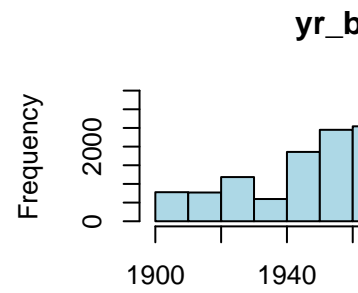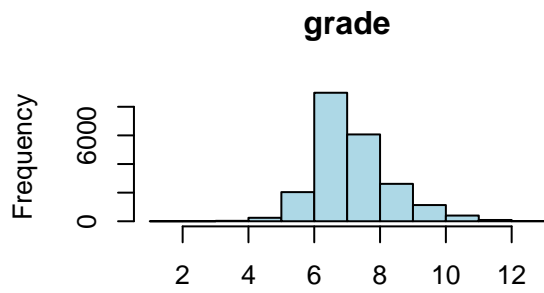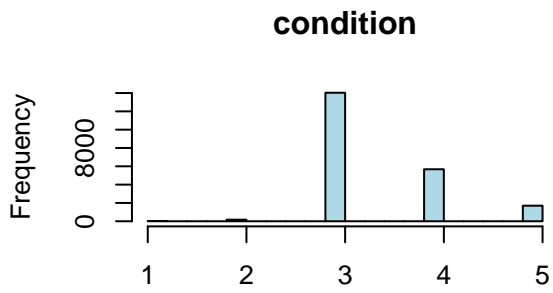
```r
corrplot(cor_matrix, method = "circle")
```
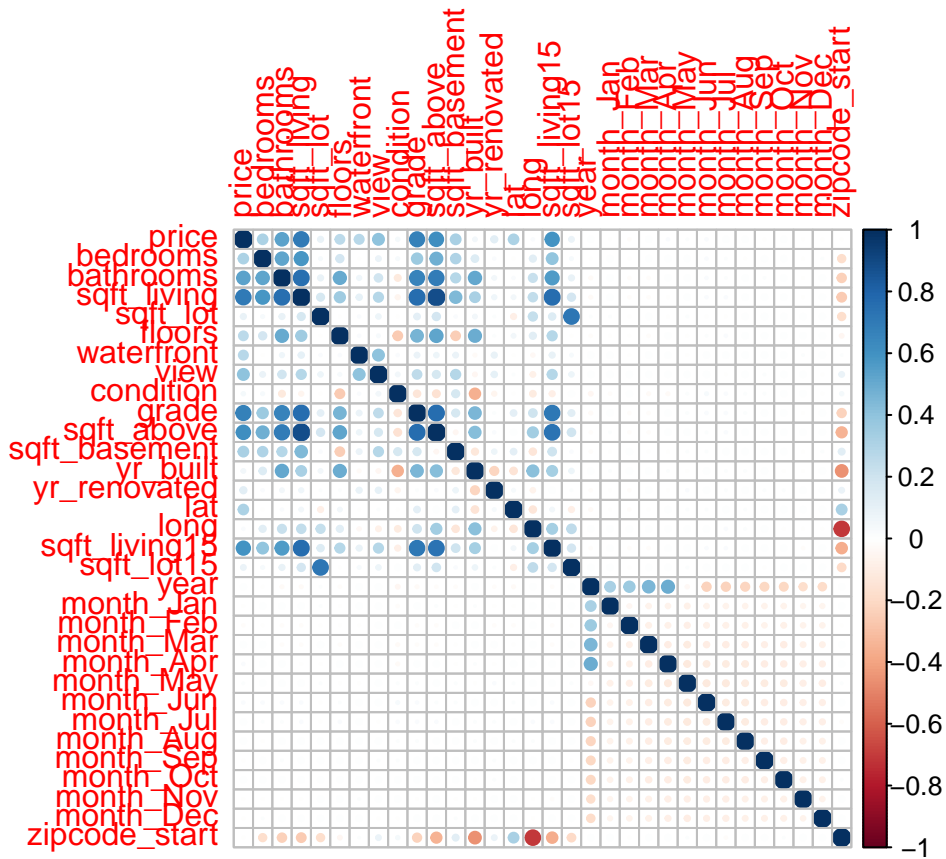
```
par(mfrow=c(2,2))

for (i in 1:ncol(df)){
  hist(df[,i], main = names(df[i]), xlab = NULL, col = "lightblue")
}
```
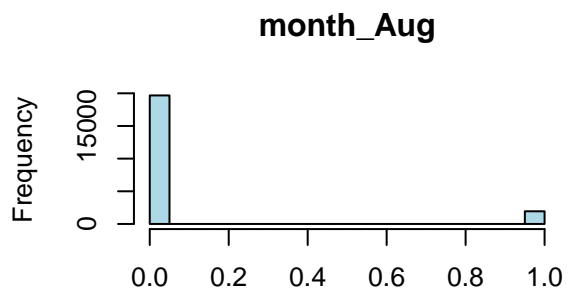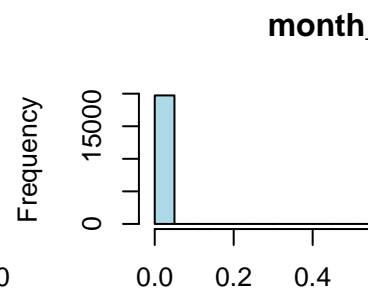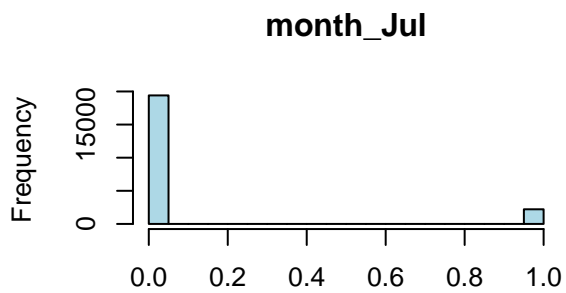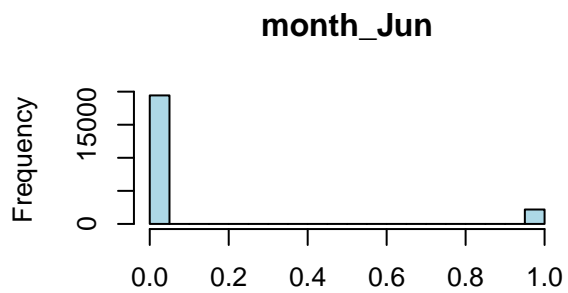
month_Jun

month_Jul

month_

month_Aug

month_Sep

month_
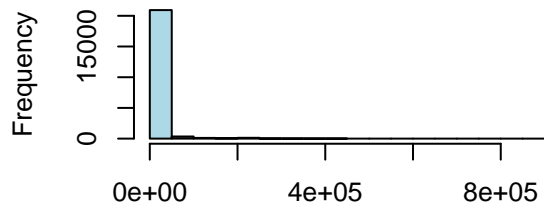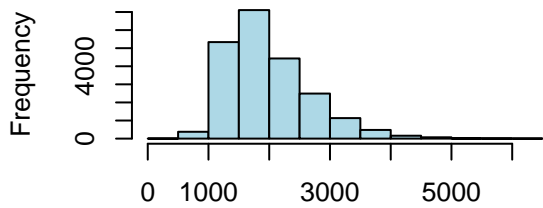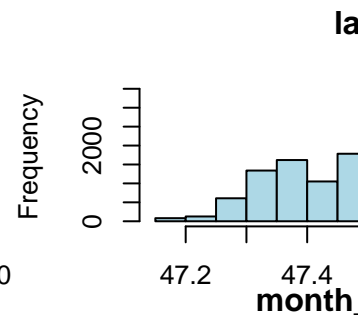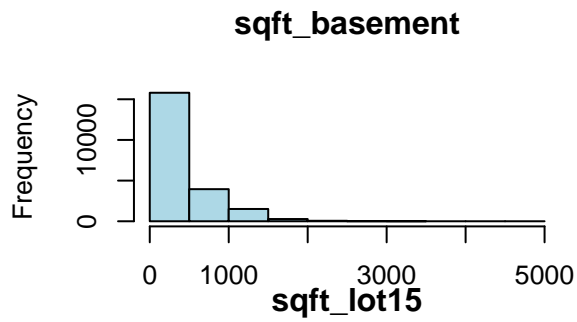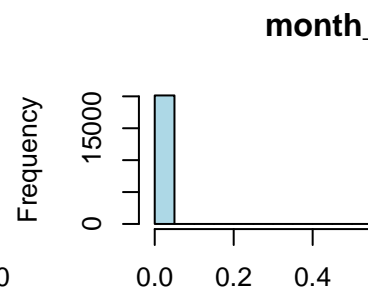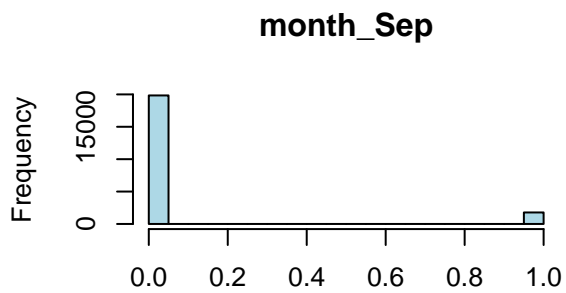
III. Model Development Process (15 points)

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.

```r
set.seed(1023)

response <- "price"

#Dropping variables as specified (id was dropped previously)
drop.vars <- c("lat", "long")
kc.house.df <- dplyr::select(df, -drop.vars)
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##    # Was:
##    data %>% select(drop.vars)
##
##    # Now:
##    data %>% select(all_of(drop.vars))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
#Renaming to conform to unified convention
kc.house.df <- kc.house.df %>% rename("year_built" = "yr_built")
```

```r
#Feature Ideas
#Quality Adjusted Features
transform_sqft_adj_grade <- function(kc.house.df){

  sqft_adj_grade <- kc.house.df[, "sqft_living"] / kc.house.df[, "grade"]

  return (sqft_adj_grade)
}


transform_sqft_adj_condition <- function(kc.house.df){

  sqft_adj_condition <- kc.house.df[, "sqft_living"] / kc.house.df[, "condition"]

  return (sqft_adj_condition)
}


transform_sqft_adj_waterfront <- function(kc.house.df){

  sqft_adj_waterfront <- kc.house.df[, "sqft_living"] * kc.house.df[, "waterfront"]

  return (sqft_adj_waterfront)
}

transform_poly_sqft_living <- function(kc.house.df){
  #Center variables for polynomial terms
  sqft_living <- (kc.house.df$sqft_living - mean(kc.house.df$sqft_living))
  sqft_living_squared <- sqft_living^2

  return (list(center = sqft_living, squared = sqft_living_squared))
}

transform_poly_floor <- function(kc.house.df){
```

```r
  floors <- (kc.house.df$floors - mean(kc.house.df$floors))
  floors_squared <- floors^2

  return(list(center = floors, squared = floors_squared))
}
```

Centering the variables do not change the interpretation of the coefficients. However they do change the interpretation of the intercept. Instead of the intercept being read as the value of price when all variables are zero. It is not read as price when all variables are zero, `floors` is equal to its average and `sqft_living` is equal to its average.

```r
#Apply transformations

kc.house.df$sqft_adj_grade <- transform_sqft_adj_grade(kc.house.df)
kc.house.df$sqft_adj_condition <- transform_sqft_adj_condition(kc.house.df)
kc.house.df$sqft_adj_waterfront <- transform_sqft_adj_waterfront(kc.house.df)

res <- transform_poly_sqft_living(kc.house.df)
kc.house.df$sqft_living <- res$center
kc.house.df$sqft_living_squared <- res$squared

res <- transform_poly_floor(kc.house.df)
kc.house.df$floors <- res$center
kc.house.df$floors_squared <- res$squared

#Remove collinear variables
# sqft_basement + sqft_above = sqft_living
kc.house.df$sqft_basement <- NULL
```

```r
set.seed(1023)

#use 70% of dataset as training set and 30% as test set
train_prop <- 0.7

index <- sample(1:nrow(HouseSales), size = round(train_prop * nrow(kc.house.df)))
kc.house.train.X <- kc.house.df[index, -which(names(kc.house.df) %in% c(response))] # modified slightly
kc.house.train.y <- kc.house.df [index, response]

kc.house.test.X <- kc.house.df [-index, -which(names(kc.house.df) %in% c(response))]
kc.house.test.y <- kc.house.df [-index, response]
```

```r
#baseline model
baseline.model <- lm(price ~ ., data = cbind(price = kc.house.train.y, kc.house.train.X))
summary(baseline.model)
```

```
##
## Call:
## lm(formula = price ~ ., data = cbind(price = kc.house.train.y,
##     kc.house.train.X))
##
## Residuals:
##       Min       1Q   Median       3Q      Max
## -1855664  -100963   -12045    82995  2443584
##
## Coefficients: (1 not defined because of singularities)
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.729e+08  2.203e+07  -7.849 4.47e-15 ***
## bedrooms        -1.060e+04  2.225e+03  -4.764 1.92e-06 ***
## bathrooms        4.770e+04  3.750e+03  12.721  < 2e-16 ***
## sqft_living      4.558e+02  3.077e+01  14.814  < 2e-16 ***
## sqft_lot         3.920e-02  5.842e-02   0.671 0.502209
## floors           2.032e+04  4.930e+03   4.122 3.78e-05 ***
```

```
## waterfront           -2.363e+05  4.602e+04  -5.136 2.84e-07 ***
## view                  3.518e+04  2.500e+03  14.076  < 2e-16 ***
## condition            -1.693e+04  5.676e+03  -2.983 0.002860 **
## grade                 3.993e+04  7.750e+03   5.152 2.61e-07 ***
## sqft_above           -1.056e+01  5.310e+00  -1.989 0.046764 *
## year_built           -2.608e+03  8.765e+01 -29.757  < 2e-16 ***
## yr_renovated          3.316e+01  4.232e+00   7.835 5.00e-15 ***
## sqft_living15         7.260e+01  4.115e+00  17.641  < 2e-16 ***
## sqft_lot15           -3.580e-01  8.455e-02  -4.234 2.30e-05 ***
## year                  5.885e+04  1.074e+04   5.480 4.31e-08 ***
## month_Jan            -5.105e+04  1.438e+04  -3.550 0.000387 ***
## month_Feb            -5.179e+04  1.400e+04  -3.700 0.000216 ***
## month_Mar            -1.990e+04  1.344e+04  -1.481 0.138628
## month_Apr            -1.677e+04  1.329e+04  -1.262 0.206824
## month_May             3.392e+03  8.232e+03   0.412 0.680333
## month_Jun             3.213e+03  7.836e+03   0.410 0.681790
## month_Jul             3.071e+03  7.820e+03   0.393 0.694578
## month_Aug             3.485e+03  8.043e+03   0.433 0.664828
## month_Sep             1.317e+03  8.229e+03   0.160 0.872890
## month_Oct             1.861e+03  8.126e+03   0.229 0.818816
## month_Nov            -4.963e+02  8.712e+03  -0.057 0.954573
## month_Dec                   NA         NA      NA       NA
## zipcode_start         6.154e+04  4.269e+03  14.414  < 2e-16 ***
## sqft_adj_grade       -2.050e+03  2.048e+02 -10.012  < 2e-16 ***
## sqft_adj_condition   -3.481e+02  3.275e+01 -10.626  < 2e-16 ***
## sqft_adj_waterfront   2.432e+02  1.309e+01  18.578  < 2e-16 ***
## sqft_living_squared   2.384e-02  1.887e-03  12.631  < 2e-16 ***
## floors_squared        3.329e+04  5.567e+03   5.980 2.28e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 195200 on 15096 degrees of freedom
## Multiple R-squared:  0.7094, Adjusted R-squared:  0.7087
## F-statistic:  1151 on 32 and 15096 DF,  p-value: < 2.2e-16
```

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.).

#TODO 1. Stepwise feature selection 2. Add plots (e.g. scale location, residual vs fitted) 3. Normality plots

TODO: The following code needs to be updated once we have the updated models from stepwise feature selection.

```r
#evaluate on test data

CalcTestMetrics <- function(pred, act, n, p) {
  SST <- var(act)*(length(act)-1)
  SSE <- sum((act-pred)^2)
  SSR <- sum(pred - mean(act))^2
  rsquared <- 1- SSE/SST

  adj.rsquared <- 1 - (((1 - rsquared)*(n-1)) / (n-p-1))
  mse <- sum((act - pred)^2) / (n-p)
  mae <- (sum(abs(act-pred))) / n

  return (
    list(
      adj.rsquared = adj.rsquared,
      rsquared = rsquared,
      mse = mse,
      mae = mae
    )
  )
}


pred <- predict(baseline.model, kc.house.test.X)
```

```
## Warning in predict.lm(baseline.model, kc.house.test.X): prediction from a
## rank-deficient fit may be misleading
```

```r
act <- kc.house.test.y
n <- dim(model.matrix(baseline.model)[, -1])[1]
p <- dim(model.matrix(baseline.model)[, -1])[2]

cbind(CalcTestMetrics(pred, act, n, p))
```

```
##                  [,1]
## adj.rsquared 0.7029118
## rsquared     0.7035598
## mse          18339297929
## mae          56354.97
```

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.

```r
# Load libraries
library(caret)
library(rpart)
library(nnet)
library(knitr)
library(dplyr)
library(stringr)
library(readr)

HouseSales <- read.csv('KC_House_Sales.csv')

# Step a: Data Splitting
set.seed(123)   #
split_ratio <- 0.7  # 70% training, 30% testing
index <- sample(1:nrow(HouseSales), size = round(split_ratio * nrow(HouseSales)))
train_data <- HouseSales[index, ]
test_data <- HouseSales[-index, ]

# Step b: Model Building

# Model 1: Logistic Regression

# Create a binary variable 'high_price' based on the chosen percentile
logisticdf <- HouseSales
logisticdf$price <- parse_number(logisticdf$price)
logisticdf$high_price <- ifelse(logisticdf$price >= 1000000, 1, 0)

# Create a new dataframe with 'high_price' column
logisticdf <- logisticdf %>%
  select(-price)

# Split data for logistic regrssion model
set.seed(123)   #
split_ratio <- 0.7  # 70% training, 30% testing
index <- sample(1:nrow(logisticdf), size = round(split_ratio * nrow(logisticdf)))
train_data_logistic <- logisticdf[index, ]
test_data_logistic <- logisticdf[-index, ]

#Build basic logistic regrssion model
logistic_model <- glm(high_price ~ ., data = train_data_logistic, family = binomial)
summary(logistic_model)

# Model 2: Regression Tree
tree_model <- rpart(price ~ ., data = train_data)
printcp(tree_model)   # Display complexity parameter plot
plot(tree_model)   # Visualize the decision tree

# Model 3: Neural Network
nn_model <- neuralnet(price ~ ., data = train_data, hidden = c(5, 2), linear.output = TRUE)

# Tune neural network hyperparameters - placehlder

# Step c: Model Evaluation
```

```r
# Model 1: Logistic Regression
logistic_predictions <- predict(logistic_model, newdata = test_data, type = 'response')
logistic_accuracy <- sum((logistic_predictions > 0.5) == test_data$high_price) / length(test_data$high_price)
cat("Logistic Regression Accuracy:", logistic_accuracy, "\n")

# Model 2: Regression Tree
tree_predictions <- predict(tree_model, newdata = test_data)
tree_rmse <- sqrt(mean((tree_predictions - test_data$price)^2))
cat("Regression Tree RMSE:", tree_rmse, "\n")

# Model 3: Neural Network
nn_predictions <- predict(nn_model, newdata = test_data)
# Calculate relevant performance metrics for the neural network model

# Step d: Model Comparison (Choose the best model based on evaluation metrics)

# Calculate evaluation metrics for all models
logistic_predictions <- predict(logistic_model, newdata = test_data, type = 'response')
logistic_accuracy <- sum((logistic_predictions > 0.5) == test_data$high_price) / length(test_data$high_price)
cat("Logistic Regression Accuracy:", logistic_accuracy, "\n")

tree_predictions <- predict(tree_model, newdata = test_data)
tree_rmse <- sqrt(mean((tree_predictions - test_data$price)^2))
cat("Regression Tree RMSE:", tree_rmse, "\n")

nn_predictions <- predict(nn_model, newdata = test_data)
cat("NN RMSE:", tree_rmse, "\n")

# Step e: Backtesting - placeholder

# Function to compare model performance
compare_model_performance <- function(models, model_names, data_train, data_test) {
  rmse_values <- c()
  rsquared_values <- c()
  mape_values <- c()

  for (i in 1:length(models)) {
    model <- models[[i]]
    predictions <- predict(model, newdata = data_test)
    rmse <- sqrt(mean((data_test$price - predictions)^2))
    rsquared <- cor(data_test$price, predictions)^2
    mape <- mean(abs((data_test$price - predictions) / data_test$price)) * 100

    rmse_values <- c(rmse_values, rmse)
    rsquared_values <- c(rsquared_values, rsquared)
    mape_values <- c(mape_values, mape)
  }

  # Create a data frame to hold the results
  model_results <- data.frame(Model = model_names, RMSE = rmse_values, R_squared = rsquared_values, MAPE = map

  # Create a table to compare model results
  kable(model_results, format = "markdown")
}
```

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?)

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?

Overview

For model monitoring there are four areas to track 1) model stability, 2) prediction performance, 3) incremental data quality and 4) data pipeline failures.

1. Prediction Stability

This subsection concerns itself with stability of predictions. Price predictions often are used by loan originators to size a loan or as an input to a consumer product. For example, assume the product is a home price recommendation for homeowners looking to see. If the homeowners aka the customers get highly variable sale price recommendations month to month, it'll result in subpar user experience, creating distrust between the customer and the firm.

To check for prediction stability the approach is to store last month's prediction values in a database table and the model itself in an S3 bucket as an RDS file. We are assuming new data arrives in monthly batches because that is the frequency of our data set.

Since we do not have infrastructure stood up we use the following code block to simulate pulling in previous month's model and corresponding predictions. We sample the data using a sliding window approach. The past 12 months of data are used to train the model and the subsequent month is the test set. This different from the world in previous parts because here we are simulating the arrival of new data.

```r
concat_date <- year*100 + month
kc.house.df$yyyymm <- concat_date

set.seed(1023)

lookback.window <- 12 #one year lookback window
#Any data before this date is in the training set
#Any data after is in the test set
cutoff_date <- 201501
train_start <- adjust_months(cutoff_date, -lookback.window)

prev_train_data <- kc.house.df[
  which(
    kc.house.df$yyyymm < cutoff_date &
    kc.house.df$yyyymm >= train_start
  ),
]

#horizon is the size of the test set
#Here one means the subsequent month, so 201501
#If it was 2 then the test set will include [201501-201502] so forth
horizon <- 1
test_start <- cutoff_date
test_end <- adjust_months(test_start, horizon)

prev_test_data <- kc.house.df[
  which(
    (kc.house.df$yyyymm < test_end) &
    (kc.house.df$yyyymm >= test_start)
  ),
]
prev_train_data <- prev_train_data %>% select(-contains("Month"))
prev_test_data <- prev_test_data %>% select(-contains("Month"))

previous.model <- lm(price ~ . -year, data = prev_train_data)
previous.pred <- predict(previous.model, prev_test_data)
```

Now we simulate getting a new batch of data and fitting the model on new data. We compare the new model predictions against the old model using

1. T-test on the means of the predictions

- To determine if prediction means are drifting from month to month

2. F-test for the variance of the predictions

- To determine if prediction variance is drifiting from month to month

3. T-test on the mean squared error.

- To determine if mean squared error is drifting from month to month

For assumption checking, we use Anderson-Darling test to check normality of predictions. Shapiro-Wilk test will not work due to the large sample size.

In practice, model stability tests tend to be overly sensitive so we will trigger Slack warnings when the p-values of the test are less than 0.01.

```r
library(nortest)

#Current Model
cutoff_date <- adjust_months(cutoff_date, 1)
train_start <- adjust_months(cutoff_date, -lookback.window)

curr_train_data <- kc.house.df[
  which(
    kc.house.df$yyyymm < cutoff_date &
    kc.house.df$yyyymm >= train_start
  ),
]

horizon <- 1
test_start <- cutoff_date
test_end <- adjust_months(test_start, horizon)

curr_test_data <- kc.house.df[
  which(
    (kc.house.df$yyyymm < test_end) &
    (kc.house.df$yyyymm >= test_start)

  ),
]
curr_train_data <- curr_train_data %>% select(-contains("Month"))
curr_test_data <- curr_test_data %>% select(-contains("Month"))

current.model <- lm(price ~ . -year, data = curr_train_data)
curr.pred <- predict(current.model, curr_test_data)

# HO: same means
# H1: means are not the same
t.test(previous.pred, curr.pred)

##
##  Welch Two Sample t-test
##
## data:  previous.pred and curr.pred
## t = -0.24506, df = 1980.9, p-value = 0.8064
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -27881.66  21687.63
## sample estimates:
```

```
## mean of x mean of y
##  497366.3  500463.3
# H0: same variance
# H1: variance are not the same
var.test(previous.pred, curr.pred)

##
##  F test to compare two variances
##
## data:  previous.pred and curr.pred
## F = 1.2417, num df = 977, denom df = 1249, p-value = 0.0003206
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  1.103482 1.398452
## sample estimates:
## ratio of variances
##           1.241703
#Check if mean squared error is equivalent
prev_sq_error <- (prev_test_data[, "price"] - previous.pred)^2
curr_sq_error <- (curr_test_data[, "price"] - curr.pred)^2
t.test(prev_sq_error, curr_sq_error)

##
##  Welch Two Sample t-test
##
## data:  prev_sq_error and curr_sq_error
## t = -0.57043, df = 2206.4, p-value = 0.5684
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -16082681642   8834641272
## sample estimates:
##    mean of x    mean of y
## 39020371656 42644391842
#Check predictions are normal
ad.test(current.model$residuals)

##
##  Anderson-Darling normality test
##
## data:  current.model$residuals
## A = 344.56, p-value < 2.2e-16
```

Looks like in our case, the tests will trigger and warn us that the variance of the predictions have deviated (increased), means have not, MSE has not and the AD test will trigger an assumption violation alert. Variance of predictions deviating is worth notifying because we want to ensure stakeholders and customers a consistent experience.

The most important of these is MSE as we do not want our model to fluctuate in terms of error. We certainly do not want MSE to deviate downwards to significant degree (alert can adjusted to trigger when difference is negative and significant). Normality is important for inference in the case that model coefficients become part of a product. However it is not as important currently since we are only concerned with predictive power.

2. Prediction Performance over Time

For this section we track model metrics over time to ensure the model continues to be on par. Metrics we choose are adjusted R^2 and RMSE. We apply a function to create a rolling window for the model to train on and then test with data outside the window. We collect the metrics and plot them over time.

If any metrics fall below the baseline production model, we will trigger a slack message.

```
library(ggplot2)

#Iterate over folds
```

```r
roll_model <- function(cutoff_date, form){

  lookback.window <- 6 #six month lookback
  train_start <- adjust_months(cutoff_date, -lookback.window)

  train_data <- kc.house.df[
    which(
      kc.house.df$yyyymm < cutoff_date &
      kc.house.df$yyyymm >= train_start
    ),
  ]
  train_data <- train_data %>% select(-contains("Month"))

  horizon <- 1
  test_start <- cutoff_date
  test_end <- adjust_months(test_start, horizon)

  test_data <- kc.house.df[
    which(
      (kc.house.df$yyyymm < test_end) &
      (kc.house.df$yyyymm >= test_start)

    ),
  ]
  test_data <- test_data %>% select(-contains("Month"))

  model <- lm(form, data = train_data)
  pred <- predict(model, test_data)

  act <- test_data[, "price"]
  n <- dim(model.matrix(model))[1]
  p <- dim(model.matrix(model))[2]

  metric <- CalcTestMetrics(pred, act, n, p)

  return (metric)
}

years <- c(201408, 201409, 201410, 201411, 201412, 201501, 201502, 201503)

ols_form <- as.formula("price ~ . -year")
metrics_table <- do.call(rbind, lapply(years, roll_model, form = ols_form))
metrics_table <- as.data.frame(metrics_table)
metrics_table$years <- years
metrics_table$rmse <- sqrt(as.numeric(metrics_table$mse))
metrics_table$model.name <- "Production Model"

#assume this it the baseline model
null_form <- as.formula("price ~ sqft_living")
null_metrics_table <- do.call(rbind, lapply(years, roll_model, form = null_form))
null_metrics_table <- as.data.frame(null_metrics_table)
null_metrics_table$years <- years
null_metrics_table$rmse <- sqrt(as.numeric(null_metrics_table$mse))
null_metrics_table$model.name <- "Baseline Model"


metrics_table <- dplyr::union(metrics_table, null_metrics_table)
```

```
# Convert date to Date class
metrics_table$date <- as.Date(paste0(metrics_table$years, "01"), format = "%Y%m%d")

# Create the ggplot
ggplot(metrics_table, aes(x = date, y = as.numeric(adj.rsquared), color = model.name)) +
  geom_point(size = 4) +
  labs(x = "Year-Month", y = "Adjusted R Squared", title = "Adjusted R Squared over Months") +
  scale_x_date(date_labels = "%Y-%m", date_breaks = "1 month") +
  theme_minimal()
```


Adjusted R Squared over Months
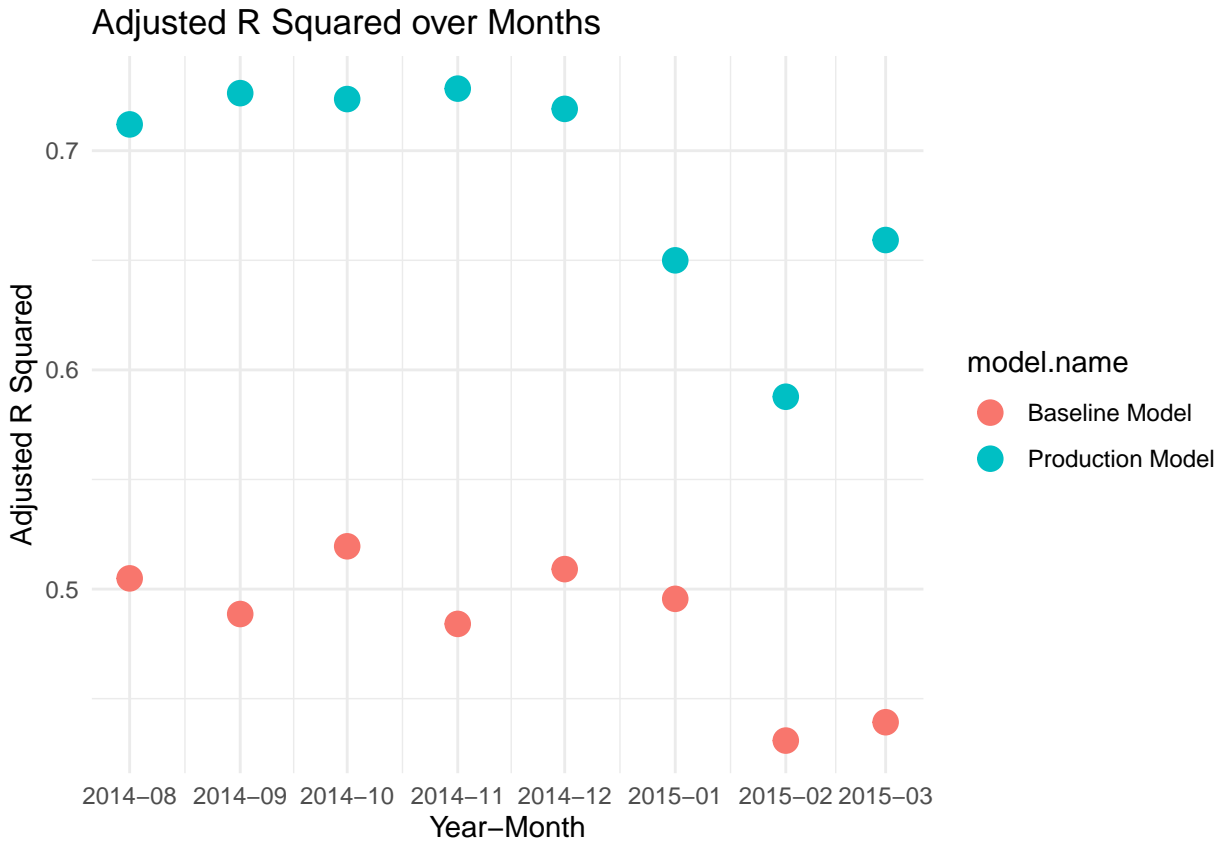
```
ggplot(metrics_table, aes(x = date, y = as.numeric(rmse), color = model.name)) +
  geom_point(size = 4) +
  labs(x = "Year-Month", y = "RMSE", title = "RMSE over Months") +
  scale_x_date(date_labels = "%Y-%m", date_breaks = "1 month") +
  theme_minimal()
```
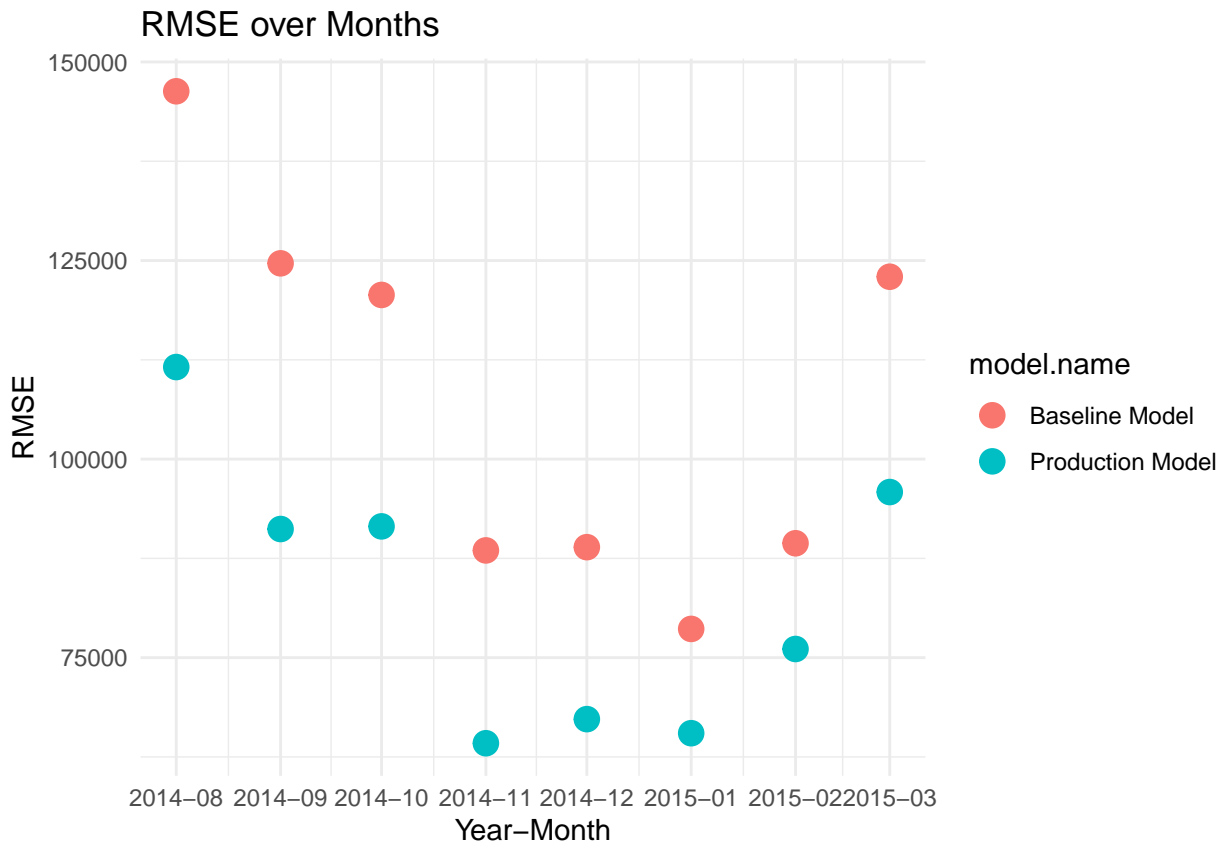
RMSE over Months

The `Adjusted R Squared over Months` plot will have no triggers. `RMSE over Months` will trigger on every month.

3. Incremental Data Quality Check

When new data comes in at monthly interval we will perform checks to prevent malformed data from entering the model. First we create graphs to add to a dashboard for visual inspection. These are the `Residuals vs Leverage` plot to detect outliers, leverage points or both and the `Cook's Distance` chart to detect influential points. The hope is to catch problematic points early, before they reach the model.
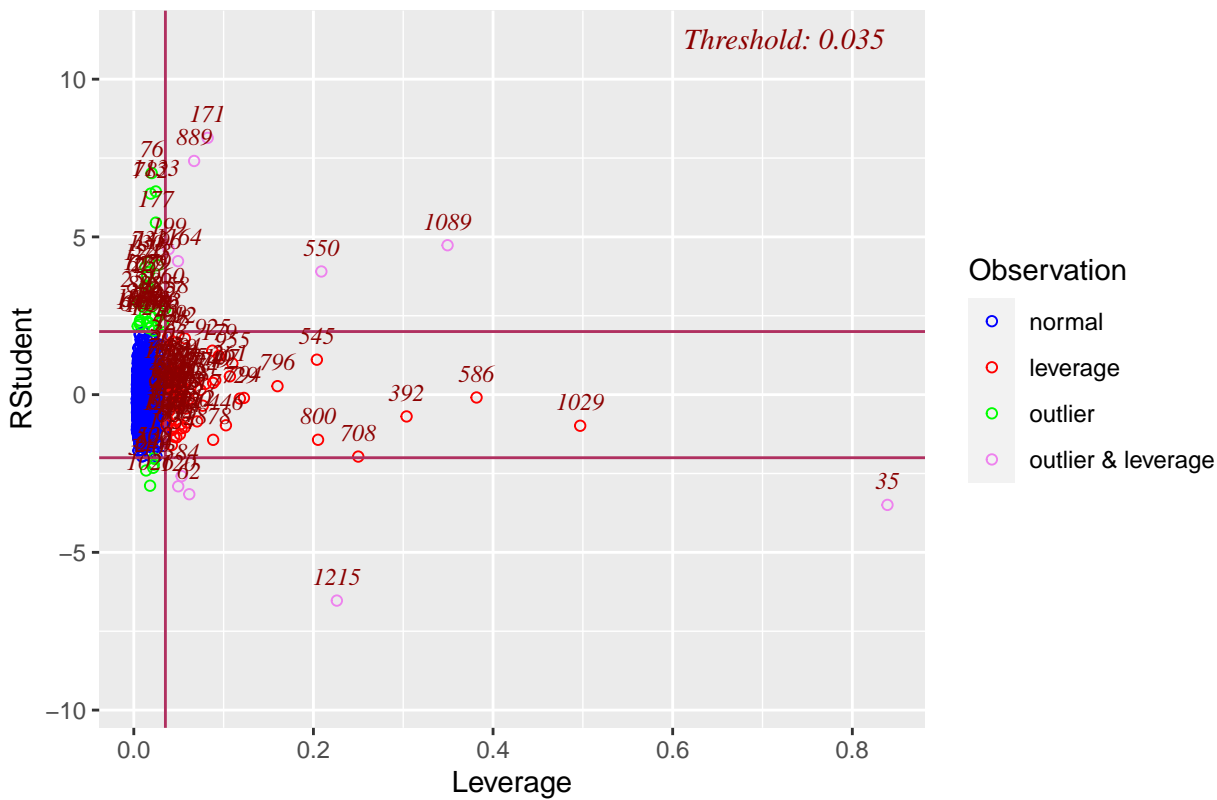
```
library(olsrr)
```

```
##
## Attaching package: 'olsrr'

## The following object is masked from 'package:datasets':
##
##     rivers
```

```
incremental_data <- curr_test_data
incremental.model <- lm(price ~ . -year, data = incremental_data)

#Residual vs Leverage
ols_plot_resid_lev(incremental.model)
```
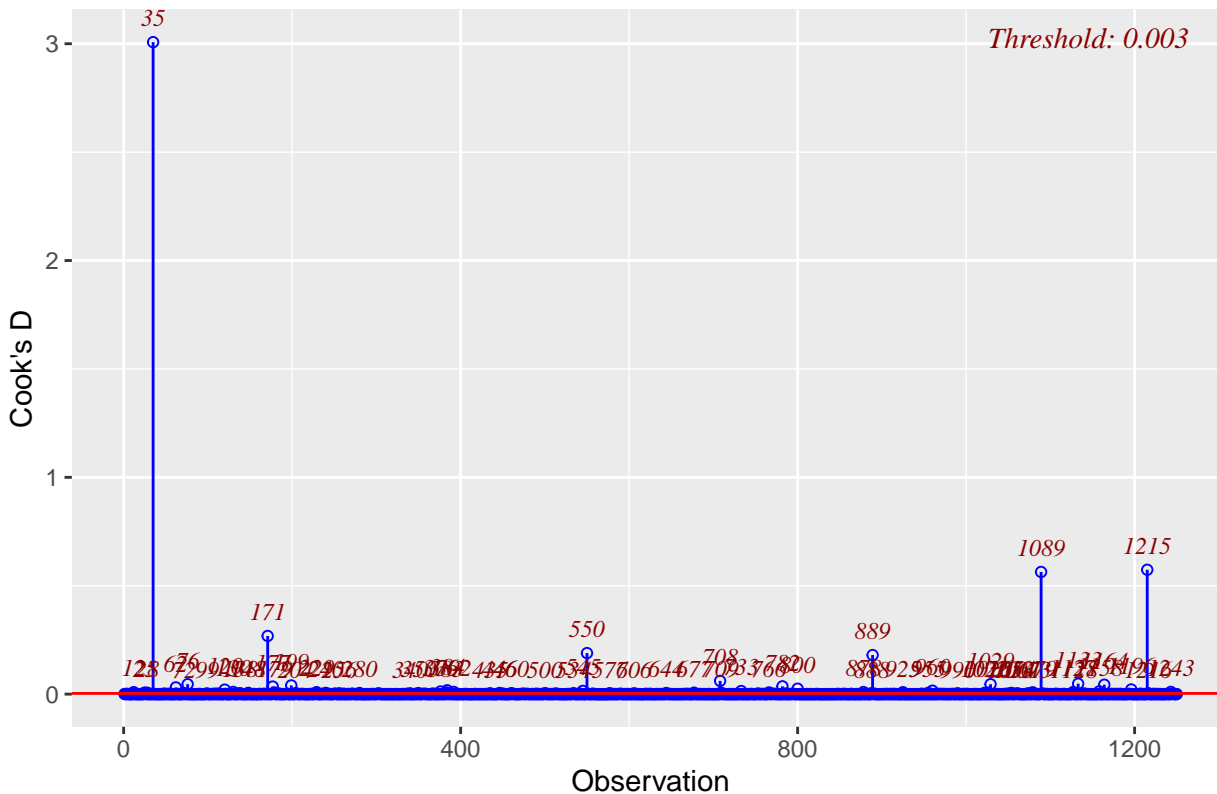
Outlier and Leverage Diagnostics for price


```
#Cook's Distance Visualized
ols_plot_cooksd_chart(incremental.model)
```

Cook's D Chart

To compare the old data with the new incremental data batch we perform a Kolmogorov–Smirnov test for continuous features and Chi-Square Goodness of Fit test for categorical data. The threshold we set will be a conservative 0.01 because these tests tend to be overly sensitive due to the variable nature of real data.

```r
old_data <- curr_train_data

continuous_features <- c(
  "price",
  "sqft_living",
  "sqft_lot",
  "sqft_above",
  "year_built",
  "yr_renovated",
  "sqft_living15",
  "sqft_lot15"
)
discrete_features <- c(
  "bedrooms",
  "bathrooms",
  "floors",
  "waterfront",
  "view",
  "condition",
  "grade",
  "zipcode_start"
)

#H0: Same Distribution
#H1: Not same distribution
ks_pvalues <- c()
for (feature in continuous_features){
  pval <- ks.test(
    old_data[, feature],
    incremental_data[, feature],
    simulate.p.value = TRUE
  )$p.value

  ks_pvalues <- c(ks_pvalues, pval)
}

#H0: Same Proportions
#H1: Not same proportions
chisq_pvalues <- c()
for (feature in discrete_features){
  # Extract the column data
  old_feature_data <- old_data[[feature]]
  n <- length(old_feature_data)
  expected_probabilities <- table(old_feature_data) / n

  incremental_feature_data <- incremental_data[[feature]]
  n <- length(incremental_feature_data)
  observed_probabilities <- table(old_feature_data) / n

  #want to compare new to old
  pval <- chisq.test(
    observed_probabilities,
    expected_probabilities,
    simulate.p.value = TRUE
  )$p.value

  chisq_pvalues <- c(chisq_pvalues, pval)
}
```

```
cbind(continuous_features, ks_pvalues)
```

```
##      continuous_features ks_pvalues
## [1,] "price"             "0.00499750124937526"
## [2,] "sqft_living"       "0.00599700149925032"
## [3,] "sqft_lot"          "0.19640179910045"
## [4,] "sqft_above"        "0.00299850074962513"
## [5,] "year_built"        "0.529735132433783"
## [6,] "yr_renovated"      "0.000999500249875007"
## [7,] "sqft_living15"     "0.217891054472764"
## [8,] "sqft_lot15"        "0.432783608195902"
```

```
cbind(discrete_features, chisq_pvalues)
```

```
##      discrete_features chisq_pvalues
## [1,] "bedrooms"        "0.0104947526236882"
## [2,] "bathrooms"       "0.000499750124937531"
## [3,] "floors"          "1"
## [4,] "waterfront"      "1"
## [5,] "view"            "1"
## [6,] "condition"       "1"
## [7,] "grade"           "0.0154922538730635"
## [8,] "zipcode_start"   "1"
```

Looks like `sqft_living`, `bedrooms`, `bathrooms` and deviated in our incremental data batch.

4. Pipeline Fail Safes

Note: No examples are shown here because we do not have data infrastructure

The plan to handle pipeline failures is to persist every production model and every test and training set of data. It is ok to store large amounts of data because storage is cheap in the modern day. Models RDS files will be stored in an S3 bucket. The data sets will be stored as compressed parquet files because production database tables usually only contain the most updated version (upserted records) and in this case we want to revert to data before any updates were made. When the pipeline fails, we can use old data and the old model to continue to generate predictions. This ensures downstream stakeholders are free from breaking changes and free from work disruption.
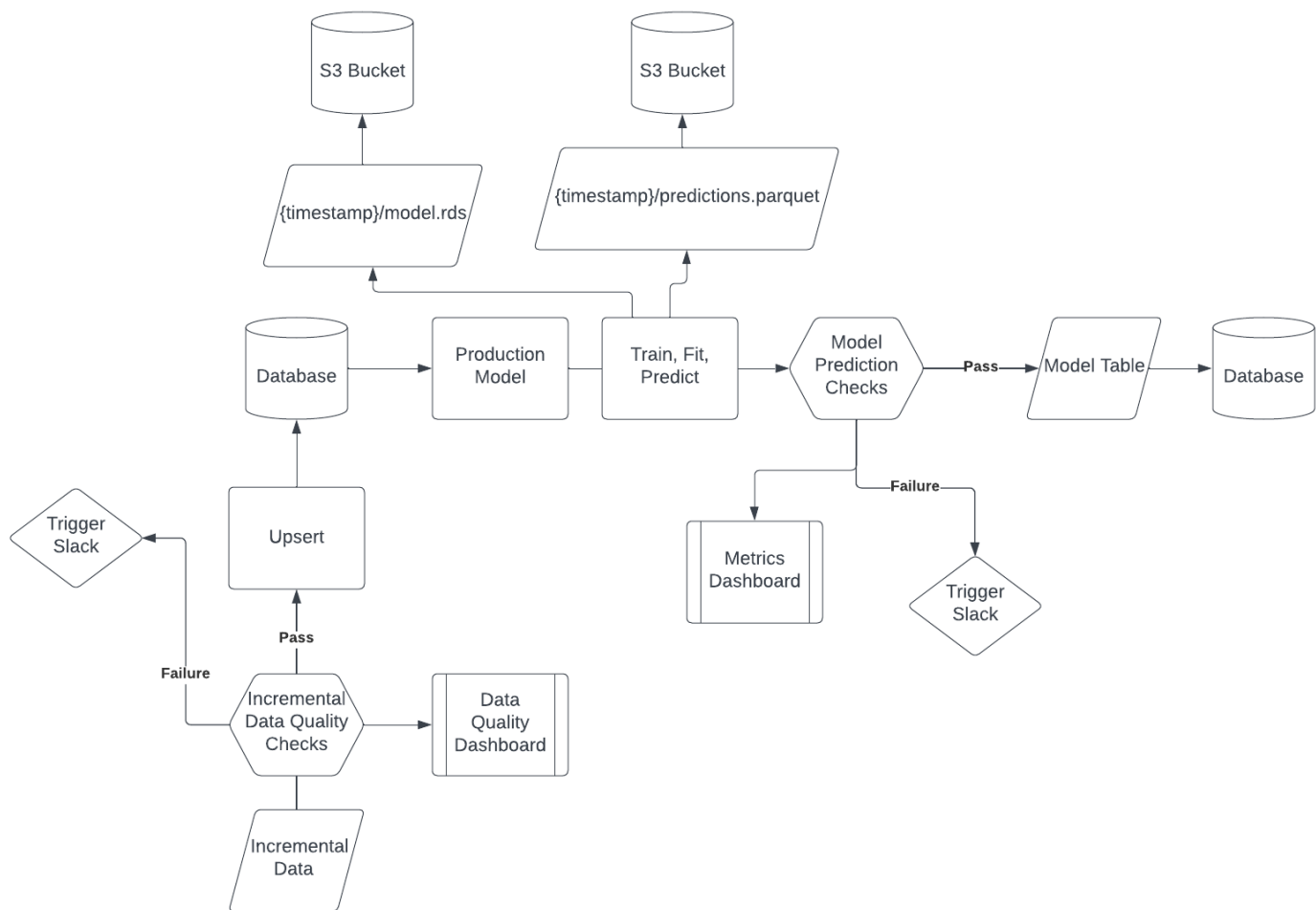
Figure 1: Model Monitoring Architecture

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.

1. McGill University

- http://www.med.mcgill.ca/epidemiology/joseph/courses/EPIB-621/centered_var.pdf

Appendix (3 points)

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