Regression Analysis of Housing Prices: A Data Mining Example Project

Chris Henson

4/13/17

Designed and Implemented In Accordance with:

CS 699 Data Mining and Business Intelligence

Spring 2017

Boston University

1. Introduction

The purpose of this paper is to show some of the steps taken in the world of data science and data mining to clean and pre-process data, use different modelling algorithms and their effectiveness, and testing these models on a test set of data to determine effectiveness. The data used for this project is from the Kaggle website. While the design intent of the project was for classification models, this dataset calls for a regression approach, and the following process has been design as such. A full copy of all code used in the project is attached to this report. An overview of the data set and the intentions are described below.

2. Data Overview and Project Intentions

The intention of this project is to estimate the sale price of a house with the same data parameters as the given training dataset (i.e. within the same city, and with the same attribute listing as the given data). The original training dataset given is attached with this project and contains 81 attributes, including an ID attribute and the class attribute of the Sale Price, and consists of 1460 tuples. Within the data are multiple missing values, and clear outliers. There are also problems with normality and co-linearity among the attributes. The methodology of cleansing the data is shown through the steps below, but given the size and scope of the data parameters, many approaches can be taken to arrive at a reasonable result. This project was completed almost exclusively in the R language, and a copy of the script is attached to this project. In the attached script, the values reached for any data calls are also included and commented out for transparency. The code should read through in the same manner as the data was processed, and should be able to be run as a complete script, if all libraries have previously been installed.

3. Pre-Processing of the Data

Missing Data

The first task in pre-processing the dataset was to address the missing values throughout. Running the script to find any "NA's" reveals that the following attributes have missing values (shown with the number of missing values):

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley
0	0	0	259	0	0	1369
LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1
0	0	0	0	0	0	0
Condition2	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	YearRemodAdd
0	0	0	0	0	0	0
RoofStyle	RoofMatl	Exterior1st	Exterior2nd	MasVnrType	MasVnrArea	ExterQual

0	0	0	0	8	8	0
ExterCond	Foundation	BsmtQual	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinSF1
0	0	37	37	38	37	0
BsmtFinType2	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	Heating	HeatingQC	CentralAir
38	0	0	0	0	0	0
Electrical	X1stFlrSF	X2ndFlrSF	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath
1	0	0	0	0	0	0
FullBath	HalfBath	BedroomAbvGr	KitchenAbvGr	KitchenQual	TotRmsAbvGrd	Functional
0	0	0	0	0	0	0
Fireplaces	FireplaceQu	GarageType	GarageYrBlt	GarageFinish	GarageCars	GarageArea
0	690	81	81	81	0	0
GarageQual	GarageCond	PavedDrive	WoodDeckSF	OpenPorchSF	EnclosedPorch	X3SsnPorch
81	81	0	0	0	0	0
ScreenPorch	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold
0	0	1453	1179	1406	0	0
YrSold	SaleType	SaleCondition	SalePrice			
0	0	0	0			

While there could be some merit in retaining the missing attributes and tuple for training purposes, most of these attributes either carry redundant information with other attributes, or they have so many missing values as to not be useful for analysis. The attributes that had greater than 15% missing information were removed: PoolQC, MiscFeature, Alley, Fence, LotFrontage, and FireplaceQu. By grouping the remaining attributes into their respective housing aspects (i.e. outside aspects, basement, garage, interior, etc.) each attribute with missing values was examined for significance. Removing some of the attributes in the group will cause little to no information loss for the characteristic. Based on preliminary correlation analysis of all the garage attributes, GarageCars and GarageArea both are highly correlated and can represent that facet of analysis well, so the others that have missing values aren't needed and were removed. Likewise, TotBsmtSF correlates highly and can represent the Basement information aspect. The basement attributes with missing attributes were then removed.

The only remaining missing values were 8 each from MasVnrArea and MasVnrType, and one from Electrical. After doing some research on what the two veneer attributes referred to (these are any sort of masonry siding shown on the outside of the house), it was determined that these can be represented by the higher-correlated outdoor attributes of ExterQual and ExterCond, and thus were removed as well. After examining the tuple with the missing electrical attribute and given the size of the dataset, the tuple was simply removed.

These actions eliminated all missing values within the set, and a function call to search for missing values proves this:

MSSubClass	MSZoning	LotArea	Street	LotShape	LandContour	Utilities
0	0	0	0	0	0	0
LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	HouseStyle
0	0	0	0	0	0	0
OverallQual	OverallCond	YearBuilt	YearRemodAdd	RoofStyle	RoofMatl	Exterior1st
0	0	0	0	0	0	0
Exterior2nd	ExterQual	ExterCond	Foundation	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF
0	0	0	0	0	0	0
TotalBsmtSF	Heating	HeatingQC	CentralAir	Electrical	X1stFlrSF	X2ndFlrSF
0	0	0	0	0	0	0
LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	BedroomAbvGr
0	0	0	0	0	0	0
KitchenAbvGr	KitchenQual	TotRmsAbvGrd	Functional	Fireplaces	GarageCars	GarageArea
0	0	0	0	0	0	0
PavedDrive	WoodDeckSF	OpenPorchSF	EnclosedPorch	X3SsnPorch	ScreenPorch	PoolArea
0	0	0	0	0	0	0
MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice	
0	0	0	0	0	0	

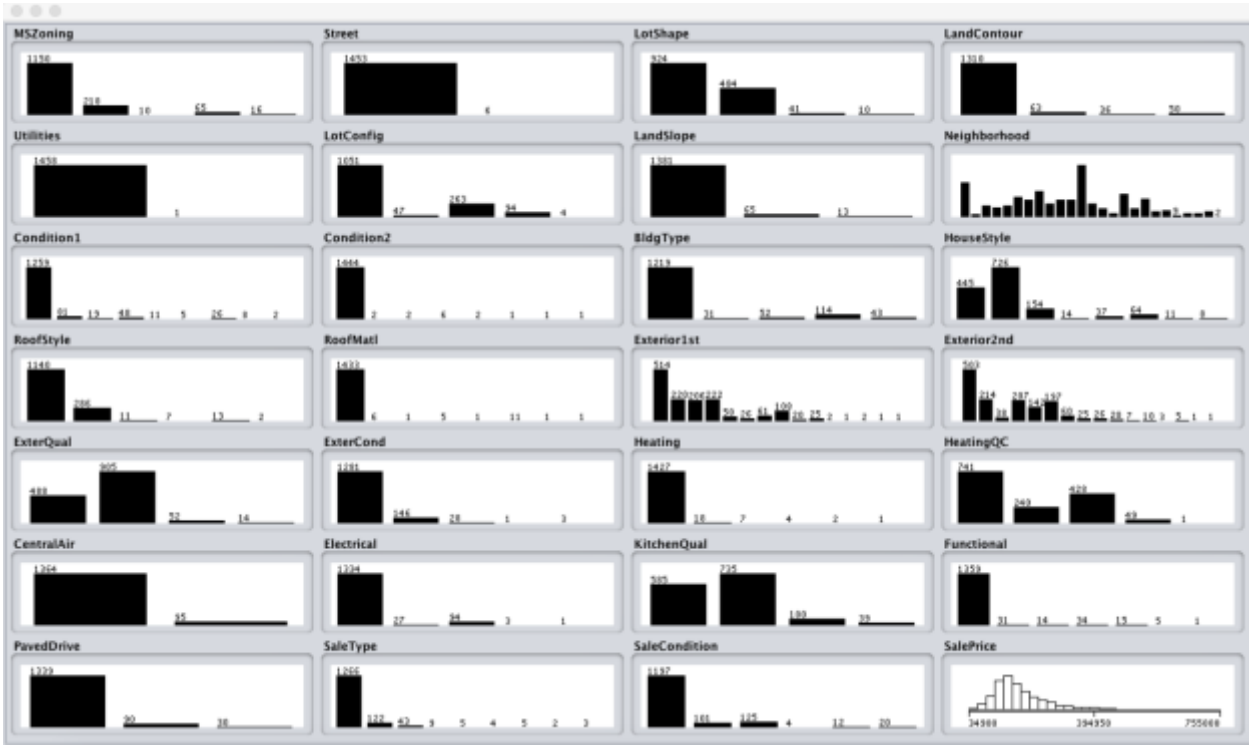
One note on missing values: if the data set and number of attributes were smaller and/or after removing these values and testing training models the results were poorer than expected, then this method would be re-evaluated. Methods such as inserting the mean value or the most dominant categorical value would be further explored. In this case, however, the values returned at the end were well within expected error values, and given the size of the dataset, removing attributes without losing information was one of the most important aspects to the pre-processing procedure.

Determining Numeric Correlation

The remaining dataset was split into those attributes that were numerical in nature, and those that were categorical. There were 34 numerical attributes, including the class attribute of Sale Price. A correlation test was conducted on each attribute comparing it to Sale Price. While the output for each was too verbose to be included here (it is included in the script code, commented under each attribute) this list is a good aggregation of which attributes are statistically associated with Sale Price. From this grouping, it was determined to remove any numerical attribute that showed less than 50% correlation. These included: MSSubClass, LotArea, OverallCond, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, X2ndFlrSF, LowQualFinSF, BsmtFullBath, BsmtHalfBath, HalfBath, BedroomAbvGr, KitchenAbvGr, Fireplaces, WoodDeckSF, OpenPorchSF, EnclosedPorch, X3SsnPorch, PoolArea, MiscVal, MoSold, YrSold. Of all of these, the only surprising non-correlations were from the number of bedrooms (correlation of 16.8%) and the month sold (correlation of 4.6% attributes. Both of these would normally be considered higher in correlation, but for this dataset it was not so.

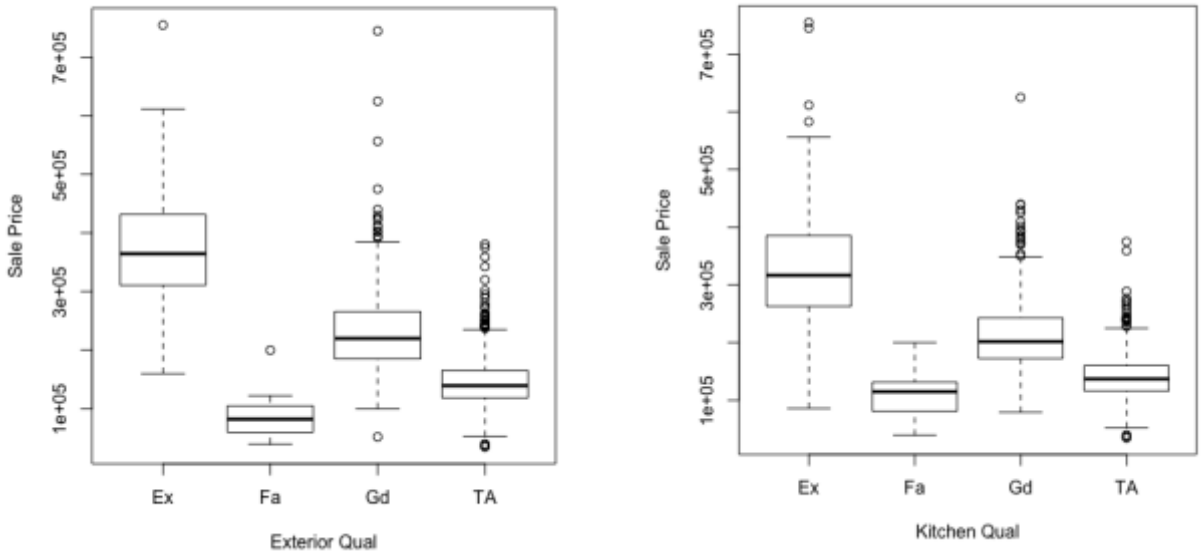
Processing Categorical Attributes

The first step taken in looking at the categorical attributes was to evaluate the variance of each attribute. Each categorical attribute is plotting for visualization in this Weka chart:

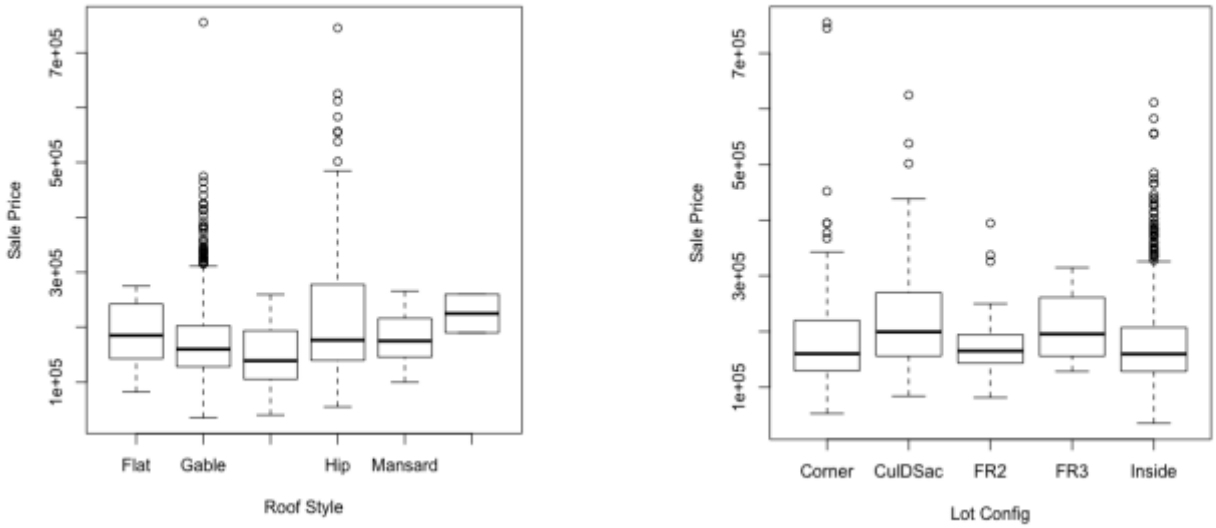


Note: While this is a convenient layout for presentation form, the function in the code at this point creates separate plots for each attribute that is arranged in descending order, which makes it easier to not only determine the variance (or lack thereof) but also linearity when combined with the boxplots, explained below.

Those attributes with a low variance (determined by if one factor accounts for > 85% of all classifications) were removed. The remaining attributes were plotted in boxplots with Sale Price. What was being determined is if there are any linear relationships with Sale Price, or if there may be any other relationship types. Some of the boxplots are shown below as examples. Most of the data was either clearly non-linear (very little difference in the mean Sale Prices across the factors) or linear in nature. If the graph was inconclusive and could be linear, then the attribute was kept and evaluated in other ways shown later.



Example of Linear Relationships

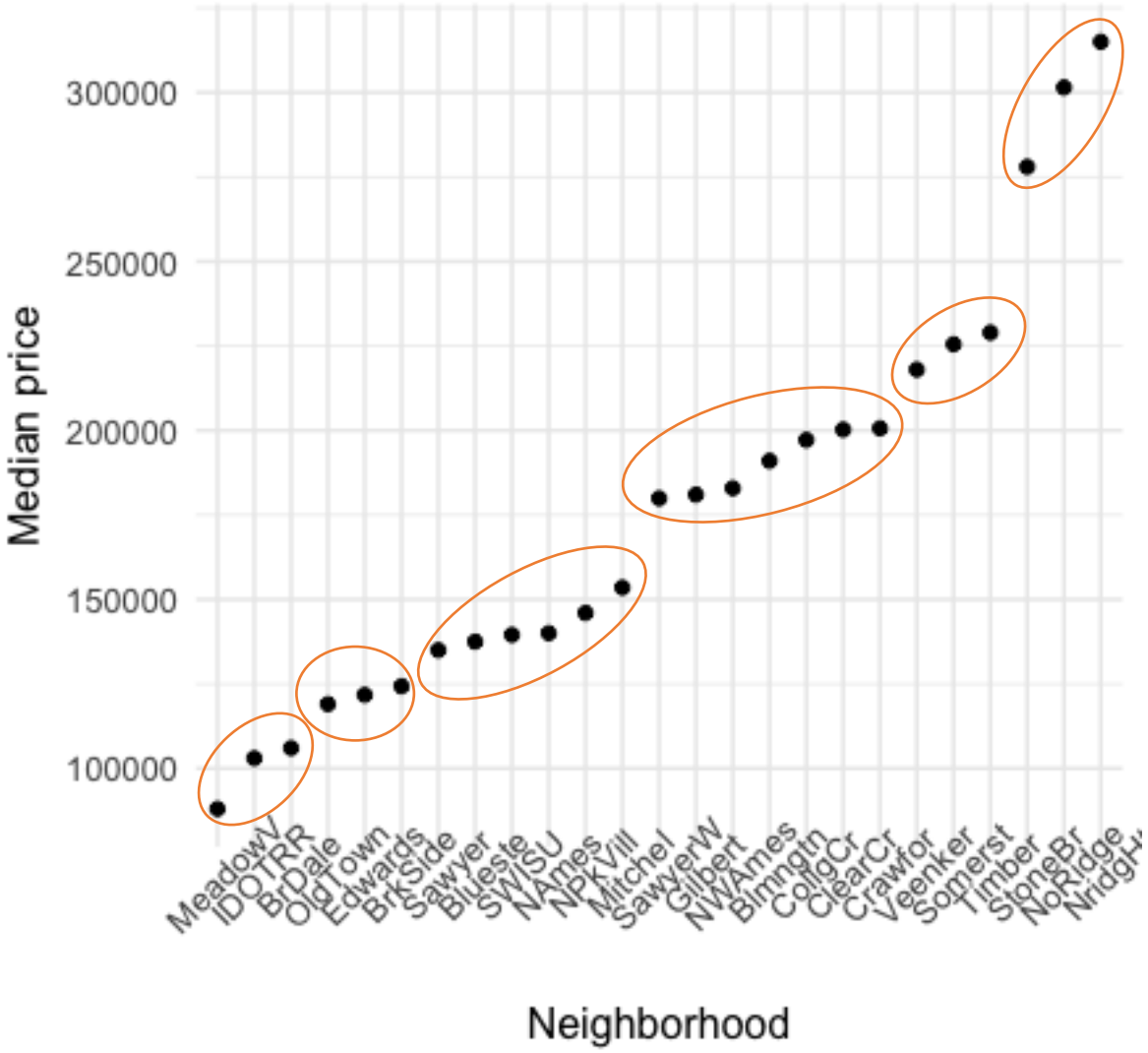


Examples of Non-Linear Relationships

For each boxplot, the evaluation was combined with the categorical plot function to determine linearity. Those that didn't show linearity (LotShape, LotConfig, RoofStyle, Exterior1st, and Exterior2nd) were removed.

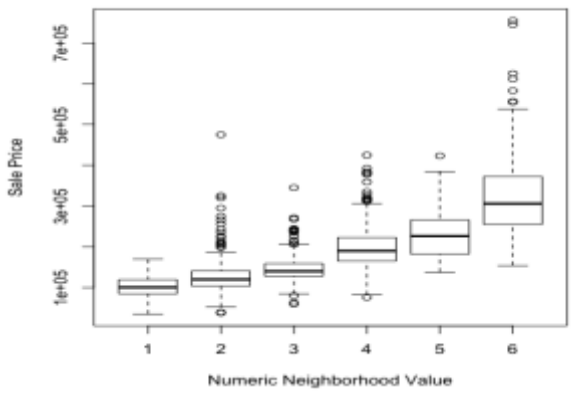
Those categorical attributes left were broken into two groups: those with ordinal factors, and those without. The non-ordinal attributes were each plotted with a function that plots the factors as a function of the median Sale Price. The function then orders the factors in an ascending order to determine if there are specific groupings among the factors. These groupings can then be assigned a numeric value, with the higher values getting higher numbers,

and vice versa. An example is the Neighborhood attribute, which was probably one of the more difficult categorical attributes to work with because of the large number of factors listed:



A plot of the Neighborhood Factors as a function of Median Price, with the groups selected highlighted in red

It was determined that the 6 grouping showed above would represent the data best, and each member of the group was assigned its numeric value accordingly. The resulting boxplot shows the strong linear aspect retained in the new grouping:

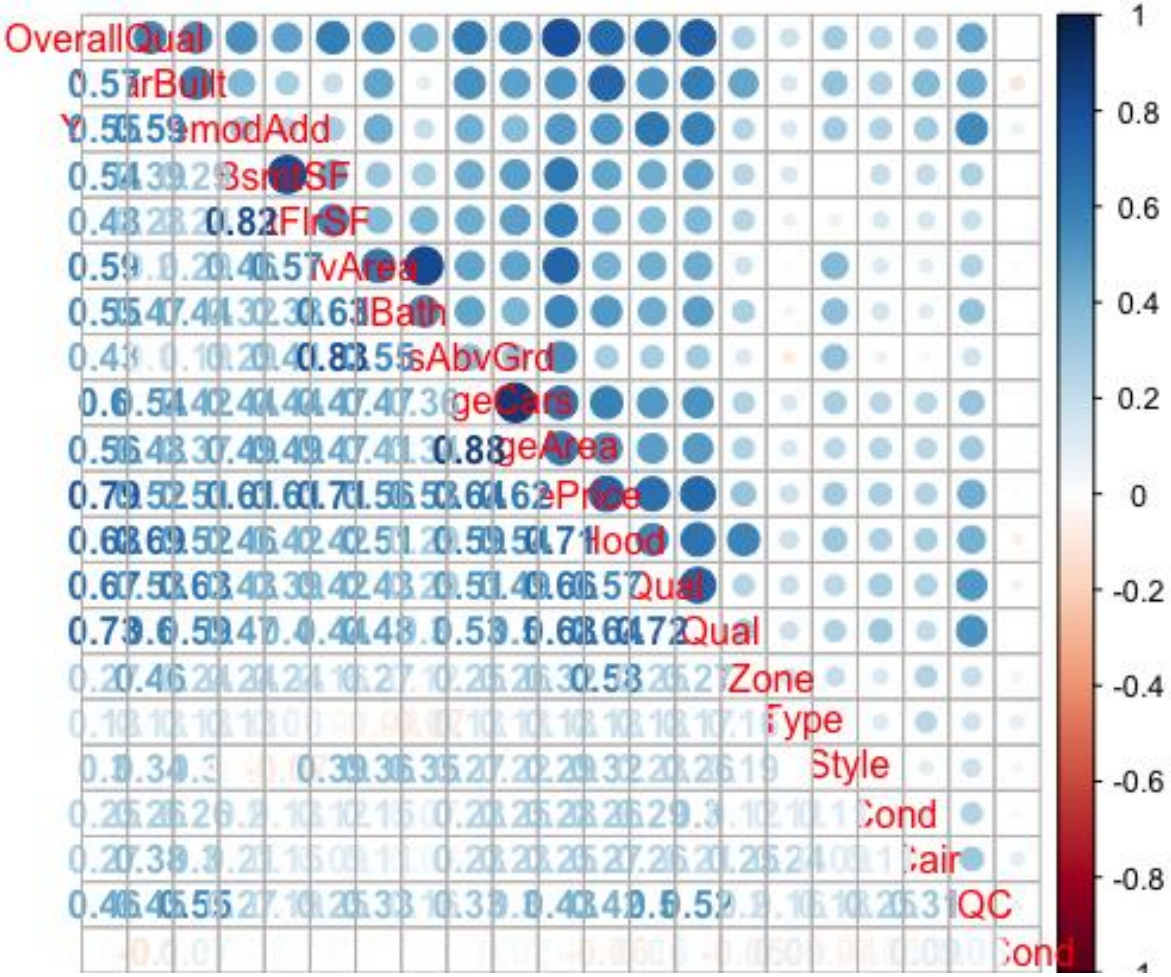


This process was repeated for each non-ordinal attribute, and the resulting groups were refactored into numerical attributes. The only exception was the Sale Condition attribute. This one showed no clear grouping, but was still linear, so each factor was retained and refactored individually.

The ordinal attributes were then numerically refactored based on their values, with the highest values being assigned the higher values, and vice versa. The only exception to this process was Electrical which was a binary attribute. It was assigned a simple 1/0 value based on the yes or no listed. With all of these reassigned, the working part of the dataset is now completely numerical in nature.

Determining New Numeric Correlation and Co-Linearity

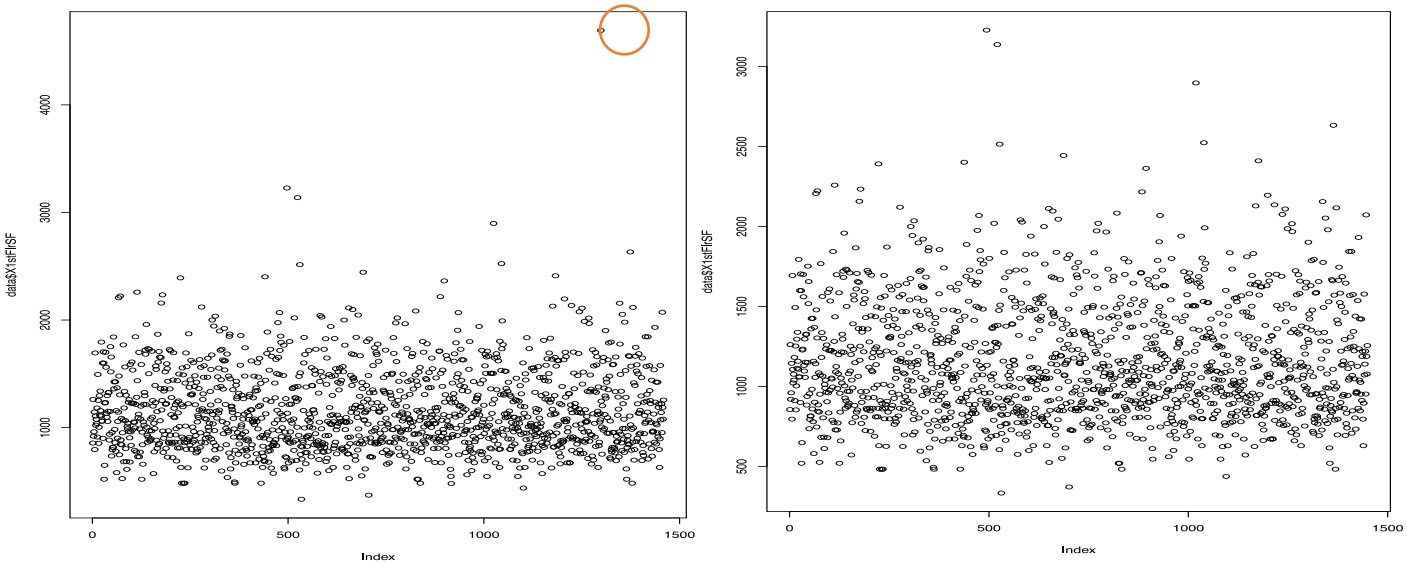
The next step in processing was to plot the new values and numerically determine their correlation with both Sale Price and with each other. The raw numbers of each correlation are shown in the code as a numerical matrix, but this form, although a little hard to read, is slightly more visual and shows some co-linearlity problems in the data:



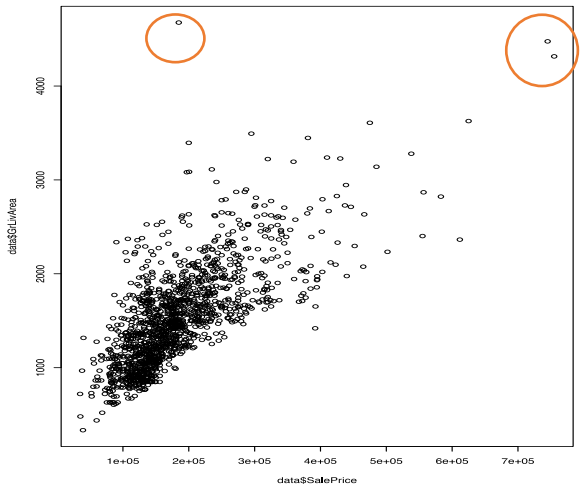
The co-linearity standard was set as an R^2 value of 0.06, which was based on research of typical values. The resulting correlation number was approximately 0.75, so any attributes above this number were singled out for further analysis. Three sets of attributes were found to be above the limit (GarageCars and GarageArea, GrLivArea and TotRmsAbvGrd, and X1stFlrSF and TotBsmtSF). These pairs were compared with each other based on their correlation with Sale Price, and the lower correlated attribute was eliminated (GarageArea, TotRmsAbvGrd were removed). The one exception was that TotBsmtSF was eliminated over X1stFlrSf, even though the basement attribute had a slightly higher correlation. The judgement call was made that the 1st floor square footage would be a more accurate representation overall of the house price over the basement.

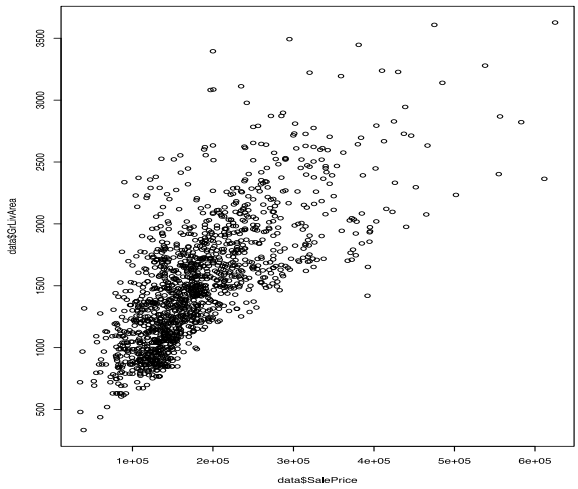
Addressing Outliers

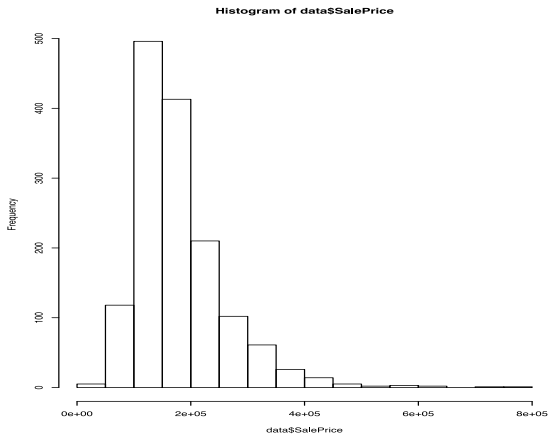
Plotting the remaining attributes, it was identified that several had outlier values that could affect the model during the training phase. The first were 9 tuples that listed that it had no bathrooms. These are clearly erroneous entries and were removed over changing the information, assuming that other entries in this tuple were erroneous. 1st Floor SF's plot showed an outlier as well:

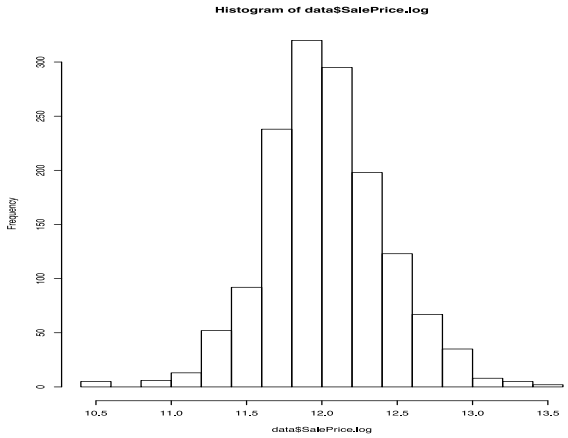


This entry was removed. The resulting scatter plot, shown in the code, results in a much more balanced grouping. There are a couple of further candidates for outliers, but it was determined that they weren't far enough away to overcome the loss of information in removing the tuples. Likewise, GrLivArea showed 3 outliers that were also removed, as shown below. Three attributes were also found to be skewed when their histogram was created. A log transformation on each, however, created a normally distributed graph:
All these steps created a dataset ready for data exploration.

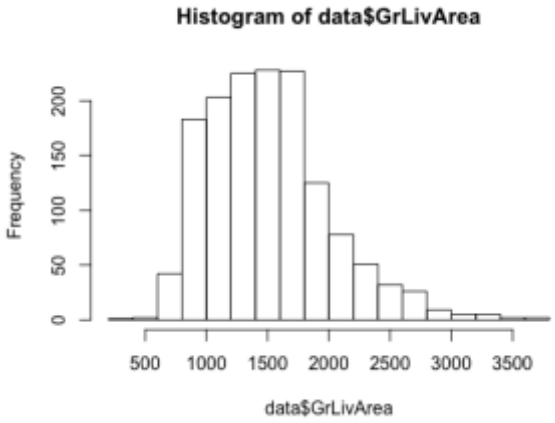


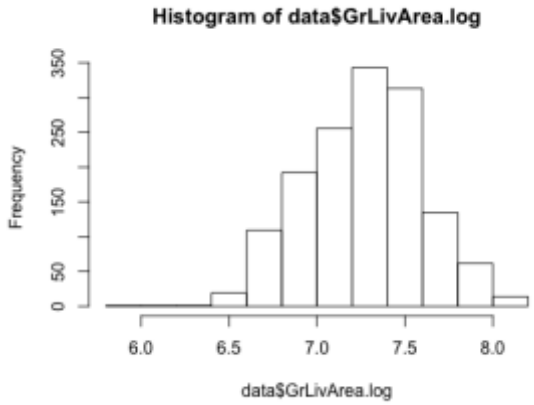




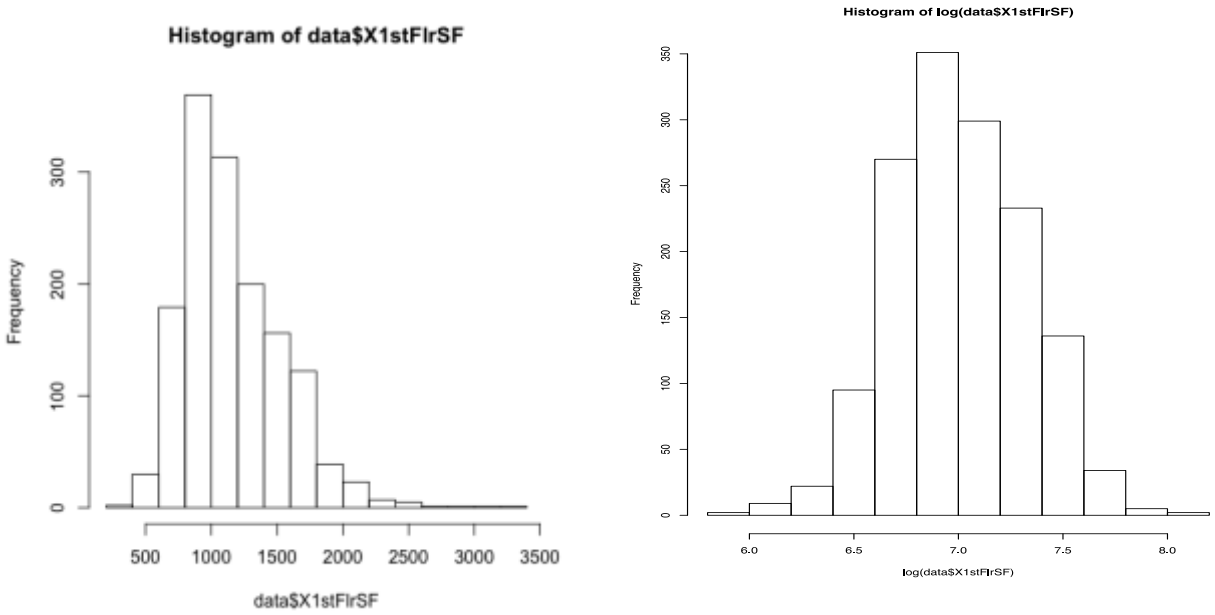


Log Transformation of Sale Price





Log Transformation of GrLivArea



Log Transformation of X1stFlrSF

The final post-processed dataset consists of 1446 tuples, and 17 attributes, including those that have been transformed and excluding their original parent attribute: GrLivArea.log, X1stFlrSF.log, GarageCars, OverallQual, FullBath, YearBuilt, Econd, HQC, CAir, ExQual, KQual, SCond, HStyle, BType, MSZone, NHood, and SalePrice.log. There is no missing information, no visible outliers, all attributes have a solid correlation to the class attribute, and the plot distributions of the attributes appear to be normal.

4. Training Data Models and Data Exploration

The first task during the data exploration was to split the data into a training set and test set. This was split as 60% training, 40% testing. These models will be evaluated on root mean square error (RMSE) in order to compare them using the same standard for determination of performance on testing data. An evaluation function was created based on the RMSE equation:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}}$$

Multiple Linear Regression

Multiple Linear Regression was the first model attempted. The provided code shows two linear regression training and testing attempts using this algorithm, with the first regression included all attributes available, and the second pairing some of the less linear aspects from the first one to see if any improvement is made. The output on the first training model was:

Residuals:

Min	1Q	Median	3Q	Max
-0.81498	-0.07830	0.01236	0.08778	0.49604

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	4.8820346	0.6028303	8.099	1.92e-15 ***
OverallQual	0.0734601	0.0067146	10.940	< 2e-16 ***
YearBuilt	0.0008601	0.0002782	3.092	0.002052 **
FullBath	-0.0230189	0.0138549	-1.661	0.096997 .
GarageCars	0.0565687	0.0097749	5.787	1.01e-08 ***
NHood	0.0442945	0.0072682	6.094	1.66e-09 ***
MSZone	0.0337883	0.0079595	4.245	2.43e-05 ***
BType	0.0221757	0.0140872	1.574	0.115819
HStyle	-0.0114143	0.0094874	-1.203	0.229272
SCond	0.0394109	0.0116417	3.385	0.000743 ***
KQual	0.0584935	0.0117772	4.967	8.23e-07 ***
ExQual	-0.0063129	0.0149774	-0.421	0.673499
HQC	0.0256137	0.0064746	3.956	8.25e-05 ***
X1stFlrSF.log	0.1664746	0.0224675	7.410	3.05e-13 ***
GrLivArea.log	0.4170317	0.0282454	14.765	< 2e-16 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1465 on 852 degrees of freedom
Multiple R-squared: 0.8695, Adjusted R-squared: 0.8673
F-statistic: 405.4 on 14 and 852 DF, p-value: < 2.2e-16

The R^2 of 0.8673 is not bad for the first attempt, and clearly there are many of the attributes strongly correlated with our class attribute, but the RMSE must be calculated for comparison with other modes. The training RMSE used is created just for comparison to the testing RMSE to make sure there are no major differences and/or potential red flags to potential problems. The training RMSE was 0.14526, and using a predict function with our trained model, the RMSE was 0.14737, which coincides with expected values.

One more multiple linear regression training model was created to explore the idea that there may be an improvement in the testing metrics. On the second run, those attributes which didn’t have individual significance (i.e. they didn’t have low p-values). The exclusions were: FullBath, BType, HStyle, and ExQual. The resulting regression model calculated is shown below:

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	5.4903263	0.5578664	9.842	< 2e-16 ***
OverallQual	0.0740926	0.0064666	11.458	< 2e-16 ***
YearBuilt	0.0006267	0.0002623	2.389	0.017100 *
GarageCars	0.0572779	0.0097442	5.878	5.94e-09 ***
NHood	0.0420817	0.0070867	5.938	4.19e-09 ***
MSZone	0.0356403	0.0078855	4.520	7.06e-06 ***
SCond	0.0401235	0.0116385	3.447	0.000593 ***
KQual	0.0581652	0.0109876	5.294	1.52e-07 ***
HQC	0.0256156	0.0064037	4.000	6.88e-05 ***
X1stFlrSF.log	0.1843498	0.0195527	9.428	< 2e-16 ***
GrLivArea.log	0.3752617	0.0217759	17.233	< 2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

#Residual standard error: 0.1469 on 856 degrees of freedom
#Multiple R-squared: 0.8683, Adjusted R-squared: 0.8667
#F-statistic: 564.2 on 10 and 856 DF, p-value: < 2.2e-16

Based off of R-value, the overall performance dropped, suggesting an over-fitting may be close or occurring in this model. This is even more likely by looking at the calculated training RMSE of 0.14594 and comparing it to the test RMSE of 0.14927. Since there is no improvement, and in fact a slightly worse outcome, other predictive models were then explored.

XGBoost

XGBoost is a popular gradient boosting algorithm that uses an ensemble of weak learners, in this case decision trees, then generalizes by optimizing the loss function created. As the attached code shows, the XGBoost process involves primarily optimizing each of the given parameters of the algorithm for best output. Most of the values in this process were the given values, with some changed based on suggestions from online posts on this process. This optimization process can take some time, and is outside the full scope of this project. An interesting aspect to this package is the option to create a k-folds training model and see how the training model RMSE compares to a testing RMSE. For continuous comparison purposes this was not the training model that was used for final comparison, it does give a good idea of how the model will perform overall given the training set. The output for the k-folds evaluation is shown here:

iter	train_rmse_mean	train_rmse_std	test_rmse_mean	test_rmse_std
1	10.1232660	0.006196933	10.1232442	0.01847364
2	8.8814865	0.004946893	8.8814628	0.01957221
3	7.7930140	0.005427486	7.7929877	0.01912428
4	6.8387332	0.005073198	6.8387045	0.01948291
5	6.0016728	0.004306680	6.0016407	0.02042116

1996	0.1103818	0.001418320	0.2036537	0.01288824
1997	0.1103750	0.001417858	0.2036495	0.01286756
1998	0.1103690	0.001422625	0.2036345	0.01283468
1999	0.1103660	0.001424034	0.2036190	0.01283953
2000	0.1103640	0.001423274	0.2036152	0.01287118

The actual predicted model was developed, and the test data was used to predict values. The resulting RMSE score was 0.175888. Given the popularity of using this approach for regression analysis such as this, it would suggest that either the data was not optimized for this particular approach, or that the parameter optimization suggested above may need some work. Clearly, the multiple linear regression model wins by a significant margin against this method.

Random Forest

The next algorithm model evaluated was the random forest approach. This is another ensemble learning method that creates multiple random decision trees and essentially averages the outputs together. As in XGBoost, there are a myriad of parameters that should be optimized in their own right to get the best model possible out of this package, but this model was run with all default parameters. A random seed was introduced into this part of the code so that it could be reproduced, as there is a stochastic aspect to this algorithm. The output from the model training is shown below:

```
 randomForest(formula = SalePrice.log ~ ., data = data.train)
       Type of random forest: regression
             Number of trees: 500
No. of variables tried at each split: 4

       Mean of squared residuals: 0.02301841
             % Var explained: 85.76
```

Because MSE is given as part of the training output, a training RMSE can be calculated as 0.1520. Using the prediction process on the test data as before, the calculated test RMSE is 0.1394, which is clearly a better outcome than any of the other models attempted so far.

Generalized Linear Models

The final model evaluated is Generalized Linear Models, which is a variation of linear regression that allows for some correction to attributes that don't quite have a normal distribution. Each variant used in this model training uses much more complex linear regression techniques: ridge focuses on an ordinary least squares approach, lasso performs variable selection and regularization, and elastic net is a more generalized version of lasso that is supposed to address some of the short-comings from lasso. These are all housed in one package, so they can easily be compared to one another.

The training process revolves around determining the best learning rate, λ. The first part of this code uses a cross validation method to determine what the best learning rate should be for each type. This information is then fed into the algorithm with the training data to create the trained model. As in the other cases, this model is tested against the test data, and the RMSE of each is measured:

Ridge RMSE: 0.06113
Lasso RMSE: 0.01216
Elastic Net RMSE: 0.06109

5. Conclusion

The results from the GLM models are much better than any of the previous model attempts by a significant margin. The outcome from this model set compared to the others suggests that perhaps the data, even after the processing steps, has some irregular data characteristics. Due to the complex nature of a dataset with all numeric regression values, it may also be advisable to perform boosting methods with differing parameter settings to determine if a better outcome can be achieved. These appear to have great influence over the model accuracy, and could yield a better outcome than the GLM's.

That being said, compared to the results from others that ran their models against a test data set provided by the site, the lasso model trained from this process, if the test RMSE output holds true, would be one of the top 10 of those submitted. Even with a small loss of accuracy due to randomness or unknown causes, it would still be one of the better choices for this particular data set.

I have learned a tremendous amount throughout this project. From experience cleaning a large data set and determining what that entails, to exploring different classifying algorithms and their strengths and weaknesses and how the algorithm parameters are implemented can greatly affect the accuracy of the model. I plan at some point to submit the lasso model online and see how it actually compares to other models.