Building A Sale Price Model On Housing Data

Javiy Wang
Eric Estrella
Glen Chu

11 May 2020

Introduction:

When a human being grows up into a full-fledged adult, they go through various events and experiences to help reach that self-fulfilling moment such as graduating college, pursuing a career, and, eventually, buying a home. The housing market in America had a crisis back during the tail end of the 2000's. Ever since then, many citizens in the country pondered on the housing market and the way it works as they slowly reached the point in their lives when buying a home was their next big step. How much would it cost to buy a house? What exactly factors into the sale price? This was part of our goal when looking at a dataset containing information on almost 1500 houses in Ames, Iowa and creating regression models that predict the price of a house in said location. Our variable of interest in this dataset is the Sales Price and, along with this, we also have 80 predictor variables that we could potentially utilize (23 nominal, 23 ordinal, 14 discrete, and 20 continuous). Our major goal with this project involves selecting the most important variables to study and creating a model that best predicts Sales Price. We utilize a training set with 1060 observations and thus compare it to the 400 observations found in the testing set to see if our model's Sales Price compares at all with the Sales Price of the 400 testing observations.

Data Cleansing and Processing:

Before we begin to implement a regression model to the data set, it is crucial for us to have clean and tidy training data, because the more reliable the training data the more improvements on the performance and reliability of our regression model. Our data set was fairly nice in the fact that there was not any missing data for the most part. In other words, we did not need to omit any columns or use imputation. However, one problem we ran into was that many columns had values of N/A when in reality the meaning of the N/A really meant either "None" or 0. For example, for the "Pool" column when there was a value of N/A, that meant that the house just did not possess a pool, thus we changed it to "None." This concept also applied to N/A values in a numeric column but their values were converted to 0. The only exception to these N/A values that we deemed was actual missing data turned out to be LotFrontage, Masonry Veneer Type, and Masonry Veneer Area. We replaced these missing values with "None" or 0 accordingly. Another thing we needed to change about the data was converting some numeric data columns into factor data columns. For example, we learned before that although things like zip code seem numeric at first, in reality, this is really just a qualitative variable. Thus, we took MSSubClass, OverallQual, OverallCond, YearSold, and MonthSold, and turned these numerical variables into qualitative variables instead.

Now, we want to remove outliers in our data. Rather than making 80 different graphs and finding outliers in each of them, we used something called "Cook's Distance." This is a way of getting rid of outliers with a multivariate model approach by computing a particular data point or row on the predicted outcome. Saying an observation as an outlier based on just one predictor could lead to unrealistic inferences when we build our final model. Therefore, when you have to decide if an individual data point is an outlier, it is better to consider a model with all the predictors (Xi's) and then analyzing if that same outlier is still far from our fitted model when all variables are considered. Generally, if a particular observation had a Cook's distance of 3 times the mean it is most likely an outlier. The result of this formula/concept allowed us to identify 40 potential outliers within the 1060 observations, thus we removed those rows from the data set to obtain a better and more accurate final model.

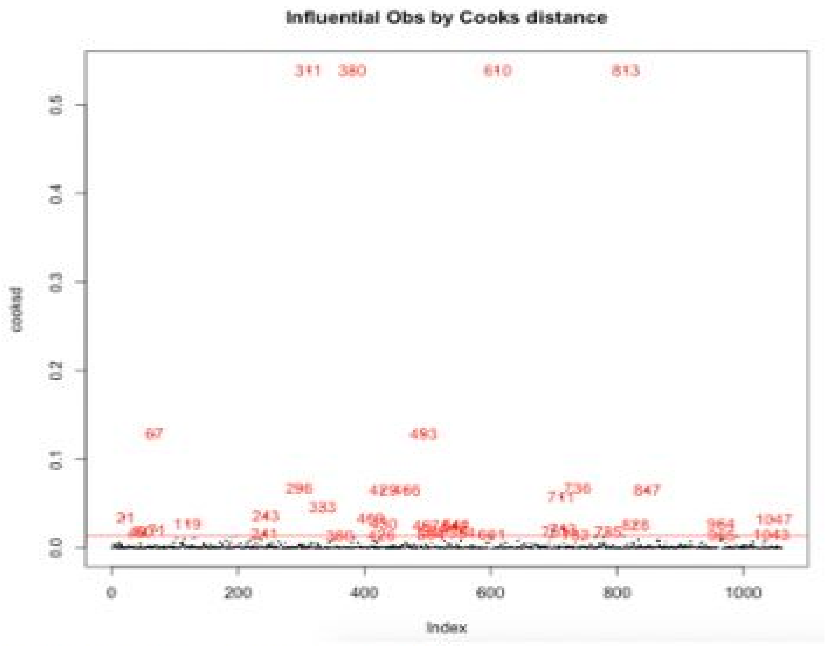


Figure 1 : Our Cook's Distance graph

Data Manipulation and Transformations:

One problem we had with our response variable "SalePrice" was that it seemed to be skewed right, therefore, we took the natural log of the response variable in order to normalize it. We plotted the histogram and Q-Q plot of both "SalePrice" and "Ln(SalesPrice)" in order to see if the transformed sales price had a normal distribution. We made this decision because one of the assumptions of linear regression is that the response is normally distributed. Therefore, the validity of performing multiple linear regression is not compromised.

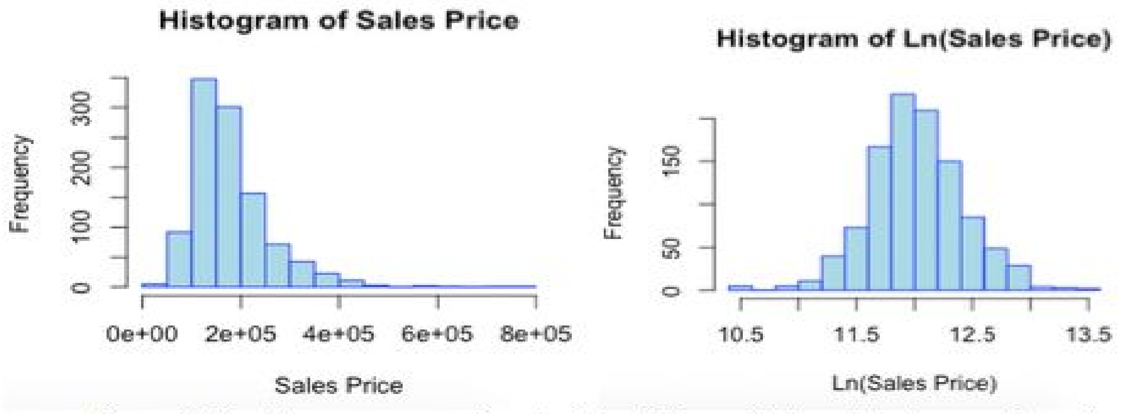
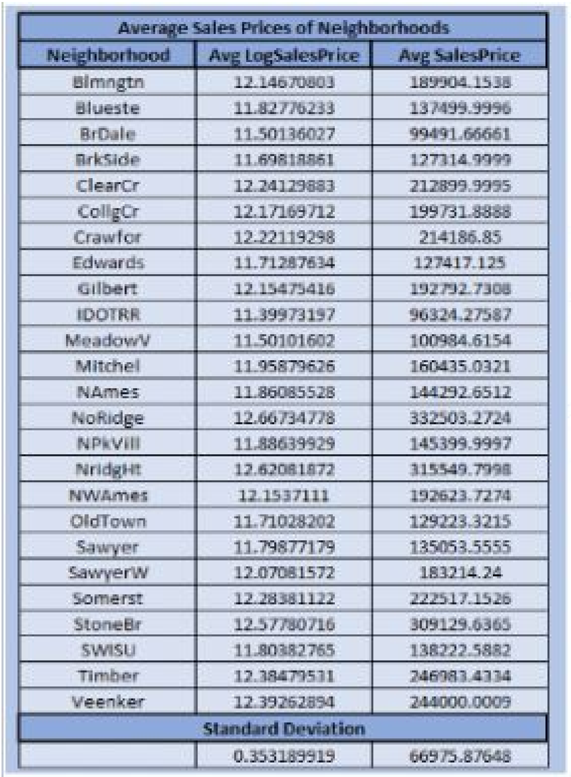


Figure 2: The histograms regarding the Sales Price variable and the Ln transformation of the Sales Price

Another column of data we manipulated was the variable "Neighborhood". We knew from experience that location is very significant in determining the sales price of a house, but our group wanted to collapse the location variable, neighborhood, into 4 wards, similar to how NYC is split up into 5 boroughs. We mostly did this because we thought that neighborhood would still yield as much if not more explanation of the variance in our response variable as well as significantly reducing the amount of parameters in our model from the initial 25 down to just 4. This was sadly not the case, but the silver lining was that since our response variable was transformed, so was the significance of our "Neighborhood" variable. We then used Excel to make a table to see the average sales price of a house in each of the 25 neighborhoods, and it



Average Sales Prices of Neighborhoods		
Neighborhood	Avg LogSalesPrice	Avg SalesPrice
Blmngtn	12.14670803	189904.1538
Blueste	11.82776233	137499.9996
BrDale	11.50136027	99491.66661
BrkSide	11.69818861	127314.9999
ClearCr	12.24129883	212899.9995
CollgCr	12.17169712	199731.8888
Crawfor	12.22119298	214186.85
Edwards	11.71287634	127417.125
Gilbert	12.15475416	192792.7308
IDOTRR	11.39973197	96324.27587
MeadowV	11.50101602	100984.6154
Mitchel	11.95879626	160435.0321
NAmes	11.86085528	144292.6512
NoRidge	12.66734778	332503.2724
NPkVill	11.88639929	145399.9997
NridgHt	12.62081872	315549.7998
NWAmes	12.1537111	192623.7274
OldTown	11.71028202	129223.3215
Sawyer	11.79877179	135053.5555
SawyerW	12.07081572	183214.24
Somerst	12.28381122	222517.1526
StoneBr	12.57780716	309129.6365
SWISU	11.80382765	138222.5882
Timber	12.38479531	246983.4334
Veenker	12.39262894	244000.0009
Standard Deviation		
	0.353189919	66975.87648

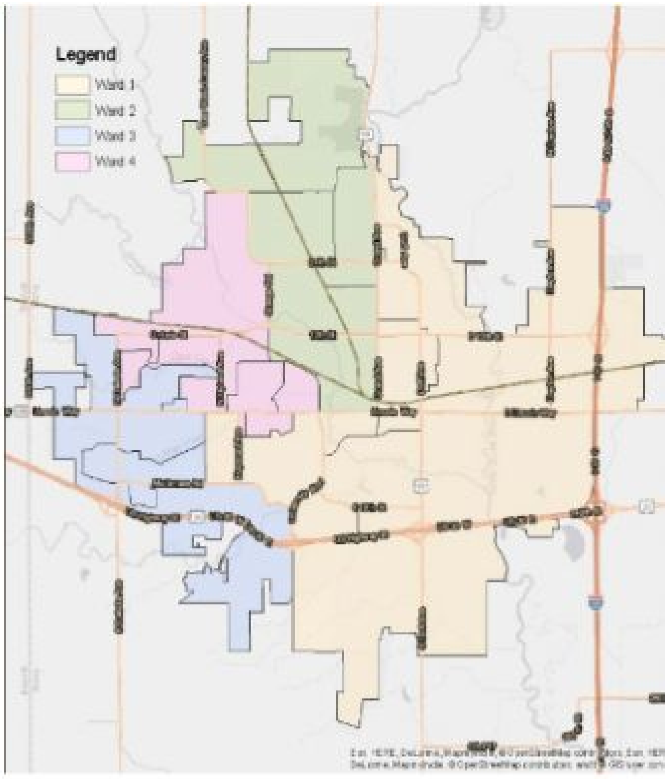
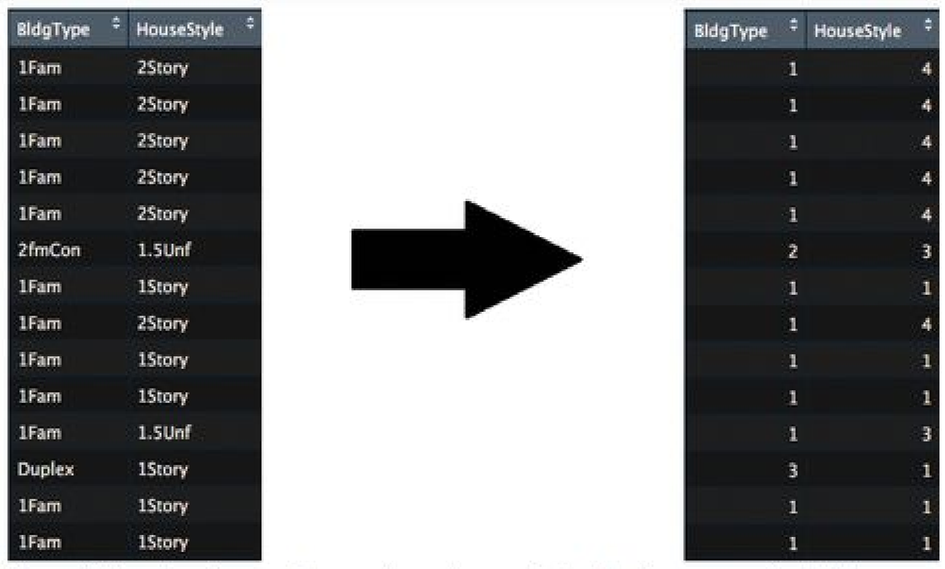


Figure 3: Here is an image of Ames, Iowa Split up into 4 wards as well as the AvgLogSalesPrice and the AvgSalesPrice respectively

proved to us what we initially believed, that neighborhood had a significant effect on "SalesPrice" but not on "Ln(SalesPrice)."

Many other columns that had categorical values were changed to numerical values in order for us to perform R functions such as a correlation plot or VIF. Once we ran those functions we obtained information such as multicollinearity. We kept the values as numbers but converted their column values back to "as.factor" so that our final model would have an accurate number of parameters. Along with this, we also went into the variable known as LotConfig and decided to condense the final two categories present in said variable, Frontage on two and three sides of property, and condense these two categories into one universal Frontage category, as we felt having two extremely similar categories was redundant. For the variables of WoodDeck Square Feet and OpenPorch Square Feet, we went ahead and manipulated them so that if a house simply had one of these accessories, the appropriate cell for the column would be populated with a "1". If the house lacked the particular accessory then it would be populated with a "0" instead. We did this because upon our initial inspection of the dataset, we realized that around half of the houses in the dataset did not even have a Wood Deck or an Open Porch. Due to this fact, we felt that the square feet of either of these were somewhat meaningless, and that just the act of having a Wood Deck or an Open Porch would suffice in affecting the price of the house.



BldgType	HouseStyle
1Fam	2Story
1Fam	2Story
1Fam	2Story
1Fam	2Story
1Fam	2Story
2fmCon	1.5Unf
1Fam	1Story
1Fam	2Story
1Fam	1Story
1Fam	1Story
1Fam	1.5Unf
Duplex	1Story
1Fam	1Story
1Fam	1Story

BldgType	HouseStyle
1	4
1	4
1	4
1	4
1	4
2	3
1	1
1	4
1	1
1	1
1	3
3	1
1	1
1	1

Figure 4: Here is an image of how we changed categorical values into a numerical label

The final major piece of data transformation/manipulation we implemented before moving on to our variable selection processes involved the four variables in the dataset that involved the bathrooms present in each home. Initially there were four variables that discuss the amount of half and full bathrooms both in the basement and above ground of each home. We felt that having four different variables discussing bathrooms was redundant because the location of the half and full bathrooms in a home are of little importance, and the distinction between half and full bathrooms itself fairly pointless. With all this in mind, we created a new variable called TotalBath that simply added the amount of basement half and full bathrooms as well as the above ground half and full bathrooms for each house, ultimately creating one universal bathroom variable that we will use once we begin creating models. This will help us in the future by limiting the amount of parameters related to the bathrooms in each house from four to just one.

Variable Selection:

There were many different ways we selected variables which caused us to create different models with different predictors. One of the ways we selected variables was to just take the predictor variables which were highly correlated to Ln(SalesPrice) in our Fit1.

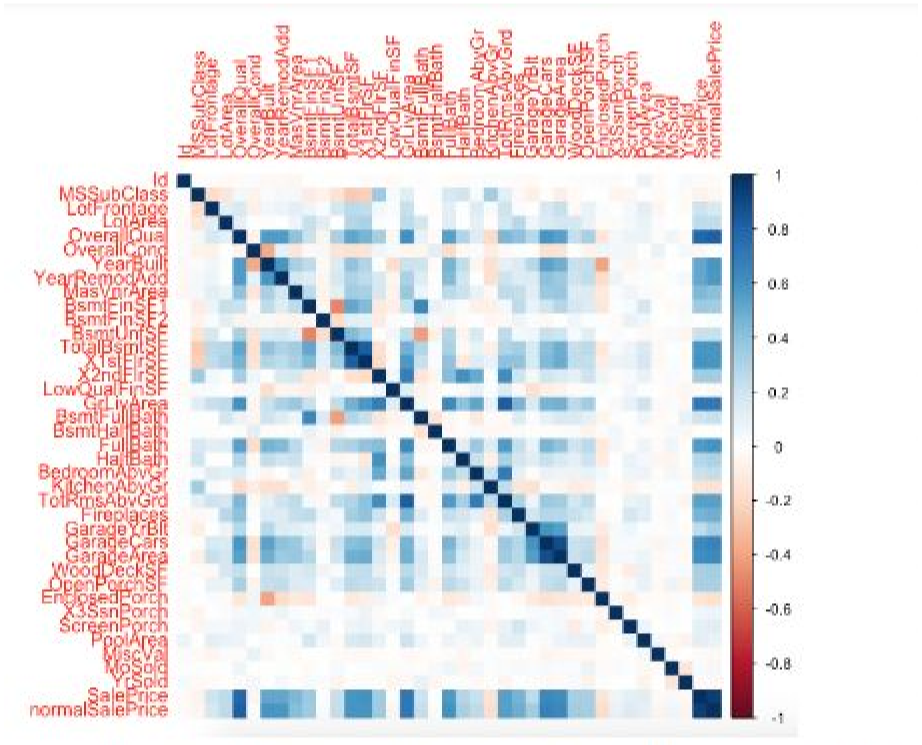
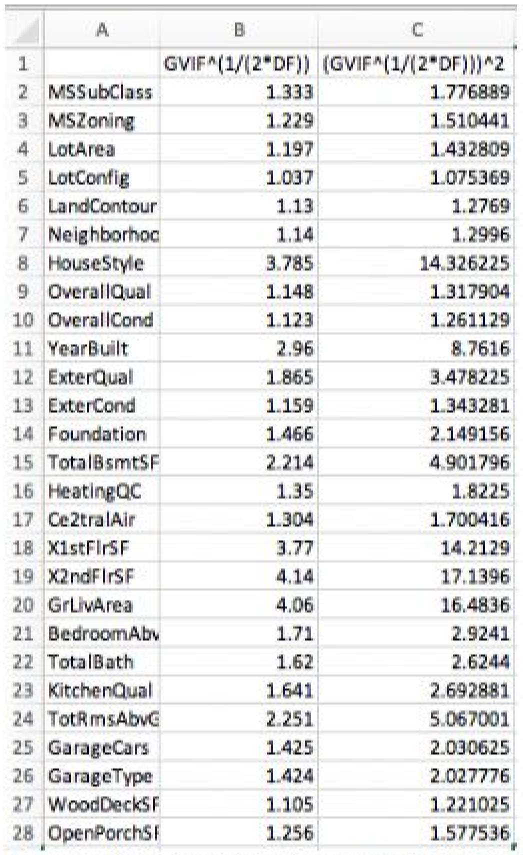


Figure 5: Here is a correlation plot of all numeric predictor variables against Ln(SalesPrice)

We also went through each and every variable and had a discussion about whether each feature would contribute to the price of the house and the magnitude of the contribution. In the end, we came up with a list of 24 "logical" variables to incorporate into our Fit2.

Then came the part where we needed to remove unnecessary predictors from our second model by looking at things such as a stepwise function and multicollinearity. For example, one way we used multicollinearity to get rid of multiple variables off the bat were variables in Fit2 that were highly correlated with one another such as TotRmsAbvGrd, X1stFlrSF, X2ndFlrSF, and basically any other variable that depicted the size of the property. So rather than keeping all those variables we decided to take all of them out except GrLivArea because this variable essentially describes all the other variables that were removed. Variables that were removed due to it barely reducing the AIC were variables such as WoodDeckSF and OpenPorchSF. When it came to VIF we used the rule of thumb of 3 as the threshold in order to remove variables from our model. We also attempted to transform our predictor variables and add interaction terms in order to get better residual plots as well as a lower MSE. However, many of these variables did not seem to help even after trying many different transformations with a combination of the transformed variables. Basically, it did not yield any significant results.

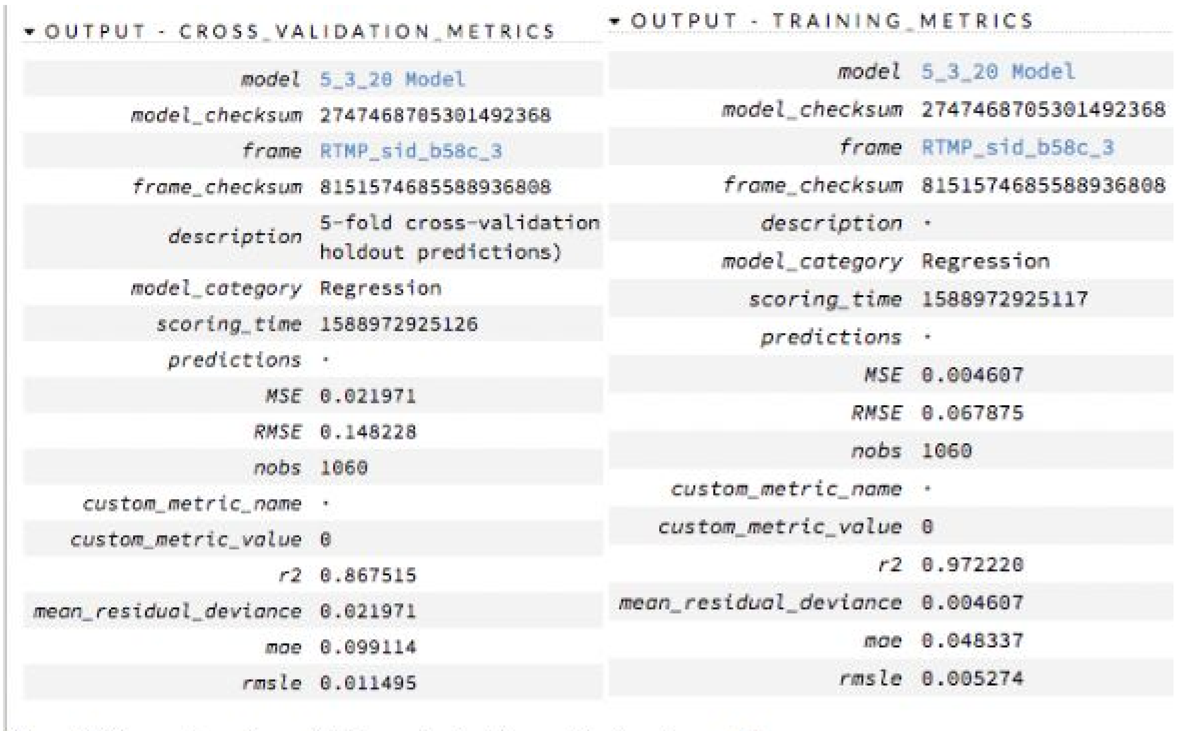


	A	B	C
1		GVIF^(1/(2*DF))	(GVIF^(1/(2*DF)))^2
2	MSSubClass	1.333	1.776889
3	MSZoning	1.229	1.510441
4	LotArea	1.197	1.432809
5	LotConfig	1.037	1.075369
6	LandContour	1.13	1.2769
7	Neighborhoo	1.14	1.2996
8	HouseStyle	3.785	14.326225
9	OverallQual	1.148	1.317904
10	OverallCond	1.123	1.261129
11	YearBuilt	2.96	8.7616
12	ExterQual	1.865	3.478225
13	ExterCond	1.159	1.343281
14	Foundation	1.466	2.149156
15	TotalBsmtSF	2.214	4.901796
16	HeatingQC	1.35	1.8225
17	Ce2tralAir	1.304	1.700416
18	X1stFlrSF	3.77	14.2129
19	X2ndFlrSF	4.14	17.1396
20	GrLivArea	4.06	16.4836
21	BedroomAbv	1.71	2.9241
22	TotalBath	1.62	2.6244
23	KitchenQual	1.641	2.692881
24	TotRmsAbvG	2.251	5.067001
25	GarageCars	1.425	2.030625
26	GarageType	1.424	2.027776
27	WoodDeckSF	1.105	1.221025
28	OpenPorchSI	1.256	1.577536

Figure 6: Presented here is a recreation of the VIF table we utilized in our variable selection that R-Studio provided us.

Using Machine Learning(GBM):

By utilizing an R package called “h2o,” we are able to use a pre-built in function called “gbm.” A GBM stands for a Gradient Boosted Machine and it is a technique for regression, for which it produces a prediction model by using multiple learning algorithms in order to obtain better predictive performance. The most common technique a GBM utilizes are decision trees, which are yes or no questions, and it is used to classify the data. Once one decision tree is used, GBM’s find outliers and continue using better decision trees in order to obtain an even better predictor model than the first decision tree. It is also not uncommon for a GBM to make thousands of decision trees in order to obtain the best model. At this point in time, we still did not know what type of regression we would be using, whether it be logistic, multiple linear, lasso, or even ridge. However, the GBM helps us here too, because as long as you do not specify the type of regression you want, the GBM function will analyze the function and manually test for all models to see which one explains the “Ln(Sales Price)” with the given predictor variables. In this case, the function gave us which type of regression we should use given the many classifiers GBM utilizes in order for it to produce the best model with the given data type of the response variable, which in our case was numeric. Such classifiers that GBM uses are: Gini Coefficient, Absolute MCC, F1, F0.5, F2, Accuracy, Logloss, AUC, AUCPR.



▾ OUTPUT - CROSS_VALIDATION_METRICS

model	5_3_20 Model
model_checksum	2747468705301492368
frame	RTMP_sid_b58c_3
frame_checksum	8151574685588936808
description	5-fold cross-validation holdout predictions)
model_category	Regression
scoring_time	1588972925126
predictions	·
MSE	0.021971
RMSE	0.148228
nobs	1060
custom_metric_name	·
custom_metric_value	0
r2	0.867515
mean_residual_deviance	0.021971
mae	0.099114
rmsle	0.011495

▾ OUTPUT - TRAINING_METRICS

model	5_3_20 Model
model_checksum	2747468705301492368
frame	RTMP_sid_b58c_3
frame_checksum	8151574685588936808
description	·
model_category	Regression
scoring_time	1588972925117
predictions	·
MSE	0.004607
RMSE	0.067875
nobs	1060
custom_metric_name	·
custom_metric_value	0
r2	0.972220
mean_residual_deviance	0.004607
mae	0.048337
rmsle	0.005274

Figure 7: Here we have the model diagnostics for the machine learning model

However, one of the downsides to using this method is that GBM’s require thousands of decision trees, and in our case much more because we did not specify what kind of regression we wanted the model to use, thus it was extremely time and memory exhaustive. Another disadvantage to using GBMs is that it will continue to improve the model in order to minimize all errors. This can overemphasize outliers and cause overfitting. In order to avoid the problems caused from overfitting we can remove outliers before running the GBM or we can use cross-validation to neutralize overfitting as well.

GBM keeps improving on decision trees until the deviance fails to continue to improve. This function actually yielded one of the best models for us; and it gave us valuable information such as variable importance. Our group needed to find an approach after the GBM model in order to reduce the number of variables we needed to predict the sales price of houses. We made sure to take note of what the GBM model deemed the most important variables when constructing one of our various models.

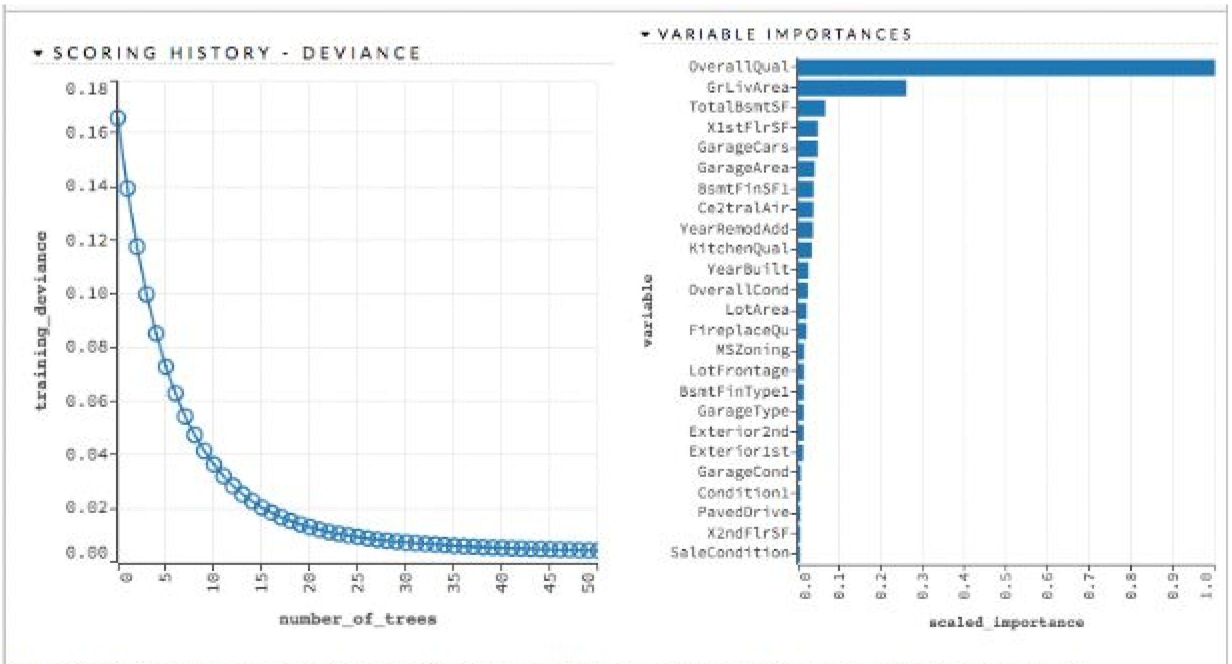
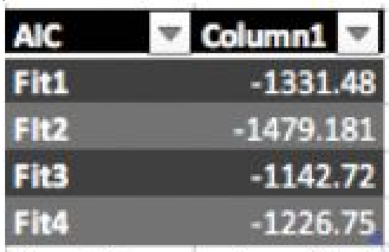


Figure 8: Deviance Scoring Graph along with the Variable Importance Graph from the machine learning model

Model Building:

We fitted a multiple linear regression model with each attempt we made because our residual plot allowed us to assume that a linear model is appropriate for predicting our response variable. Our first fit was a model that only included variables that were highly correlated to our response variable "Ln(SalesPrice)". In our second fit we included all variables we all considered to be important when predicting the sales price of a house in Ames Iowa. Then, our third model was just an improvement of the second model by using functions like VIF, AIC, and ols. With this we were able to reduce the number of our parameters and predictors of our model by more than half.



AIC	Column1
Fit1	-1331.48
Fit2	-1479.181
Fit3	-1142.72
Fit4	-1226.75

Figure 9: A comparison of the AIC's for each model

Although the increase in AIC was not ideal going from Fit2 to Fit3, we justify these actions because we have a model with 80 less parameters which allows our model to be simpler than when it contained all the variables we considered logical. So our third model is more interpretable than the second one, at the expense of accuracy.

Lastly we turned to our machine learning model, which told us the 24 most important predictor variables and graphically mapped each of the relative importance. After we created the multiple linear regression model for the GBM we used methods discussed in the previous 2 models in order to cut down the number of predictor variables for our final model. For our final model we have 9 predictor variables which are: Neighborhood, OverallQual, GrLivArea, TotalBsmtSF, GarageArea, KitchenQual, BsmtFinSF1, YearRemodAdd, CentralAir. This model also contained 20 parameters as well as an adjusted R squared value of .89 and an AIC of -1226.75. One interesting thing about our final model is that the Neighborhood variable barely makes our model any more accurate than it already is. Our group speculated 2 reasons why Neighborhood was not a good predictor. As stated before we collapsed the variable Neighborhood into 4 wards, and with the previous tests to check if this variable was still meaningful to us, we concluded that it is meaningful in the case of Sales Price but not Ln(SalesPrice). However, we chose to keep Neighborhood in the model because we deemed it to be a control variable. Control variables are usually variables that we are not particularly interested in, but that are related to the dependent variable. So even though the numbers said that Neighborhood was not important, we know from personal experience that location is a huge factor in the sales price of houses and we include it in the model to control for confounders.

Diagnostics:

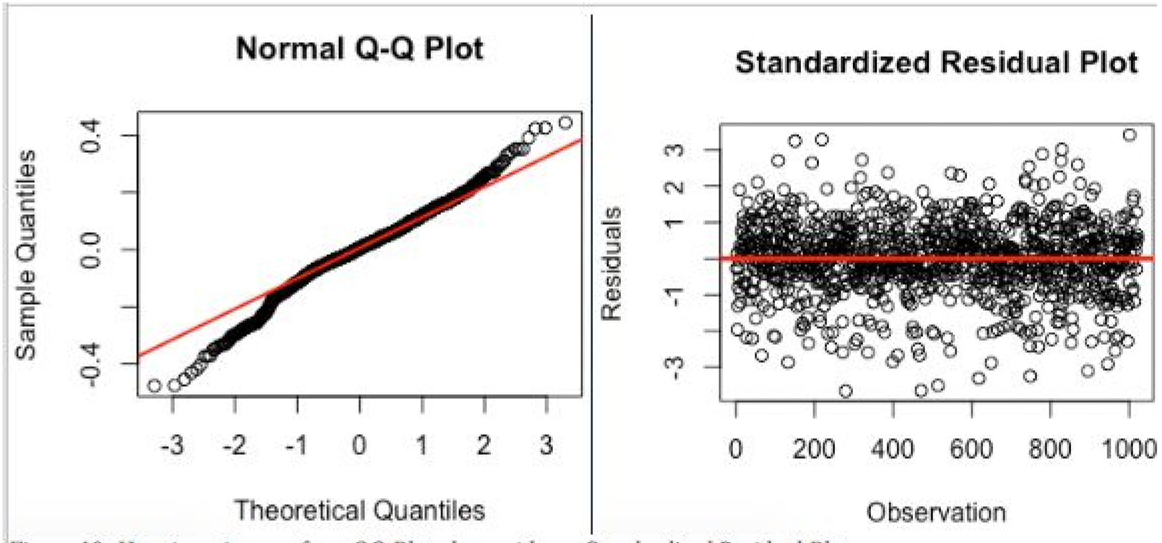


Figure 10: Here is an image of our QQ Plot along with our Standardized Residual Plot

We see from the residual Q-Q plot that the distribution of the residuals is not normal since most of the points in the tails are not on the red line. However, this will not be an issue since our criteria for selecting a model, AIC and MSE, both do not depend on the distribution of the residuals and neither does fitting a regression line to the data. We will keep this in mind however for any required inference procedures later on in the diagnostics.

In the standardized residual plot, we see that the residuals seem to be randomly centered around 0, as well that there are few residuals that have an absolute magnitude of 3 or greater, and that the variance of the residuals are constant. In order to formally test for homoscedasticity, we performed the Brown-Forsythe test, which is robust and does not depend on residual normality. The observations were broken up into two groups consisting of the first 510 observations and the last 510 observations. Since we have a p-value of .736, we failed to reject the null hypothesis that the error variance is constant at a 95% confidence level. This conclusion is supported by the absolute standardized residual plot, which also shows the residual outliers, since there is a clear rectangular pattern in the plot.

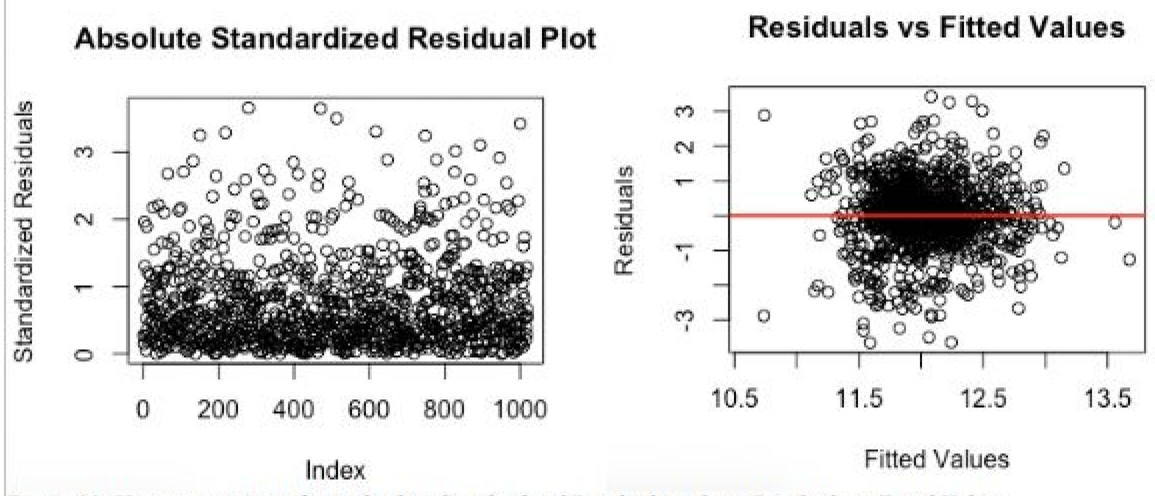
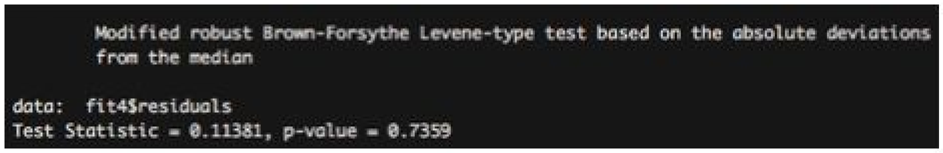


Figure 11: Here is an image of our Absolute Standardized Residuals and our Residuals vs Fitted Values

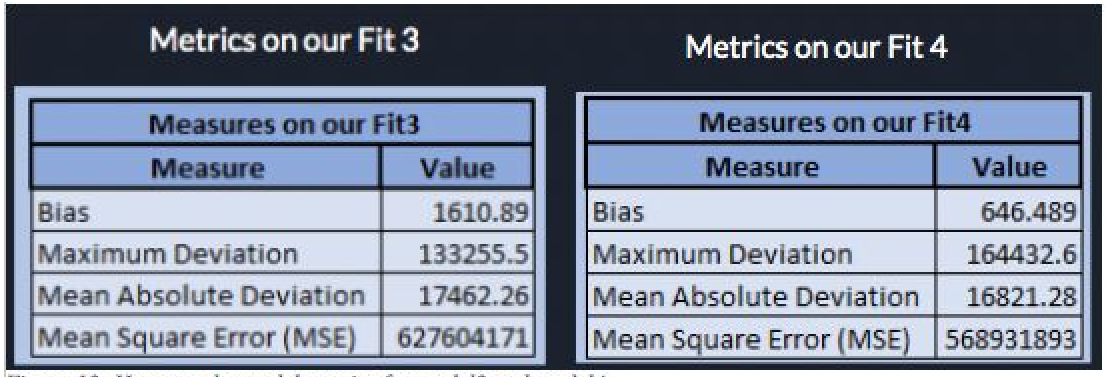

```
        Modified robust Brown-Forsythe Levene-type test based on the absolute deviations
        from the median

data:  fit4$residuals
Test Statistic = 0.11381, p-value = 0.7359
```
Figure 12: Here is the output of the Brown-Forsythe Test

In the residuals vs fitted plot, we see, again, that the residuals are randomly centered around 0 with very few outliers. Therefore, since the plot shows no definite pattern, a linear response function is appropriate in modelling the data.

We, however, could not produce a meaningful residual vs time plot. The data only has information on the month and year of when the house was sold. Since multiple houses were sold in such a general time frame, we could not index each observation in time order and therefore, the residual vs time plot could not be meaningfully interpreted.

Prediction on Test Set:

In our metrics on both Fit3 and Fit4, we see that Fit4 has a smaller bias. But this measure is not as important since model bias can be corrected. The maximum deviation for Fit4 is larger than Fit3 by about 30,000 dollars. However, the mean absolute deviation and MSE is lower for Fit4. Mean square error is the most important metric in evaluating the predictions from our model since it is equivalent to the bias plus the variance of the model. We see that our Fit4 has a lower variance than Fit3, which means that Fit4 is more accurate in its predictions than Fit3 and this is why we chose Fit4 to be our final model.



Metrics on our Fit 3

Measures on our Fit3	
Measure	Value
Bias	1610.89
Maximum Deviation	133255.5
Mean Absolute Deviation	17462.26
Mean Square Error (MSE)	627604171

Metrics on our Fit 4

Measures on our Fit4	
Measure	Value
Bias	646.489
Maximum Deviation	164432.6
Mean Absolute Deviation	16821.28
Mean Square Error (MSE)	568931893

Figure 13: Here are the model metrics for model3 and model4

Conclusion:

In conclusion, by running our final linear model with 9 variables, or 21 parameters, in the end we had an MSE of 568,931,893 squared dollars. Our adjusted R-squared turned out to be .89 and our AIC was at a low -1226.75 as well. As mentioned before, we also ran various model diagnostic tests to determine whether or not the residuals in our final model were normal, and we determined fairly quickly that they were indeed not normal. Despite this, we did manage to find out through diagnostics that our relationship was linear, which supported our final model of multiple linear regression. The final model ended up utilizing a mix of both categorical and numerical variables, the most important ones being OverallQual, GrLivArea, and BsmtFinSF1. The model performance on the testing set of the data is shown above and as one can see, due to the lower bias and MSE, our Fit4 was also the best model of the whole bunch when it came to running predictions on the testing set.

If we were given more time to play around with the dataset and create even more models than we already did, we would have also attempted to add in some interaction terms in some models to see how much that would have affected things, since we did not really try as many as we wanted to. Along with this, with more time we also would have attempted to seek out a professional who works in the real estate field, preferably someone near Ames, Iowa to begin with, who could shed some light on the various variables in the dataset we ultimately passed on due to lack of knowledge on how this market works, such as SaleType. We also would have properly investigated the outliers that showed up in some of the residual plots as well as invoked some more robust regression methods such as LASSO.

Code Appendix:

```
#Title: General_Linear_Models_Project.R
#Date: 4/9/20
install.packages("TSA")
library(car)
library(carData)
library(tidyr)
library(stats)
library(readxl)
library(alr3)
library(rio)
library(dlpyr)
library(corrplot)
library(olsrr)
library(dplyr)

trainingSet <- import('~/Desktop/Importing Data In R/training.csv')
testingSet <- import('~/Desktop/Importing Data In R/test.csv')
df1 = import('~/Desktop/Importing Data In R/trainingnumbers.csv')
testnumeric = import('~/Desktop/Importing Data In R/testnumeric.csv')

View(trainingnumbers)
#There were many predictor variables that seemed numeric but in reality it was categorical
#e.g zip code therefore i changed all 5 columns into ordinal values instead
df1 = mutate(df1,MSSubClass = as.factor(MSSubClass),
        OverallCond = as.factor(OverallCond),
        YrSold = as.factor(YrSold),
        MoSold = as.factor(MoSold),
        OverallQual = as.factor(OverallQual))
```

```
#There were many columns that had the value of N/A in the entry however in reality some of
these
#N/A's are actually either "None" or 0 thus I needed to change those values either
#manually in the CSV or with R

#Here is a vector of all the columns that needed the N/A's to be chnaged to "none"
NAColumnsNone = c("PoolQC","MiscFeature","Alley","Fence","FireplaceQu",
"GarageType","GarageFinish",
                "GarageQual","GarageCond","BsmtQual","BsmtCond","BsmtExposure",
"BsmtFinType1",
                "BsmtFinType2","MasVnrType","MSSubClass")
#Here is a vector of all the columns that needed the N/A's to be chnaged to 0
NAColumns0 = c("GarageYrBlt","GarageArea","GarageCars","BsmtFinSF1","BsmtFinSF2",
  "BsmtUnfSF","TotalBsmtSF","BsmtFullBath","BsmtHalfBath","MasVnrArea","LotFrontage")

#This is where i actually changed the N/A values to 0 or None
#[,NAColumnsNone] and [,NAColumns0] refers to all the rows but only some columns that
must be checked
#[is.na(df[,NAColumnsNone])] = checked for if each entry was N/A but needed to recall the data
frame in order to search for N/A values
df1[,NAColumnsNone][is.na(df1[,NAColumnsNone])] = "None"
df1[,NAColumns0][is.na(df1[,NAColumns0])] = 0

#Transformation of multiple variables
df1$Neighborhood[which(df1$Neighborhood == "BrDale"|df1$Neighborhood ==
"Crawfor"|df1$Neighborhood == "IDOTRR"|df1$Neighborhood ==
"Timber"|df1$Neighborhood == "Veenker"|df1$Neighborhood == "Mitchel"|df1$Neighborhood
== "OldTown")] = 1
df1$Neighborhood[which(df1$Neighborhood == "Blmngtn"|df1$Neighborhood ==
"BrkSide"|df1$Neighborhood == "Gilbert"|df1$Neighborhood ==
"MeadowV"|df1$Neighborhood == "NAmes"|df1$Neighborhood ==
"NoRidge"|df1$Neighborhood == "NPkVill"|df1$Neighborhood ==
"NridgHt"|df1$Neighborhood == "NWAmes")] = 2
df1$Neighborhood[which(df1$Neighborhood == "Sawyer"|df1$Neighborhood ==
"SWISU"|df1$Neighborhood == "Edwards"|df1$Neighborhood ==
"SawyerW"|df1$Neighborhood == "Somerst"|df1$Neighborhood == "StoneBr")] = 3
df1$Neighborhood[which(df1$Neighborhood == "Blueste"|df1$Neighborhood ==
"ClearCr"|df1$Neighborhood == "CollgCr")] = 4
df1$LotConfig[which(df1$LotConfig == "FR2")] = "FR"
```

```
df1$LotConfig[which(df1$LotConfig == "FR3")] = "FR"
df1$X2ndFlrSF[which(df1$X2ndFlrSF > "0")] = "1"
df1$TotalBath = df1$BsmtFullBath+df1$BsmtHalfBath+df1$FullBath+df1$HalfBath
df1$WoodDeckSF[which(df1$WoodDeckSF > "0")] = "1"
df1$OpenPorchSF[which(df1$OpenPorchSF > "0")] = "1"
#Removed rows/observations from Cook's Distance
df1 = df1[-c(21,45,50,67,71,119,241,243,296,311,333,360,380,409,426,429
        ,430,466,493,493,497,504,509,542,544,554,601,610,701,
        711,713,736,785,813,828,847,945,964,965,1043,1047), ]
#We changed all the variables into numbers in order to run VIF, thus we mutated all the
variables bac into factors
df1 = mutate(df1,MSSubClass = as.factor(MSSubClass),
          ExterQual = as.factor(ExterQual),
          Ce2tralAir = as.factor(Ce2tralAir),
          HeatingQC = as.factor(HeatingQC),
         OverallQual = as.factor(OverallQual),
         Neighborhood = as.factor(Neighborhood),
         OverallCond = as.factor(OverallCond),
         MSZoning = as.factor(MSZoning),
         LotConfig = as.factor(LotConfig),
         LandContour = as.factor(LandContour),
         ExterCond = as.factor(ExterCond),
         Foundation = as.factor(Foundation),
         KitchenQual = as.factor(KitchenQual),
         GarageType = as.factor(GarageType),
         WoodDeckSF = as.factor(WoodDeckSF),
         OpenPorchSF = as.factor(OpenPorchSF),
         HouseStyle = as.factor(HouseStyle),
         BldgType = as.factor(BldgType),
         RoofMatl = as.factor(RoofMatl))
df1$TotalBath = df1$BsmtFullBath+df1$BsmtHalfBath+df1$FullBath+df1$HalfBath
cbind(df1,df1$normalSalePrice)

#Did the same thing for the test set
testnumeric = mutate(testnumeric,WoodDeckSF = as.numeric(WoodDeckSF),
              OpenPorchSF = as.numeric(OpenPorchSF))
testnumeric$WoodDeckSF[which(testnumeric$WoodDeckSF > "0")] = "1"
testnumeric$OpenPorchSF[which(testnumeric$OpenPorchSF > "0")] = "1"
testnumeric = mutate(testnumeric,MSSubClass = as.factor(MSSubClass),
              ExterQual = as.factor(ExterQual),
              Ce2tralAir = as.factor(Ce2tralAir),
              HeatingQC = as.factor(HeatingQC),
              OverallQual = as.factor(OverallQual),
              Neighborhood = as.factor(Neighborhood),
              OverallCond = as.factor(OverallCond),
              MSZoning = as.factor(MSZoning),
              LotConfig = as.factor(LotConfig),
              LandContour = as.factor(LandContour),
              ExterCond = as.factor(ExterCond),
              Foundation = as.factor(Foundation),
              KitchenQual = as.factor(KitchenQual),
              GarageType = as.factor(GarageType),
              WoodDeckSF = as.factor(WoodDeckSF),
              OpenPorchSF = as.factor(OpenPorchSF),
              HouseStyle = as.factor(HouseStyle),
              BldgType = as.factor(BldgType),
              RoofMatl = as.factor(RoofMatl))
#Created a new column where we added all the bathrooms in the house
testnumeric$TotalBath =
testnumeric$BsmtFullBath+testnumeric$BsmtHalfBath+testnumeric$FullBath+testnumeric$Hal
fBath
cbind(testnumeric,testnumeric$TotalBath)
#Created column where we took the log of the SalePrice
testnumeric$normalSalePrice = log(testnumeric$SalePrice)
cbind(testnumeric,testnumeric$normalSalePrice)
testnumeric$OverallCond[which(testnumeric$OverallCond == "1")] = "2"

#Transforming our repsonse variable and normalizing it
normalSalePrice = log(df$SalePrice)
hist(normalSalePrice,xlab = "ln(SalePrice)", col = "light blue", border = "blue")
hist(trainingSet$SalePrice,xlab = "SalePrice", col = "light blue", border = "blue")

#Multivariate Model Approach(Cook's Distance)
#Getting rid of outliers
fmla
=newDataSet$normalSalePrice~newDataSet[,1]+newDataSet[,2]+newDataSet[,3]+newDataSet[,
4]+newDataSet[,5]+newDataSet[,6]+newDataSet[,8]+newDataSet[,9]+newDataSet[,11]+newDa
taSet[,12]+newDataSet[,13]+newDataSet[,14]+newDataSet[,15]+newDataSet[,16]+newDataSet[
,17]+newDataSet[,18]+newDataSet[,19]+newDataSet[,20]+newDataSet[,21]+newDataSet[,22]+
newDataSet[,23]+newDataSet[,24]+newDataSet[,25]+newDataSet[,26]+newDataSet[,27]+newD
ataSet[,28]+newDataSet[,29]+newDataSet[,30]+newDataSet[,31]+newDataSet[,32]+newDataSe
t[,33]+newDataSet[,34]+newDataSet[,35]+newDataSet[,36]+newDataSet[,37]+newDataSet[,38]
+newDataSet[,39]+newDataSet[,40]+newDataSet[,41]+newDataSet[,42]+newDataSet[,43]+new
DataSet[,44]+newDataSet[,45]+newDataSet[,46]+newDataSet[,47]+newDataSet[,48]+newData
Set[,49]+newDataSet[,50]+newDataSet[,51]+newDataSet[,52]+newDataSet[,53]+newDataSet[,5
4]+newDataSet[,55]+newDataSet[,56]+newDataSet[,57]+newDataSet[,58]+newDataSet[,59]+ne
wDataSet[,60]+newDataSet[,61]+newDataSet[,62]+newDataSet[,63]+newDataSet[,64]+newDat
aSet[,65]+newDataSet[,66]+newDataSet[,67]+newDataSet[,68]+newDataSet[,69]+newDataSet[,
70]+newDataSet[,71]+newDataSet[,72]+newDataSet[,73]+newDataSet[,74]+newDataSet[,75]+n
ewDataSet[,76]+newDataSet[,77]+newDataSet[,78]+newDataSet[,79]+newDataSet[,80]
mod = lm(fmla, data=newDataSet)
summary(mod)
cooksd <- cooks.distance(mod)
plot(cooksd, pch=".", cex=2, main="Influential Obs by Cooks distance")  # plot cook's distance
abline(h = 3*mean(cooksd, na.rm=T), col="red")  # add cutoff line
text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>3*mean(cooksd,
na.rm=T),names(cooksd),""), col="red")  # add labels

#df.numeric = df[,sapply(df, is.numeric)]
#Did this to get the corrplot
df.numeric = df.numeric[,colSums(is.na(df.numeric))<nrow(df.numeric)]
cordf = cor(df.numeric)
corrplot(cordf, method="color")

#Here is the fit of the model with the variables we decided were important ONLY from the
Corrplot
fit1 =
lm(normalSalePrice~OverallQual+YearBuilt+YearRemodAdd+GrLivArea+FullBath+GarageCa
rs
        +BsmtFinSF1+TotalBsmtSF+Neighborhood,data = df1)
summary(fit1)#40 Parameters
AIC(fit1)

#fit2 will be all the variables we found logical important with transformed variables
#Remove BsmtFinType1,
#Remove these b/c didnt contribute to R squared BsmtCond, BldgType,
#Remove these bc VIF higher than 5
TotRmsAbvGrd,TotalBsmtSF,X1stFlrSF,X2ndFlrSF,YearBuilt,HouseStyle
fit2 = lm(normalSalePrice~MSZoning+TotRmsAbvGrd+X1stFlrSF+
          +LotArea+LotConfig+LandContour+X2ndFlrSF+YearBuilt+
          +Neighborhood+OverallQual+OverallCond+HouseStyle+
          +ExterQual+ExterCond+Foundation+
          +HeatingQC+Ce2tralAir+GrLivArea+BedroomAbvGr+TotalBath+KitchenQual+
          +GarageCars+GarageType+WoodDeckSF+OpenPorchSF,data = df1)

#Removed due to multicolineary and high p value
#MSZoning,OverallCond
fit3 =
lm(normalSalePrice~Time+Neighborhood+LotArea+GrLivArea+OverallQual+MSSubClass+To
talBath+GarageCars,data = df1)

fit4 =
lm(normalSalePrice~Neighborhood+OverallQual+GrLivArea+TotalBsmtSF+GarageArea+
          KitchenQual+BsmtFinSF1+
          YearRemodAdd+Ce2tralAir,data = df1)
summary(fit4)
vif(fit4)
step.model = ols_step_forward_aic(fit4)
step.model
#Resiuals for fit 4
resid(fit4)
plot(rstandard(fit4),xlab = "Observation", ylab = "Residuals", main = "Standardized Residual
Plot")
abline(h=0, col = "red", lwd = 3)

#Histogram of resid
res_hist = hist(fit4$residuals,col = "light blue", border = "blue",xlab = "Residuals", ylab =
"Frequency", main = "Histogram of Residuals")

summary(fit4$residuals)
sd(fit4$residuals)

#QQ Plot
qplot = qqnorm(fit4$residuals)
qqline(fit4$residuals,col = "red", lwd = 2)
```

```
cor(qplot$x,qplot$y)

#Res vs Fitted
plot(fit4$fitted.values,rstandard(fit4), main = "Residuals vs Fitted Values", xlab = "Fitted
Values",ylab = "Residuals")
abline(h = 0, col = "red", lwd = 2)

plot(df1$normalSalePrice,fit4$fitted.values, main = "Fitted Sales Price vs Ln(Salesprice)")
abline(lm(fit4$fitted.values~df1$normalSalePrice),col = "red", lwd = 2)
cor(fit4$fitted.values,df1$normalSalePrice)

#Residuals vs TIme
df1 = mutate(df1,Time = as.yearmon(Time))
plot(df1$Time,fit4$residuals, main = "Date Sold")
abline(median(fit4$residuals), col = "red")
df1 = cbind(df1,fit4$residuals)

plot(df1$Time, df1$`fit4$residuals`)

plot(ts(fit4$residuals,start=c(2006, 1), end=c(2010, 7), frequency=12))
abline(h=0, col = "red")
#y-hat log predicted values
log_y_hat = predict(fit4, testnumeric)
y_hat = exp(log_y_hat)

#Formulas for prediction
biased = function(Y_hat, Y){
  mean(Y_hat - Y)
}

Max_Dev <- function(Y_hat, Y){
  max(abs(Y_hat - Y))
}
Mean_Abs_Dev <- function(Y_hat, Y){
  mean(abs(Y_hat - Y))
}
Mean_SQ_ERR <- function(Y_hat , Y){
  mean((Y_hat - Y)^2)
}
```

```
#Finding Y_Hats and using it to find Bias as well as MSE
log_y_hat = predict(fit4, testnumeric)
y_hat = exp(log_y_hat)
biased(y_hat,testnumeric$SalePrice)
Max_Dev(y_hat,testnumeric$SalePrice)
Mean_Abs_Dev(y_hat,testnumeric$SalePrice)
Mean_SQ_ERR(y_hat,testnumeric$SalePrice)
plot(testnumeric$SalePrice,y_hat, xlab = "Actual Sales Price",ylab = "Predicted Sales Price",
main = "Predicted vs Actual", pch = 16)
abline(lm(testnumeric$SalePrice~y_hat), col = "red", lwd = 2)
cor(testnumeric$SalePrice,y_hat)

#Machine Learning Model
h2o.init(nthreads = -1)
path = "/Users/JaviyWang/Desktop/Importing Data In R/trainingnumbers.csv"
secondpath = "/Users/JaviyWang/Desktop/Importing Data In R/test.csv"
training = h2o.importFile(path)
validate = h2o.importFile(secondpath)

myY = "normalSalePrice"
myX = setdiff(names(training), myY)

training[,myY] = as.numeric(training[,myY])
#validate[,myY] = as.factor(validate[,myY])
#
gbm_m = h2o.gbm(x = myX, y = myY, training_frame = training, nfolds = 5, model_id =
"5/3/20 Model", distribution = "gaussian")
```