Raag Patel & Ranjan Karki

Professor Dr. Bivin Sadler

MSDS 6371: Statistical Foundations for Data Science

August 1, 2022

MSDS 6371 Final Project

The dataset given to us, is compiled of a little less than 1500 home sales in Ames, Iowa. The data set contains 81 variables, detailing important details about the houses sold in that area, including the value of the sale, the amenities, features, and physical properties of the house. In the first analysis question, we utilized the price of the sale, the neighborhood, and the square footage of the living area, to fully answer the question given to us from Century 21 Ames.

Our model building for the entry into the Kaggle competition started by utilizing all 80 variables (not including the ID). We selected using statistical measures certain columns that included categorical and continuous data to build a model that is indicative of the price of home sales in Ames, Iowa.

We highlight some column names throughout the report, such as GrLiveAreaPer100, which means the square footage of the living area, in increments of 100.

Analysis Question 1:

Century 21 Ames, a real estate company in Ames Iowa, has commissioned us to help them answer some questions they had about prices in certain neighborhoods. We were asked to use a dataset that contained about 1500 home sales and narrow it down to certain neighborhoods. Where we then should build a model that highlights the relation between sale price of a home and the square footage of the living area of the house, depending on the neighborhood the house is located in. We were asked to provide confidence intervals in our analysis, and a thorough statistical analysis of the question of interest.

We ran 4 different models which are linear model, log-linear, linear log, model with outliers addressed. The model with outliers addressed is the best fitting model.

Model with outliers addressed:

PredictedSalesPrice = ß0 + ß1 * (GrLivAreaPer100) + ß2 * (Edwards) + ß3 * (North Ames) + ß4 * (Edwards * GrLivAreaPer100) + ß5 * (North Ames * GrLivAreaPer100)

Normality: Looking at the histogram of residuals there is no evidence that the residuals do not follow normal distribution. Given the size of the sample, it is enough to follow the central limit theorem.

Linearity: Log transformed, and untransformed model residual plot looks random cloud providing enough evidence that linearity assumptions are met. It is obvious that transformation didn't help. So proceeded with untransformed data with outliers addressed. This evidence is backed by a QQ plot which shows most of the data are linearly related.

Equal Variance: From the residual plots most of the data points are scattered above and below the reference line indicating equal variance. However, there is a clustering effect seen. It is most likely most of the houses are in the same price ranges. Also, from the QQ plot the data points are normally distributed in relation with the x-axis.

Independence: We will assume the data points are independent for the purpose of the question. House prices are often related to each other, as certain areas are deemed more "wealthy". Since we cannot measure the impact of that, we can and will assume independence.

Outliers: There were few outliers that were addressed by using Cook's D-method. Any data points that were above 3 times the mean was removed. There were only 9 observations as outliers, when we removed these outliers adjusted R2 improved significantly.

(Please refer to figures 1.1 and 1.2 in the appendix, to highlight the before (1.1) and after (1.2) of the outliers being removed.)

The influential points were analyzed using Cook's D. Any data points that were 3 times the mean were considered outliers and a new model was fitted after removing these data points. This significantly improved Residual vs Leverage plot. After removing there were no data points that were high leverage low studentized residual. Please refer to the plot above.

Normality: Judging from the histogram of the residuals, with the outliers removed, there is no evidence that residuals do not follow normal distribution.

Linearity: The residual plot looks randomly distributed and there is sufficient evidence for linearity.

Equal variance: QQ plot - that data points are normally distributed in relation with x-axis, sufficient evidence of equal variance.

Independence: We will assume the data is independent of each other, by ignoring the clustering effect.

Model	Adj R2
Original	0.44
Log-Log	0.5056
Linear-Log	0.4587
Outliers Addressed	0.5226

Setting Brookside as reference;
PredictedSalesPrice = ß0 + ß1 * (GrLivAreaPer100) + ß2 * (Edwards) + ß3 * (North Ames) + ß4 * (Edwards * GrLivAreaPer100) + ß5 * (North Ames * GrLivAreaPer100)

Fitted Model;
PredictedSalesPrice = 22396.3 + 8416.6* (GrLivAreaPer100) + 35252.2* (Edwards) + 55475.3 * (North Ames) - -3235.5 * (Edwards * GrLivAreaPer100) -3247.7 * (North Ames * GrLivAreaPer100)

Three regression equations:

Predicted (SalesPrice|Brookside) = 22396.3 + 8416.6 * (GrLivAreaPer100)
Predicted (SalesPrice|Edwards) = 57648.5 + 5181.1 * (GrLivAreaPer100)
Predicted (SalesPrice|North Ames) = 77871.6 + 5168.9* (GrLivAreaPer100)

To answer the question about how the square footage of the living area relates to the sale price of the home, depending on the neighborhood, we can refer to the regression equations we provided earlier.

For homes in Brookside, we estimate that for every 100 feet of square footage added, the price of houses sold increased by $8,416.60, from a base of $22,396.30.

For homes in Edwards, we estimate that for every 100 feet of square footage added, the price of houses sold increased by $5,181.10 from a base of $57,648.50.

For homes sold in North Ames, we estimate that for every 100 feet of square footage added, the price of houses sold increased by $5,168.90 from a base of $77,871.60.

From the above analysis, client Century 21 targets the average sales price in relation to living area for the three neighborhoods. Removing outliers helped us to narrow down the average living space with average sales price, because there were the houses with more than 4000 square feet for less than $170,000 which is a possibility of foreclosure. Whereas some houses with a smaller than 1700 square foot living area were more than $300,000.

R Shiny: Price v. Living Area Chart

Link to RShiny App. *(Please refer to the appendix if the hyperlink does not work).* The app is useful to show scatterplots of the sale price of the home vs the square footage of the home, including the outliers we had removed to build the model. There are 3 plots for the 3 neighborhoods Century 21 Ames asked us to look at regarding their model. The sale price of the homes is not adjusted, as the model utilized the standard data as well.

Analysis Question 2:

We were also asked to create the most predictive model we can for the sales price of Ames Iowa. We are limited by the technical knowledge we have, when it comes to building models like these, but we are more than confident we can generate a sufficient model. We are building 4 models, using 4 techniques we will discuss in detail, one of them being a custom model that we will discuss in detail.

We transformed the data, without touching the data too much. The first time we went through this dataset we edited many things; got rid of columns we deemed unnecessary via statistical methods, and rows that had too many NA's for us to get anything useful. Our models were, frankly, useless and uninterpretable. The second time we went around the dataframe, we tried to keep the data as true as possible. By hand, we went through each and every column and transformed it into a numerical factor. Which treats the column as a factor, and assigns numbers to the levels, so that we can mathematically interpret what the data means. We also made sure we did the same things to both of the given datasets, the data with 1460 rows and 81 columns, including Sale Price, and the data with 1459 rows and 80 columns, not having the Sale Price. To address the NA's this time around, instead of deleting them and missing crucial information, we decided that it would make sense for us to input 0's in the categorical columns, because it would be safe to assume that if the data is not there, it means it's missing from the home. We also looked at the NA's for the continuous data columns, and found that there were almost none, or at least a negligible amount. When we built the models however, and outputted the predicted data, we ran into some NA's there. We believe that they were attributed to some of the columns that had NA values, but we believed it was such an inconsequential amount, that inputting the mean Sale Price into those data points would not hamper the results too much.

The first model *(Please refer to figure 2.1 in the appendix)* we built had everything in it. This model did not exclude a single column no matter the significance of the column. This was

our baseline model, and we had to compare it to everything else. The model yielded solid results. The Adj R^2 value came in at about .88. That number may be inflated due to the gross overfitting, but it is surprising how high that value is since all 80 columns are thrown into the model. As we can see from the residual plots, and the QQ plot, there are a lot of outliers in the data. It's something that we are keenly aware of, but since this is just an exploratory model, we decided to keep it in.

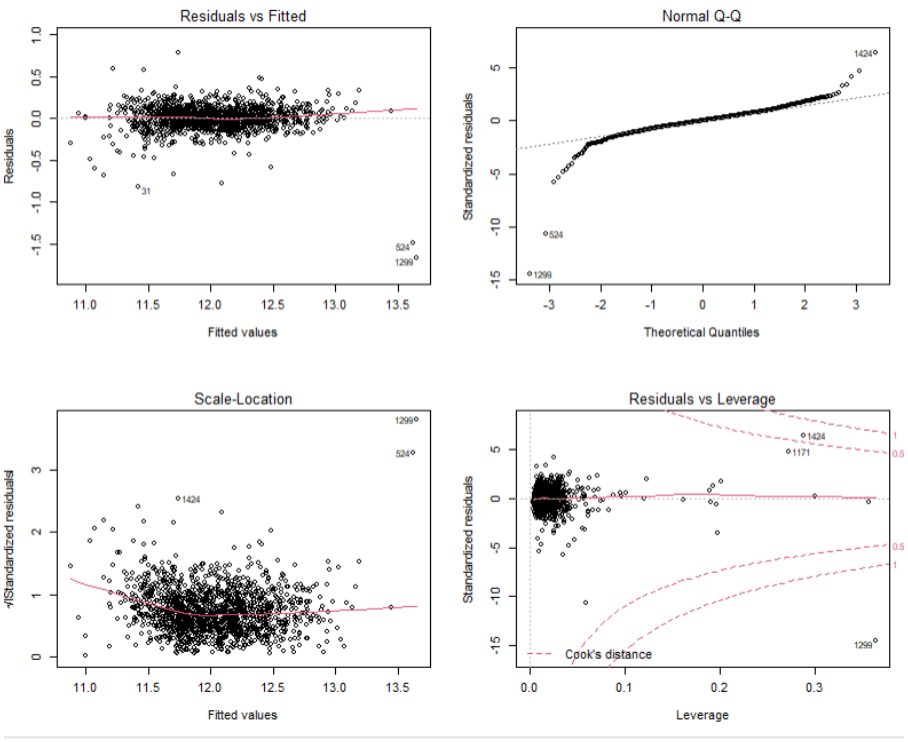
The stepwise model *(Please refer to figure 2.2 in the appendix)* is built using a function in the "olsrr" package. That package has a pre-built function for this and the next 2 models, which were incredibly useful. We got a really high Adj R^2, at .8817. This model was built by the function where it went through and added variables with a p value between .01 and .02. We chose extremely specific and low p values, because we wanted to test how accurate the model gets when we restrict the variables it can select. The CV value is .0243, which tells us that there is little variation from the mean.

The forward selection model *(Please refer to figure 2.3 in the appendix)* works similarly to the stepwise model, but instead this time it goes until it reaches a p value that it cannot add to the model, and starts from the first variable it encounters. This alongside backwards selection is not the greatest tool we can use to build a model, but it does provide some insightful results. The models we have talked about so far all had pretty strong outliers. We explain why we decide to keep them in when we address the assumptions of our custom model. This model has an Adj R^2 value of .8817, which is identical to the stepwise model.

The backwards model *(Please refer to figure 2.4 in the appendix)* does the same process as the forwards model, but this time starting from the end of the dataframe. Both of these methods together create the strength that is the stepwise model, so it is expected that these models are slightly inferior to the stepwise model. This model has an incredibly similar Adj R^2 of .8835, but has 6 more parameters than the forward model. That can mean that the data is overfitted, and does not entail that those extra parameters are necessary.

Our custom model was hand built, by looking only at the significant p values from the full model. We went through and manually added parameters that were statistically significant at the .01 level, and then went through that model to add parameters that were significant on the .005 level. This extremely rigorous selection progress created our best kaggle score output.

The residual plots for our chosen custom model show some conflicting messages. We did see the outliers, but we decided to keep them there, because some of the outliers held crucial information about the coefficients about certain regressors at certain factor levels.



The influential points were checked using Cook's D method. There were some data points that had high leverage and low studentized residual. But removing these outliers data points gave us high adjusted R2 and high CV press as well. So to keep CV press low we decided to put the outliers to get better inference in the final custom model. Getting rid of outliers impacted the accuracy of our model because it got rid of rare categorical information.

Normality : Judging from the histogram of the residuals ,there is no evidence that the residuals do not follow normal distribution. 95% of the data points are scattered randomly, and spread out enough to deem normality.

Linearity: The residual plot looks randomly distributed and there is sufficient evidence for linearity. 95% of the data points are spread and scattered enough to assume linearity.

Equal Variance: From the QQ plot the data points look normally distributed in relation to x-axis . Since more than 95% data points fall in this line, suggesting normal distribution.

Independence: We will assume the data is independent of each other, by taking account of clustering effect. Because the data points fall under the same price range.

Predicted Model	Adjusted R2	CV Press	Kaggle Score
Stepwise	0.88172	.02433418	0.16792
Forward	0.88172	.02433418	0.16792
Backward	0.883572	.0292174	0.16792
Custom	0.86978	.02488314	0.16489

(Please refer to the appendix for the kaggle team name, for verification)

Our custom model had the best kaggle score, at a value of .16489. Interestingly it had the "worst" Adj R^2 value, but we believe the other models are inflated due to the overfitting of them. We believe that in this case, in the case of the comparison between these models, that a lower Adj R^2 is better.

We believe that we have created a strong model, which does a good job at predicting the sales price of homes being sold in Ames, Iowa. We created a model that kaggle rated a .16489, which was our strongest showing. This model included these variables:

MSZoning LotArea Street LotShape BldgType BsmtCond BsmtFinSF1 OverallQual YearBuilt YearRemodAdd RoofMatl BsmtFinType1 BsmtFinSF2 HeatingQC CentralAir X1stFlrSF X2ndFlrSF BsmtFullBath KitchenQual TotRmsAbvGrd Functional Fireplaces GarageCars ScreenPorch PoolQC SaleCondition

This model was built by manually editing the class of the columns, and then manually adding columns that met our criteria of an extremely low p value. This is the best model with our technical knowledge capabilities that we can create.

APPENDIX

Figure 1.1

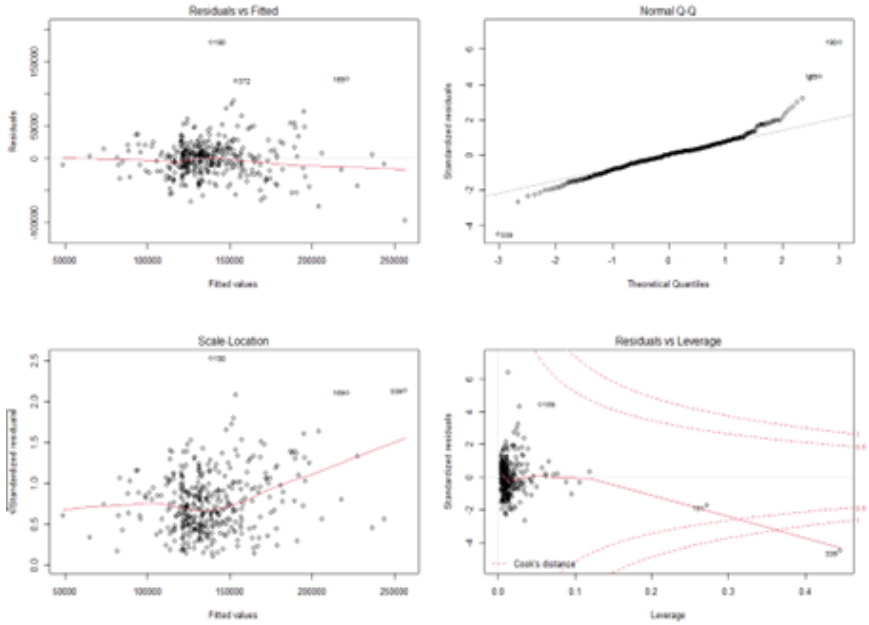


Figure 1.2

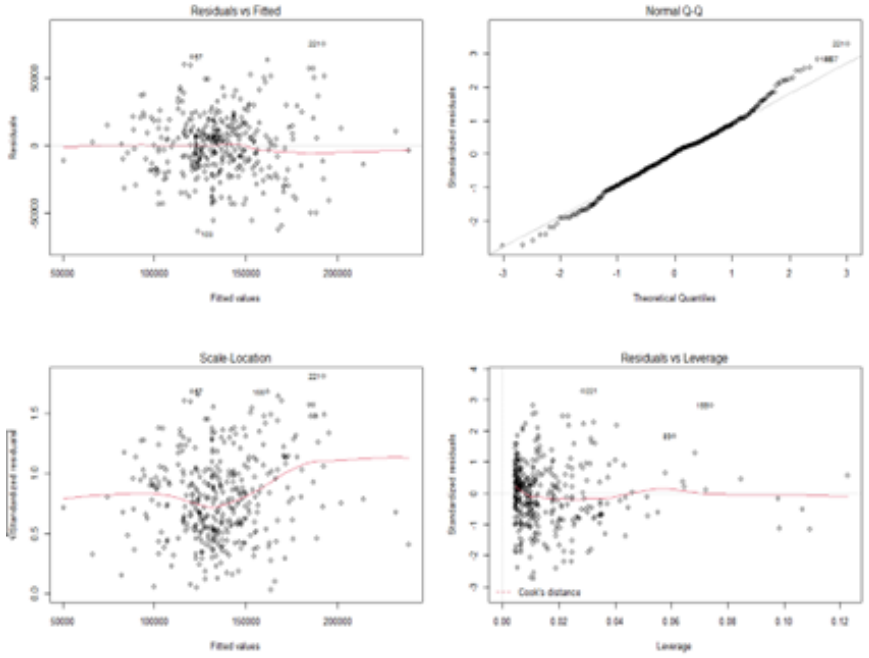


Figure 2.1

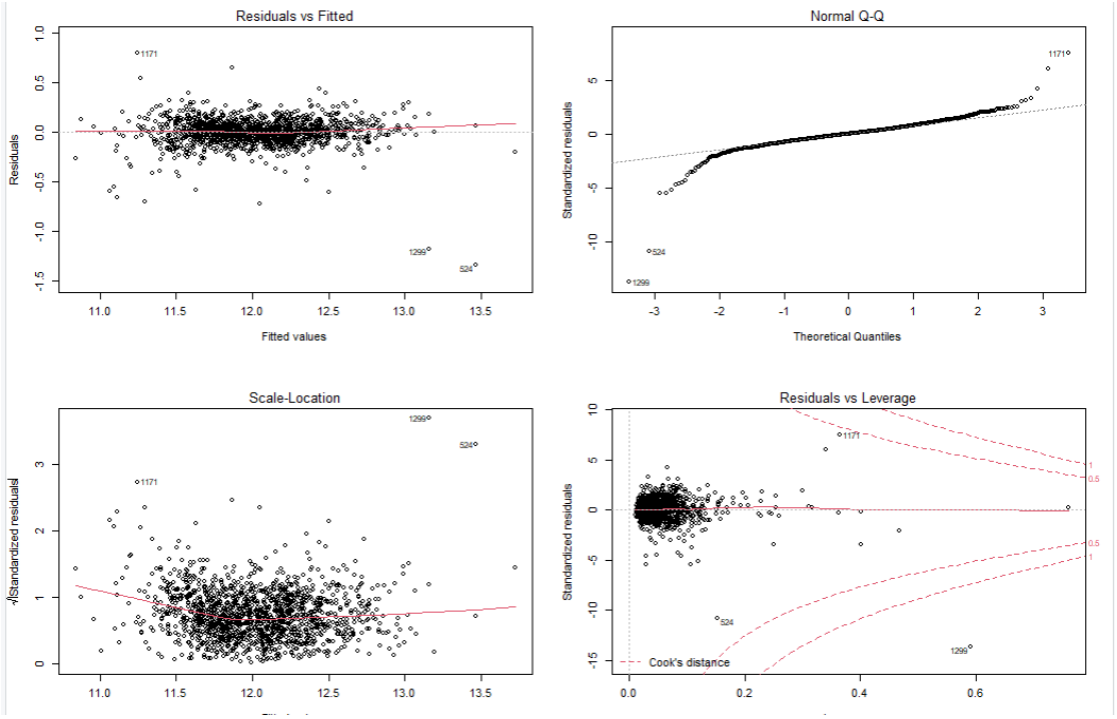


Figure 2.2

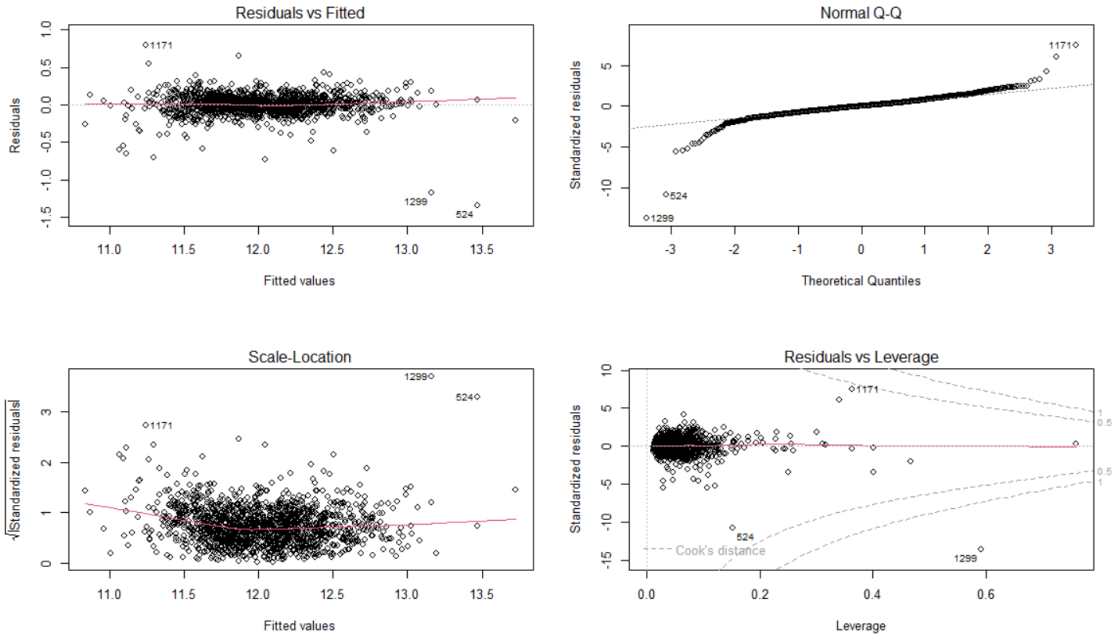


Figure 2.3

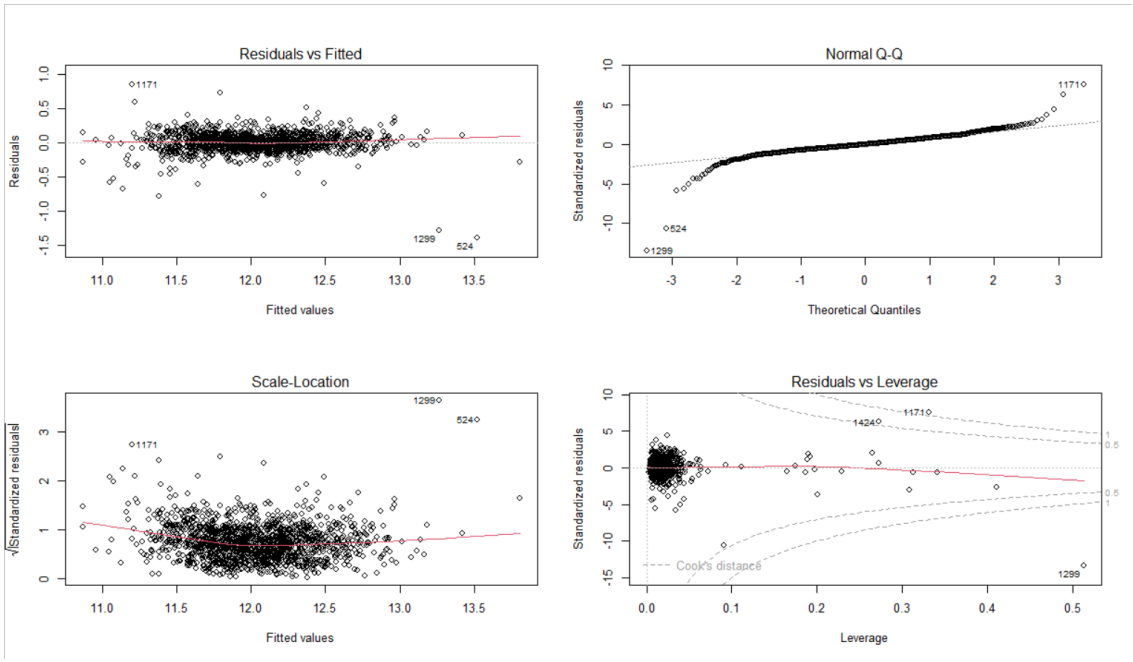
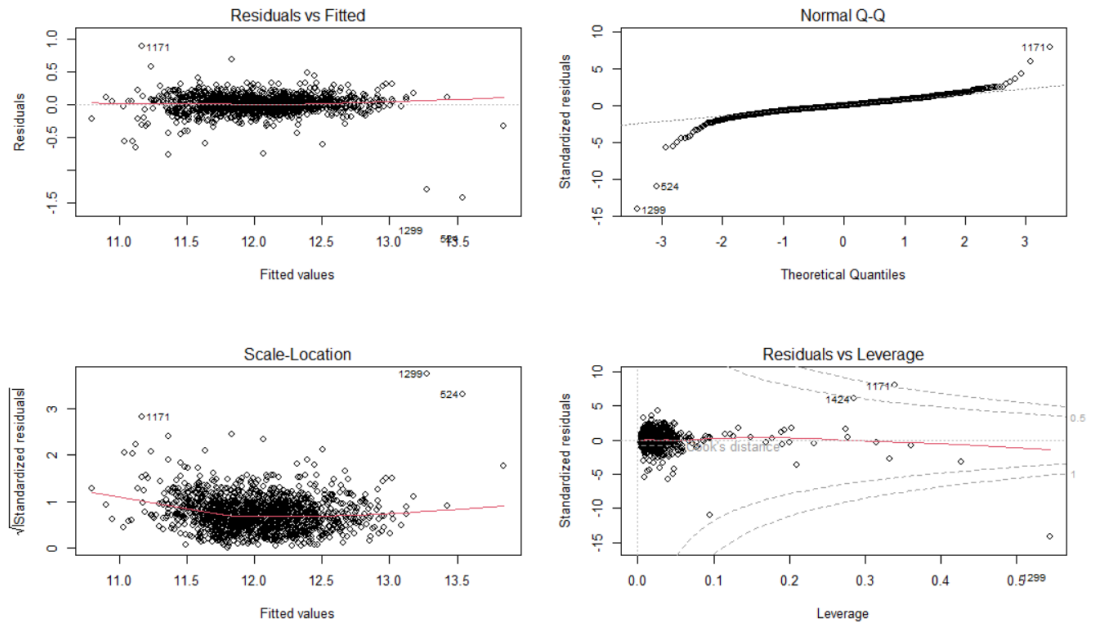


Figure 2.4



R Shiny Link: https://raagpatel1.shinyapps.io/RShiny/

Kaggle Team Name: DS6371RR

RStudio Code:

We kept our mistakes in the code, it is commented out. Both so that we can learn from our mistakes, but also highlight where we learned more about data cleaning and model building. Also, since this is a lot of code, here is the github file: https://github.com/raagpatel1/MSDS_6371_Stat_Foundations/blob/master/FinalProject/6371_ProjectRCode_Raag.R

```
library(tidyverse)
library(dplyr)
library(car)
library(readr)
library(readxl)
library(knitr)

library(ggplot2)
library(ggpubr)
library(ggcorrplot)
library(ggthemes)

library(stringr)
library(corrplot)
library(corrr)
library(multcomp)
library(pairwiseCI)
library(effectsize)
library(investr)
library(broom)
library(olsrr)
library(forecast)


setwd("D:/School/GitHub/MSDS_6371_Stat_Foundations/FinalProject")

FullData = read.csv("Data/train.csv", na.strings = "NA", strip.white = T)

Clean_FullData = FullData

Submission_TestData  = read.csv("Data/test.csv", na.strings = "NA", strip.white = T)

######################################## Data Cleaning

summary(Clean_FullData)

# Test if na.strings worked
# There are 81 NA's in the column BM "GarageCond" according to Excel

sum(is.na(Clean_FullData$GarageCond))

# R counted 81 na's, therefore it worked.

# This is a loop that will go through the columns and decide
# whether to delete the column based off of the amount of na's. Then it will
# go through each row and eleminate the row if it has more than 1 na.

yCol = 0
yRow = 0
```

```
for (i in 1:length(Clean_FullData)){
  x = sum(is.na(Clean_FullData[,i])) / length(Clean_FullData[,i])
  if (x > 0.3150685){ # 1460 - (.3150685 * 1460) = 1000
    yCol = append(yCol,i)
  }
}
yCol
length(yCol)
Clean_FullData = subset(Clean_FullData, select = -yCol)


for (i in 1:nrow(Clean_FullData)){
  x = as.numeric(rowSums(is.na(Clean_FullData[i,])))
  if (x > 0){
    yRow = append(yRow,i)
  }
}
yRow
length(yRow)
Clean_FullData = Clean_FullData[-yRow,]

# Now that the dataset is cleaned

rm(i,x,yCol,yRow)

summary(Clean_FullData)
dim(Clean_FullData)

# We can see that there are 76 variables, and 1094 rows, where each row has
# complete information on a house. This sample is more than plenty to configure
# a model.

# Need to assign certain columns to be categorical/continuous. So we will output
# the dataframe to a csv, and go through the columns, using the kaggle website
# to help us.

write.csv(Clean_FullData,"Data/Clean_FullData.csv", row.names = T)

# Have to do it manually:

Clean_FullData$Id = as.numeric(Clean_FullData$Id)

sapply(Clean_FullData, class)

Clean_FullData[,c(2,3,6:25,27:33,35,39:42,53,55,57:59,62:64,73:75)] <-
lapply(Clean_FullData[,c(2,3,6:25,27:33,35,39:42,53,55,57:59,62:64,73:75)], factor)

sapply(Clean_FullData, class)

knitr::opts_chunk$set(echo = TRUE)

source("./Functions.R")


#loading train data

hpdf<- read.csv("train.csv",header = TRUE, ",")
head(hpdf)
str(hpdf)
colSums(is.na(hpdf))  ## no NA value on required columns
```

```
#adding column name GrLivArea/100 in hpdf

hpdf["GrLivAreaper100"] <- hpdf$GrLivArea / 100
head(hpdf)




## merging didnt work. so proceeded with train data set only.

#loading testing data
## added missing salepricecolumn in R

#test <- read.csv("test.csv",header = TRUE, ",")

#head(test)
#str(test)
#colSums(is.na(test))


# merging both traning and test data set to get more number of observations

#hpdf <- merge(train,test,by="Id",all.x=TRUE, all.y = TRUE)
#head(hpdf)
#str(hpdf)



# changing Neighborhood column to factor


hpdf$Neighborhood<-as.factor(hpdf$Neighborhood)

str(hpdf$Neighborhood)



## filtering only the neighborhood of interest and assigning it to new data frame.

library(dplyr)

Cent21 = hpdf %>% filter(Neighborhood == "NAmes"|Neighborhood=="Edwards"|Neighborhood=="BrkSide" )
Cent21
head(Cent21)
count(Cent21) ## 383
colSums(is.na(Cent21))



##QOI sale price related to the square footage of the house in GrLIvArea?
##how the SalePrice of the house is related to the square footage of the living area of the house (GrLIvArea) and if the SalesPrice (and its relationship to square footage) depends on which neighborhood the house is located in.

##separate fit lines with interaction
```

```
sfit <- lm(SalePrice~GrLivAreaper100+Neighborhood+Neighborhood*GrLivAreaper100,data = Cent21)
summary(sfit)
confint(sfit)



##plotting separate fit lines for untransformed model

library(tidyverse)
ggplot(Cent21,aes(x= (GrLivAreaper100),y=SalePrice,colour = Neighborhood))+geom_point()+geom_smooth(method="lm")




##performing log-log model

slfit <- lm(log(SalePrice)~log(GrLivAreaper100)+Neighborhood+Neighborhood*log(GrLivAreaper100),data =Cent21)
summary(slfit)



## plotting separate fit lines for log -log model


ggplot(Cent21,aes(x=log( GrLivAreaper100),y=log(SalePrice),colour = Neighborhood))+geom_point()+geom_smooth(method="lm")




### performing linear-log model


slfit1 <- lm((SalePrice)~log(GrLivAreaper100)+Neighborhood+Neighborhood*log(GrLivAreaper100),data =Cent21)
summary(slfit1)



## plotting separate fit lines for linear -log model


ggplot(Cent21,aes(x=log( GrLivAreaper100),y=SalePrice,colour = Neighborhood))+geom_point()+geom_smooth(method="lm")



### performing log-linear model

slfit2 <- lm((log(SalePrice)~GrLivAreaper100+Neighborhood+Neighborhood*GrLivAreaper100),data = Cent21)
summary(slfit2)
```

```
## plotting separate fit lines for log-linear model


ggplot(Cent21,aes(x= GrLivAreaper100,y=log(SalePrice),colour = Neighborhood))+geom_point()+geom_smooth(method="lm")




## checking the assumptions of separate lines model

##Linearity for untransformed model


library(olsrr)
ols_plot_resid_fit(sfit)



#transformed model residual plot.

ols_plot_resid_fit(slfit) ## similar output
ols_plot_resid_fit(slfit1)## similar output
ols_plot_resid_fit(slfit2)## similar output


##Both transformed and regular model looks random cloud providing enough evidence that the linearity assumptions are met.It is obvious that trasnformation didnt help. So proceeded with untransformed data.Next we need to find potential outliers in the data set. Also we will proceed with caution that independence assumptions are met.

library(car)
outlierTest(sfit)



## these observations  are potential outliers . more than 3 times the mean.


Cent21[c("190"),] ## 1698 sq ft for 320 K -Neighborhood Edwards
Cent21[c("339"),] ## 5642 sq ft for 160 K -Neighborhood Edwards
Cent21[c("169"),] ##2704 sq ft for 345 K -Neighborhood NAmes
Cent21[c("372"),] ##  2201  sq ft for 274 K -Neighborhood Edwards
Cent21[c("131"),] ## 4676  sq ft for 184 K -Neighborhood Edwards
Cent21[c("48"),] #2158 243K
Cent21[c("180"),] #2360 for 129k
Cent21[c("205"),] #1576 223500




Cent21N <- Cent21[-c(339,131,190,169,372),] ## these are influential outliers
str(Cent21N$Id)



## creating a model by removing outliers  #Multiple R-squared:  0.4474,         Adjusted R-squared:   0.44

sfitN <- lm(SalePrice~GrLivAreaper100+Neighborhood+Neighborhood*GrLivAreaper100,data =Cent21N)
```

```
summary(sfitN)
```

removing two outliers gives us better R2 of 0.532 vs 0.4474.
#the 1698 sq ft house might be fully upgraded so the selling price (320K) could be higher, whereas the 5642 ,4676 sq ft house might be on foreclosure, so the selling price is only 160 K and 184 k repectively . But it gave us inference that neighborhood Edward is statistically insignificant to predict the sales price. SO procceded with model that has outlier

```
library(MASS)
studen_resi <- studres(sfit)
studen_resi
plot(studen_resi)
abline(0,0)
```

#Histogram of residuals for regular model

```
ols_plot_resid_hist(sfit)
```

Normality : From the histogram of the residuals there is no sufficient evidence that the residuals do not follow a normal distribution.

QQ plot of residual

```
resi <- resid(sfit)
resi
plot(fitted(sfitN),resi)
abline(0,0)

qqnorm(resi)
qqline(resi)
```

from QQ plot most of the data are linearily related . There are still some observations that are conerning that suggest evidence against normality. It looks like most the residuals are clustered along the line. So proceeded with assumptions the data set is normal

Equal variance

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

##From the residual plot , that the data points scattered randomly above and below the line 0 , however clusttering effect is seen but this could be due to majority of houses in same price range in all these neighborhood. So proceeded with the assumption of equal variance and this is backed by QQ plot - that data points are normally distributed in relation with x-axis but there are still some outliers that extremly low and extremly high sales price. But the more than 95 % of the data falls in the dash line so proceeded with assumption of equal variance.

removing the outliers gives us inference that the neighborhood Edward is statistically insignificant, So procceded with model that includes ouliiers

##Model
Setting Brkside as reference

#PredictedSalesPrice = ß0 + ß1 * (GrLivAreaper100) + ß2 * (Edwards) + ß3 * (NAmes) + ß4 * (Edwards * GrLivAreaper100) + ß5 * (NAmes * GrLivAreaper100)

Fitted Model

#PredictedSalesPrice = 19971.5 + 8716.3 * (GrLivAreaper100) + 68381.6 * (Edwards) + 54704.9 * (NAmes) - 5741.2 * (Edwards * GrLivAreaper100) -3284.7 * (NAmes * GrLivAreaper100)

Three regression equation:
Predicted(SalesPrice|Brkside) = 19971.5 + 8716.3 * (GrLivAreaper100)
Predicted(SalesPrice|Edwards) = 88353.1 + 2975.1 * (GrLivAreaper100)
Predicted(SalesPrice|NAmes) = 74676.4 + 5431.6 * (GrLivAreaper100)

Interpretations

The estimated average sale price for houses in the Brkside neighborhood is $19971.5 with no living area considered.
For every 100 unit in living area in Brkside neighborhood, the estimated sales price increases by $8716.3

The estimated average sale price for houses in the Edwards neighborhood is $88353.1 with no living area considered.
For every 100 unit in living area in Edwards neighborhood, the estimated sales price increases by $2975.1

The estimated average sale price for houses in the NAmes neighborhood is $74676.4 with no living area considered.
For every 100 unit in living area in NAmes neighborhood, the estimated sales price increases by $5431.6.

#> Confint(sfit)
Estimate   2.5 %   97.5 %
#(Intercept)   19971.514 -4314.212 44257.239
#GrLivAreaper100   8716.253 6792.850 10639.657
#NeighborhoodEdwards   68381.591 40913.670 95849.512
#NeighborhoodNAmes   54704.888 27408.383 82001.393
#GrLivAreaper100:NeighborhoodEdwards -5741.223 -7848.612 -3633.834
#GrLivAreaper100:NeighborhoodNAmes   -3284.667 -5411.269 -1158.065

Confidence interval
#95 % CI for Brkside is from ($-4314.212,$44257.239)
#95 % CI for Brkside and Edwards is from ($40913.670,$95849.512)
#95 % CI for Brkside and NAmes is from ($27408.383,$82001.393)

#dataf <- filter(Cent21,GrLivAreaper100 >=40) ?? Do you think we need to remove these as outlier to have statistically significant output for intercept. ## 524 $ 1299
#dataf

#dataf1 <- filter(Cent21,SalePrice >=300000) ## 643 $ 725
#dataf1

MeanSalePrice <-Cent21 %>% group_by(Neighborhood) %>% summarise(SalePrice_mean = mean(SalePrice))
MeanSalePrice
#after outliers removed
MeanSalePrice <-Cent21WO %>% group_by(Neighborhood) %>% summarise(SalePrice_mean = mean(SalePrice))
MeanSalePrice

```
# after outliers removed
MeanLivingArea <-Cent21 %>% group_by(Neighborhood) %>% summarise(Livigarea_mean = mean(GrLivArea))
MeanLivingArea

MeanSalePrice <-Cent21WO %>% group_by(Neighborhood) %>% summarise(SalePrice_mean = mean(SalePrice))
MeanSalePrice
```

#Conclusion :
THe intercept of the model provides the estimated average sales price for Brkside neighboorhood is 19971.5 when living area is not considered. And we are 95% confident that price is between ($-4314.21 to $44257.239) for BrkSide neighboorhood. For every 100 unit in living area in Brkside neighborhood, the estimated sales price increases by $8716.3

#For house without living area considered the average estimated sales price in Edwards neighborhood is increased by $68381.591 with 95 % confidence interval of ($40913.670,$95849.512) with respect to Brkside.
For every 100 unit in living area in Edwards neighborhood, the estimated sales price decreases by $5741.2 than BrKSide.

#For house without living area considered the average estimated sales price in NAmes neighborhood is increased by $54704.888 with 95 % confidence interval of ($27408.383,$82001.393) with respect to Brkside. For every 100 unit in living area in NAmes neighborhood, the estimated sales price decreases by $3284.7 than Brkside.

##observational study

outliers that are 3 times the mean

```
cooksD <- cooks.distance(sfit)
cooksD

influential <- cooksD[(cooksD >(3 * mean(cooksD,na.rm = TRUE)))]
influential
```

removing the influential data points

```
names_of_influential <- names(influential)
outliers <- Cent21[names_of_influential,]
Cent21WO <- Cent21 %>% anti_join(outliers)
```

fitting a model with outliers addressed and CI## Multiple R-squared: 0.529, Adjusted R-squared: 0.5226

```
sfitWO <-lm(SalePrice~GrLivAreaper100+Neighborhood+Neighborhood*GrLivAreaper100,data =Cent21WO)
summary(sfitWO)
confint(sfitWO)
```

Better R2 with outliers removed

```
# plots after outliers removed.

par(mfrow = c(2, 2))
plot(sfitWO)
ols_plot_resid_hist(sfitWO)
```

NOrmality : JUdging from the hsitogram of the residuals, with the ouliers removed , there is no evidence that residual do not follow normal distribution.
Linearity : The residual plot looks randomly distributed and there is sufficient evidence for linearity.
Equal variance : QQ plot - that data points are normally distributed in relation with x-axis, sufficient evidence of equal variance.
##independence: We will assume the data is independent to eachother, by ignoring the clustering effect.
Outliers: The outliers were addressed by follwing the cooks distance method. any values greater than 3 times the mean were excluded in order to get proper estimation of living area related to sales price.

##Model 2 without outliers

```
#Coefficients:
#                                Estimate Std. Error t value Pr(>|t|)
#(Intercept)                      22396.3    10171.0   2.202 0.028286 *
#GrLivAreaper100                   8416.6      809.5  10.397  < 2e-16 ***
#NeighborhoodEdwards              35252.2    13995.4   2.519 0.012197 *
#NeighborhoodNAmes                55475.3    11500.0   4.824 2.06e-06 ***
#GrLivAreaper100:NeighborhoodEdwards  -3235.5     1108.4  -2.919 0.003727 **
#GrLivAreaper100:NeighborhoodNAmes    -3247.7      900.7  -3.606 0.000354 ***
---
 #Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 #Residual standard error: 23430 on 368 degrees of freedom
 #Multiple R-squared:  0.529,     Adjusted R-squared:  0.5226
 #F-statistic: 82.68 on 5 and 368 DF,  p-value: < 2.2e-16
```

 ## Setting Brkside as reference

 #PredictedSalesPrice = ß0 + ß1 * (GrLivAreaper100) + ß2 * (Edwards) + ß3 * (NAmes) + ß4 * (Edwards * GrLivAreaper100) + ß5 * (NAmes * GrLivAreaper100)

Fitted Model

#PredictedSalesPrice = 22396.3 +   8416.6* (GrLivAreaper100) + 35252.2* (Edwards) + 55475.3 * (NAmes) - -3235.5 * (Edwards * GrLivAreaper100) -3247.7 * (NAmes * GrLivAreaper100)

Three regression equation:
Predicted(SalesPrice|Brkside) = 22396.3 + 8416.6 * (GrLivAreaper100)
Predicted(SalesPrice|Edwards) = 57648.5 + 5181.1 * (GrLivAreaper100)
Predicted(SalesPrice|NAmes) = 77871.6 + 5168.9* (GrLivAreaper100)

Interpretations

The estimated average sale price for houses in the Brkside neighborhood is $22396.35 with no living area considered.
For every 100 unit in living area in Brkside neighborhood, the estimated sales price increases by $8416.6

The estimated average sale price for houses in the Edwards neighborhood is $57648.5 with no living area considered.
For every 100 unit in living area in Edwards neighborhood, the estimated sales price increases by $5181.1

The estimated average sale price for houses in the NAmes neighborhood is $77871.6 with no living area considered.
For every 100 unit in living area in NAmes neighborhood, the estimated sales price increases by $5168.9.

#> confint(sfitWO)
2.5 %   97.5 %
  #(Intercept)   2395.768 42396.774
  #GrLivAreaper100   6824.769 10008.425
  #NeighborhoodEdwards   7731.154 62773.281
  #NeighborhoodNAmes   32861.295 78089.322
  #GrLivAreaper100:NeighborhoodEdwards -5415.079 -1055.857
  #GrLivAreaper100:NeighborhoodNAmes   -5018.911 -1476.437

  #Conclusion

  # THe intercept of the model provides the estimated average sales price for Brkside neighboorhood is $22396.3 when living area is not considered. And we are 95% confident that price is between ($2395.768, $42396.774) for BrkSide neighboorhood. For every 100 unit in living area in Brkside neighborhood, the estimated sales price increases by $8416.6

#For house without living area considered the average estimated sales price in Edwards neighborhood is increased by $57648.5 with 95 % confidence interval of ($ 7731.154,$62773.281) with respect to Brkside.
For every 100 unit in living area in Edwards neighborhood, the estimated sales price decreases by $3235.5 than BrKSide.

#For house without living area considered the average estimated sales price in NAmes neighborhood is increased by $54704.888 with 95 % confidence interval of ($32861.295,$78089.322) with respect to Brkside. For every 100 unit in living area in NAmes neighborhood, the estimated sales price decreases by $3247.7 than Brkside.

##observational study

Data Selection // Analysis Question 1

Setting the pattern allows us to easily select 3 neighborhood
#
pattern = c("NAmes","Edwards","BrkSide")
#
AQ1_Data = Clean_FullData %>% dplyr::select(Neighborhood,GrLivArea,SalePrice) %>%
  filter(grepl(paste(pattern, collapse="|"), Neighborhood)) %>%
  mutate(GrLivArea = round(GrLivArea / 100))
#
summary(AQ1_Data)
#
Plotting the data
#
ggplot(AQ1_Data, aes(x = GrLivArea, y = SalePrice)) + geom_point(aes(color = Neighborhood), alpha = .8) +
  geom_smooth(method='lm', se = F, color = 'black')
#
#
Model
#
AQ1Model = lm((SalePrice~GrLivArea+Neighborhood+Neighborhood*GrLivArea),data = AQ1_Data)
summary(AQ1Model)
#
par(mfrow = c(2,2))

plot(AQ1Model)

par(mfrow = c(1,1))
#
Looking at the plots, it's clear to see that there are points that are neccesary

```
# # to remove. Looking at Resids V Lev, we can see that are points that should be
# # looked at.
#
# # The ADJ r^2 value seems pretty low, at .4206, let see if removing the points mentioned
# # will increase it.
#

# Statistically removed outliers
cooksD <- cooks.distance(AQ1Model)
# cooksD

influential <- cooksD[(cooksD >(3 * mean(cooksD,na.rm = TRUE)))]
# influential

names_of_influential <- names(influential)
outliers <- AQ1_Data[names_of_influential,]
OutRem_AQ1_Data <- AQ1_Data %>% anti_join(outliers)


AQ1Model <-lm(SalePrice~GrLivArea+Neighborhood+Neighborhood*GrLivArea,data =OutRem_AQ1_Data)

summary(AQ1Model)

# confint(AQ1Model)
#
# par(mfrow = c(2,2))
#
# plot(AQ1Model)
#
# par(mfrow = c(1,1))
#
# # Okay, these graphs look a lot better, and the ADJ R^2, went up to .49. The model still isn't
# # great, but that's also because of the exclusion of so many other variables.
#
# # Let's remove the outliers from our main dataset now.
#
# Clean_FullData = Clean_FullData[-c(339,131,190,169,372),]


######################################## Data Selection // Analysis Question 2

# # Building a correlation matrix, where we only take data that has high ( > |.5|) correlation
# # with SalePrice
#
# CorrMatData = Clean_FullData %>% dplyr::select(where(is.numeric)) %>% correlate() %>% focus(SalePrice) %>%
#   mutate(term = factor(term, levels = term[order(SalePrice)])) %>% filter(SalePrice > abs(.5))
#
# CorrMatData  %>%
#   ggplot(aes(x = term, y = SalePrice)) +
#   geom_bar(stat = "identity", color = "black", fill = "turquoise") +
#   ylab("Correlation with Sale Price") + xlab("Variable") + theme_tufte() +
#   geom_text(aes(label = paste(round((SalePrice*100),2), "%", sep = "")),
#                               parse = F, vjust=1.6, color="black", size=3.5)
#
#
# # Looking to see if the saleprice has significant changes over the years
#
# ggplot(Clean_FullData, aes(x=YrSold, y=SalePrice, fill = YrSold)) + geom_boxplot() +
#   theme(legend.position="none") +
#   scale_fill_brewer(palette="Dark2") +
#   labs(title = "Price of Home Sales Over the Years", x = "Year Sold", y = "SalePrice")
```

```
#
#
# # Let's test the correlation between the categorical variables and the sales price
# # to see which variables we should keep there.
#
# # First we will create a dataframe with just the catgeorical variables and
# # SalePrice
#
# CatModelData =  Clean_FullData %>% dplyr::select(!where(is.numeric), SalePrice, -Utilities, Id)
#
# # We need sale price obviously, and all homes have the same value for utilities.
# # Id is for a later full join
# sapply(CatModelData, class)
#
# Model_Cat_SP = aov(SalePrice ~ YrSold, data = CatModelData)
# summary(Model_Cat_SP)
#
# # Where is the p value stored in the AOV model?
# summary(aov(SalePrice ~ YrSold, data = CatModelData))[[1]][["Pr(>F)"]][1]
#
# # Attempting to build a function where I loop all of the cat variables
# # that we have, and eliminate those that have little to no effect on the data.
# # We can use the AOV since we have met the assumptions, and it will tell us if
# # atleast one of the factor levels in each column have some effect on the mean.
#
# yCol = NULL
# x = 0
#
# for (i in 1:length(CatModelData[,1:44])){
#   x = x + 1
#   # print(x)
#   if (summary(aov(SalePrice ~ CatModelData[,x], data = CatModelData))[[1]][["Pr(>F)"]][1] > .01){ # alpha level
#     yCol = append(yCol,x)
#     # print(summary(aov(SalePrice ~ CatModelData[,x], data = CatModelData))[[1]][["Pr(>F)"]][1])
#   }
# }
#
# yCol
# length(yCol)
#
# CatModelData = subset(CatModelData, select = -yCol)
#
# sapply(CatModelData, class)
#
# # I've noticed that some columns have similar data, so I will manually pit them against
# # each other, to see which column to keep.
#
# summary(aov(SalePrice ~ GarageYrBlt  , data = CatModelData))
#
# summary(aov(SalePrice ~ GarageCond, data = CatModelData))
#
# TukeyHSD(aov(SalePrice ~ BsmtQual    , data = CatModelData))
#
#
#
# # lets just build a model
#
# # combine the 2 cat/cont data
#
# ContData = Clean_FullData %>% dplyr::select(Id,TotalBsmtSF, X1stFlrSF, GrLivArea, FullBath, TotRmsAbvGrd, GarageCars, GarageArea)
#
```

```
# AQ2_CleanData = ContData %>% full_join(CatModelData, by = "Id")
#
# AQ2_CleanData = AQ2_CleanData %>% dplyr::select(-Id,-YearRemodAdd,-Electrical,-GarageYrBlt,-GarageQual)
#
# AQ2_CleanData$SalePrice = log(AQ2_CleanData$SalePrice)
#
# sapply(AQ2_CleanData, class)
#
# dim(AQ2_CleanData)
#
# AQ2_Model = lm(SalePrice ~ ., data = AQ2_CleanData)
#
# summary(AQ2_Model)
#
# dim(summary(AQ2_Model)$coefficients)
#
# # # Now let's get rid of everything :)
# #
# #
# # xRow = NULL
# # x = 0
# #
# # for (i in 1:length(summary(AQ2_Model)$coefficients[,4])){
# #   x = x + 1
# #   if (summary(AQ2_Model)$coefficients[x,4] > .01){ # alpha level
# #     xRow = append(xRow,x)
# #   }
# # }
# #
# # xRow
# # length(xRow)
#
# # test = AQ2_Model
# # summary(test)
# # test = update(test,~.-YearBuilt2008)
# # summary(test)
# # dim(summary(test)$coefficients)
#
# # variables <- labels(test)[attr(terms(test), "order") == 1]
# # factors <- sapply(names(test$xlevels), function(x) paste0(x, test$xlevels[[x]])[-1])
# # setdiff(colnames(model.matrix(test)), c("(Intercept)", variables, unlist(factors)))
#
#
# # AQ2_Model = update(AQ2_Model,~.-Id)
# # summary(AQ2_Model)
#
# xtestx = ols_step_both_p(AQ2_Model,pent = .02,prem = .03, progress = T, details = F)
#
# #
# # Model Summary
# # ----------------------------------------------------------------------
# #   R                  0.952       RMSE                26548.362
# # R-Squared            0.907       Coef. Var              14.194
# # Adj. R-Squared       0.898       MSE                704815501.583
# # Pred R-Squared        -Inf       MAE                  15822.379
# # ----------------------------------------------------------------------
# #
#
# # apply the model
#
#
#
```

```
# # row.number = sample(1:nrow(AQ2_CleanData), 0.8*nrow(AQ2_CleanData))
# # train = AQ2_CleanData[row.number,]
# # test = AQ2_CleanData[-row.number,]
# # dim(train)
# # dim(test)
# #
# # pred1 <- predict(xtestx$model, newdata = test)
# # rmse <- sqrt(sum((exp(pred1) - test$SalePrice)^2)/length(test$SalePrice))
# # c(RMSE = rmse, R2=summary(xtestx$model)$r.squared)
# #
# # IdSalePrice = Clean_FullData %>% dplyr::select(Id,SalePrice)
#
# # new attempt
#
# SalePrice_train = AQ2_CleanData[2:1094,43]
#
# Affectors_train = as.matrix(AQ2_CleanData[2:1094,1:42])
#
# # SalePrice_test = as.matrix(Submission_TestData[1461:2919,2:78])
# #
# # SalePrice_test = as.matrix(Submission_TestData[1461:2919,79])
#
# full = lm(SalePrice ~ ., data = AQ2_CleanData)
# null = lm(SalePrice ~ 1, data = AQ2_CleanData)
#
# fullStep = step(null, scope = list(upper=full), direction="both")
#
# summary(fullStep)
# #
# # Residual standard error: 0.1101 on 947 degrees of freedom
# # Multiple R-squared:  0.9326,  Adjusted R-squared:  0.9222
# # F-statistic: 89.72 on 146 and 947 DF,  p-value: < 2.2e-16
# #
#
# a = predict(fullStep, newdata = test)
#
# x = colnames(AQ2_CleanData)
# x = x[1:42]
# Submission_TestData_Filtered = Submission_TestData %>% dplyr::select(x)
#
# Submission_TestData_Filtered[,c(8:42)] = lapply(Submission_TestData_Filtered[,c(8:42)], factor)
#
#
# sapply(AQ2_CleanData, class)
# sapply(Submission_TestData_Filtered, class)
#
# TestSubmission_1 = predict(fullStep, newdata = Submission_TestData_Filtered)
#
# ## test
# AQ2_CleanData$ExterQual
# AQ2_CleanData$ExterQual = as.numeric(factor(AQ2_CleanData$ExterQual))
#

#
# I am keeping the above code, as a way to show that this next part is us
# learning from our many mistakes.
#

FullData = read.csv("Data/train.csv", na.strings = "NA", strip.white = T)

Submission_TestData  = read.csv("Data/test.csv", na.strings = "NA", strip.white = T)
```

```
Submission_TestData$SalePrice = NA
AllData = rbind(FullData,Submission_TestData)

# Starting over, as you can see. Let's just start from the beginning using all
# of the original data.

# Combining the data so that we can easily modify the columns, and can easily
# split it up earlier.

str(AllData)

# We also know to make SalePrice log transformed to adress normality.

qqnorm(AllData$SalePrice)
qqline(AllData$SalePrice)

qqnorm(log(AllData$SalePrice))
qqline(log(AllData$SalePrice))

AllData$SalePrice = log(AllData$SalePrice)

# c = c(3,6:8)
# c
# AllData[,c] = as.factor(AllData[,c])
#
# AllData[,c(3,6:18)] = as.factor(AllData[,c(3,6:18)])
#
# Warning message:
#   In xtfrm.data.frame(x) : cannot xtfrm data frames

# Will need to change the columns to numeric/categorical. We tried
# somehow automating it, but to no avail. By hand we go;

AllData$MSZoning = as.numeric(factor(AllData$MSZoning))
AllData$Street = as.numeric(factor(AllData$Street))
AllData$Alley = as.numeric(factor(AllData$Alley))
AllData$LotShape = as.numeric(factor(AllData$LotShape))
AllData$LandContour = as.numeric(factor(AllData$LandContour))
AllData$Utilities = as.numeric(factor(AllData$Utilities))
AllData$LotConfig = as.numeric(factor(AllData$LotConfig))
AllData$LandSlope = as.numeric(factor(AllData$LandSlope))
AllData$Neighborhood = as.numeric(factor(AllData$Neighborhood))
AllData$Condition1 = as.numeric(factor(AllData$Condition1))
AllData$Condition2 = as.numeric(factor(AllData$Condition2))
AllData$BldgType = as.numeric(factor(AllData$BldgType))
AllData$HouseStyle = as.numeric(factor(AllData$HouseStyle))
AllData$OverallQual = as.numeric(factor(AllData$OverallQual))
AllData$OverallCond = as.numeric(factor(AllData$OverallCond))
AllData$YearBuilt = as.numeric(as.factor(AllData$YearBuilt))
AllData$YearRemodAdd = as.numeric(as.factor(AllData$YearRemodAdd))
AllData$RoofStyle = as.numeric(factor(AllData$RoofStyle))
AllData$RoofMatl = as.numeric(factor(AllData$RoofMatl))
AllData$Exterior1st = as.numeric(factor(AllData$Exterior1st))
AllData$Exterior2nd = as.numeric(factor(AllData$Exterior2nd))
AllData$MasVnrType = as.numeric(factor(AllData$MasVnrType))
AllData$ExterQual = as.numeric(factor(AllData$ExterQual))
AllData$ExterCond = as.numeric(factor(AllData$ExterCond))
AllData$Foundation = as.numeric(factor(AllData$Foundation))
AllData$BsmtQual = as.numeric(factor(AllData$BsmtQual))
AllData$BsmtCond = as.numeric(factor(AllData$BsmtCond))
```

```
AllData$BsmtExposure = as.numeric(factor(AllData$BsmtExposure))
AllData$BsmtFinType1 = as.numeric(factor(AllData$BsmtFinType1))
AllData$BsmtFinType2 = as.numeric(factor(AllData$BsmtFinType2))
AllData$Heating = as.numeric(factor(AllData$Heating))
AllData$HeatingQC = as.numeric(factor(AllData$HeatingQC))
AllData$CentralAir = as.numeric(factor(AllData$CentralAir))
AllData$Electrical = as.numeric(factor(AllData$Electrical))
AllData$KitchenQual = as.numeric(factor(AllData$KitchenQual))
AllData$TotRmsAbvGrd = as.numeric(factor(AllData$TotRmsAbvGrd))
AllData$Functional = as.numeric(factor(AllData$Functional))
AllData$FireplaceQu = as.numeric(factor(AllData$FireplaceQu))
AllData$GarageType = as.numeric(factor(AllData$GarageType))
AllData$GarageYrBlt = as.numeric(as.factor(AllData$GarageYrBlt))
AllData$GarageFinish = as.numeric(factor(AllData$GarageFinish))
AllData$GarageQual = as.numeric(factor(AllData$GarageQual))
AllData$GarageCond = as.numeric(factor(AllData$GarageCond))
AllData$PavedDrive = as.numeric(factor(AllData$PavedDrive))
AllData$PoolQC = as.numeric(factor(AllData$PoolQC))
AllData$Fence = as.numeric(factor(AllData$Fence))
AllData$SaleType = as.numeric(factor(AllData$SaleType))
AllData$YrSold = as.numeric(as.factor(AllData$YrSold))
AllData$MiscFeature = as.numeric(factor(AllData$MiscFeature))
AllData$SaleCondition = as.numeric(factor(AllData$SaleCondition))


sapply(AllData, class)

# Instead of deleting NA values, as we did before, we will throw 0's in for the
# categorical variables. Mainly looking at data with a lot of NA's.

summary(AllData)

AllData$Alley[which(is.na(AllData$Alley))] = 0
AllData$PoolQC[which(is.na(AllData$PoolQC))] = 0
AllData$Fence[which(is.na(AllData$Fence))] = 0
AllData$MiscFeature[which(is.na(AllData$MiscFeature))] = 0
AllData$FireplaceQu[which(is.na(AllData$FireplaceQu))] = 0
AllData$LotFrontage[which(is.na(AllData$LotFrontage))] = 0
AllData$GarageType[which(is.na(AllData$GarageType))] = 0
AllData$GarageFinish[which(is.na(AllData$GarageFinish))] = 0
AllData$GarageQual[which(is.na(AllData$GarageQual))] = 0
AllData$GarageCond[which(is.na(AllData$GarageCond))] = 0
AllData$GarageYrBlt[which(is.na(AllData$GarageYrBlt))] = 0
AllData$BsmtFinType2[which(is.na(AllData$BsmtFinType2))] = 0
AllData$BsmtFinType1[which(is.na(AllData$BsmtFinType1))] = 0
AllData$BsmtExposure[which(is.na(AllData$BsmtExposure))] = 0
AllData$BsmtCond[which(is.na(AllData$BsmtCond))] = 0
AllData$BsmtFullBath[which(is.na(AllData$BsmtFullBath))] = 0
AllData$BsmtQual[which(is.na(AllData$BsmtQual))] = 0
AllData$MasVnrType[which(is.na(AllData$MasVnrType))] = 0
AllData$MasVnrArea[which(is.na(AllData$MasVnrArea))] = 0

summary(AllData)

# Unsure what to do about the SalePrice NA's, we have seen methods of using the
# mean, but that is hard with all the categorical variable differences. We are
# also trying not to delete any data, as that has screwed us in our first
# attempt of this problem.


# Time to split the data up.
```

```
FinalTestData = AllData[1461:2919,1:80]

FinalTrainData = AllData[1:1460,1:81]

sum(is.na(AllData$SalePrice))

# Let's create a model with everything in it, then use olss stepwise to
# fix it.


FirstModel = lm(SalePrice ~  ., data = FinalTrainData)
par(mfrow = c(2,2))
plot(FirstModel)
summary(FirstModel)
CV(FirstModel)
# CV          AIC          AICc          BIC        AdjR2
# Inf -5779.2668772 -5769.8619426 -5356.4263534     0.8870119

# Residual standard error: 0.1343 on 1380 degrees of freedom
# (1 observation deleted due to missingness)
# Multiple R-squared:  0.8931,    Adjusted R-squared:  0.887
# F-statistic: 147.7 on 78 and 1380 DF,  p-value: < 2.2e-16

# High AdjR^2, probably bc of the overfitting. Though we are surprised that
# the AdjR^2 is not lower with these many variables in the model.

StepwiseModel = ols_step_both_p(FirstModel, prem = 0.01, pent = 0.02, details = F, progress = T)
plot(StepwiseModel)
StepwiseModel = StepwiseModel$model
summary(StepwiseModel)
# Residual standard error: 0.1374 on 1435 degrees of freedom
# Multiple R-squared:  0.8837,    Adjusted R-squared:  0.8817
# F-statistic: 454.2 on 24 and 1435 DF,  p-value: < 2.2e-16
CV(StepwiseModel)
#       CV          AIC          AICc          BIC        AdjR2
# 2.433418e-02 -5.769467e+03 -5.768487e+03 -5.632026e+03  .8817212



BackwardsModel = ols_step_backward_p(FirstModel, prem = 0.01, details = F, progress = T)
plot(BackwardsModel)
BackwardsModel = BackwardsModel$model
summary(BackwardsModel)
# Residual standard error: 0.1363 on 1429 degrees of freedom
# Multiple R-squared:  0.886,      Adjusted R-squared:  0.8836
# F-statistic: 370.1 on 30 and 1429 DF,  p-value: < 2.2e-16
CV(BackwardsModel)
#     CV          AIC          AICc          BIC        AdjR2
# 2.492174e-02 -5.786626e+03 -5.785146e+03 -5.617468e+03 .8835732



ForwardModel = ols_step_forward_p(FirstModel, penter = 0.01, details = F, progress = T)
plot(ForwardModel)
ForwardModel = ForwardModel$model
summary(ForwardModel)
# Residual standard error: 0.1374 on 1435 degrees of freedom
# Multiple R-squared:  0.8837,    Adjusted R-squared:  0.8817
# F-statistic: 454.2 on 24 and 1435 DF,  p-value: < 2.2e-16
CV(ForwardModel)
```

```
#    CV          AIC         AICc          BIC        AdjR2
# 2.433418e-02 -5.769467e+03 -5.768487e+03 -5.632026e+03  .8817212



CustomModel = lm(SalePrice ~ MSZoning+LotArea+Street+
                 LotShape+
                 BldgType+OverallQual+YearBuilt+YearRemodAdd+
                 RoofMatl+BsmtCond+BsmtFinType1+BsmtFinSF1+
                 BsmtFinSF2+HeatingQC+CentralAir+X1stFlrSF+X2ndFlrSF+
                 BsmtFullBath+KitchenQual+TotRmsAbvGrd+Functional+Fireplaces+
                 GarageCars+ScreenPorch+PoolQC+SaleCondition, data = FinalTrainData)
plot(CustomModel)
summary(CustomModel)
CV(CustomModel)
#     CV          AIC         AICc          BIC        AdjR2
# 2.488314e-02 -5.627182e+03 -5.626048e+03 -5.479169e+03  8.697892e-01


######################################## Applying the models to get kaggle scores

Predict_Stepwise = predict(StepwiseModel, FinalTestData)
Predict_Stepwise = exp(Predict_Stepwise)
Predict_Stepwise = as.data.frame(Predict_Stepwise)

StepwisePredicted = cbind(FinalTestData$Id,Predict_Stepwise)
names(StepwisePredicted)[1] = "Id"
names(StepwisePredicted)[2] = "SalePrice"

summary(StepwisePredicted)
# There are 8 NA's, no idea how or why they are there. We will replace them with
# the average, since that will have a negligible impact on the data.
StepwisePredicted$SalePrice[which(is.na(StepwisePredicted$SalePrice))] = 178591

summary(StepwisePredicted)

write.csv(StepwisePredicted, "SubmissionData/StepwisePredicted.csv", row.names =FALSE)



Predict_Forward = predict(ForwardModel, FinalTestData)
Predict_Forward = exp(Predict_Forward)
Predict_Forward = as.data.frame(Predict_Forward)

ForwardPredicted = cbind(FinalTestData$Id,Predict_Forward)
names(ForwardPredicted)[1] = "Id"
names(ForwardPredicted)[2] = "SalePrice"

summary(ForwardPredicted)
# There are 8 NA's, again, which means it is somewhere in our data.
ForwardPredicted$SalePrice[which(is.na(ForwardPredicted$SalePrice))] = 178591

summary(ForwardPredicted)

write.csv(ForwardPredicted, "SubmissionData/ForwardPredicted.csv", row.names =FALSE)



Predict_Backwards = predict(BackwardsModel, FinalTestData)
Predict_Backwards = exp(Predict_Backwards)
```

```
Predict_Backwards = as.data.frame(Predict_Backwards)

BackwardPredicted = cbind(FinalTestData$Id,Predict_Backwards)
names(BackwardPredicted)[1] = "Id"
names(BackwardPredicted)[2] = "SalePrice"

summary(BackwardPredicted)
# There are 8 NA's, again, which means it is somewhere in our data. This time, the
# mean is different.
BackwardPredicted$SalePrice[which(is.na(BackwardPredicted$SalePrice))] = 178936

summary(BackwardPredicted)

write.csv(BackwardPredicted, "SubmissionData/BackwardPredicted.csv", row.names =FALSE)



Predict_Custom = predict(CustomModel, FinalTestData)
Predict_Custom = exp(Predict_Custom)
Predict_Custom = as.data.frame(Predict_Custom)

CustomPredicted = cbind(FinalTestData$Id,Predict_Custom)
names(CustomPredicted)[1] = "Id"
names(CustomPredicted)[2] = "SalePrice"

summary(CustomPredicted)
# There are 8 NA's, again, which means it is somewhere in our data. This time, the
# mean is different.
CustomPredicted$SalePrice[which(is.na(CustomPredicted$SalePrice))] = 178182

summary(CustomPredicted)

write.csv(CustomPredicted, "SubmissionData/CustomPredicted.csv", row.names =FALSE)
```