HARVARD EXTENSION SCHOOL

EXT CSCI E-106 Model Data Class Group Project Template

Sheri Cunningham Daniel Yinanc Hannah Halvorsen Sravan Katepalli
Swapnil Daingade Phu Thai

12 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

```
data <- read.csv("KC_House_Sales.csv")
original_data <- data
summary(data)
```

```
##        id                 date               price              bedrooms
##  Min.   :1.000e+06   Length:21613       Length:21613       Min.   : 0.000
##  1st Qu.:2.123e+09   Class :character   Class :character   1st Qu.: 3.000
##  Median :3.905e+09   Mode  :character   Mode  :character   Median : 3.000
##  Mean   :4.580e+09                                         Mean   : 3.371
##  3rd Qu.:7.309e+09                                         3rd Qu.: 4.000
##  Max.   :9.900e+09                                         Max.   :33.000
##    bathrooms      sqft_living       sqft_lot           floors
##  Min.   :0.000   Min.   :  290   Min.   :    520   Min.   :1.000
##  1st Qu.:1.750   1st Qu.: 1427   1st Qu.:   5040   1st Qu.:1.000
##  Median :2.250   Median : 1910   Median :   7618   Median :1.500
##  Mean   :2.115   Mean   : 2080   Mean   :  15107   Mean   :1.494
##  3rd Qu.:2.500   3rd Qu.: 2550   3rd Qu.:  10688   3rd Qu.:2.000
##  Max.   :8.000   Max.   :13540   Max.   :1651359   Max.   :3.500
##    waterfront           view          condition         grade
##  Min.   :0.000000   Min.   :0.0000   Min.   :1.000   Min.   : 1.000
##  1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.: 7.000
##  Median :0.000000   Median :0.0000   Median :3.000   Median : 7.000
##  Mean   :0.007542   Mean   :0.2343   Mean   :3.409   Mean   : 7.657
##  3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.: 8.000
##  Max.   :1.000000   Max.   :4.0000   Max.   :5.000   Max.   :13.000
##    sqft_above   sqft_basement       yr_built     yr_renovated
##  Min.   : 290   Min.   :   0.0   Min.   :1900   Min.   :   0.0
##  1st Qu.:1190   1st Qu.:   0.0   1st Qu.:1951   1st Qu.:   0.0
##  Median :1560   Median :   0.0   Median :1975   Median :   0.0
##  Mean   :1788   Mean   : 291.5   Mean   :1971   Mean   :  84.4
##  3rd Qu.:2210   3rd Qu.: 560.0   3rd Qu.:1997   3rd Qu.:   0.0
##  Max.   :9410   Max.   :4820.0   Max.   :2015   Max.   :2015.0
##     zipcode           lat             long        sqft_living15
##  Min.   :98001   Min.   :47.16   Min.   :-122.5   Min.   : 399
##  1st Qu.:98033   1st Qu.:47.47   1st Qu.:-122.3   1st Qu.:1490
##  Median :98065   Median :47.57   Median :-122.2   Median :1840
##  Mean   :98078   Mean   :47.56   Mean   :-122.2   Mean   :1987
##  3rd Qu.:98118   3rd Qu.:47.68   3rd Qu.:-122.1   3rd Qu.:2360
##  Max.   :98199   Max.   :47.78   Max.   :-121.3   Max.   :6210
##    sqft_lot15
##  Min.   :   651
##  1st Qu.:  5100
##  Median :  7620
##  Mean   : 12768
##  3rd Qu.: 10083
##  Max.   :871200
```

```
str(data)
```

```
## '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 ...
```

```
#install.packages('TeachingDemos')

#library(dplyr)
#check to see if few enough zipcodes to turn into categorical variable. otherwise
#drop zipcode in later chunk
#n_distinct(data$zipcode)
```

```
#convert date to usable format for calculations
#substr(date, start = 1, stop = 4)
#substr(date, 1, 4)
#str(data)
```

```
#built_age <- date - yr_built
#renovated_age <- date-yr_renovated


#turn price into numeric
data$price <- as.numeric(gsub('[$,]', '', data$price))

str(data)
```

```
## '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        : 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 ...
```

```
#drop the noted variables
#
library(fastDummies)
```

```
## Thank you for using fastDummies!

## To acknowledge our work, please cite the package:

## Kaplan, J. & Schlegel, B. (2023). fastDummies: Fast Creation of Dummy (Binary) Columns and Rows from Catego
```

```
data$house_age = as.integer(substr(data$date, 0, 4)) - data$yr_built
data$renovated = as.numeric(data$yr_renovated != 0)
data <- dummy_cols(data,
                   select_columns = "zipcode", remove_first_dummy =TRUE)

# yr_built and yr_renovated is transformed in this way to house_age and renovated, removed to prevent collinea
# sqft_basement removed due to high collinearity with sqft_above
# zipcode_98199 removed to high collinearity with other zipcodes
data = subset(data, select = -c(id, lat, long, date, yr_built, yr_renovated, sqft_basement, zipcode_98199, zip
```

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.

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

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.

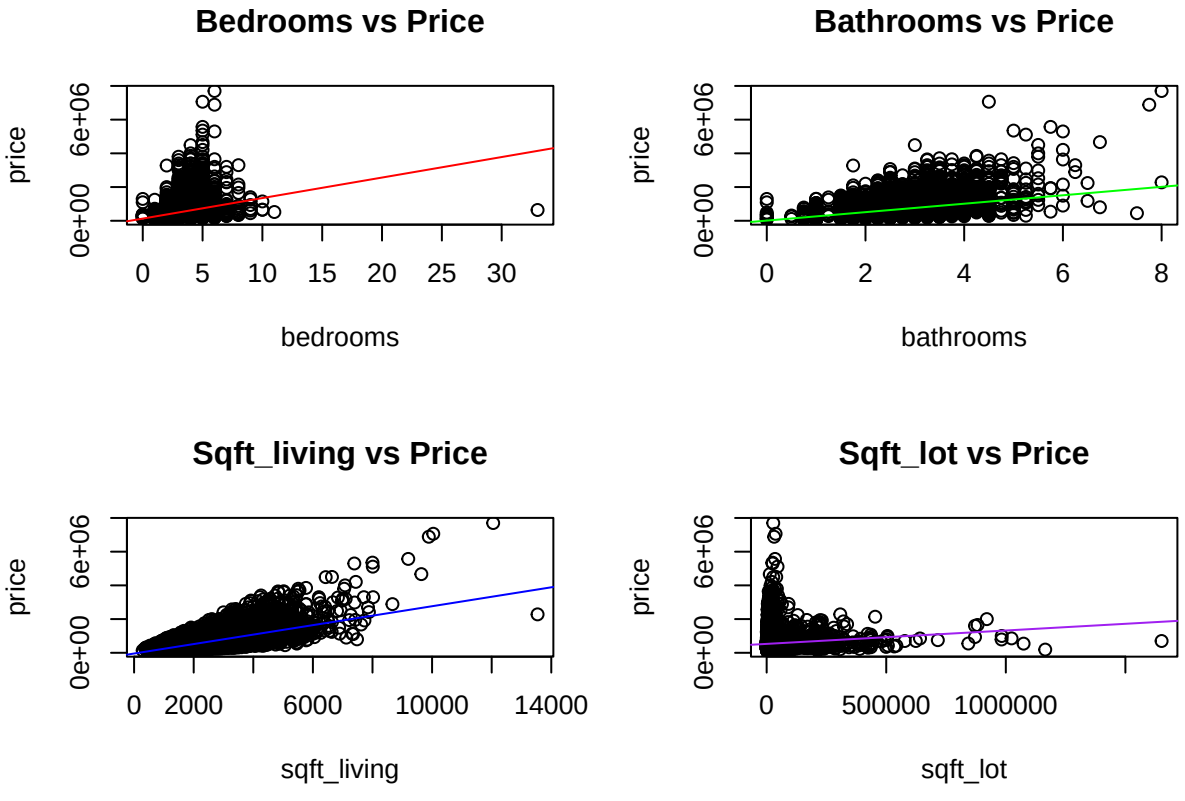
```
#exploring the data and no issues with NAs; no NAs found
colSums(is.na(data))
```

```
##         price      bedrooms     bathrooms   sqft_living      sqft_lot
##             0             0             0             0             0
##        floors    waterfront          view     condition         grade
##             0             0             0             0             0
##    sqft_above sqft_living15    sqft_lot15     house_age     renovated
##             0             0             0             0             0
## zipcode_98002 zipcode_98003 zipcode_98004 zipcode_98005 zipcode_98006
##             0             0             0             0             0
## zipcode_98007 zipcode_98008 zipcode_98010 zipcode_98011 zipcode_98014
##             0             0             0             0             0
## zipcode_98019 zipcode_98022 zipcode_98023 zipcode_98024 zipcode_98027
##             0             0             0             0             0
## zipcode_98028 zipcode_98029 zipcode_98030 zipcode_98031 zipcode_98032
##             0             0             0             0             0
## zipcode_98033 zipcode_98034 zipcode_98038 zipcode_98039 zipcode_98040
##             0             0             0             0             0
## zipcode_98042 zipcode_98045 zipcode_98052 zipcode_98053 zipcode_98055
##             0             0             0             0             0
## zipcode_98056 zipcode_98058 zipcode_98059 zipcode_98065 zipcode_98070
##             0             0             0             0             0
## zipcode_98072 zipcode_98074 zipcode_98075 zipcode_98077 zipcode_98092
##             0             0             0             0             0
## zipcode_98102 zipcode_98103 zipcode_98105 zipcode_98106 zipcode_98107
##             0             0             0             0             0
## zipcode_98108 zipcode_98109 zipcode_98112 zipcode_98115 zipcode_98116
##             0             0             0             0             0
## zipcode_98117 zipcode_98118 zipcode_98119 zipcode_98122 zipcode_98125
##             0             0             0             0             0
## zipcode_98126 zipcode_98133 zipcode_98136 zipcode_98144 zipcode_98146
##             0             0             0             0             0
## zipcode_98148 zipcode_98155 zipcode_98166 zipcode_98168 zipcode_98177
##             0             0             0             0             0
```

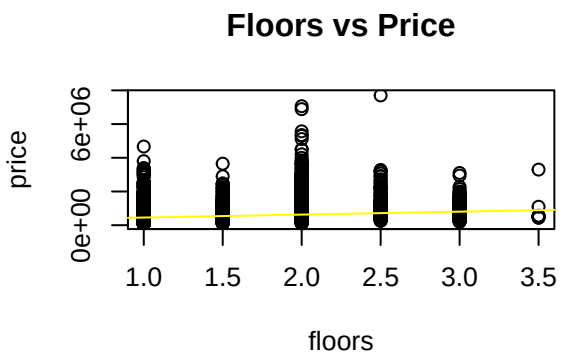
```
## zipcode_98178 zipcode_98188 zipcode_98198
##             0             0             0
```

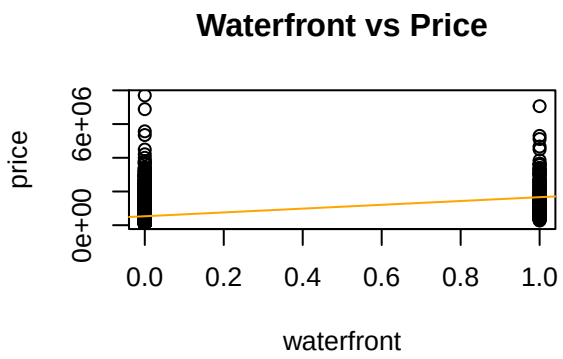
```
#create new variable to simplify yr_renovated to simply renovated or note since most properties are not renova
#is_renovated = data$is_renovated
#make is_renovated = 1 if the property has been renovated
#data$is_renovated =I(data$yr_renovated>0)*1
```

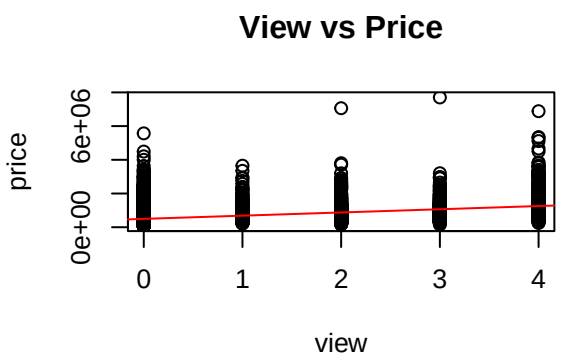
```
price<-data$price
bedrooms<-data$bedrooms
bathrooms<-data$bathrooms
sqft_living<-data$sqft_living
sqft_lot<-data$sqft_lot
floors<-data$floors
waterfront<-data$waterfront
view<-data$view
condition<-data$condition
grade<-data$grade
sqft_above<-data$sqft_above
sqft_basement<-data$sqft_basement
yr_renovated<-data$yr_renovated
sqft_living15<-data$sqft_living15
sqft_lot15<-data$sqft_lot15
house_age <- data$house_age
renovated <- data$renovated

par(mfrow=c(2,2))
plot(bedrooms, price,
main="Bedrooms vs Price"); abline(lm(price~bedrooms),col="red")
plot(bathrooms, price,
main="Bathrooms vs Price"); abline(lm(price~bathrooms),col="green")
plot(sqft_living, price,
main="Sqft_living vs Price"); abline(lm(price~sqft_living),col="blue")
plot(sqft_lot, price,
main="Sqft_lot vs Price"); abline(lm(price~sqft_lot),col="purple")
```

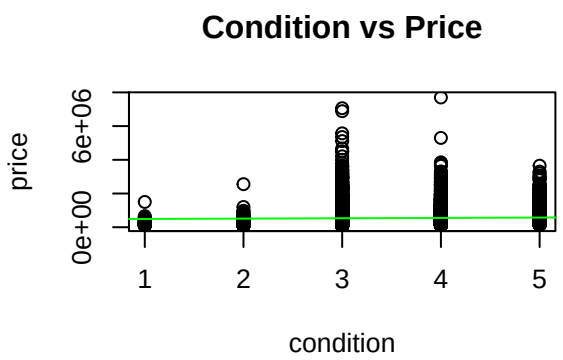


```
plot(floors, price,
main="Floors vs Price"); abline(lm(price~floors),col="yellow")
plot(waterfront, price,
main="Waterfront vs Price"); abline(lm(price~waterfront),col="orange")
plot(view, price,
main="View vs Price"); abline(lm(price~view),col="red")
plot(condition, price,
main="Condition vs Price"); abline(lm(price~condition),col="green")
```

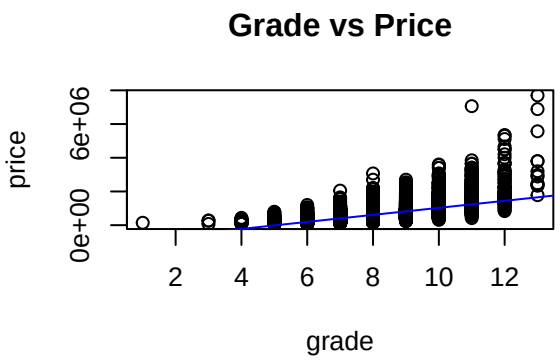


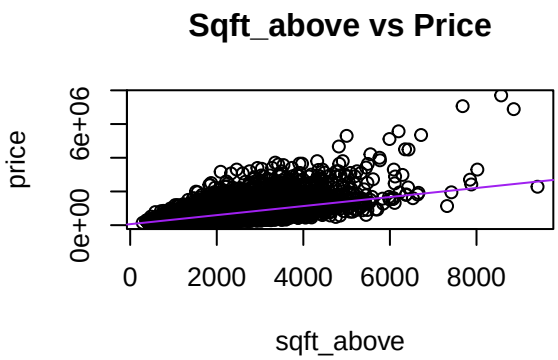


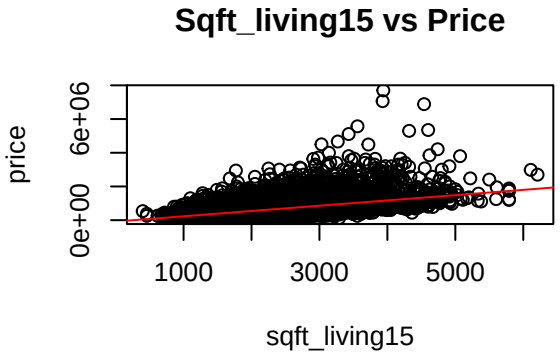


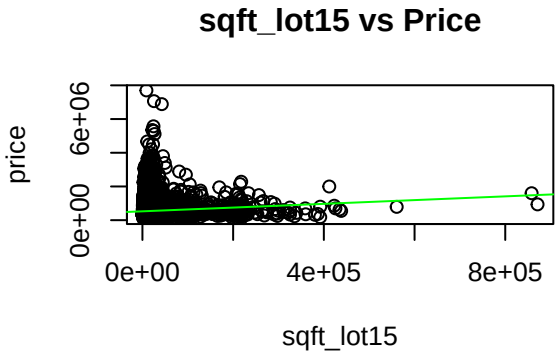



```
plot(grade, price,
main="Grade vs Price"); abline(lm(price~grade),col="blue")
plot(sqft_above, price,
main="Sqft_above vs Price"); abline(lm(price~sqft_above),col="purple")
#plot(sqft_basement, price,
#main="Sqft_basement vs Price"); abline(lm(price~sqft_basement),col="yellow")
#plot(is_renovated, price,
#main="is_renovated vs Price"); abline(lm(price~is_renovated),col="orange")
plot(sqft_living15, price,
main="Sqft_living15 vs Price"); abline(lm(price~sqft_living15),col="red")
plot(sqft_lot15, price,
main="sqft_lot15 vs Price"); abline(lm(price~sqft_lot15),col="green")
```

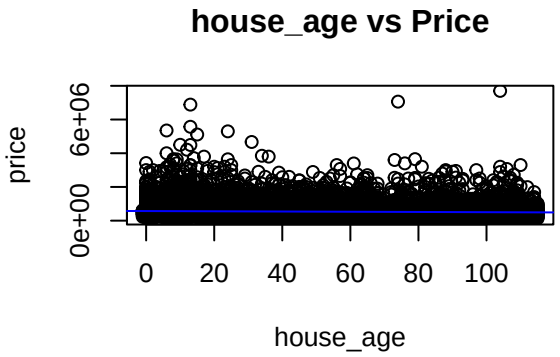


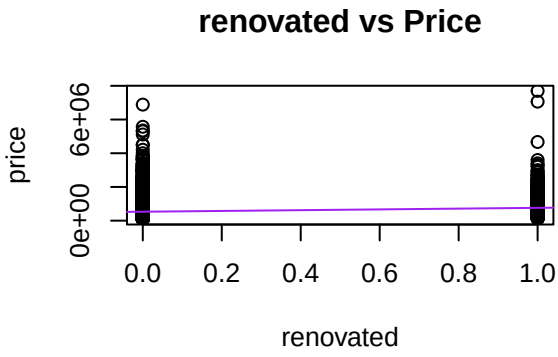







```
plot(house_age, price,
main="house_age vs Price"); abline(lm(price~house_age),col="blue")
plot(renovated, price,
main="renovated vs Price"); abline(lm(price~renovated),col="purple")
```





4. Build a regression model to predict price.

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

```
##
## Call:
## lm(formula = price ~ ., data = train.dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1124879   -73697     -802    64115  4394100
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -4.474e+05  1.895e+04 -23.604  < 2e-16 ***
## bedrooms      -2.696e+04  1.828e+03 -14.746  < 2e-16 ***
## bathrooms      2.495e+04  3.160e+03   7.894 3.12e-15 ***
## sqft_living    1.358e+02  4.300e+00  31.576  < 2e-16 ***
```

```
## sqft_lot       2.383e-01  4.935e-02   4.829 1.39e-06 ***
## floors        -3.188e+04  3.849e+03  -8.283  < 2e-16 ***
## waterfront     6.075e+05  1.798e+04  33.785  < 2e-16 ***
## view           5.884e+04  2.154e+03  27.319  < 2e-16 ***
## condition      2.494e+04  2.336e+03  10.679  < 2e-16 ***
## grade          6.373e+04  2.192e+03  29.082  < 2e-16 ***
## sqft_above     5.873e+01  4.391e+00  13.376  < 2e-16 ***
## sqft_living15  1.517e+01  3.539e+00   4.285 1.84e-05 ***
## sqft_lot15    -1.066e-01  7.322e-02  -1.456 0.145546
## house_age      1.026e+03  7.766e+01  13.211  < 2e-16 ***
## renovated      3.654e+04  7.110e+03   5.138 2.81e-07 ***
## zipcode_98002 -1.370e+05  1.553e+04  -8.821  < 2e-16 ***
## zipcode_98003 -1.836e+05  1.401e+04 -13.099  < 2e-16 ***
## zipcode_98004  5.978e+05  1.346e+04  44.405  < 2e-16 ***
## zipcode_98005  1.158e+05  1.665e+04   6.957 3.62e-12 ***
## zipcode_98006  9.138e+04  1.179e+04   7.753 9.55e-15 ***
## zipcode_98007  7.064e+04  1.814e+04   3.894 9.89e-05 ***
## zipcode_98008  7.199e+04  1.395e+04   5.161 2.48e-07 ***
## zipcode_98010 -9.164e+04  2.097e+04  -4.371 1.25e-05 ***
## zipcode_98011 -4.823e+04  1.610e+04  -2.996 0.002738 **
## zipcode_98014 -4.210e+04  1.994e+04  -2.111 0.034755 *
## zipcode_98019 -6.997e+04  1.590e+04  -4.400 1.09e-05 ***
## zipcode_98022 -1.772e+05  1.520e+04 -11.655  < 2e-16 ***
## zipcode_98023 -2.000e+05  1.158e+04 -17.271  < 2e-16 ***
## zipcode_98024 -2.621e+04  2.374e+04  -1.104 0.269611
## zipcode_98027  4.773e+03  1.223e+04   0.390 0.696384
## zipcode_98028 -5.361e+04  1.418e+04  -3.780 0.000158 ***
## zipcode_98029  4.227e+04  1.320e+04   3.202 0.001369 **
## zipcode_98030 -1.578e+05  1.435e+04 -10.997  < 2e-16 ***
## zipcode_98031 -1.529e+05  1.391e+04 -10.998  < 2e-16 ***
## zipcode_98032 -1.738e+05  1.924e+04  -9.036  < 2e-16 ***
## zipcode_98033  1.892e+05  1.215e+04  15.565  < 2e-16 ***
## zipcode_98034  3.512e+04  1.129e+04   3.111 0.001866 **
## zipcode_98038 -1.287e+05  1.105e+04 -11.650  < 2e-16 ***
## zipcode_98039  9.533e+05  2.921e+04  32.640  < 2e-16 ***
## zipcode_98040  3.321e+05  1.413e+04  23.506  < 2e-16 ***
## zipcode_98042 -1.579e+05  1.129e+04 -13.991  < 2e-16 ***
## zipcode_98045 -6.969e+04  1.519e+04  -4.589 4.48e-06 ***
## zipcode_98052  5.850e+04  1.098e+04   5.329 1.00e-07 ***
## zipcode_98053  2.632e+04  1.275e+04   2.064 0.039046 *
## zipcode_98055 -1.174e+05  1.397e+04  -8.406  < 2e-16 ***
## zipcode_98056 -6.538e+04  1.216e+04  -5.378 7.63e-08 ***
## zipcode_98058 -1.385e+05  1.190e+04 -11.631  < 2e-16 ***
## zipcode_98059 -8.326e+04  1.182e+04  -7.045 1.93e-12 ***
## zipcode_98065 -7.562e+04  1.344e+04  -5.628 1.86e-08 ***
## zipcode_98070 -1.836e+05  2.015e+04  -9.113  < 2e-16 ***
## zipcode_98072 -2.156e+04  1.425e+04  -1.513 0.130278
## zipcode_98074  4.035e+03  1.201e+04   0.336 0.736944
## zipcode_98075 -2.477e+03  1.296e+04  -0.191 0.848472
## zipcode_98077 -5.274e+04  1.645e+04  -3.206 0.001351 **
## zipcode_98092 -2.042e+05  1.272e+04 -16.055  < 2e-16 ***
## zipcode_98102  3.422e+05  1.946e+04  17.581  < 2e-16 ***
## zipcode_98103  1.612e+05  1.115e+04  14.450  < 2e-16 ***
## zipcode_98105  2.938e+05  1.516e+04  19.377  < 2e-16 ***
## zipcode_98106 -2.469e+04  1.282e+04  -1.926 0.054159 .
## zipcode_98107  1.709e+05  1.422e+04  12.012  < 2e-16 ***
## zipcode_98108 -3.704e+04  1.571e+04  -2.358 0.018380 *
## zipcode_98109  3.186e+05  1.999e+04  15.938  < 2e-16 ***
## zipcode_98112  4.561e+05  1.447e+04  31.509  < 2e-16 ***
## zipcode_98115  1.512e+05  1.115e+04  13.561  < 2e-16 ***
```

```
## zipcode_98116  1.092e+05  1.288e+04   8.477  < 2e-16 ***
## zipcode_98117  1.430e+05  1.129e+04  12.672  < 2e-16 ***
## zipcode_98118  2.202e+03  1.150e+04   0.191 0.848143
## zipcode_98119  3.005e+05  1.582e+04  18.991  < 2e-16 ***
## zipcode_98122  1.401e+05  1.372e+04  10.211  < 2e-16 ***
## zipcode_98125  3.017e+04  1.223e+04   2.467 0.013649 *
## zipcode_98126  2.448e+04  1.297e+04   1.888 0.059068 .
## zipcode_98133  4.068e+03  1.151e+04   0.354 0.723675
## zipcode_98136  7.117e+04  1.434e+04   4.964 6.99e-07 ***
## zipcode_98144  1.017e+05  1.283e+04   7.928 2.39e-15 ***
## zipcode_98146 -6.053e+04  1.398e+04  -4.331 1.50e-05 ***
## zipcode_98148 -9.701e+04  2.479e+04  -3.914 9.12e-05 ***
## zipcode_98155 -2.082e+04  1.186e+04  -1.755 0.079321 .
## zipcode_98166 -1.164e+05  1.405e+04  -8.288  < 2e-16 ***
## zipcode_98168 -9.050e+04  1.429e+04  -6.334 2.46e-10 ***
## zipcode_98177  4.565e+04  1.447e+04   3.156 0.001604 **
## zipcode_98178 -1.195e+05  1.435e+04  -8.331  < 2e-16 ***
## zipcode_98188 -1.323e+05  1.878e+04  -7.044 1.96e-12 ***
## zipcode_98198 -1.643e+05  1.424e+04 -11.538  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 163600 on 15046 degrees of freedom
## Multiple R-squared:  0.7966, Adjusted R-squared:  0.7955
## F-statistic: 718.4 on 82 and 15046 DF,  p-value: < 2.2e-16
```

```
#NA for sqft_basement may be linearly related to another variable and therefore
#shows NA. Thus dropping sqft_basement
#data = subset(data, select = -c(sqft_basement))
#model <-lm(price ~ ., data = data)
#summary(model)
```

```
library(car)
```

```
## Loading required package: carData
```

```
#Check for outliers
outlierTest(model)
```

```
##        rstudent unadjusted p-value Bonferroni p
## 7253  27.964803        8.2474e-168  1.2477e-163
## 1449  16.352307         1.3671e-59   2.0683e-55
## 1316  14.860709         1.3327e-49   2.0162e-45
## 12371 14.512385         2.1139e-47   3.1981e-43
## 8639  14.030386         1.9361e-44   2.9291e-40
## 2627  12.455871         1.9429e-35   2.9394e-31
## 10447 10.691927         1.3834e-26   2.0930e-22
## 18483 10.574575         4.8251e-26   7.2999e-22
## 5450   9.388080         6.9690e-21   1.0543e-16
## 20155  9.335792         1.1400e-20   1.7247e-16
```

5. Create scatter plots and a correlation matrix for the train data set. Interpret the possible relationship between the response.

Conclusion

Price is highly correlated with bedrooms, sqft_living, bathrooms, grade, sqft_above and sqft_living15.

We can see multicollinearity problems with bedrooms to bathrooms and sqft_living

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
library("ggplot2")                      # Load ggplot2 package
library("GGally")                       # Load GGally package
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
# Quick Correlation Matrix Visualization shows that sqft_above, sqft_living, grade and price are highly correl
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:car':
##
##     recode
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
correl <- round(cor(data),2)
cor_df <- as.data.frame(as.table(correl))
cor_df %>%  arrange(desc(Freq)) %>% filter(abs(Freq)>0.6 & abs(Freq)<1)
```

```
##             Var1          Var2 Freq
## 1     sqft_above   sqft_living 0.88
## 2    sqft_living    sqft_above 0.88
## 3          grade   sqft_living 0.76
## 4  sqft_living15   sqft_living 0.76
## 5    sqft_living         grade 0.76
## 6     sqft_above         grade 0.76
## 7          grade    sqft_above 0.76
## 8    sqft_living sqft_living15 0.76
## 9    sqft_living     bathrooms 0.75
## 10     bathrooms   sqft_living 0.75
## 11 sqft_living15    sqft_above 0.73
## 12    sqft_above sqft_living15 0.73
## 13    sqft_lot15      sqft_lot 0.72
## 14      sqft_lot    sqft_lot15 0.72
## 15 sqft_living15         grade 0.71
## 16         grade sqft_living15 0.71
## 17   sqft_living         price 0.70
## 18         price   sqft_living 0.70
## 19    sqft_above     bathrooms 0.69
## 20     bathrooms    sqft_above 0.69
## 21         grade         price 0.67
## 22         price         grade 0.67
## 23         grade     bathrooms 0.66
## 24     bathrooms         grade 0.66
## 25    sqft_above         price 0.61
## 26         price    sqft_above 0.61
```

```r
# None of the independent variables have VIF > 5 indicating multi-collinearity
# As we removed them at data manipulation section
vif(model)
```

```
##      bedrooms     bathrooms   sqft_living      sqft_lot        floors
##      1.669263      3.347101      8.705147      2.097052      2.438024
```

```
##    waterfront          view     condition         grade    sqft_above
##      1.236127      1.499140      1.318613      3.734085      7.390360
## sqft_living15    sqft_lot15     house_age     renovated zipcode_98002
##      3.356274      2.229039      2.951630      1.166794      1.302581
## zipcode_98003 zipcode_98004 zipcode_98005 zipcode_98006 zipcode_98007
##      1.369955      1.468317      1.263901      1.715494      1.197191
## zipcode_98008 zipcode_98010 zipcode_98011 zipcode_98014 zipcode_98019
##      1.385393      1.161093      1.267157      1.226490      1.301805
## zipcode_98022 zipcode_98023 zipcode_98024 zipcode_98027 zipcode_98028
##      1.358955      1.666150      1.154278      1.590312      1.359298
## zipcode_98029 zipcode_98030 zipcode_98031 zipcode_98032 zipcode_98033
##      1.469349      1.360757      1.391149      1.169172      1.569976
## zipcode_98034 zipcode_98038 zipcode_98039 zipcode_98040 zipcode_98042
##      1.718410      1.892322      1.081570      1.428859      1.781554
## zipcode_98045 zipcode_98052 zipcode_98053 zipcode_98055 zipcode_98056
##      1.330500      1.834374      1.611625      1.374646      1.587036
## zipcode_98058 zipcode_98059 zipcode_98065 zipcode_98070 zipcode_98072
##      1.618259      1.694973      1.495810      1.267651      1.379280
## zipcode_98074 zipcode_98075 zipcode_98077 zipcode_98092 zipcode_98102
##      1.653200      1.586637      1.303795      1.533376      1.196702
## zipcode_98103 zipcode_98105 zipcode_98106 zipcode_98107 zipcode_98108
##      1.881027      1.335244      1.492847      1.404028      1.279262
## zipcode_98109 zipcode_98112 zipcode_98115 zipcode_98116 zipcode_98117
##      1.173385      1.422998      1.782757      1.500022      1.768324
## zipcode_98118 zipcode_98119 zipcode_98122 zipcode_98125 zipcode_98126
##      1.689650      1.316304      1.457821      1.547273      1.484105
## zipcode_98133 zipcode_98136 zipcode_98144 zipcode_98146 zipcode_98148
##      1.673046      1.358826      1.507715      1.390878      1.098601
## zipcode_98155 zipcode_98166 zipcode_98168 zipcode_98177 zipcode_98178
##      1.602186      1.376895      1.371780      1.345095      1.361121
## zipcode_98188 zipcode_98198
##      1.179633      1.363340
```

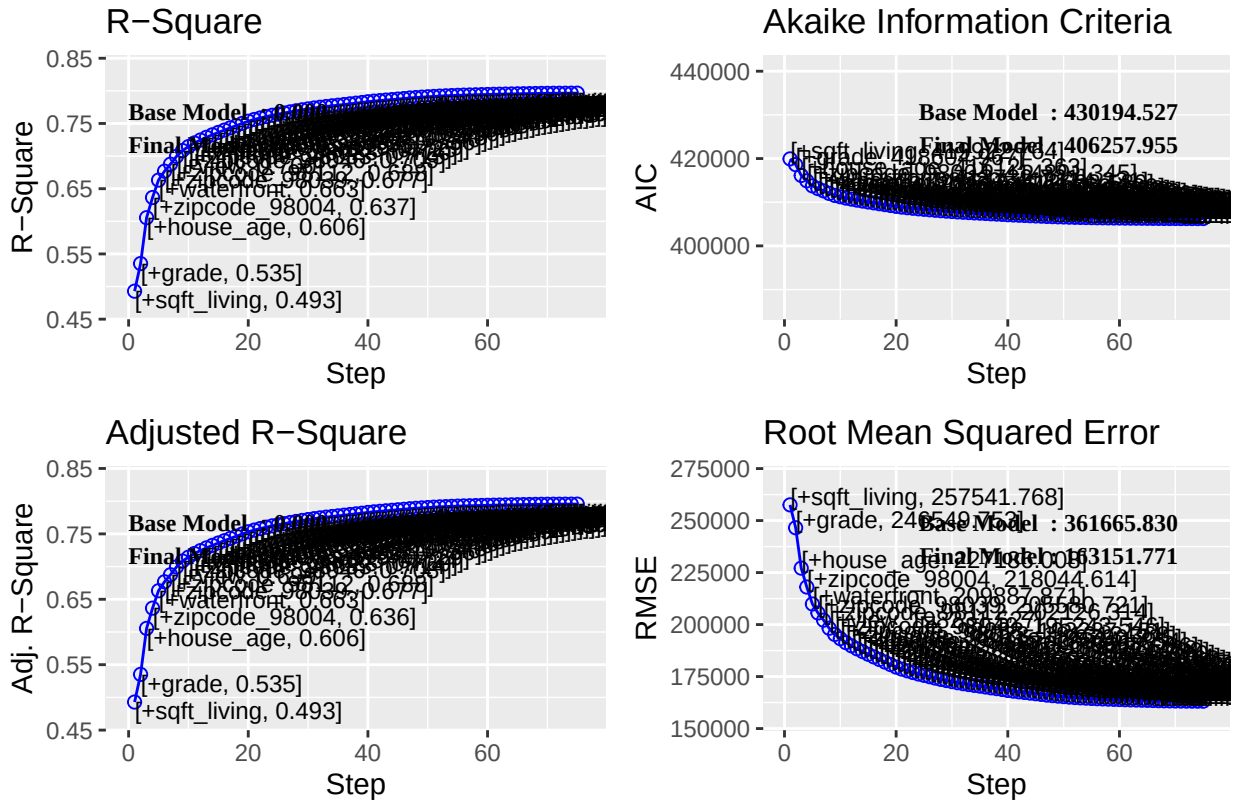
6. Build the best multiple linear models by using the stepwise selection method. Compare the performance of the best two linear models.

```
library(olsrr)
```

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

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

```
library(car)

# Stepwise regression yielded bathrooms + sqft_living +  view + grade + sqft_above as the predictor set to kee
# We name that as model_2 which is the best two linear models we have.
# takes about an hour to run
k1 <- ols_step_both_p(model, pent=0.05, prem=0.05)
plot(k1)
```



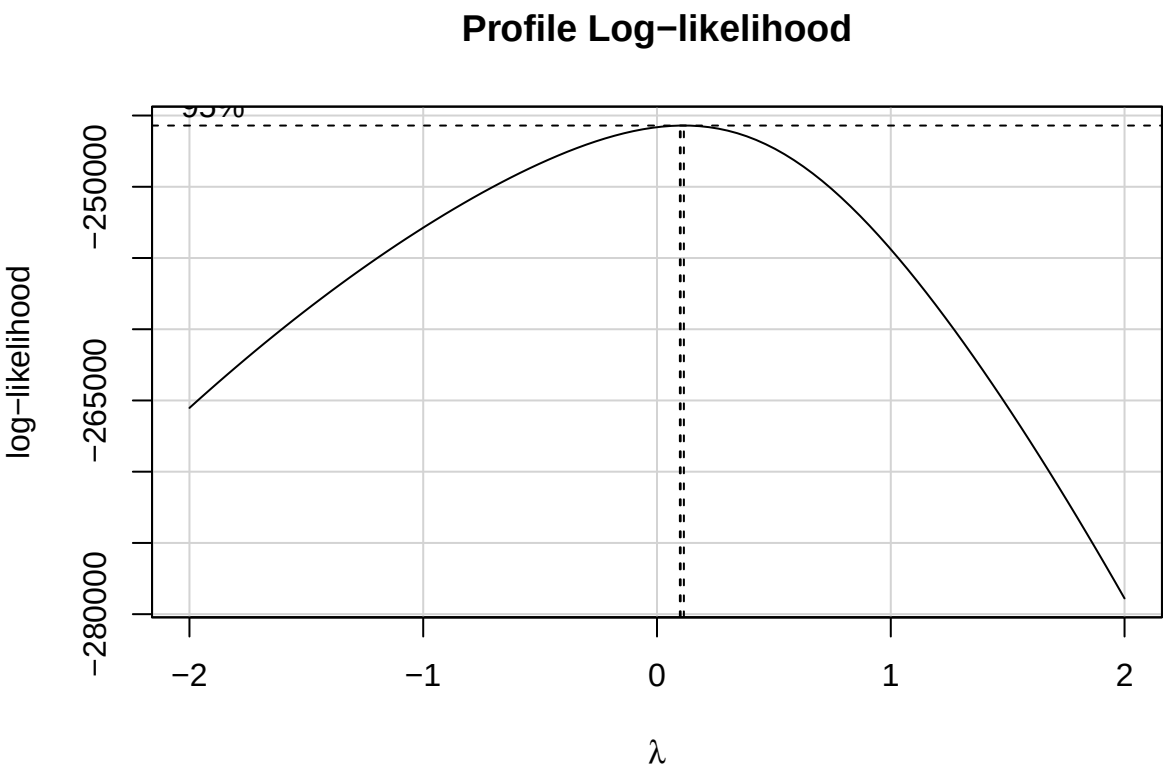

```
model_2 <- lm(price ~ . - zipcode_98024 - zipcode_98027 - zipcode_98074 - zipcode_98075 - zipcode_98118 - zipc
summary(model_2)
```

```
##
## Call:
## lm(formula = price ~ . - zipcode_98024 - zipcode_98027 - zipcode_98074 -
##     zipcode_98075 - zipcode_98118 - zipcode_98133 - sqft_lot15,
##     data = train.dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1125344   -73788     -770    64182  4397283
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -4.474e+05  1.782e+04 -25.099  < 2e-16 ***
## bedrooms      -2.685e+04  1.827e+03 -14.699  < 2e-16 ***
## bathrooms      2.503e+04  3.156e+03   7.931 2.33e-15 ***
## sqft_living    1.358e+02  4.281e+00  31.712  < 2e-16 ***
## sqft_lot       1.845e-01  3.730e-02   4.946 7.64e-07 ***
## floors        -3.156e+04  3.829e+03  -8.243  < 2e-16 ***
## waterfront     6.081e+05  1.797e+04  33.849  < 2e-16 ***
## view           5.881e+04  2.145e+03  27.418  < 2e-16 ***
## condition      2.491e+04  2.327e+03  10.701  < 2e-16 ***
## grade          6.391e+04  2.187e+03  29.228  < 2e-16 ***
## sqft_above     5.829e+01  4.355e+00  13.384  < 2e-16 ***
## sqft_living15  1.472e+01  3.461e+00   4.254 2.12e-05 ***
## house_age      1.029e+03  7.658e+01  13.437  < 2e-16 ***
## renovated      3.618e+04  7.100e+03   5.096 3.51e-07 ***
## zipcode_98002 -1.382e+05  1.415e+04  -9.766  < 2e-16 ***
## zipcode_98003 -1.848e+05  1.246e+04 -14.832  < 2e-16 ***
## zipcode_98004  5.968e+05  1.175e+04  50.783  < 2e-16 ***
## zipcode_98005  1.144e+05  1.530e+04   7.478 7.95e-14 ***
```

```
## zipcode_98006  9.045e+04  9.795e+03   9.234  < 2e-16 ***
## zipcode_98007  6.949e+04  1.694e+04   4.103 4.10e-05 ***
## zipcode_98008  7.086e+04  1.239e+04   5.720 1.09e-08 ***
## zipcode_98010 -9.443e+04  1.986e+04  -4.754 2.01e-06 ***
## zipcode_98011 -4.917e+04  1.471e+04  -3.343 0.000832 ***
## zipcode_98014 -4.529e+04  1.870e+04  -2.422 0.015446 *
## zipcode_98019 -7.229e+04  1.444e+04  -5.005 5.65e-07 ***
## zipcode_98022 -1.794e+05  1.373e+04 -13.065  < 2e-16 ***
## zipcode_98023 -2.012e+05  9.616e+03 -20.924  < 2e-16 ***
## zipcode_98028 -5.473e+04  1.262e+04  -4.336 1.46e-05 ***
## zipcode_98029  4.121e+04  1.147e+04   3.592 0.000329 ***
## zipcode_98030 -1.588e+05  1.280e+04 -12.404  < 2e-16 ***
## zipcode_98031 -1.539e+05  1.231e+04 -12.503  < 2e-16 ***
## zipcode_98032 -1.750e+05  1.814e+04  -9.647  < 2e-16 ***
## zipcode_98033  1.881e+05  1.028e+04  18.296  < 2e-16 ***
## zipcode_98034  3.399e+04  9.273e+03   3.666 0.000248 ***
## zipcode_98038 -1.299e+05  8.880e+03 -14.633  < 2e-16 ***
## zipcode_98039  9.524e+05  2.844e+04  33.491  < 2e-16 ***
## zipcode_98040  3.312e+05  1.252e+04  26.456  < 2e-16 ***
## zipcode_98042 -1.590e+05  9.217e+03 -17.248  < 2e-16 ***
## zipcode_98045 -7.098e+04  1.372e+04  -5.175 2.30e-07 ***
## zipcode_98052  5.746e+04  8.803e+03   6.527 6.91e-11 ***
## zipcode_98053  2.471e+04  1.083e+04   2.283 0.022465 *
## zipcode_98055 -1.185e+05  1.241e+04  -9.549  < 2e-16 ***
## zipcode_98056 -6.636e+04  1.030e+04  -6.440 1.23e-10 ***
## zipcode_98058 -1.395e+05  9.981e+03 -13.974  < 2e-16 ***
## zipcode_98059 -8.431e+04  9.804e+03  -8.600  < 2e-16 ***
## zipcode_98065 -7.626e+04  1.169e+04  -6.522 7.14e-11 ***
## zipcode_98070 -1.884e+05  1.885e+04  -9.998  < 2e-16 ***
## zipcode_98072 -2.286e+04  1.263e+04  -1.810 0.070266 .
## zipcode_98077 -5.487e+04  1.496e+04  -3.666 0.000247 ***
## zipcode_98092 -2.062e+05  1.088e+04 -18.957  < 2e-16 ***
## zipcode_98102  3.410e+05  1.840e+04  18.534  < 2e-16 ***
## zipcode_98103  1.598e+05  9.189e+03  17.394  < 2e-16 ***
## zipcode_98105  2.927e+05  1.376e+04  21.268  < 2e-16 ***
## zipcode_98106 -2.600e+04  1.115e+04  -2.331 0.019770 *
## zipcode_98107  1.695e+05  1.275e+04  13.293  < 2e-16 ***
## zipcode_98108 -3.829e+04  1.437e+04  -2.664 0.007728 **
## zipcode_98109  3.174e+05  1.896e+04  16.736  < 2e-16 ***
## zipcode_98112  4.550e+05  1.298e+04  35.063  < 2e-16 ***
## zipcode_98115  1.500e+05  9.178e+03  16.340  < 2e-16 ***
## zipcode_98116  1.079e+05  1.125e+04   9.597  < 2e-16 ***
## zipcode_98117  1.417e+05  9.358e+03  15.147  < 2e-16 ***
## zipcode_98119  2.992e+05  1.452e+04  20.614  < 2e-16 ***
## zipcode_98122  1.389e+05  1.218e+04  11.400  < 2e-16 ***
## zipcode_98125  2.890e+04  1.045e+04   2.765 0.005698 **
## zipcode_98126  2.323e+04  1.134e+04   2.048 0.040551 *
## zipcode_98136  6.991e+04  1.288e+04   5.427 5.81e-08 ***
## zipcode_98144  1.006e+05  1.118e+04   9.001  < 2e-16 ***
## zipcode_98146 -6.175e+04  1.245e+04  -4.958 7.18e-07 ***
## zipcode_98148 -9.826e+04  2.394e+04  -4.104 4.09e-05 ***
## zipcode_98155 -2.211e+04  9.998e+03  -2.212 0.027004 *
## zipcode_98166 -1.178e+05  1.252e+04  -9.406  < 2e-16 ***
## zipcode_98168 -9.177e+04  1.279e+04  -7.177 7.48e-13 ***
## zipcode_98177  4.443e+04  1.299e+04   3.421 0.000626 ***
## zipcode_98178 -1.208e+05  1.287e+04  -9.386  < 2e-16 ***
## zipcode_98188 -1.336e+05  1.767e+04  -7.562 4.18e-14 ***
## zipcode_98198 -1.656e+05  1.275e+04 -12.988  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 163600 on 15053 degrees of freedom
## Multiple R-squared:  0.7965, Adjusted R-squared:  0.7955
## F-statistic: 785.6 on 75 and 15053 DF,  p-value: < 2.2e-16
```

```
# Lambda of zero means a log transform of Y is in order
boxCox(model_2)
```




```
# Result improved R^2 from 0.7966 to 0.855
model_log <-lm(log(price) ~ . - zipcode_98024 - zipcode_98027 - zipcode_98074 - zipcode_98075 - zipcode_98118
summary(model_log)
```

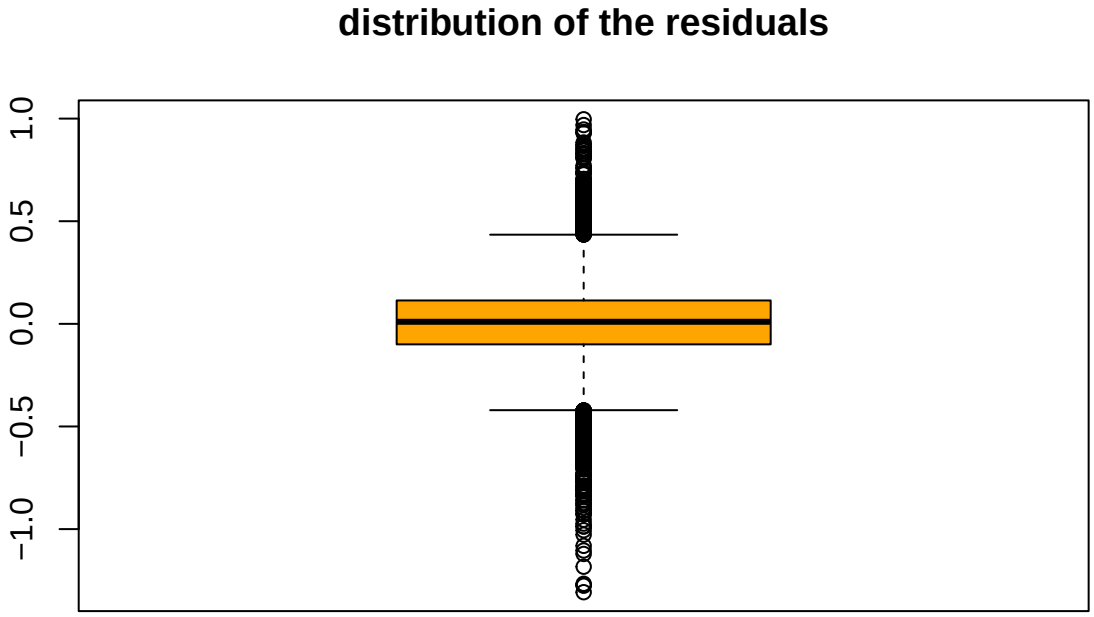
```
##
## Call:
## lm(formula = log(price) ~ . - zipcode_98024 - zipcode_98027 -
##     zipcode_98074 - zipcode_98075 - zipcode_98118 - zipcode_98133 -
##     sqft_lot15, data = train.dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.30736 -0.09994  0.01011  0.11384  0.99651
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.137e+01  2.182e-02 521.045  < 2e-16 ***
## bedrooms      -1.645e-03  2.236e-03  -0.736 0.462009
## bathrooms      4.309e-02  3.863e-03  11.153  < 2e-16 ***
## sqft_living    1.374e-04  5.240e-06  26.221  < 2e-16 ***
## sqft_lot       6.694e-07  4.565e-08  14.664  < 2e-16 ***
## floors        -1.570e-02  4.686e-03  -3.350 0.000811 ***
## waterfront     4.316e-01  2.199e-02  19.626  < 2e-16 ***
## view           5.852e-02  2.626e-03  22.290  < 2e-16 ***
## condition      5.765e-02  2.849e-03  20.237  < 2e-16 ***
## grade          1.036e-01  2.676e-03  38.725  < 2e-16 ***
## sqft_above     5.057e-05  5.331e-06   9.486  < 2e-16 ***
## sqft_living15  9.398e-05  4.236e-06  22.187  < 2e-16 ***
## house_age      8.784e-04  9.374e-05   9.370  < 2e-16 ***
```

```
## renovated      7.474e-02  8.690e-03   8.600  < 2e-16 ***
## zipcode_98002 -4.812e-01  1.732e-02 -27.781  < 2e-16 ***
## zipcode_98003 -4.346e-01  1.525e-02 -28.499  < 2e-16 ***
## zipcode_98004  6.019e-01  1.438e-02  41.846  < 2e-16 ***
## zipcode_98005  2.356e-01  1.872e-02  12.586  < 2e-16 ***
## zipcode_98006  1.260e-01  1.199e-02  10.510  < 2e-16 ***
## zipcode_98007  1.737e-01  2.073e-02   8.378  < 2e-16 ***
## zipcode_98008  1.801e-01  1.516e-02  11.879  < 2e-16 ***
## zipcode_98010 -2.002e-01  2.431e-02  -8.235  < 2e-16 ***
## zipcode_98011 -1.665e-02  1.801e-02  -0.925 0.355056
## zipcode_98014 -1.206e-01  2.289e-02  -5.270 1.38e-07 ***
## zipcode_98019 -1.140e-01  1.768e-02  -6.449 1.16e-10 ***
## zipcode_98022 -4.203e-01  1.681e-02 -25.012  < 2e-16 ***
## zipcode_98023 -4.889e-01  1.177e-02 -41.541  < 2e-16 ***
## zipcode_98028 -4.214e-02  1.545e-02  -2.728 0.006385 **
## zipcode_98029  1.314e-01  1.404e-02   9.357  < 2e-16 ***
## zipcode_98030 -4.040e-01  1.567e-02 -25.789  < 2e-16 ***
## zipcode_98031 -3.818e-01  1.507e-02 -25.341  < 2e-16 ***
## zipcode_98032 -4.829e-01  2.221e-02 -21.744  < 2e-16 ***
## zipcode_98033  3.006e-01  1.258e-02  23.897  < 2e-16 ***
## zipcode_98034  8.003e-02  1.135e-02   7.052 1.84e-12 ***
## zipcode_98038 -2.775e-01  1.087e-02 -25.534  < 2e-16 ***
## zipcode_98039  7.549e-01  3.481e-02  21.690  < 2e-16 ***
## zipcode_98040  3.480e-01  1.532e-02  22.713  < 2e-16 ***
## zipcode_98042 -3.954e-01  1.128e-02 -35.049  < 2e-16 ***
## zipcode_98045 -1.231e-01  1.679e-02  -7.335 2.33e-13 ***
## zipcode_98052  1.674e-01  1.077e-02  15.534  < 2e-16 ***
## zipcode_98053  1.084e-01  1.325e-02   8.183 2.98e-16 ***
## zipcode_98055 -3.262e-01  1.519e-02 -21.474  < 2e-16 ***
## zipcode_98056 -1.441e-01  1.261e-02 -11.424  < 2e-16 ***
## zipcode_98058 -3.042e-01  1.222e-02 -24.905  < 2e-16 ***
## zipcode_98059 -1.308e-01  1.200e-02 -10.899  < 2e-16 ***
## zipcode_98065 -6.563e-02  1.431e-02  -4.586 4.55e-06 ***
## zipcode_98070 -1.409e-01  2.307e-02  -6.109 1.02e-09 ***
## zipcode_98072  2.151e-02  1.546e-02   1.391 0.164121
## zipcode_98077 -3.647e-02  1.832e-02  -1.991 0.046492 *
## zipcode_98092 -4.375e-01  1.331e-02 -32.865  < 2e-16 ***
## zipcode_98102  4.388e-01  2.252e-02  19.485  < 2e-16 ***
## zipcode_98103  3.225e-01  1.125e-02  28.677  < 2e-16 ***
## zipcode_98105  4.449e-01  1.684e-02  26.413  < 2e-16 ***
## zipcode_98106 -1.612e-01  1.365e-02 -11.811  < 2e-16 ***
## zipcode_98107  3.392e-01  1.561e-02  21.735  < 2e-16 ***
## zipcode_98108 -1.023e-01  1.759e-02  -5.812 6.30e-09 ***
## zipcode_98109  4.764e-01  2.321e-02  20.525  < 2e-16 ***
## zipcode_98112  5.346e-01  1.588e-02  33.661  < 2e-16 ***
## zipcode_98115  3.301e-01  1.123e-02  29.382  < 2e-16 ***
## zipcode_98116  2.609e-01  1.377e-02  18.952  < 2e-16 ***
## zipcode_98117  3.196e-01  1.145e-02  27.905  < 2e-16 ***
## zipcode_98119  4.818e-01  1.777e-02  27.120  < 2e-16 ***
## zipcode_98122  2.927e-01  1.491e-02  19.631  < 2e-16 ***
## zipcode_98125  9.784e-02  1.279e-02   7.648 2.16e-14 ***
## zipcode_98126  5.713e-02  1.388e-02   4.116 3.87e-05 ***
## zipcode_98136  1.930e-01  1.577e-02  12.244  < 2e-16 ***
## zipcode_98144  1.649e-01  1.368e-02  12.055  < 2e-16 ***
## zipcode_98146 -1.933e-01  1.524e-02 -12.682  < 2e-16 ***
## zipcode_98148 -3.138e-01  2.931e-02 -10.708  < 2e-16 ***
## zipcode_98155 -4.657e-02  1.224e-02  -3.806 0.000142 ***
## zipcode_98166 -1.722e-01  1.532e-02 -11.240  < 2e-16 ***
## zipcode_98168 -3.822e-01  1.565e-02 -24.418  < 2e-16 ***
## zipcode_98177  1.181e-01  1.590e-02   7.426 1.18e-13 ***
```

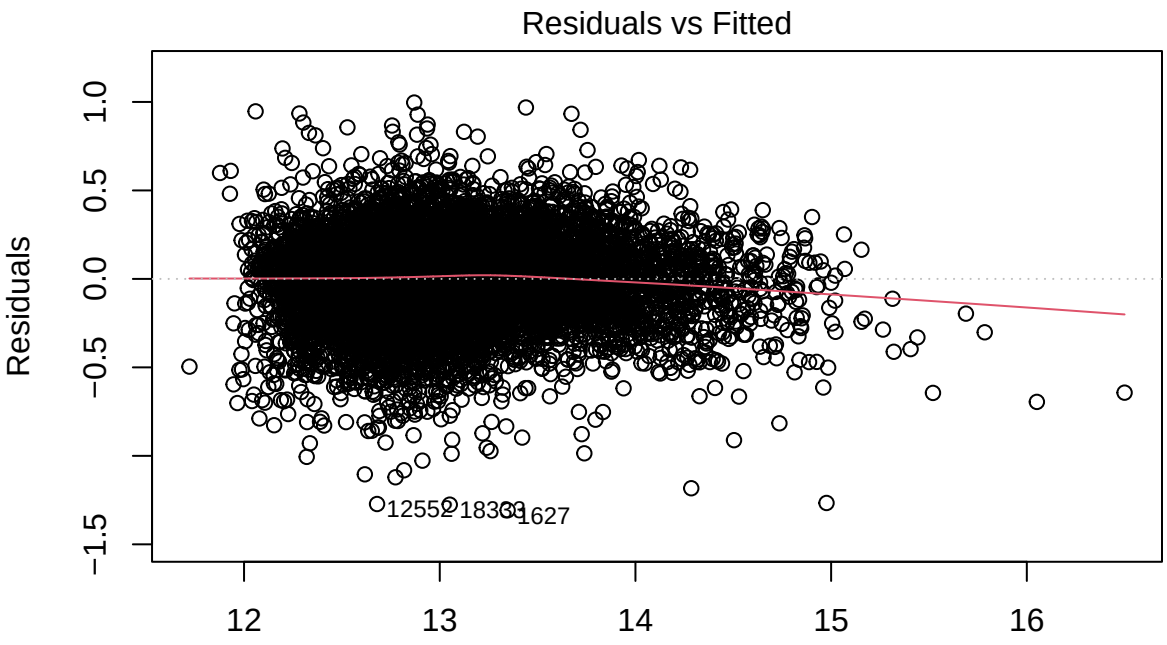
```
## zipcode_98178 -3.264e-01  1.575e-02 -20.719  < 2e-16 ***
## zipcode_98188 -3.733e-01  2.162e-02 -17.263  < 2e-16 ***
## zipcode_98198 -3.855e-01  1.560e-02 -24.707  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2002 on 15053 degrees of freedom
## Multiple R-squared:  0.855,  Adjusted R-squared:  0.8542
## F-statistic:  1183 on 75 and 15053 DF,  p-value: < 2.2e-16
```

7. Make sure that model assumption(s) are checked for the final model. Apply remedy measures (transformation, etc.) that helps satisfy the assumptions.

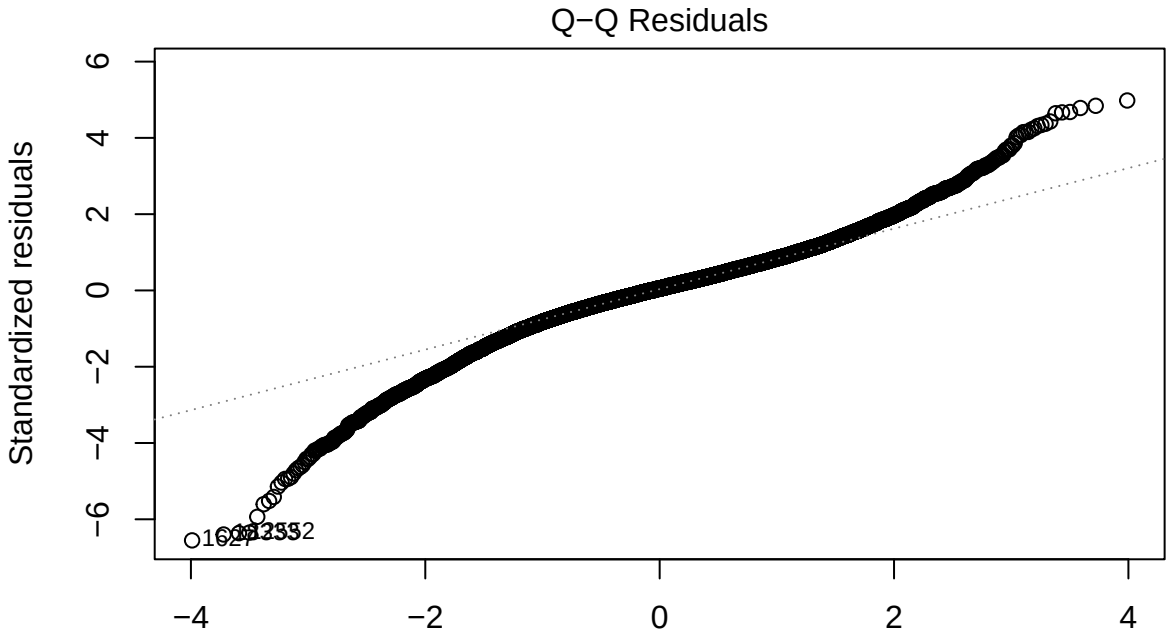
```
# Graph the distribution of the residuals. This returns a large amount of outliers
boxplot(model_log$residuals,col="orange",
main="distribution of the residuals")
```



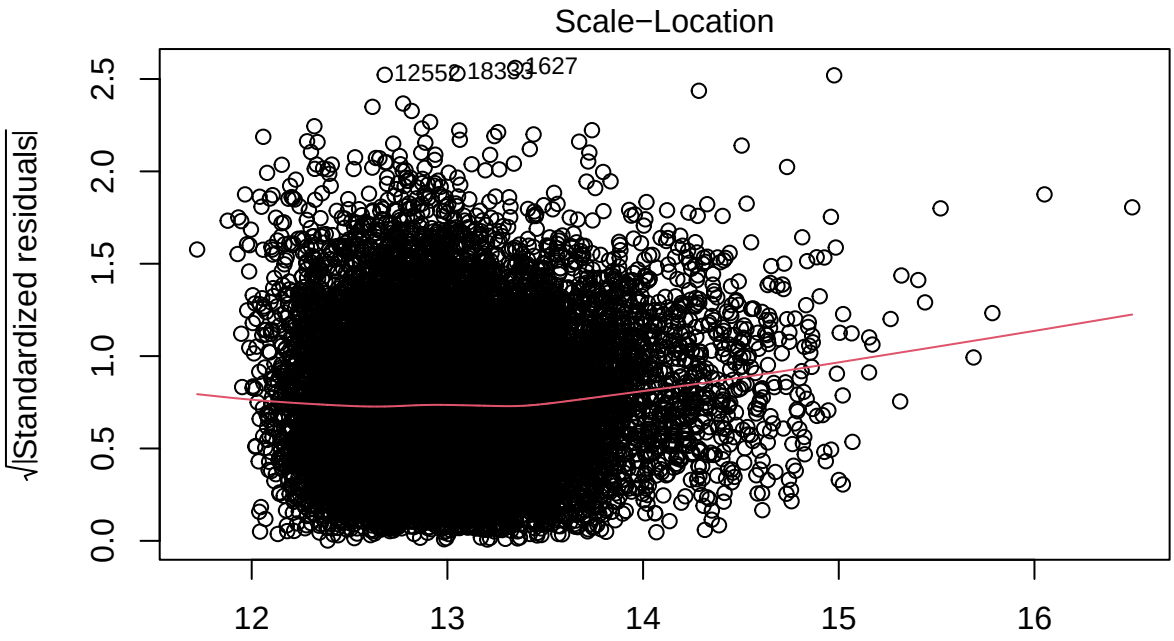

```
# Plot Residuals vs Fitted. The residuals are far from the line and demonstrate a funnel pattern that starts t
par(mfrow=c(1,1))
plot(model_log,which=1)
```



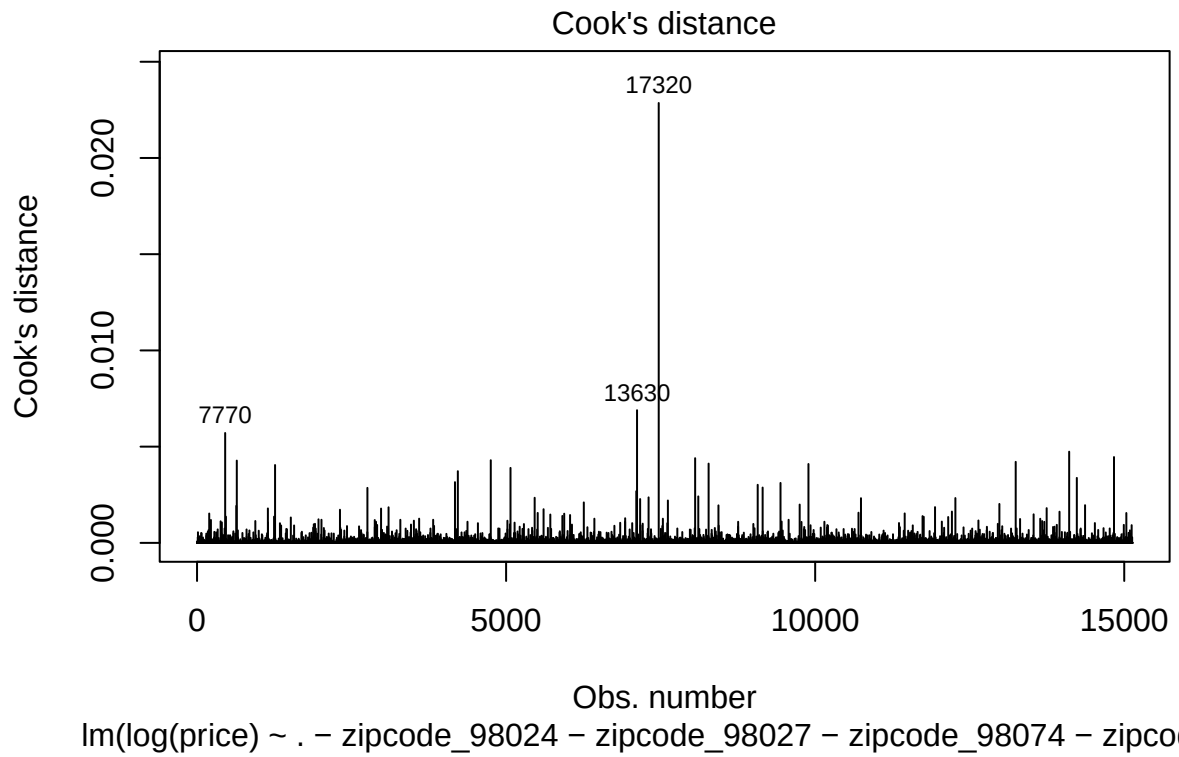

```
# Plot a Normal Q-Q. There are outliers at both ends indicating skewness and heavy tails and that our residual
plot(model_log,which=2)
```



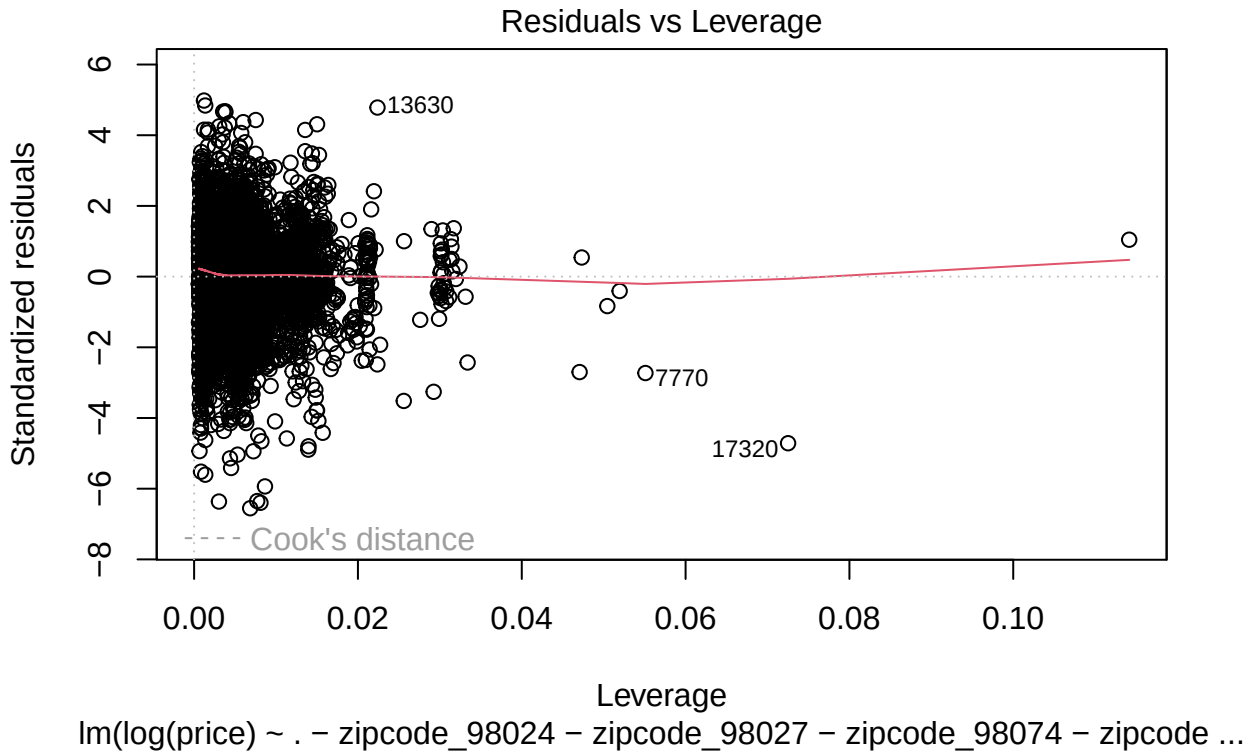

```
# Plot Scale-Location. Points are roughly spread if the ouliers in the top right quadrant are disregarded.
plot(model_log,which=3)
```




```
# Plot Cook's Distance. This shows one point above .05 that should be acknowlged and potentially removed becau
plot(model_log,which=4)
```




```
# Plot Residuals vs Leverage. There are some points with high leverage but they don't have high residuals as w
plot(model_log,which=5)
```



8. Investigate unequal variances and multicollinearity. If necessary, apply remedial methods (WLS, Ridge, Elastic Net, Lasso, etc.).

```
library(glmnet)
```

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

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

```
# Step 1: Investigate Unequal Variances

# Check for heteroscedasticity using a formal test (e.g., Breusch-Pagan test). Results in a small p-value (e.g

bp_test <- lmtest::bptest(model_log)
print(bp_test)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model_log
## BP = 1253.9, df = 75, p-value < 2.2e-16
```

```
# Step 2: Investigate Multicollinearity
# Check Variance Inflation Factors (VIF)
# Results in variables "sqft_living: 8.63" and "sqft_above: 7.27" having moderate multicollinearity
vif_values <- car::vif(model_log)

# Print VIF values above 5
high_vif_variables <- names(vif_values[vif_values > 5])
high_vif_values <- vif_values[vif_values > 5]

cat("Variables with VIF > 5:\n")
```

```
## Variables with VIF > 5:
```

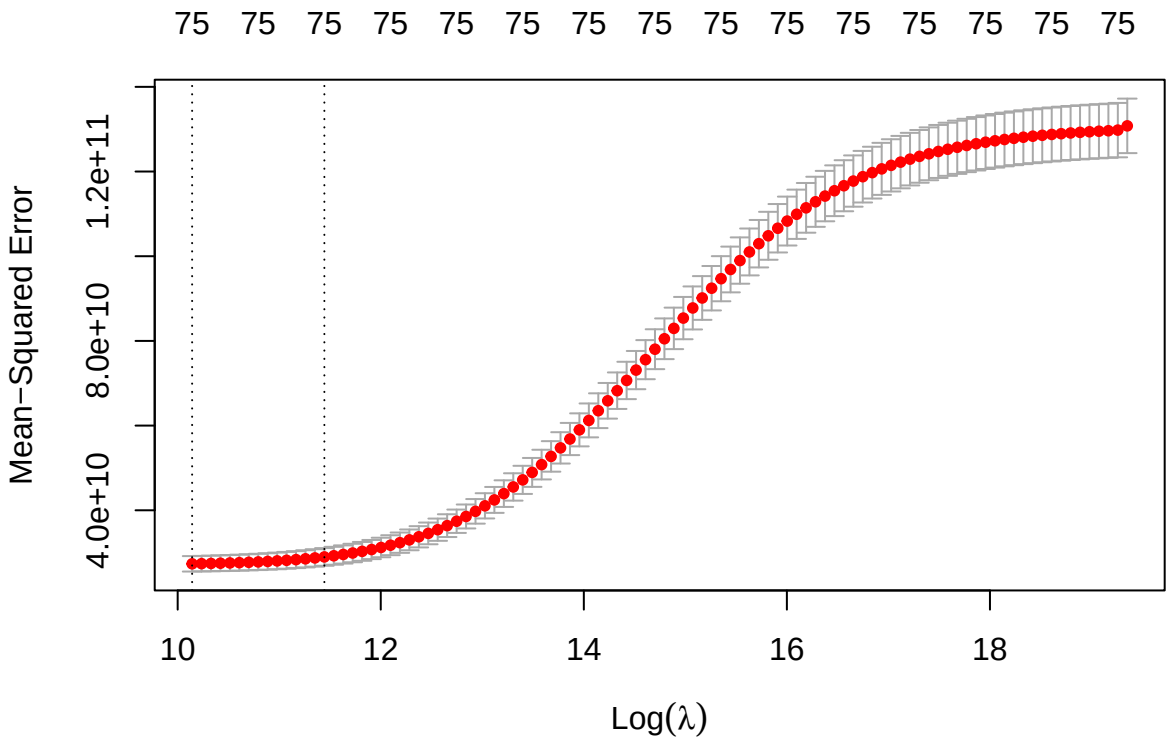
```
for (i in seq_along(high_vif_variables)) {
  cat(sprintf("%s: %.2f\n", high_vif_variables[i], high_vif_values[i]))
}
```

```
## sqft_living: 8.63
## sqft_above: 7.27
```

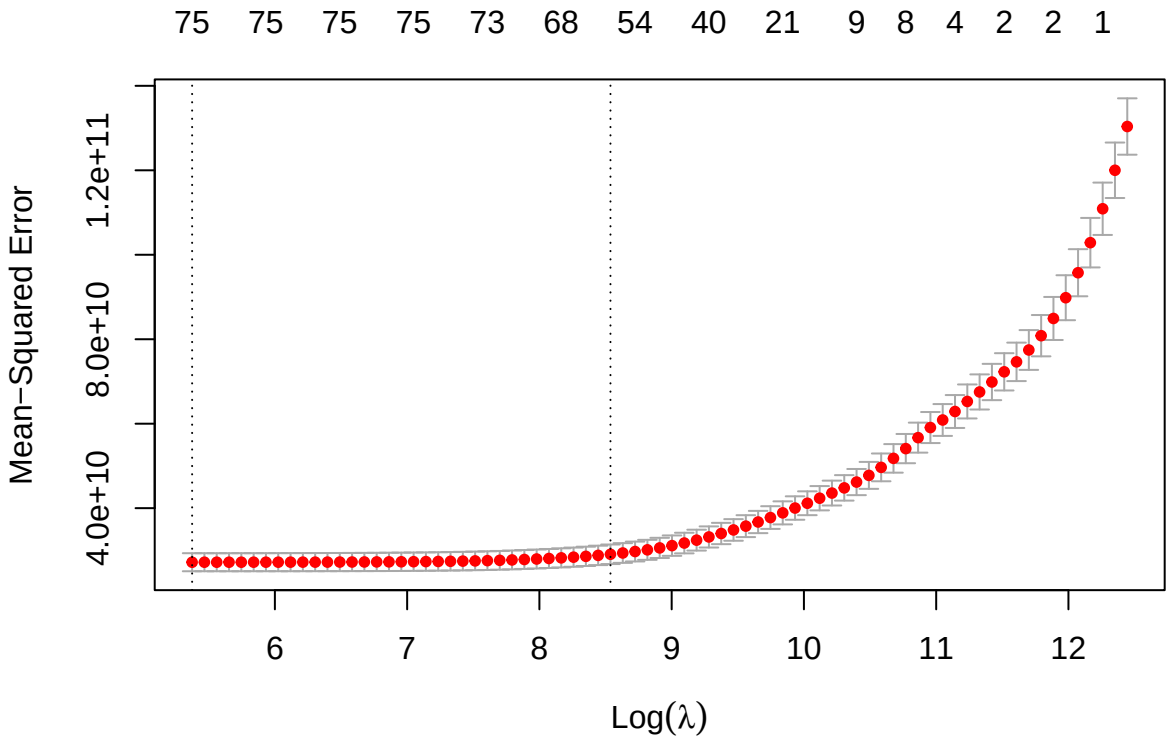
```
# Step 3: Apply Remedial Methods - Ridge, Lasso, and WLS Regression
x <- model.matrix(model_log)[, -1]  # Exclude intercept column
y <- train.dat$price

# Ridge Regression
ridge_model <- glmnet(x, y, alpha = 0, lambda = 0.1)  # Choose an appropriate lambda
ridge_predictions <- predict(ridge_model, newx = x)
ridge_residuals <- y - ridge_predictions

# Lasso Regression
lasso_model <- glmnet(x, y, alpha = 1, lambda = 0.1)  # Choose an appropriate lambda

# Weighted Least Squares (WLS)
wls_residuals <- residuals(model_log)
weights <- 1 / wls_residuals^2
wls_model <- lm(y ~ ., data = train.dat, weights = weights)


# Evaluate and compare the models
ridge_predictions <- predict(ridge_model, newx = x)
ridge_residuals <- y - ridge_predictions
ridge_mse <- mean(ridge_residuals^2)
ridge_rmse <- sqrt(ridge_mse)
ridge_r_squared <- 1 - ridge_mse / var(y)

lasso_predictions <- predict(lasso_model, newx = x)
lasso_residuals <- y - lasso_predictions
lasso_mse <- mean(lasso_residuals^2)
lasso_rmse <- sqrt(lasso_mse)
lasso_r_squared <- 1 - lasso_mse / var(y)

wls_residuals <- residuals(wls_model)
wls_mse <- mean(wls_residuals^2)
wls_rmse <- sqrt(wls_mse)
wls_r_squared <- 1 - wls_mse / var(y)

# Compare the metrics
comparison_table <- data.frame(
  Model = c("Ridge", "Lasso", "WLS"),
  RMSE = c(ridge_rmse, lasso_rmse, wls_rmse),
  R_squared = c(ridge_r_squared, lasso_r_squared, wls_r_squared)
)

print(comparison_table)
```

```
##   Model         RMSE R_squared
## 1 Ridge 1.631519e+05 0.7965111
## 2 Lasso 1.631519e+05 0.7965111
## 3   WLS 1.139190e-08 1.0000000
```

```
# Cross-validated Ridge model
cv_ridge <- cv.glmnet(x, y, alpha = 0)

# Cross-validated Lasso model
cv_lasso <- cv.glmnet(x, y, alpha = 1)

# Plot the results. Results in a
plot(cv_ridge)
```




```
plot(cv_lasso)
```



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.

Predicting Price using Regression Tree.

```
#Building Regression Tree to Predict Prices
library(rpart)
#Build Regression Model with Pruning
m.rpart <- rpart(price ~ ., data = train.dat,cp=0.001 )
m.rpart
```

```
## n= 15129
```

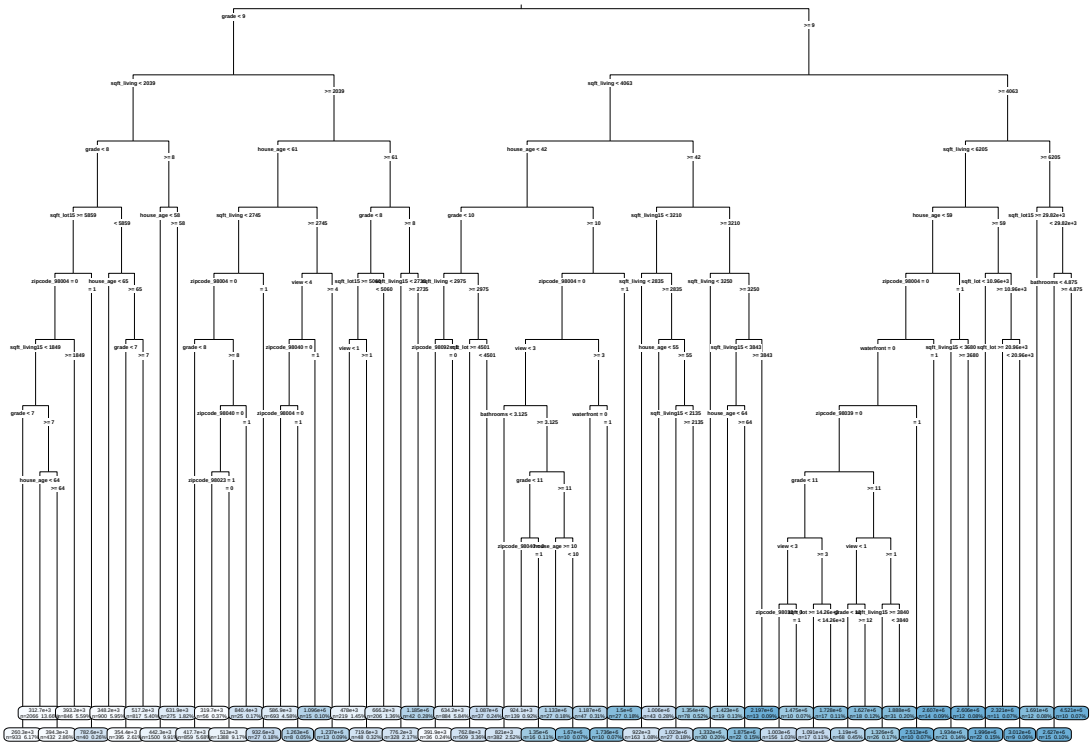
```
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##    1) root 15129 1.978906e+15  538713.5
##      2) grade< 8.5 12131 4.601037e+14  437022.5
##        4) sqft_living< 2039 8204 1.865147e+14  382300.8
##          8) grade< 7.5 6429 1.268167e+14  357619.7
##           16) sqft_lot15>=5859 4317 7.047085e+13  329676.0
##             32) zipcode_98004< 0.5 4277 6.139597e+13  325439.8
##               64) sqft_living15< 1848.5 3431 4.059058e+13  308736.1
##                128) grade< 6.5 933 8.038034e+12  260262.6 *
##                129) grade>=6.5 2498 2.954149e+13  326841.0
##                  258) house_age< 63.5 2066 1.941024e+13  312730.8 *
##                  259) house_age>=63.5 432 7.752745e+12  394321.6 *
##               65) sqft_living15>=1848.5 846 1.596573e+13  393182.6 *
##             33) zipcode_98004>=0.5 40 7.913916e+11  782631.2 *
##           17) sqft_lot15< 5859 2112 4.608458e+13  414737.6
##             34) house_age< 64.5 900 1.195536e+13  348228.0 *
##             35) house_age>=64.5 1212 2.719173e+13  464125.9
##               70) grade< 6.5 395 5.403370e+12  354395.0 *
##               71) grade>=6.5 817 1.473273e+13  517178.2 *
##          9) grade>=7.5 1775 4.159731e+13  471694.7
##           18) house_age< 57.5 1500 2.428175e+13  442323.9 *
##           19) house_age>=57.5 275 8.963615e+12  631898.9 *
##        5) sqft_living>=2039 3927 1.976996e+14  551343.0
##         10) house_age< 60.5 3084 1.140782e+14  513712.2
##           20) sqft_living< 2745 2355 6.170347e+13  481928.4
##             40) zipcode_98004< 0.5 2328 5.501049e+13  476701.1
##               80) grade< 7.5 859 1.374830e+13  417652.3 *
##               81) grade>=7.5 1469 3.651566e+13  511229.9
##                162) zipcode_98040< 0.5 1444 3.317923e+13  505531.2
##                  324) zipcode_98023>=0.5 56 2.033892e+11  319725.0 *
##                  325) zipcode_98023< 0.5 1388 3.096450e+13  513027.7 *
##                163) zipcode_98040>=0.5 25 5.808465e+11  840391.6 *
##             41) zipcode_98004>=0.5 27 1.144552e+12  932640.7 *
##           21) sqft_living>=2745 729 4.231029e+13  616388.2
##             42) view< 3.5 716 3.483868e+13  605112.4
##               84) zipcode_98040< 0.5 701 3.036403e+13  594601.4
##                168) zipcode_98004< 0.5 693 2.534507e+13  586890.6 *
##                169) zipcode_98004>=0.5 8 1.408523e+12 1262550.0 *
##               85) zipcode_98040>=0.5 15 7.778252e+11 1096327.0 *
##             43) view>=3.5 13 2.366681e+12 1237423.0 *
##         11) house_age>=60.5 843 6.327745e+13  689010.4
##           22) grade< 7.5 473 2.113671e+13  584507.6
##             44) sqft_lot15>=5060 267 1.351832e+13  521472.9
##               88) view< 0.5 219 6.448736e+12  478042.7 *
##               89) view>=0.5 48 4.771859e+12  719623.1 *
##             45) sqft_lot15< 5060 206 5.182468e+12  666207.9 *
##           23) grade>=7.5 370 3.037165e+13  822604.5
##             46) sqft_living15< 2735 328 1.785103e+13  776197.3 *
##             47) sqft_living15>=2735 42 6.297658e+12 1185023.0 *
##      3) grade>=8.5 2998 8.857481e+14  950192.7
##        6) sqft_living< 4062.5 2519 3.159003e+14  830227.4
##         12) house_age< 41.5 2124 1.802983e+14  766395.5
##           24) grade< 9.5 1466 7.369014e+13  684326.3
##             48) sqft_living< 2975 920 3.350411e+13  624720.8
##               96) zipcode_98092>=0.5 36 8.095363e+10  391883.4 *
##               97) zipcode_98092< 0.5 884 3.139200e+13  634202.8 *
##             49) sqft_living>=2975 546 3.140993e+13  784760.5
```

```
##               98) sqft_lot>=4501 509 2.433624e+13  762764.2 *
##               99) sqft_lot< 4501 37 3.439502e+12 1087358.0 *
##           25) grade>=9.5 658 7.473518e+13  949242.6
##             50) zipcode_98004< 0.5 631 6.324444e+13  925682.5
##              100) view< 2.5 574 4.379523e+13  890148.2
##                200) bathrooms< 3.125 382 1.734380e+13  820970.7 *
##                201) bathrooms>=3.125 192 2.098626e+13 1027782.0
##                  402) grade< 10.5 155 1.137528e+13  968072.8
##                    804) zipcode_98040< 0.5 139 8.063946e+12  924125.1 *
##                    805) zipcode_98040>=0.5 16 7.105882e+11 1349868.0 *
##                  403) grade>=10.5 37 6.743363e+12 1277918.0
##                    806) house_age>=10 27 1.669977e+12 1132776.0 *
##                    807) house_age< 10 10 2.968884e+12 1669800.0 *
##              101) view>=2.5 57 1.142575e+13 1283519.0
##                202) waterfront< 0.5 47 6.176174e+12 1187161.0 *
##                203) waterfront>=0.5 10 2.762176e+12 1736400.0 *
##             51) zipcode_98004>=0.5 27 2.954878e+12 1499852.0 *
##          13) house_age>=41.5 395 8.041189e+13 1173465.0
##            26) sqft_living15< 3210 311 3.568058e+13 1050558.0
##              52) sqft_living< 2835 163 1.166820e+13  921970.1 *
##              53) sqft_living>=2835 148 1.834885e+13 1192179.0
##               106) house_age< 54.5 43 5.367233e+12 1005835.0 *
##               107) house_age>=54.5 105 1.087700e+13 1268491.0
##                 214) sqft_living15< 2135 27 1.232161e+12 1022840.0 *
##                 215) sqft_living15>=2135 78 7.451550e+12 1353524.0 *
##            27) sqft_living15>=3210 84 2.263945e+13 1628514.0
##              54) sqft_living< 3250 30 3.600600e+12 1331620.0 *
##              55) sqft_living>=3250 54 1.492536e+13 1793456.0
##               110) sqft_living15< 3843 41 7.946041e+12 1665400.0
##                 220) house_age< 63.5 19 1.469966e+12 1422994.0 *
##                 221) house_age>=63.5 22 4.395422e+12 1874750.0 *
##               111) sqft_living15>=3843 13 4.186565e+12 2197324.0 *
##        7) sqft_living>=4062.5 479 3.429471e+14 1581075.0
##          14) sqft_living< 6205 442 2.030485e+14 1476089.0
##            28) house_age< 58.5 400 1.562764e+14 1389698.0
##              56) zipcode_98004< 0.5 367 1.215545e+14 1318783.0
##               112) waterfront< 0.5 353 9.339697e+13 1267694.0
##                 224) zipcode_98039< 0.5 343 7.335211e+13 1231397.0
##                   448) grade< 10.5 200 3.473657e+13 1095910.0
##                     896) view< 2.5 166 1.860566e+13 1031676.0
##                      1792) zipcode_98033< 0.5 156 1.578246e+13 1003245.0 *
##                      1793) zipcode_98033>=0.5 10 7.299696e+11 1475200.0 *
##                     897) view>=2.5 34 1.210205e+13 1409520.0
##                      1794) sqft_lot>=14263.5 17 3.964215e+12 1091456.0 *
##                      1795) sqft_lot< 14263.5 17 4.698227e+12 1727585.0 *
##                   449) grade>=10.5 143 2.980935e+13 1420891.0
##                     898) view< 0.5 86 1.047070e+13 1281255.0
##                      1796) grade< 11.5 68 5.594428e+12 1189602.0 *
##                      1797) grade>=11.5 18 2.147106e+12 1627500.0 *
##                     899) view>=0.5 57 1.513183e+13 1631569.0
##                      1798) sqft_living15>=3840 26 2.646860e+12 1325577.0 *
##                      1799) sqft_living15< 3840 31 8.008787e+12 1888208.0 *
##                 225) zipcode_98039>=0.5 10 4.093755e+12 2512653.0 *
##               113) waterfront>=0.5 14 4.004377e+12 2606964.0 *
##              57) zipcode_98004>=0.5 33 1.235053e+13 2178363.0
##               114) sqft_living15< 3680 21 3.116424e+12 1934082.0 *
##               115) sqft_living15>=3680 12 5.787981e+12 2605854.0 *
##            29) house_age>=58.5 42 1.535474e+13 2298861.0
##              58) sqft_lot< 10960 22 4.160452e+12 1995883.0 *
##              59) sqft_lot>=10960 20 6.953314e+12 2632138.0
```

```
##               118) sqft_lot>=20959.5 11 2.956318e+12 2321000.0 *
##               119) sqft_lot< 20959.5 9 1.630614e+12 3012417.0 *
##           15) sqft_living>=6205 37 7.682943e+13 2835229.0
##             30) sqft_lot15>=29816 12 4.606470e+12 1691012.0 *
##             31) sqft_lot15< 29816 25 4.897095e+13 3384454.0
##               62) bathrooms< 4.875 15 1.005289e+13 2626823.0 *
##               63) bathrooms>=4.875 10 1.739291e+13 4520900.0 *
```

```
library(rpart.plot)
#Regression Tree Plot
 rpart.plot(m.rpart, digits = 4, fallen.leaves = TRUE,type = 3, extra = 101)
```

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



```
#Print Variables used in Tree Construction
printcp(m.rpart)
```

```
##
## Regression tree:
## rpart(formula = price ~ ., data = train.dat, cp = 0.001)
##
## Variables actually used in tree construction:
##  [1] bathrooms     grade         house_age     sqft_living   sqft_living15
##  [6] sqft_lot      sqft_lot15    view          waterfront    zipcode_98004
## [11] zipcode_98023 zipcode_98033 zipcode_98039 zipcode_98040 zipcode_98092
##
## Root node error: 1.9789e+15/15129 = 1.308e+11
##
## n= 15129
##
##           CP nsplit rel error  xerror     xstd
## 1  0.3199011      0   1.00000 1.00009 0.045768
## 2  0.1146597      1   0.68010 0.68969 0.035962
## 3  0.0383491      2   0.56544 0.57836 0.027347
## 4  0.0318707      3   0.52709 0.54353 0.027200
## 5  0.0278892      4   0.49522 0.53348 0.026323
## 6  0.0161064      5   0.46733 0.49344 0.020238
```

```
## 7  0.0158762      6   0.45122 0.47822 0.020109
## 8  0.0117551      7   0.43535 0.46260 0.019380
## 9  0.0117499      9   0.41184 0.45516 0.019486
## 10 0.0111637     10   0.40009 0.45205 0.019258
## 11 0.0108773     11   0.38892 0.44851 0.019115
## 12 0.0102804     12   0.37805 0.44365 0.019049
## 13 0.0091469     13   0.36777 0.43658 0.018854
## 14 0.0080606     14   0.35862 0.41074 0.016828
## 15 0.0059473     15   0.35056 0.40067 0.016918
## 16 0.0051853     16   0.34461 0.39337 0.016765
## 17 0.0050859     17   0.33943 0.38262 0.016233
## 18 0.0044500     18   0.33434 0.37396 0.015593
## 19 0.0044348     19   0.32989 0.36639 0.013348
## 20 0.0043134     20   0.32546 0.36534 0.013335
## 21 0.0042205     21   0.32114 0.36301 0.013333
## 22 0.0041859     22   0.31692 0.36163 0.013372
## 23 0.0040545     23   0.31274 0.35767 0.013341
## 24 0.0035356     24   0.30868 0.35065 0.013258
## 25 0.0031446     26   0.30161 0.34245 0.013168
## 26 0.0028620     27   0.29847 0.33885 0.013101
## 27 0.0028038     28   0.29560 0.33677 0.013060
## 28 0.0027617     29   0.29280 0.33650 0.013060
## 29 0.0025797     30   0.29004 0.33536 0.013049
## 30 0.0024456     31   0.28746 0.33117 0.012822
## 31 0.0023986     32   0.28501 0.32972 0.012782
## 32 0.0021939     33   0.28261 0.32737 0.012757
## 33 0.0021431     35   0.27823 0.32385 0.012685
## 34 0.0020787     36   0.27608 0.32464 0.012739
## 35 0.0020359     37   0.27400 0.32440 0.012733
## 36 0.0018681     38   0.27197 0.32416 0.012728
## 37 0.0018365     39   0.27010 0.32232 0.012728
## 38 0.0018245     40   0.26826 0.32336 0.012754
## 39 0.0017414     41   0.26644 0.32168 0.012729
## 40 0.0017381     42   0.26470 0.32120 0.012737
## 41 0.0015216     43   0.26296 0.31807 0.012652
## 42 0.0014491     44   0.26144 0.31650 0.012666
## 43 0.0014113     45   0.25999 0.31603 0.012678
## 44 0.0013925     46   0.25858 0.31598 0.012679
## 45 0.0013791     47   0.25719 0.31554 0.012678
## 46 0.0013142     48   0.25581 0.31545 0.012746
## 47 0.0012570     49   0.25449 0.31574 0.012791
## 48 0.0012309     50   0.25324 0.31505 0.012854
## 49 0.0012019     51   0.25200 0.31309 0.012836
## 50 0.0011958     52   0.25080 0.31316 0.012844
## 51 0.0011611     53   0.24961 0.31314 0.012845
## 52 0.0010859     54   0.24845 0.31154 0.012867
## 53 0.0010635     56   0.24627 0.31040 0.012937
## 54 0.0010578     57   0.24521 0.30912 0.012901
## 55 0.0010514     58   0.24415 0.30912 0.012901
## 56 0.0010264     59   0.24310 0.30850 0.012902
## 57 0.0010164     60   0.24207 0.30759 0.012894
## 58 0.0010000     61   0.24106 0.30732 0.012897
```

Determining Price using SVM Model :

```r
library(e1071)

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

# Print the model summary
```

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

```
##
## Call:
## svm(formula = price ~ ., data = train.dat)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.01219512
##     epsilon:  0.1
##
##
## Number of Support Vectors:  8459
```

Logistic Regression. Convert Price to a binary 1(High) or 0(Low) to Predict Price. Predict Price_cat using the GLM method

```r
# Convert price to a binary variable for Logistic Regression
train.dat$price_cat <- ifelse(train.dat$price > median(train.dat$price), 1, 0)
# Convert price to a binary variable for Logistic Regression
test.dat$price_cat <- ifelse(test.dat$price > median(test.dat$price), 1, 0)
```

```r
# Build a logistic regression model
log_model <- glm(price_cat ~ ., data = train.dat, family = binomial())
```

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

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

```r
# Print the model summary
print(summary(log_model))
```

```
##
## Call:
## glm(formula = price_cat ~ ., family = binomial(), data = train.dat)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -9.233e+03  2.077e+04  -0.445    0.657
## price          2.050e-02  4.591e-02   0.447    0.655
## bedrooms       1.002e+00  7.941e+01   0.013    0.990
## bathrooms     -7.126e+00  1.912e+02  -0.037    0.970
## sqft_living   -9.980e-03  2.451e-01  -0.041    0.968
## sqft_lot       2.871e-05  3.295e-03   0.009    0.993
## floors        -3.006e+00  1.156e+02  -0.026    0.979
## waterfront    -4.466e+02  1.807e+04  -0.025    0.980
## view           1.522e+01  2.740e+03   0.006    0.996
## condition     -1.500e+00  1.249e+02  -0.012    0.990
## grade          4.731e-01  9.781e+01   0.005    0.996
## sqft_above     2.021e-02  2.077e-01   0.097    0.922
## sqft_living15 -1.054e-02  2.516e-01  -0.042    0.967
## sqft_lot15    -1.917e-05  8.387e-03  -0.002    0.998
## house_age     -3.158e-02  3.689e+00  -0.009    0.993
## renovated     -1.689e+00  2.441e+02  -0.007    0.994
## zipcode_98002  1.649e+03  1.608e+04   0.103    0.918
## zipcode_98003 -4.338e+00  7.562e+03  -0.001    1.000
## zipcode_98004 -2.399e+03  1.019e+04  -0.236    0.814
## zipcode_98005 -4.707e+02  1.360e+04  -0.035    0.972
## zipcode_98006 -1.321e+01  8.989e+03  -0.001    0.999
## zipcode_98007  1.145e+01  2.450e+03   0.005    0.996
```

```
## zipcode_98008 -7.012e+00  3.535e+03  -0.002    0.998
## zipcode_98010  2.050e+02  1.696e+05   0.001    0.999
## zipcode_98011  7.486e-01  6.628e+03   0.000    1.000
## zipcode_98014  1.054e+02  4.638e+04   0.002    0.998
## zipcode_98019  1.543e+02  1.033e+06   0.000    1.000
## zipcode_98022 -1.717e+02  9.007e+03  -0.019    0.985
## zipcode_98023  2.158e+01  1.133e+04   0.002    0.998
## zipcode_98024 -1.523e+02  3.791e+05   0.000    1.000
## zipcode_98027  1.771e+01  2.454e+03   0.007    0.994
## zipcode_98028  1.127e+01  2.452e+03   0.005    0.996
## zipcode_98029  1.760e+01  2.452e+03   0.007    0.994
## zipcode_98030  8.628e+00  1.009e+05   0.000    1.000
## zipcode_98031  4.329e+02  1.204e+04   0.036    0.971
## zipcode_98032  9.683e+02  2.331e+04   0.042    0.967
## zipcode_98033 -1.580e+02  6.019e+03  -0.026    0.979
## zipcode_98034 -1.496e+01  3.712e+03  -0.004    0.997
## zipcode_98038  1.274e+01  4.042e+03   0.003    0.997
## zipcode_98039 -8.462e+03  3.161e+04  -0.268    0.789
## zipcode_98040 -9.598e+02  9.876e+03  -0.097    0.923
## zipcode_98042 -4.990e+00  1.845e+04   0.000    1.000
## zipcode_98045  1.101e+00  4.473e+03   0.000    1.000
## zipcode_98052  9.085e+00  3.971e+03   0.002    0.998
## zipcode_98053  6.816e+00  2.658e+03   0.003    0.998
## zipcode_98055  2.007e+02  8.007e+04   0.003    0.998
## zipcode_98056  1.599e+01  2.444e+03   0.007    0.995
## zipcode_98058 -3.868e+01  1.254e+05   0.000    1.000
## zipcode_98059 -4.897e+00  3.268e+03  -0.001    0.999
## zipcode_98065  1.541e+01  2.450e+03   0.006    0.995
## zipcode_98070  1.283e+02  3.707e+03   0.035    0.972
## zipcode_98072  2.251e+01  2.466e+03   0.009    0.993
## zipcode_98074 -7.422e+01  1.569e+04  -0.005    0.996
## zipcode_98075 -6.348e+02  4.186e+06   0.000    1.000
## zipcode_98077 -1.116e+02  4.523e+04  -0.002    0.998
## zipcode_98092 -1.079e+02  2.410e+05   0.000    1.000
## zipcode_98102 -4.603e+02  2.483e+04  -0.019    0.985
## zipcode_98103  9.223e+00  2.948e+03   0.003    0.998
## zipcode_98105 -6.406e+01  6.536e+03  -0.010    0.992
## zipcode_98106  5.518e+00  2.449e+03   0.002    0.998
## zipcode_98107  7.883e+00  2.573e+03   0.003    0.998
## zipcode_98108  4.279e+01  3.191e+03   0.013    0.989
## zipcode_98109 -4.882e+00  2.500e+04   0.000    1.000
## zipcode_98112 -2.234e+02  3.269e+04  -0.007    0.995
## zipcode_98115  1.634e+01  2.466e+03   0.007    0.995
## zipcode_98116 -3.924e+01  1.358e+04  -0.003    0.998
## zipcode_98117 -6.648e+00  3.504e+03  -0.002    0.998
## zipcode_98118  3.979e+00  2.587e+03   0.002    0.999
## zipcode_98119 -4.213e+01  4.271e+03  -0.010    0.992
## zipcode_98122  9.613e+00  8.265e+03   0.001    0.999
## zipcode_98125  1.096e+01  2.453e+03   0.004    0.996
## zipcode_98126 -1.197e+02  4.293e+04  -0.003    0.998
## zipcode_98133  2.416e+01  2.455e+03   0.010    0.992
## zipcode_98136  2.446e+01  2.465e+03   0.010    0.992
## zipcode_98144 -2.139e+01  7.585e+03  -0.003    0.998
## zipcode_98146 -4.229e+01  1.908e+05   0.000    1.000
## zipcode_98148  9.963e+01  3.838e+05   0.000    1.000
## zipcode_98155  1.240e+01  2.446e+03   0.005    0.996
## zipcode_98166 -8.878e-02  3.317e+03   0.000    1.000
## zipcode_98168  5.089e+02  1.413e+04   0.036    0.971
## zipcode_98177  1.369e+02  8.627e+04   0.002    0.999
## zipcode_98178  1.080e+01  4.183e+03   0.003    0.998
```

```
## zipcode_98188  1.028e+03  2.085e+04   0.049    0.961
## zipcode_98198  5.059e+02  4.264e+04   0.012    0.991
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 20973.16  on 15128  degrees of freedom
## Residual deviance:   288.35  on 15045  degrees of freedom
## AIC: 456.35
##
## Number of Fisher Scoring iterations: 25
```

The logistic regression model will predict the price of the home in to two categories (high/low price) rather than actual price values. So will continue with the SVM/Regression Tree as we will need actual Price values which will be valuable.

10. Use the test data set to assess the model performances from above.

Performance Testing for Regression Tree

```
#evaluating model performance for Regression
p.rpart <- predict(m.rpart, test.dat)
#Summary for Predicted Price
summary(p.rpart)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  260263  348228  442324  540161  631899 4520900
```

```
#Summary for Price in Test Data
summary(test.dat$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   81000  325000  450000  543295  648000 7062500
```

```
#Correlation between Predicted Price and Test Data
cor(p.rpart, test.dat$price)
```

```
## [1] 0.8101315
```

```
#Measuring performance with the mean absolute error
MAE <- function(actual, predicted) {mean(abs(actual - predicted))}

#The MAE for our predictions is then:
MAE(test.dat$price,p.rpart)
```

```
## [1] 131801.2
```

```
#Measuring performance with the SSE
SSE <- function(actual, predicted) {sum((actual - predicted)^2)}
SSE(test.dat$price,p.rpart)
```

```
## [1] 3.234463e+14
```

```
#Measuring performance with the RSquare
R2 <- function(actual, predicted) {sum((actual - predicted)^2)/((length(actual)-1)*var(actual))}
1-R2(test.dat$price,p.rpart)
```

```
## [1] 0.6536664
```

The R^2 for the Regression Tree is 66% indicating decent performance for the model but the MAE is significantly high.

Performance Testing for SVM Model

```
#evaluating model performance for Regression
p.svm_model <- predict(svm_model, test.dat)
#Summary for Predicted Price
summary(p.svm_model)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  130188  327562  455807  530568  631687 2801316
```

```
#Summary for Price in Test Data
summary(test.dat$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   81000  325000  450000  543295  648000 7062500
```

```
#Correlation between Predicted Price and Test Data
cor(p.svm_model, test.dat$price)
```

```
## [1] 0.8919623
```

```
#Measuring performance with the mean absolute error
MAE <- function(actual, predicted) {mean(abs(actual - predicted))}

#The MAE for our predictions is then:
MAE(test.dat$price,p.svm_model)
```

```
## [1] 75162.14
```

```
#Measuring performance with the SSE
SSE <- function(actual, predicted) {sum((actual - predicted)^2)}
SSE(test.dat$price,p.svm_model)
```

```
## [1] 1.990518e+14
```

```
#Measuring performance with the RSquare
R2 <- function(actual, predicted) {sum((actual - predicted)^2)/((length(actual)-1)*var(actual))}
1-R2(test.dat$price,p.svm_model)
```

```
## [1] 0.7868631
```

SVM Model Performed Better than Regression Tree and it has an R-squared value of 79% compared with 66%. We will use the SVM Model as our Alternate Model to Predict Price.

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.

```
#evaluating model performance for Regression
model_log.predicted_values <- exp(predict(model_log, test.dat))
#Summary for Predicted Price
summary(model_log.predicted_values)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   128677   328704   444396   537413   607591 13739743
```

```
#Summary for Price in Test Data
summary(test.dat$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   81000  325000  450000  543295  648000 7062500
```

```
#Correlation between Predicted Price and Test Data
cor(model_log.predicted_values, test.dat$price)
```

```
## [1] 0.8664888
```

```
#Measuring performance with the mean absolute error
MAE <- function(actual, predicted) {mean(abs(actual - predicted))}
#The MAE for our predictions is then:
MAE(test.dat$price, model_log.predicted_values)
```

```
## [1] 84891.57
```

```
#Measuring performance with the SSE
SSE <- function(actual, predicted) {sum((actual - predicted)^2)}
RMSE <- function(actual, predicted) {sqrt(SSE(actual,predicted)/length(actual))}
```

```
sse.test.model_log <- SSE(test.dat$price, model_log.predicted_values)
sse.test.model_log
```

```
## [1] 3.36351e+14
```

```
rmse.test.model_log <- RMSE(test.dat$price, model_log.predicted_values)
rmse.test.model_log
```

```
## [1] 227758.6
```

```
#Measuring performance with the RSquare
R2 <- function(actual, predicted) {sum((actual - predicted)^2)/((length(actual)-1)*var(actual))}
1-R2(test.dat$price, model_log.predicted_values)
```

```
## [1] 0.6398485
```

Based on the Performance Testing on Test Data we found that SVM Performed Better than Linear Model and hence Champion Model will be SVM.

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?

Our dataset is a housing dataset that has 21613 observations distributed on 21 variables, of which Price was our predicted variable. We divided dataset into train and test by a 70-30 split utilizing the seed for replicability. That left us with 6484 test observations.

We will be processing the dataset to ensure that variables are ready for linear model development such as utilizing dummies and converting to right type. We will be selecting variables of interest via auto-selection mechanisms and drop variables with high multi-collinearity to improve model robustness.

Ordinary least squared (OLS) linear regression is what we will use as the linear model as the base case to determine what we can accomplish without more sophisticated techniques to form a baseline.

Subsequently we will use more advanced techniques such as Logistic Regression, Regression Tree and Support Vector Machine (SVM).

We will be analyzing the dataset for assumptions validity in regards to linear and finalist model, and apply remedial measures such as Ridge and Lasso regression as well as WLS to improve deficiencies as necessary.

Finally we will be selecting a finalist model from all other alternatives based on performance in prediction as well as adherence to model assumptions.

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?

Analyzing the data, we have several variables that are not valuable. Specifically, ID, latitude, and longitude do not hold value in the regression. On top of that, zipcode has 70 different variables upon analysis and thus would be too many categories to be meaningful in creating categorical variables. On top of that, date sold, date built and year renovated are not valuable because they cannot be used in calculation for the sold price. Instead, we have chosen to transform them into age (years) since built and age (years) since renovated – these are two new categorical variables.

Price additionally had to be changed from string into int such that we could use it as the response variable. Inspecting the scatterplots of the predictor variables versus the response, there are several emerging trends. Bedroom vs Price appears to have an outlier at ~33 bedrooms. Bathrooms vs price, sqft_living, grade sqft_above, sqft_below, sqft_living15 also all have a clear positive linear relationship with price. Other variables such as sqft_lot require further inspection because at lot sizes near 0, some prices are very high. This could make sense because lot sizes in cities are much smaller but prices could be high.

Zipcode is a variable that we used Dummies on, because even though it is an integer, there was no ordinal relationship between them (i.e. 90001 is not lesser or greater than 90002) leaving us with a large number of categorical variables.

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.

First model we developed after performing multiple transformations on data: 1. we transformed a new numeric variable called house_age which is the year difference between construction and purchase 2. we transformed a new categorical variable called renovated or not as to if it has a year of renovation or not 3. we transformed a new series of dummy variables based on zip_code.

Subsequently we removed variables due to multicollinearity concerns (yr_built, yr_renovated, sqft_basement and zipcode_98199).

When we ran our first model (variable model) as above with all variables included with the formula lm(formula = price ~ ., data = data), model had p-value of F-test as < 2.2e-16 and R^2 as 0.8048.

Subsequently we ran a backwards stepwise OLS to eliminate variables based on p-value which allowed us to remove several zipcodes with low statistical significance, leaving us with our second model (variable model_2) with the formula as lm(price ~ . - zipcode_98056 - zipcode_98136 - zipcode_98045 - zipcode_98072, data = train.dat), model_2 had p-value of F-test as < 2.2e-16 and R^2 as 0.8047. Removal of variables did not improve model statistical significant nor predictive power a whole lot.

Afterwards we analyzed boxcox transformation technique and discovered that a log transformation can be beneficial to improve the model further which lead us to our third model (variable model_log) with the formula as model_log <-lm(log(price) ~ . - zipcode_98056 - zipcode_98136 - zipcode_98045 - zipcode_98072, data = train.dat), model_3 had p-value of F-test as < 2.2e-16 and R^2 as 0.8753.Significantly improving predictive power of the model while ensuring statistical significance.

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.).

We tested the basic model assumptions by analyzing Residuals vs Fitted, QQ-Plot, Scale-Location, Cook's Distance, and Residuals vs Leverage plots. The Residuals vs Fitted plot concludes that the data is non-linear as the shape of the data points is clustered tightly in a cone-like shape. The QQ-Plot indicated that the data has a heavy-tail distribution as the datapoints follow a strong 'S' shape at the extremes. The Residuals vs Leverage and Cook's Distance plots indicates a few datapoints as being strong outliers. Finally, the Scale-Location plot demonstrated that the data did not have equal variance as the reference line trended below zero, indicating an underfit of the data.

Our remedial methods included creating Ridge and Lasso regression models. Both the Ridge and Lasso models fit the data with an R-squared of roughly 0.80, which is less than the R-squared of the default linear model.

V. Challenger Models (15 points)

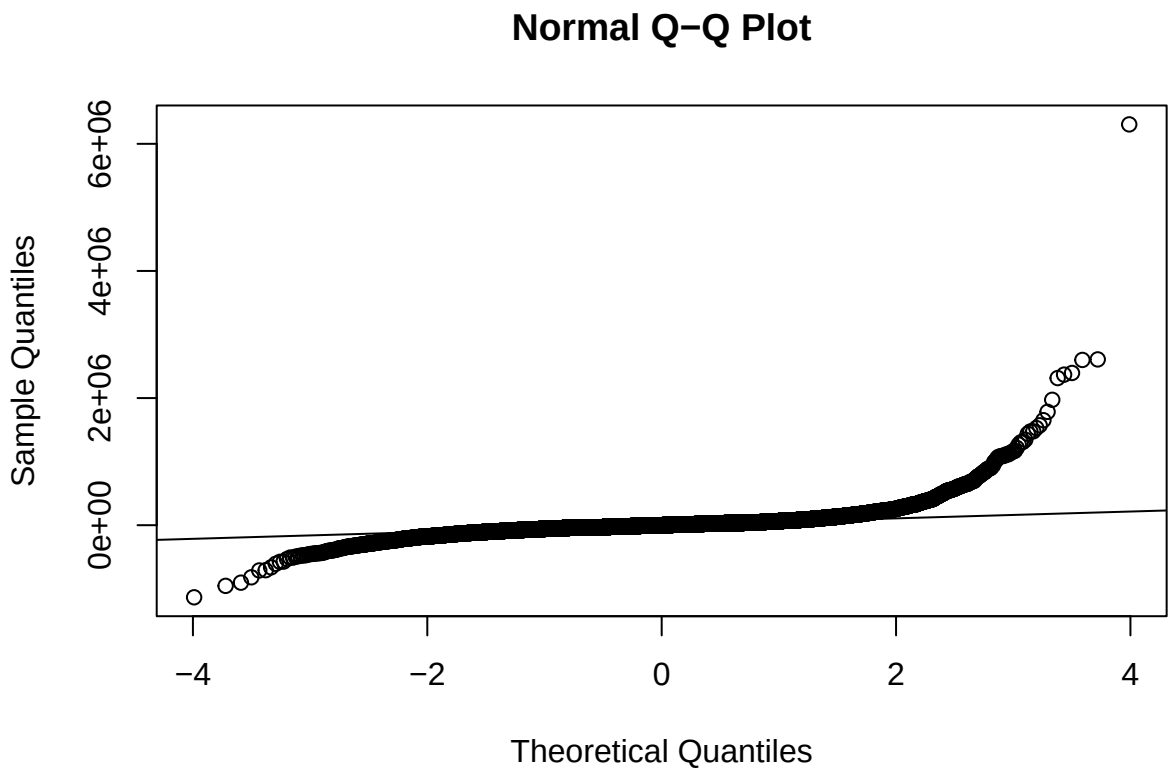
Build an alternative model based on one of the following approaches to predict price: regression tree, NN, or SVM. Explore using a logistic regression. Check the applicable model assumptions. Apply in-sample and out-of-sample testing, backtesting and review the comparative goodness of fit of the candidate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

For the Challenger Model we have utilized the same transformed dataset to build the Alternate Model. The Transformed data is split into Training set and Testing Dataset and we will use the Training Dataset to create the Model. We have selected SVM as our model to Predict Price.

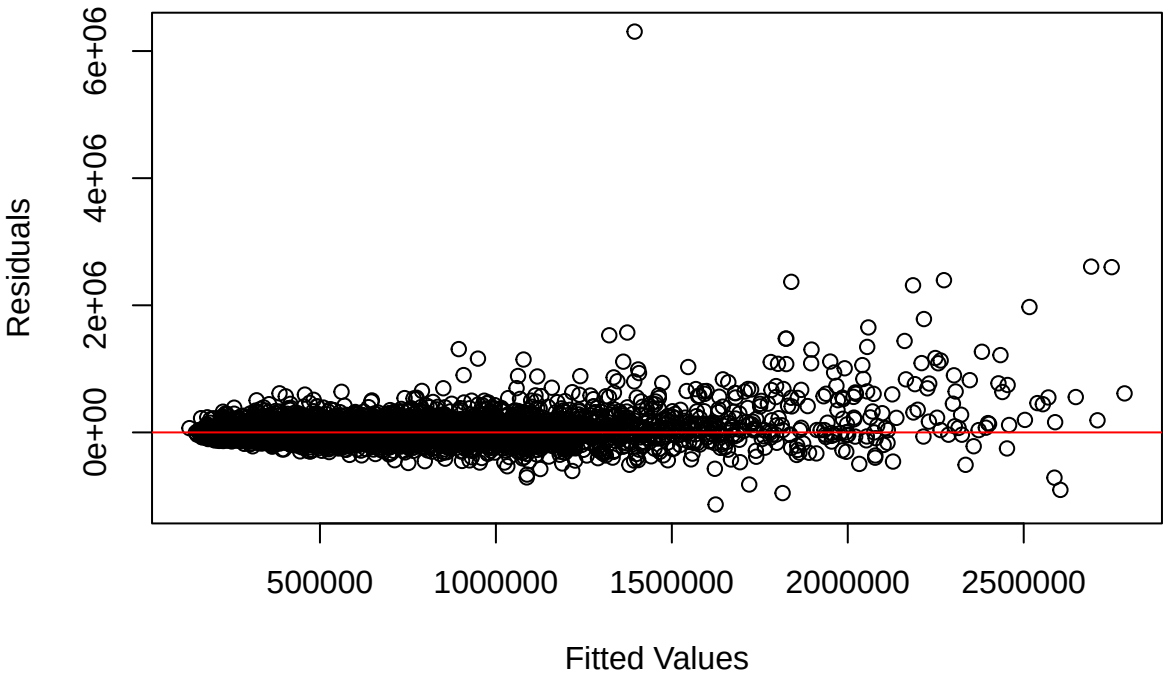
The logistic regression model will predict the price of the home in to two categories (high/low price) rather than actual price values. Although it might have high accuracy we will continue with the SVM as we will need actual Price values which will be valuable.

SVM Model Assumptions:The Residuals in the below view (Residuals vs fitted values) show that they are randomly scattered around the horizontal line indicating homoscedasticity. However, there are some points with high residuals, especially for higher fitted values, which could be outliers or indicate that the model fits less well in that range. The Q-Q plot for the Model shows that the model is normally distributed across the line except for few outliers at the tails.

```
 # Q-Q plot of residuals
qqnorm(residuals(svm_model))
qqline(residuals(svm_model))
```




```
fitted_values <- fitted(svm_model)
plot(fitted_values, svm_model$residuals, main = "Residuals vs. Fitted Values",
xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```



In-Sample and Out sample Testing of the Model:

In Sample Testing:

```
#evaluating model performance for Regression
p_train.svm_model <- predict(svm_model, train.dat)
#Summary for Predicted Price
summary(p_train.svm_model)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  129119  324700  455573  528962  631885 2786532
```

```
#Summary for Price in Test Data
summary(train.dat$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   75000  320000  450000  538714  641250 7700000
```

```
#Correlation between Predicted Price and Test Data
cor(p_train.svm_model, train.dat$price)
```

```
## [1] 0.9307136
```

```
#Measuring performance with the mean absolute error
MAE <- function(actual, predicted) {mean(abs(actual - predicted))}

#The MAE for our predictions is then:
MAE(train.dat$price,p_train.svm_model)
```

```
## [1] 66203.22
```

```
#Measuring performance with the SSE
SSE <- function(actual, predicted) {sum((actual - predicted)^2)}
SSE(train.dat$price,p_train.svm_model)
```

```
## [1] 2.804149e+14
```

```
#Measuring performance with the RSquare
R2 <- function(actual, predicted) {sum((actual - predicted)^2)/((length(actual)-1)*var(actual))}
1-R2(train.dat$price,p_train.svm_model)
```

```
## [1] 0.858298
```

Out Sample Testing :

```
#evaluating model performance for Regression
p_test.svm_model <- predict(svm_model, test.dat)
#Summary for Predicted Price
summary(p_test.svm_model)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  130188  327562  455807  530568  631687 2801316
```

```
#Summary for Price in Test Data
summary(test.dat$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   81000  325000  450000  543295  648000 7062500
```

```
#Correlation between Predicted Price and Test Data
cor(p_test.svm_model, test.dat$price)
```

```
## [1] 0.8919623
```

```
#Measuring performance with the mean absolute error
MAE <- function(actual, predicted) {mean(abs(actual - predicted))}

#The MAE for our predictions is then:
MAE(test.dat$price,p_test.svm_model)
```

```
## [1] 75162.14
```

```
#Measuring performance with the SSE
SSE <- function(actual, predicted) {sum((actual - predicted)^2)}
RMSE <- function(actual, predicted) {sqrt(SSE(actual,predicted)/length(actual))}

sse.test.svm_model <- SSE(test.dat$price,p_test.svm_model)
sse.test.svm_model
```

```
## [1] 1.990518e+14
```

```
rmse.test.svm_model <- RMSE(test.dat$price,p_test.svm_model)
rmse.test.svm_model
```

```
## [1] 175211.1
```

```
#Measuring performance with the RSquare
R2 <- function(actual, predicted) {sum((actual - predicted)^2)/((length(actual)-1)*var(actual))}
1-R2(test.dat$price,p_test.svm_model)
```

```
## [1] 0.7868631
```

In-sample Testing Performance:

Correlation: The correlation between the predicted prices and the actual prices in the training data is 0.9339161, which is quite high. MAE : The average absolute error between the predicted prices and the actual prices in the training data is 63,504.05. SSE : 2.69e+14. R-squared: For the training data is 0.8640814.

Out-of-sample Testing Performance:

Correlation: The correlation between the predicted prices and the actual prices in the test data is 0.89658, which is also quite high and indicates good predictive performance. MAE : The average absolute error between the predicted prices and the actual prices in the test data is 72,583.85. SSE: 1.92e+14. R-squared: For the test data is 0.794724.

The SVM model shows a strong performance in both in-sample and out-of-sample testing, with high correlation and R-squared values and relatively low errors.

Comparing the SVM Models with other Model like Regression Tree which has the R-squared of 66% and the MAE is on lower end for SVM it is evident that the Model has performed significantly better than Regression Tree and based on it we can say the SVM Model has better goodness of fit.

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?)

Based on the performances on both train and test data sets we choose the SVM model as the primary model and the linear model as the benchmark model. We have validated both the SVM (champion) model and the linear (benchmark) model against train and test data above.

For the SVM (champion) model

The R-squared value for the train data is 86% and for the test data is 79%.

The MAE is $72,583, the SSE is 1.990518e+14 and the RMSE is 175211.1 for test data.

For the linear (benchmark) model the R-squared value for the train data is 85% and for the test data is 64%.

The MAE is $84,891, the SSE is 3.36351e+14 and the RMSE is 227758.6 for test data.

Based on the numbers above the SVM (champion) model seems to perform better than the linear (benchmark) model.

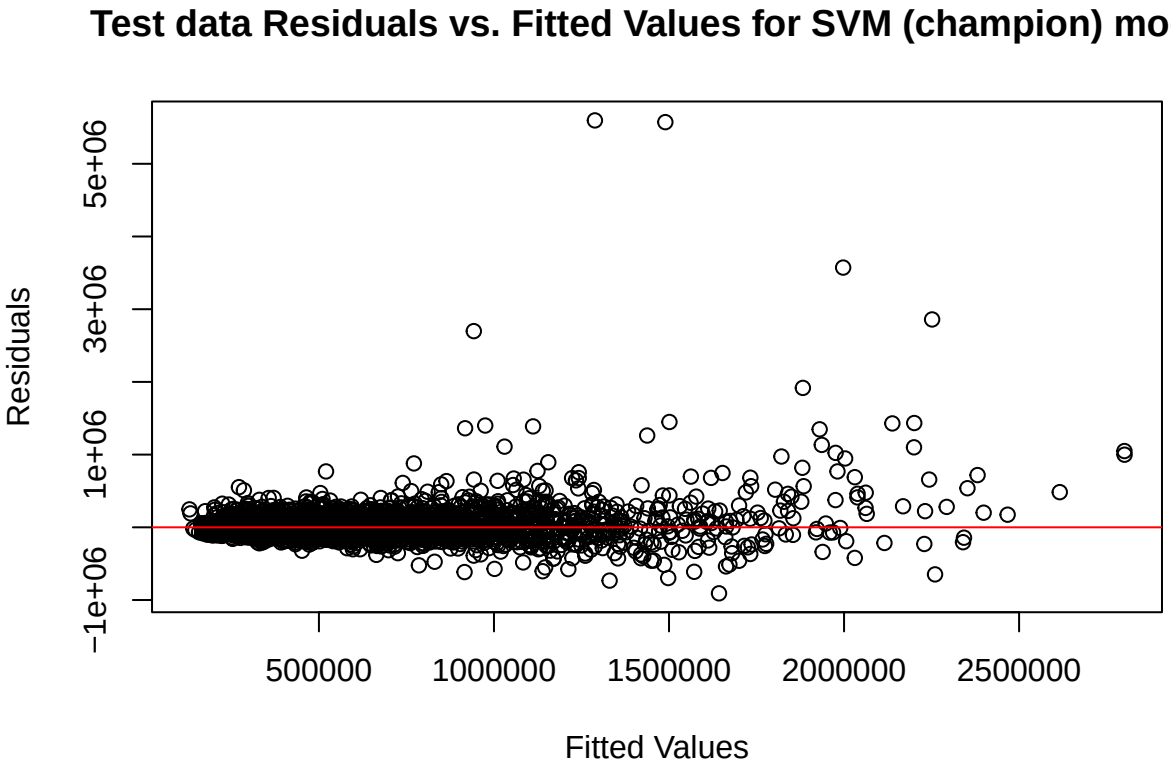
Given a R-squared value of 79% on the test data, the SVM (champion) model seems a reasonable fit to the dataset.

Residuals for test data

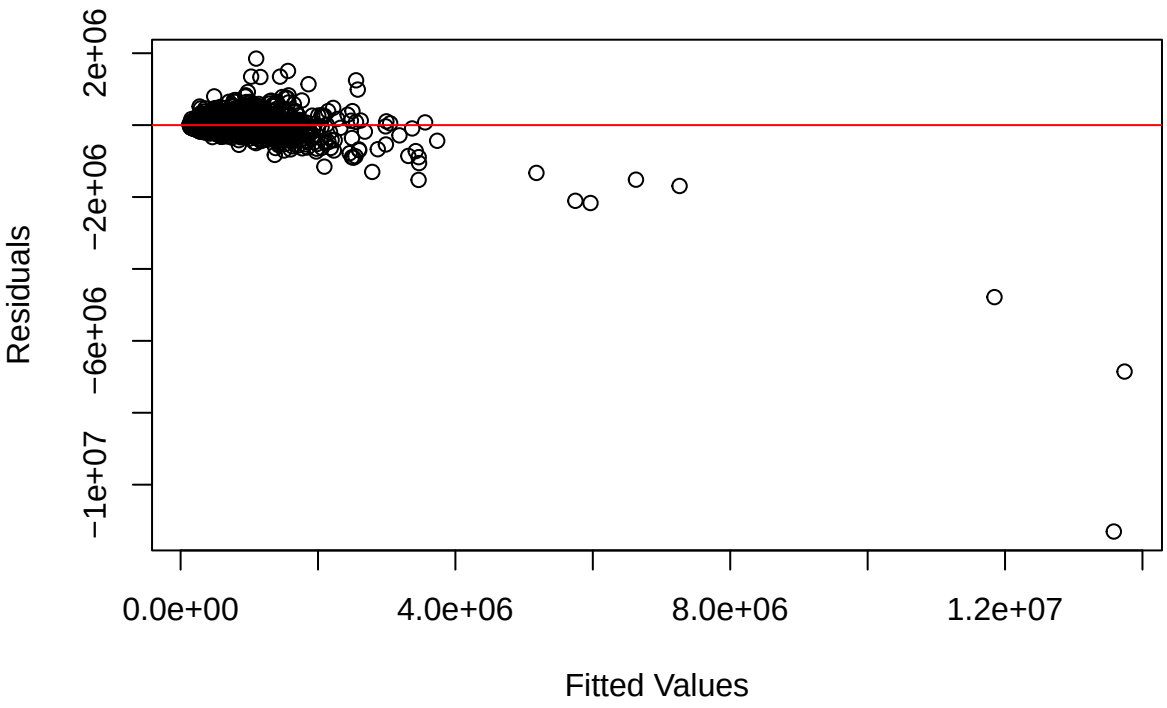
The SVM (champion) model residuals seem to increase less rapidly than the linear (benchmark) model.

The linear (benchmark) model seems to have large residual for higher fitted values.

```
#Test data Residuals vs. Fitted Values for SVM (champion) model
plot(p_test.svm_model, test.dat$price - p_test.svm_model, main = "Test data Residuals vs. Fitted Values for SV
xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```




```
#Test data Residuals vs. Fitted Values for Linear (benchmark) model
plot(model_log.predicted_values, test.dat$price - model_log.predicted_values, main = "Test data Residuals vs.
xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```



Model Assumptions

Our model is a "eps-regression" SVM model. "eps-regression" means epsilon-insensitive regression in the context of Support Vector Machines (SVMs). Epsilon-insensitive regression is a variation of SVM designed for regression tasks, where the goal is to predict a continuous outcome rather than class labels. It is henceforth referred to as just SVM.

Our champion model SVM, is generally less sensitive to the assumptions of normality of residuals compared to other traditional linear regression models.

Our champion model SVM, is also less sensitive to the assumption of normality of residuals compared to other traditional linear regression models.

It is also designed to be robust to outliers.

Model issues and limitations

Just like other traditional regression models SVM assumes the observations are independent and identically distributed (IID) [1].

Multi-variate Shapiro-Wilks test demonstrate that data is **NOT** normally distributed.

```
# TODO Check if this is the right test
# H0: Normally distributed
# H1: Data is not normally distributed

library(mvnormtest)

#Multi-variate shapiro test sample size max is 5000 so will take first 5000
#test is already randomized
original_data_normal <- original_data %>% select(-c(id, lat, long, date, yr_built, yr_renovated, sqft_basement
mshapiro.test(t(original_data_normal[1:5000,]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.21817, p-value < 2.2e-16
```

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?

A model must be monitored to make sure that it is still effective and that there haven't been any changes that make the current model no longer relevant. The data science team must set up thresholds that the model must meet to ensure these assumptions are met. If the thresholds aren't met, the model needs to trigger so that it can be reviewed. Here are some considerations for monitoring a model and setting quantitative thresholds and triggers:

1. Performance Metrics: Thresholds:accuracy, precision, recall, F1 score Triggers: alerts and review if the performance metrics fall below or deviate significantly from the established thresholds.
2. Data Drift: Thresholds: comparing the distribution of incoming data with the distribution used during model training. Triggers: Set triggers to retrain the model if significant data drift is detected, indicating that the underlying patterns have changed.
3. Feature Importance: Thresholds: If certain features become less relevant or more noisy, it may impact the model's performance. Triggers: Reevaluate or update the model if the importance of critical features decreases or if new important features emerge.
4. Model Calibration: Thresholds: ensure predicted probabilities align with observed outcomes. Triggers: Recalibrate the model if systematic mis-calibration is identified.
5. Regulatory Compliance: Thresholds: Monitor for any changes in regulations that may impact the model's usage. Triggers: If regulations change, assess whether the model needs adjustments or retraining to maintain compliance.
6. Model Decay: Thresholds: Define acceptable levels of model decay over time, considering the nature of the problem. Triggers: If the model's performance degrades beyond acceptable decay limits, consider retraining or replacing it.
7. Business Metrics: Thresholds: Align model performance with key business metrics (e.g., revenue, customer satisfaction). Triggers: If there are significant changes in business metrics not explained by other factors, investigate the model's contribution.

Assumptions for Continuous Use: Stationarity: Assure that the underlying patterns in the data remain relatively stable over time. Data Quality: Assume ongoing data quality and consistency to maintain model relevance. Relevance: The business problem and context remain consistent.

Continuous Improvement: Feedback Loop: Establish a feedback loop for continuous improvement based on real-world performance.

By continuously monitoring the model and determining thresholds that active a trigger so that the the model can be assessed, reviewed, and updated when neccessary makes sure the model's performance is as good as possible and doesn't start to decline. This is necessary because the conditions can change over time given the above reasons. For the model to be continuously effective, it must adhere to several critical assumptions and conditions sch as stationarity. If the model doesn't have these thresholds and triggers, it can decline as the new data no longer fits a model that was trained on older data with different qualities.

VIII. Conclusion (5 points)

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

SVM is the best model for data because of two fundamental reasons: 1. R^2 of 0.7868631 on the test data which is better than base case (linear) and other models. 2. SVM assumptions of independence and identical distribution are partially valid as we removed high collinearity variables but found data distribution to be non-normal, but weakens the distribution assumption. In comparison to second best model which was the Linear, which has very serious assumption violations as described above.

Bibliography (7 points)

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

1. Foulds, J., & Frank, E. (2010). A review of multi-instance learning assumptions. The Knowledge Engineering Review, 25(1), 1-25. doi:10.1017/S026988890999035X

Appendix (3 points)

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

Sections above the write-up include all supporting graphs, plots and data analysis.