Ames Iowa House Prediction Project

Jordan Kassof and Jose Torres

2/17/2018

Contents

Introduction

Buying a house is one of the largest financial decisions many people will make. So many factors go into someone's decision, but it can be hard to really explain why one house "felt right" and another didn't. We want to quantify the factors that add up to someone making the decision to purchase a house.

Data Description

We will be using the Ames, Iowa individual residential property sales data set freely available on Kaggle.com. The data set contains 2,930 observations with 79 explanatory variables. All observations occur between 2006 and 2010. Given the geography dependent nature of home value, the results of below analyses can't be applied nationally. For more information on the data, or to download it yourself, visit https://www.kaggle.com/c/house-prices-advanced-regression-techniques.

See the codebook.txt file in the github repository for complete information about all variables.

Exploratory Analysis

Our exploratory analysis was focused on getting a sense for the (numerous) variables in the dataset. We wanted to understand the marginal distributions of individual variables, the relationship between those individual variables and the sale price, and the correlation between the variables themselves.

In `cleaner_script.R` (Appendix C), the training and test sets are cleaned up and plots of variable histograms and scatter/box-plots are created for Sale Price vs each variable. View the plots here.

There are several features of a home that are not present for all homes in the data set, most notably various area measurements for home that don't have those features. There are 14 indicator variables which were created and added to the data set. The purpose of these variables is to allow flexible fits for features which are missing in some homes.

Questions of Interest

We focused on two approaches, one geared towards model performance and one towards model interpretability by one of the parties involves in a home purchase.

Interpretable Models

For the interpretable model approach, rather than just taking a handful of easily understood parameters and building a model, we wanted to take a different approach. There are many adages when it comes to home buying, we wanted to see which are the most true. The three ideas about what drives the price of a house that we looked at are as follows:

- Location, location, location!
  - For this model, we used parameters that are related to the physical location of the property. For example, neighborhood, zoning, frontage, lot size, etc.
- It's all about the curb appeal
  - For this model, we used parameters related to the external appearance of the property. For example, house style, roof style, external veneer materials, etc.
- It's what's on the inside that counts
  - For this model, we used parameters related to the internals of the property, the bones if you will. For example, the foundation, the electrical and heating system, etc.

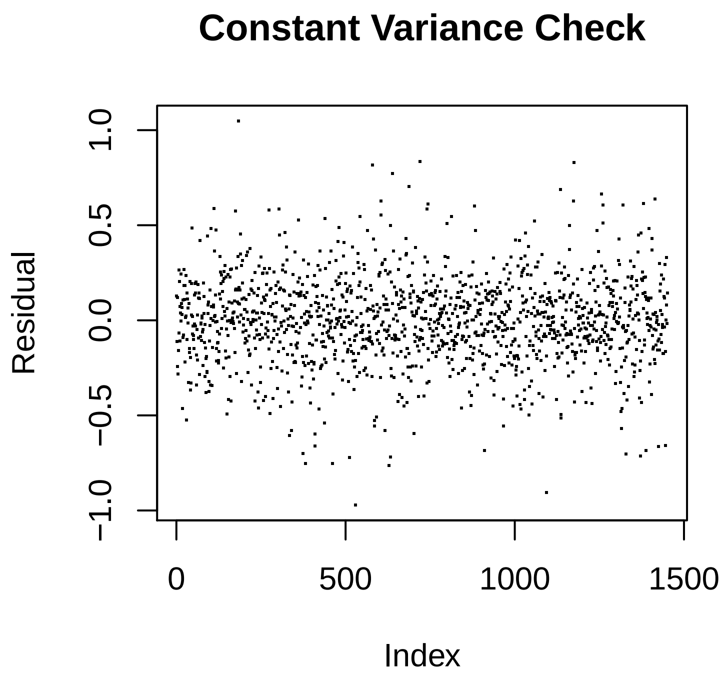
We also included the Sale Condition variable in all three models because the context of the sale seems like way too key of a factor to leave out of any model that is meant to be easily interpreted.

Model Selection

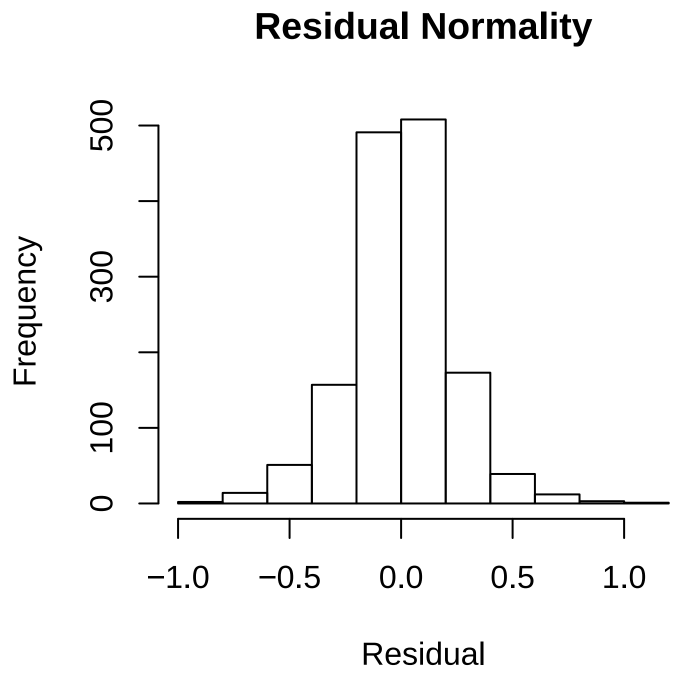
In `scripts/interp_models.R` we went through the process of selecting out variables related to each category, running OLS regression on those sets of variables. We extracted parameter estimates and diagnostic metrics for each model and compared across several values.

ModelName	adj.r.squared	AIC	BIC	df
location	0.6561550	-107.59984	130.0005	44
inside	0.6460567	-43.51478	315.5258	67
outside	0.5968436	117.47741	323.3977	38

Based on R^2, AIC, and BIC, the best model appears to be the location model. Let's examine some diagnostic plots to make sure the assumptions of linear regression are met.



There is not strong evidence against the assumption of constant variance in our residuals. I see a few points that have potentially high leverage, but nothing too egregious so we can proceed.

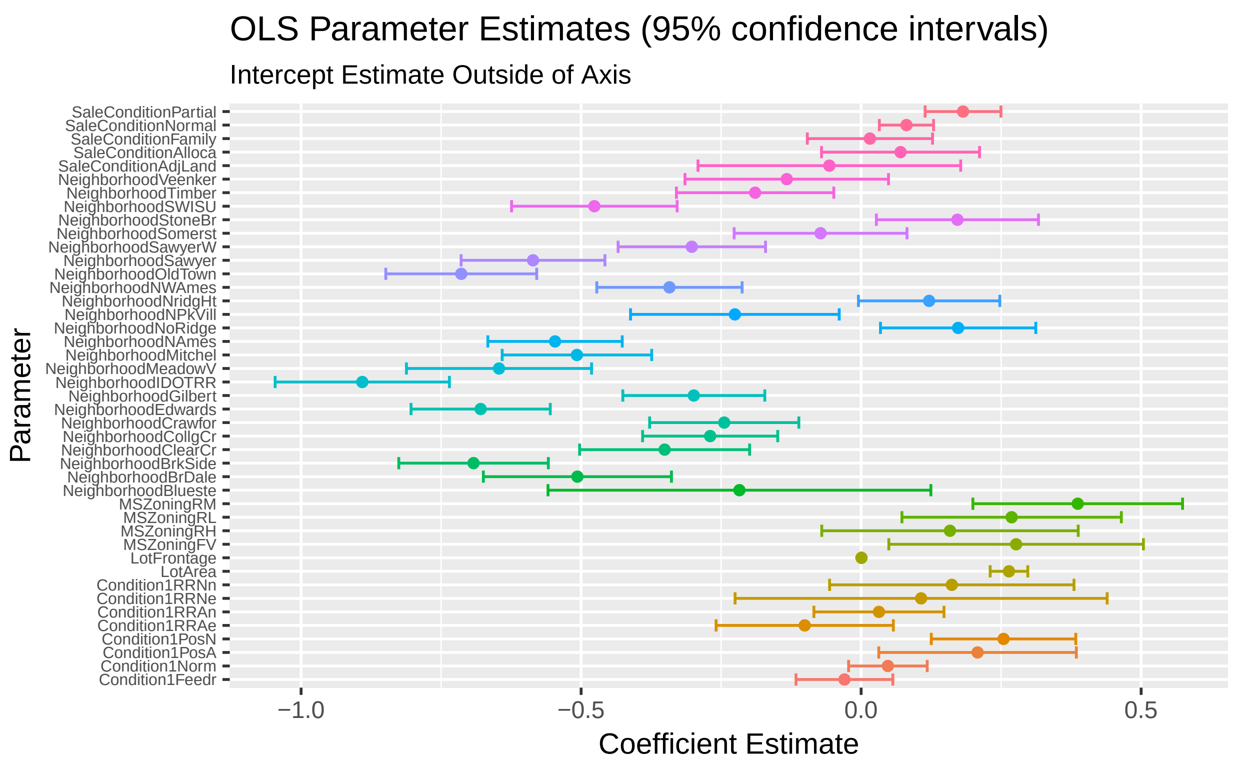


This histogram shows a symmetric distribution, and does not provide strong evidence against normality.

Our final assumption of independent observations, we will assume the data collection was conducted in a way that will provide independent observations. This assumption is probably the weakest as property values within a given city (and more so within a given neighborhood) are fairly dependent on each other. We will proceed with caution.

Parameter Interpretation

Below we will take a cursory look at all parameter estimates, then discuss a few of the parameters with the strongest influence on sale price.



The neighborhood variable has quite a lot of levels, but we can get an idea here for all the parameters estimates and a 95% confidence interval. Let's take a closer look at the five variables with the largest coefficient estimates.

term	estimate	std.error	statistic	p.value	conf.low	conf.high
MSZoningRM	0.3867477	0.0954183	4.053181	0.0000533	0.1995702	0.5739251
MSZoningFV	0.2767382	0.1158749	2.388251	0.0170600	0.0494321	0.5040443
MSZoningRL	0.2687524	0.0998678	2.691081	0.0072065	0.0728465	0.4646583
LotArea	0.2640079	0.0171241	15.417344	0.0000000	0.2304164	0.2975994
Condition1PosN	0.2542930	0.0657607	3.866943	0.0001152	0.1252934	0.3832927

It appears that the zoning variable has the strongest effect on price, as does the size of the lot. The RM Level of the zoning variable means "Residential Medium Density." Bearing in mind that we did a log transform to the sale price (which makes this a log-linear model), we can estimate that a property in this zoning area increases the median sale price by a multiplicative factor of $e^{.468} = 1.5968$. A 95% confidence interval for that effect is $[e^{.301} = 1.3512, e^{.634} = 1.8851]$ Similarly, zoning classification FV (floating village residential) indicates an $e^{.354} = 1.4248$ multiplier to the median sale price. Our most influential continuous variable, the area of the lot the proper is built on, gives a $e^{.27} = 1.31$ multiplier to the median sale price for each unit (acre) increase.

Predictive Models

Introduction and Type of Selection

The goal of this section is to make as performant a model as possible. We are not trying to be strictly interpretable or parsimonious. We are primarily focused on optimizing on Kaggle score and model fit statistics.

The process of model optimization begins with an early analysis and review of basic regression assumptions, followed by comparing models across various fit statistics: adjusted R^2, Akaike Information Criterion (AIC), Bayesian Information Criterion (SBC), internal and external average squared error (ASE), and internal cross validation partial residual sum of squares (cv press). Outliers are excluded where necessary, and interaction terms and polynomial flexibility is considered.

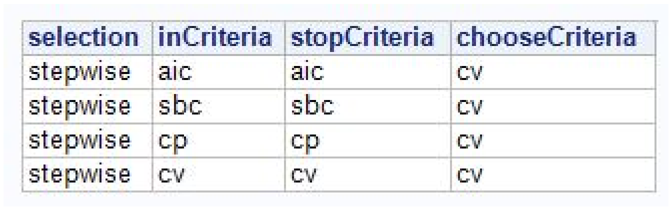
Predictive models analyzed here are limited to one of four types of penalty-based regression estimation: stepwise selection (penalty-free least-squares estimation), modified forward selection via least angle regression selection (lar/lars), least absolute shrinkage and selection operator regression (lasso), and elastic net regression. More information can be found on these regression types by opening the `data/lars.pdf` file in this repo.

SAS and R will both be used throughout this process. The latter is used to clean and merge training and test data sets which are exported into SAS for regression estimation. As mentioned above in the Exploratory Analysis section, The R script used for this is `cleaner_script.R`.

Early Analysis and Assumption Review

Because of the high number of possible explanatory variables, an initial regression estimation phase is performed in order to ascertain which regression selections tend to minimize key statistics, which will reveal a tentative model to analyze for the purpose of evaluating assumptions.

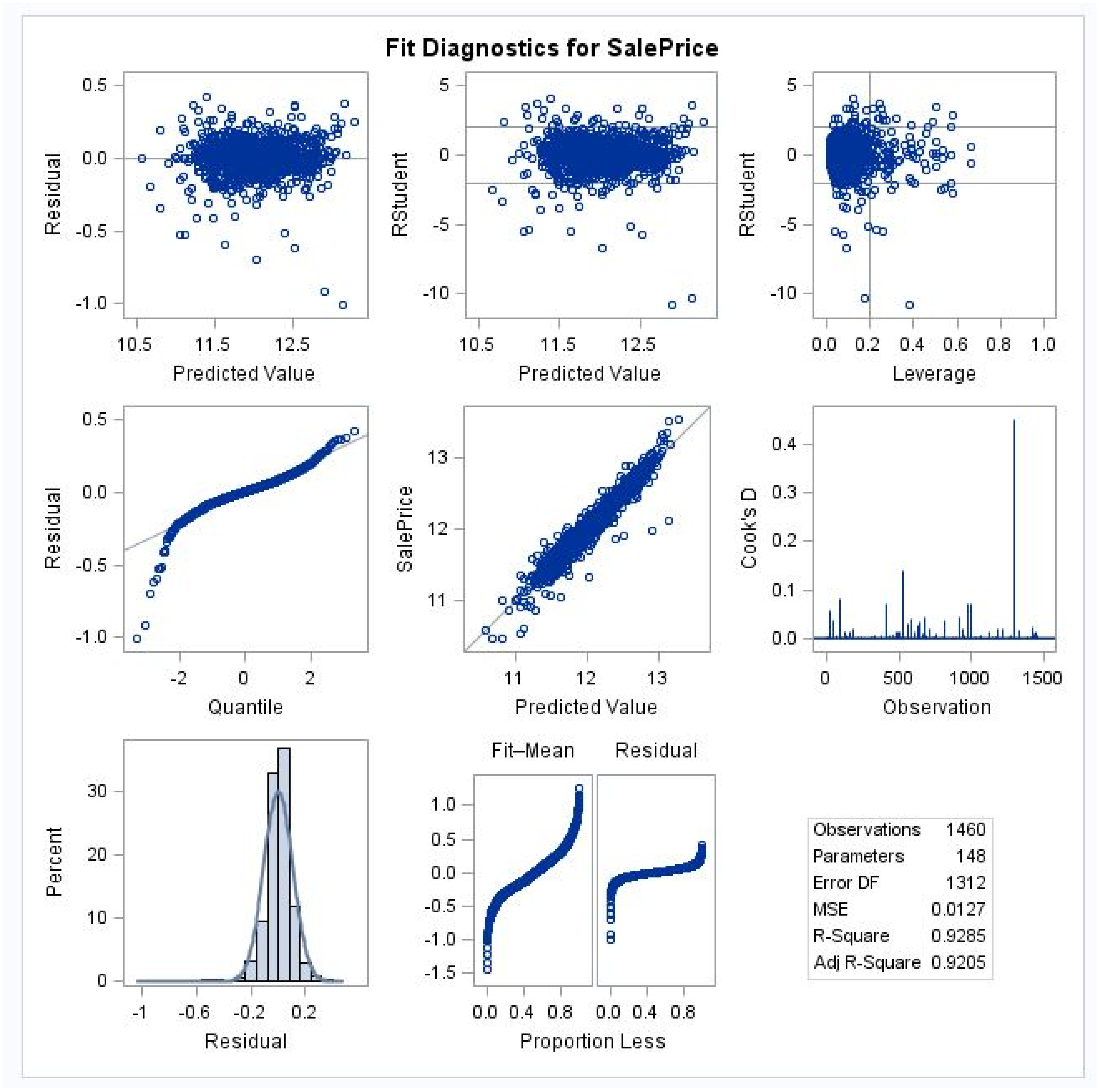
The following 4 models are compared to identify most common predictors to use for assumption review. We will use the SPLIT option in SAS to allow for each factor level to enter or exit the model independently.



selection	inCriteria	stopCriteria	chooseCriteria
stepwise	aic	aic	cv
stepwise	sbc	sbc	cv
stepwise	cp	cp	cv
stepwise	cv	cv	cv

Since the second model, based on selecting according to lowest BIC, has the fewest predictors, we will proceed with that and check residuals for normality and constant variance. These predictors are shown in Appendix D, Figure 2.

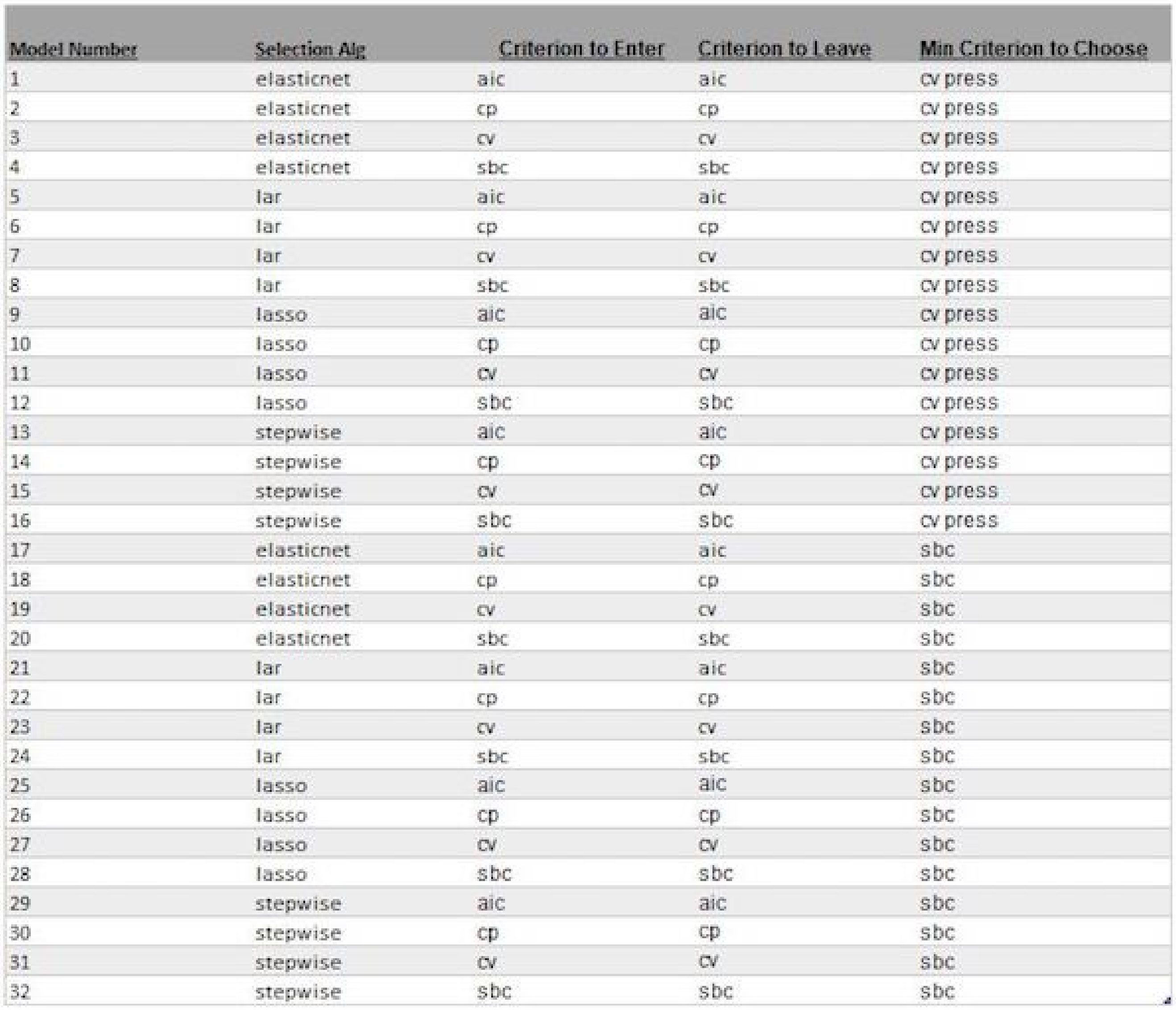
The residuals displayed in the panel below show excellent conformity with the constant variance and normally-distributed residual assumptions. There is some evidence of outlier presence in the studentized residual vs predicted value plot (Row 1, Column 2), which will be addressed in the section below on outlier analysis. We also assumed that the homes included in the training set are independent. More on this is discussed in the section `Additional Improvements` below.



Observations	1460
Parameters	148
Error DF	1312
MSE	0.0127
R-Square	0.9285
Adj R-Square	0.9205

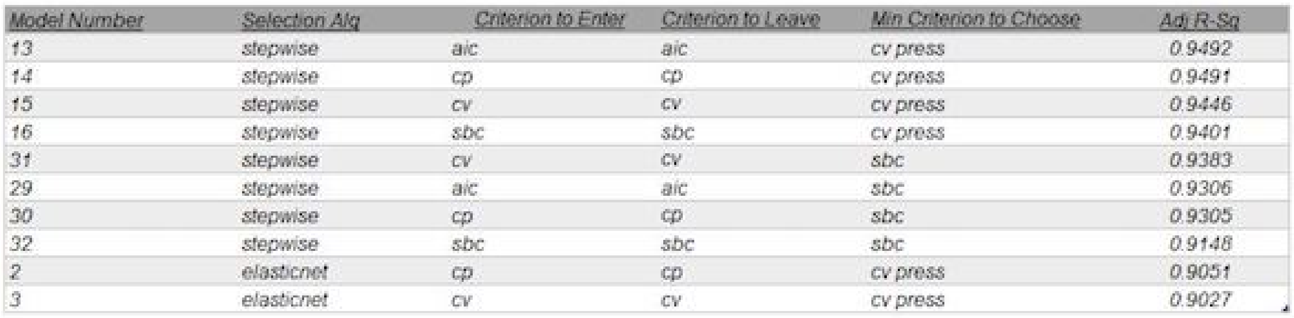
Model Selection: First Analysis

For the first pass model selection phase, the regression permutations in Figure 1 above are expanded to include LARS, LASSO, and Elastic Net. There are two choose options used: internal cv press or BIC. The table below shows all of the model selection permutations used in this modeling phase.

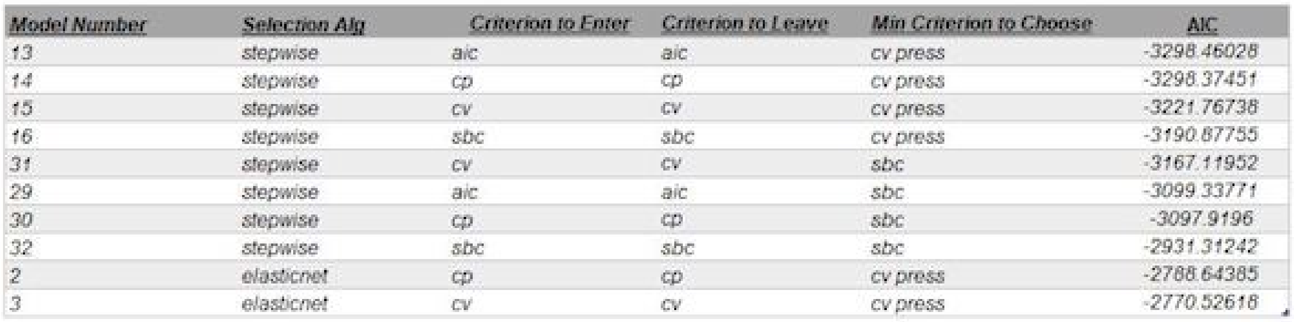


Model Number	Selection Alg	Criterion to Enter	Criterion to Leave	Min Criterion to Choose
1	elasticnet	aic	aic	cv press
2	elasticnet	cp	cp	cv press
3	elasticnet	cv	cv	cv press
4	elasticnet	sbc	sbc	cv press
5	lar	aic	aic	cv press
6	lar	cp	cp	cv press
7	lar	cv	cv	cv press
8	lar	sbc	sbc	cv press
9	lasso	aic	aic	cv press
10	lasso	cp	cp	cv press
11	lasso	cv	cv	cv press
12	lasso	sbc	sbc	cv press
13	stepwise	aic	aic	cv press
14	stepwise	cp	cp	cv press
15	stepwise	cv	cv	cv press
16	stepwise	sbc	sbc	cv press
17	elasticnet	aic	aic	sbc
18	elasticnet	cp	cp	sbc
19	elasticnet	cv	cv	sbc
20	elasticnet	sbc	sbc	sbc
21	lar	aic	aic	sbc
22	lar	cp	cp	sbc
23	lar	cv	cv	sbc
24	lar	sbc	sbc	sbc
25	lasso	aic	aic	sbc
26	lasso	cp	cp	sbc
27	lasso	cv	cv	sbc
28	lasso	sbc	sbc	sbc
29	stepwise	aic	aic	sbc
30	stepwise	cp	cp	sbc
31	stepwise	cv	cv	sbc
32	stepwise	sbc	sbc	sbc

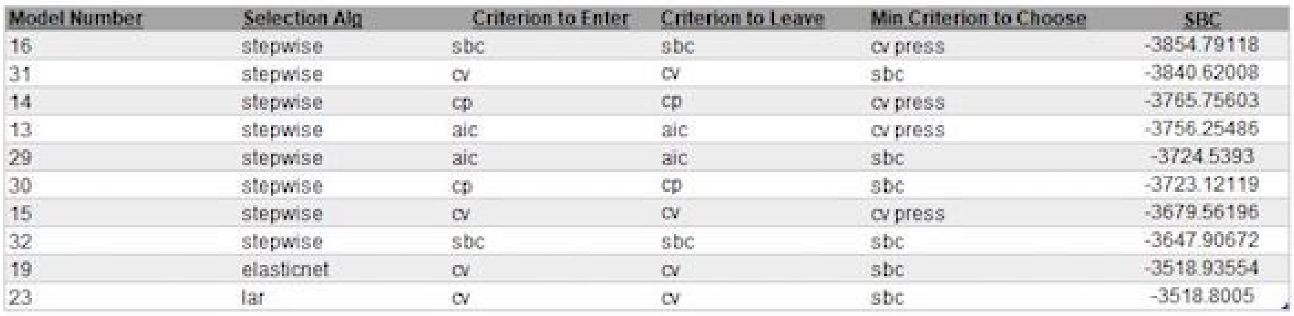
The next set of tables showcase the top 10 models with respect to each of the six target statistics. Note that 40% of the training data was held out as test data.



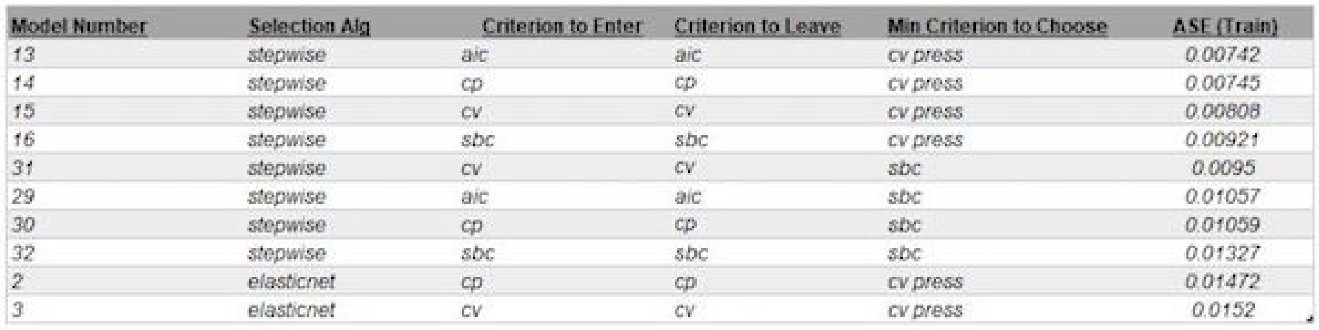
Model Number	Selection Alg	Criterion to Enter	Criterion to Leave	Min Criterion to Choose	Adj R-Sq
13	stepwise	aic	aic	cv press	0.9492
14	stepwise	cp	cp	cv press	0.9491
15	stepwise	cv	cv	cv press	0.9446
16	stepwise	sbc	sbc	cv press	0.9401
31	stepwise	cv	cv	sbc	0.9383
29	stepwise	aic	aic	sbc	0.9306
30	stepwise	cp	cp	sbc	0.9305
32	stepwise	sbc	sbc	sbc	0.9148
2	elasticnet	cp	cp	cv press	0.9051
3	elasticnet	cv	cv	cv press	0.9027



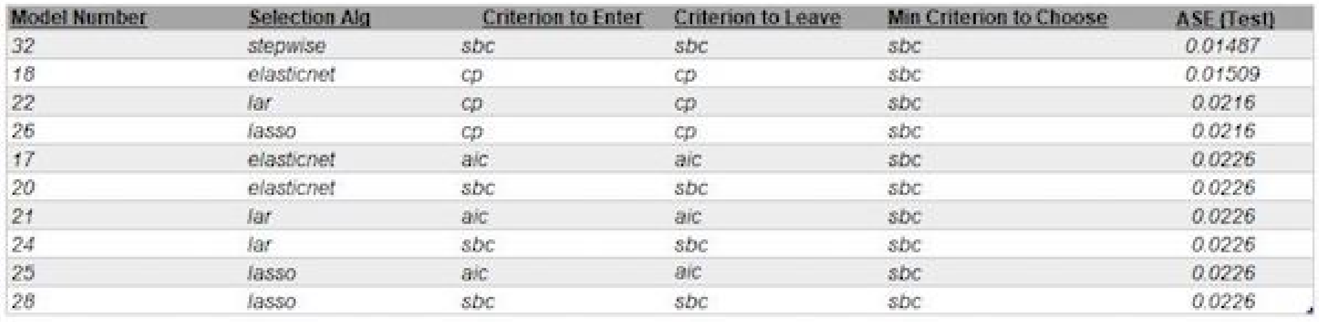
Model Number	Selection Alg	Criterion to Enter	Criterion to Leave	Min Criterion to Choose	AIC
13	stepwise	aic	aic	cv press	-3298.46028
14	stepwise	cp	cp	cv press	-3298.37451
15	stepwise	cv	cv	cv press	-3221.76738
16	stepwise	sbc	sbc	cv press	-3190.87755
31	stepwise	cv	cv	sbc	-3167.11952
29	stepwise	aic	aic	sbc	-3099.33771
30	stepwise	cp	cp	sbc	-3097.9196
32	stepwise	sbc	sbc	sbc	-2931.31242
2	elasticnet	cp	cp	cv press	-2788.64385
3	elasticnet	cv	cv	cv press	-2770.52618



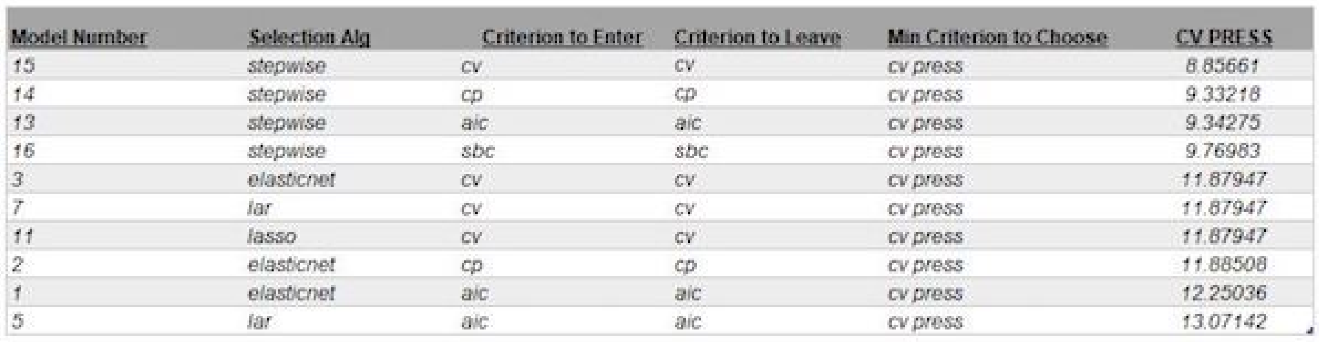
Model Number	Selection Alg	Criterion to Enter	Criterion to Leave	Min Criterion to Choose	SBC
16	stepwise	sbc	sbc	cv press	-3854.79118
31	stepwise	cv	cv	sbc	-3840.62008
14	stepwise	cp	cp	cv press	-3765.75603
13	stepwise	aic	aic	cv press	-3756.25486
29	stepwise	aic	aic	sbc	-3724.5393
30	stepwise	cp	cp	sbc	-3723.12119
15	stepwise	cv	cv	cv press	-3679.56196
32	stepwise	sbc	sbc	sbc	-3647.90672
19	elasticnet	cv	cv	sbc	-3518.93554
23	lar	cv	cv	sbc	-3518.8005



Model Number	Selection Alg	Criterion to Enter	Criterion to Leave	Min Criterion to Choose	ASE (Train)
13	stepwise	aic	aic	cv press	0.00742
14	stepwise	cp	cp	cv press	0.00745
15	stepwise	cv	cv	cv press	0.00808
16	stepwise	sbc	sbc	cv press	0.00921
31	stepwise	cv	cv	sbc	0.0095
29	stepwise	aic	aic	sbc	0.01057
30	stepwise	cp	cp	sbc	0.01059
32	stepwise	sbc	sbc	sbc	0.01327
2	elasticnet	cp	cp	cv press	0.01472
3	elasticnet	cv	cv	cv press	0.0152



Model Number	Selection Alg	Criterion to Enter	Criterion to Leave	Min Criterion to Choose	ASE (Test)
32	stepwise	sbc	sbc	sbc	0.01487
18	elasticnet	cp	cp	sbc	0.01509
22	lar	cp	cp	sbc	0.0216
26	lasso	cp	cp	sbc	0.0216
17	elasticnet	aic	aic	sbc	0.0226
20	elasticnet	sbc	sbc	sbc	0.0226
21	lar	aic	aic	sbc	0.0226
24	lar	sbc	sbc	sbc	0.0226
25	lasso	aic	aic	sbc	0.0226
28	lasso	sbc	sbc	sbc	0.0226



Model Number	Selection Alg	Criterion to Enter	Criterion to Leave	Min Criterion to Choose	CV PRESS
15	stepwise	cv	cv	cv press	8.85661
14	stepwise	cp	cp	cv press	9.33218
13	stepwise	aic	aic	cv press	9.34275
16	stepwise	sbc	sbc	cv press	9.76983
3	elasticnet	cv	cv	cv press	11.87947
7	lar	cv	cv	cv press	11.87947
11	lasso	cv	cv	cv press	11.87947
2	elasticnet	cp	cp	cv press	11.88508
1	elasticnet	aic	aic	cv press	12.25036
5	lar	aic	aic	cv press	13.07142

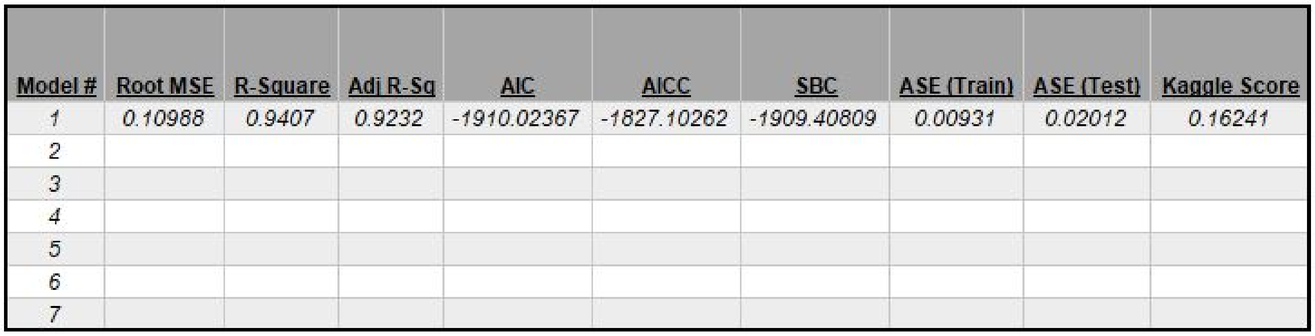
Surprisingly, the basic stepwise algorithm that is often heralded as being inferior to penalty-based regression selection methods performed the best overall on all accounts except for the Test ASE. A figure containing all of the predictors for the above 32 model fits is not included because this is considered exploratory. However, the SAS code provided in `scripts/SAS_Early_Examination.txt` can be used to recreate these exact fits using `SEED=12345` option.

Because the Test ASE is an important statistic to optimize, we cannot only rely on Stepwise selection. The approach used was an "average predictor" approach in which the frequency that each predictor is selected contributes to its score, with the top scoring predictors across all model selection permutations being used as the set of predictors to include in the regression model. Furthermore, the training data was not partitioned here to ensure that as much data as possible is being used to establish a set of initial predictors.

Top predictors are defined as having a score of 4 or more. The score is the average number of times the predictor was selected. The levels for any categorical predictor that meets this criteria will be examined to identify potential groupings of levels that offer additional degrees of freedom. This also helps reduce the possibility that our factor level estimates represent training data noise and not a real difference in the predicted sale price.

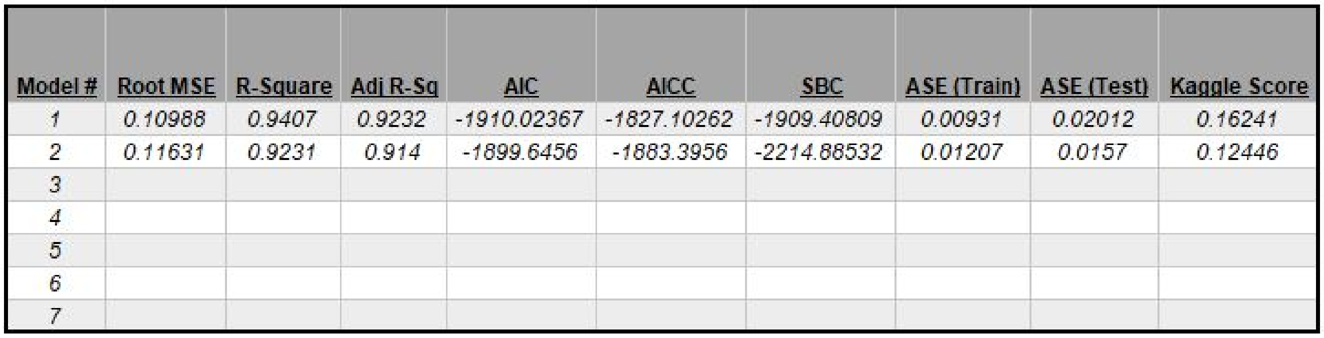
The unique list of factors in Appendix D, Figure 6 contains every predictor selected in at least one of the 32 regression fits, along with the selection frequency. Frequency A represents the number of times each factor was selected when the `CHOOSE=CV` option was used, and Frequency B represents the number of times each factor was selected when the `CHOOSE=SBC` option was used.

The table below show the fit statistics for all predictors that scored 4 or higher. This is labeled model 1.



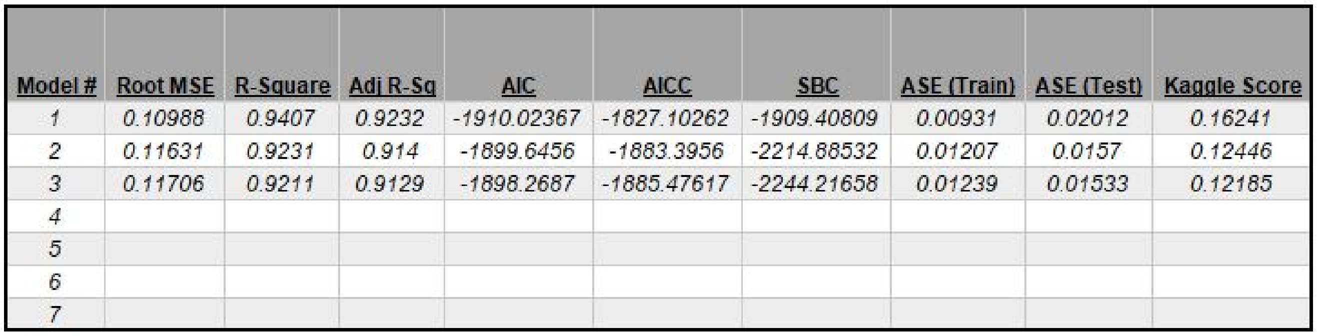
Model #	Root MSE	R-Square	Adj R-Sq	AIC	AICC	SBC	ASE (Train)	ASE (Test)	Kaggle Score
1	0.10988	0.9407	0.9232	-1910.02367	-1827.10262	-1909.40809	0.00931	0.02012	0.16241
2									
3									
4									
5									
6									
7									

Next, univariate plot analysis revealed useful factor level groups that would create a more parsimonious model. `PROC SQL` in SAS is used to create these groupings. The groupings are judgmentally selected for each factor based on patterns seen in Box Plots. These manual groupings, along with the same predictors in Model 1, form Model 2. You can find a list of the factor level groupings in `SAS_model2.txt` within the repo scripts folder.



Model #	Root MSE	R-Square	Adj R-Sq	AIC	AICC	SBC	ASE (Train)	ASE (Test)	Kaggle Score
1	0.10988	0.9407	0.9232	-1910.02367	-1827.10262	-1909.40809	0.00931	0.02012	0.16241
2	0.11631	0.9231	0.914	-1899.6456	-1883.3956	-2214.88532	0.01207	0.0157	0.12446
3									
4									
5									
6									
7									

Model 3 was developed by dropping `GarageArea` because of it's near-zero estimate and adding `ExterCond` and `GarageQual` in order to capture additional quality-based information about the home. This led to improved test ASE and Kaggle scores in Model 3.



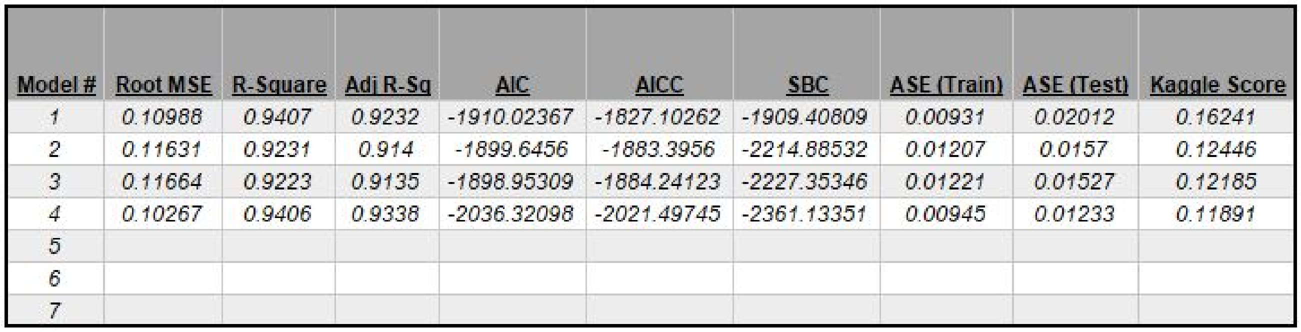
Model #	Root MSE	R-Square	Adj R-Sq	AIC	AICC	SBC	ASE (Train)	ASE (Test)	Kaggle Score
1	0.10988	0.9407	0.9232	-1910.02367	-1827.10262	-1909.40809	0.00931	0.02012	0.16241
2	0.11631	0.9231	0.914	-1899.6456	-1883.3956	-2214.88532	0.01207	0.0157	0.12446
3	0.11706	0.9211	0.9129	-1898.2687	-1885.47617	-2244.21658	0.01239	0.01533	0.12185
4									
5									
6									
7									

Before adding complexity to the model via interaction terms or attempting to model discontinuities in variables, outliers must be accounted for, which is discussed in the following section.

Outlier Analysis

Figure 3 above revealed at least two significantly leveraged outliers that had very high studentized residuals (ID 1299 and 524). In order to investigate these and other outliers, studentized residuals greater in absolute value than 3.0 are examined. There are 9 homes which met this criteria that were removed from the analysis because either they were deemed outside of the scope of the predictive model, or they were in multiple sparesely populated factor levels which made it difficult to obtain a reliable mean estimate of Sale Price.

The excluded outliers, along with the reasons for doing so, are in Appendix D, Figure 7. The resulting model fit statistics are below (Model 4).



Model #	Root MSE	R-Square	Adj R-Sq	AIC	AICC	SBC	ASE (Train)	ASE (Test)	Kaggle Score
1	0.10988	0.9407	0.9232	-1910.02367	-1827.10262	-1909.40809	0.00931	0.02012	0.16241
2	0.11631	0.9231	0.914	-1899.6456	-1883.3956	-2214.88532	0.01207	0.0157	0.12446
3	0.11664	0.9223	0.9135	-1898.95309	-1884.24123	-2227.35346	0.01221	0.01527	0.12185
4	0.10267	0.9406	0.9338	-2036.32098	-2021.49745	-2361.13351	0.00945	0.01233	0.11891
5									
6									
7									

Model Selection: Interaction Terms and Indicator Variables

With outliers removed, the model was once again studied for opportunities to optimize model fit. One thing that had not been considered is interactions between any predictors that can explain additional variability or offer more flexible parameterizations.

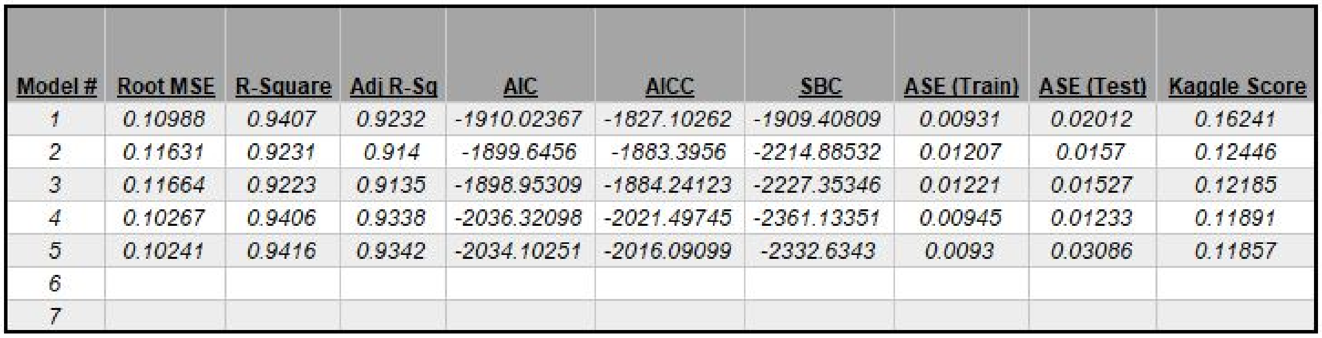
There are a few variables with strong linear correlations in the training data set. `GarageCars/Garage Area`, `TotRmsAbvGrd`, and `GrLivArea` will be variables that probably should never have interactions included in the model if the correlated terms are present. A table of all Pearson correlations greater than .5 is shown in Appendix D, Figure 8. These were considered significant enough to be mindful of when selecting interactions.

A first attempt to discover interactions was made by regressing partial residuals from Model 4 on all possible interaction combinations using `PROC GLMSELECT`. However, this yielded no useful results because no terms were identified. It could be that the selection methods were not sensitive enough to changes in partial residuals that were based on an already-decent model. In other words, they may have been too small.

By manually trying potential interactions and examining fit statistics, the following two interactions were identified by their respectable p-values and Model III sum of squares (the marginal sum of squares obtained if the variable were entered last into the model).

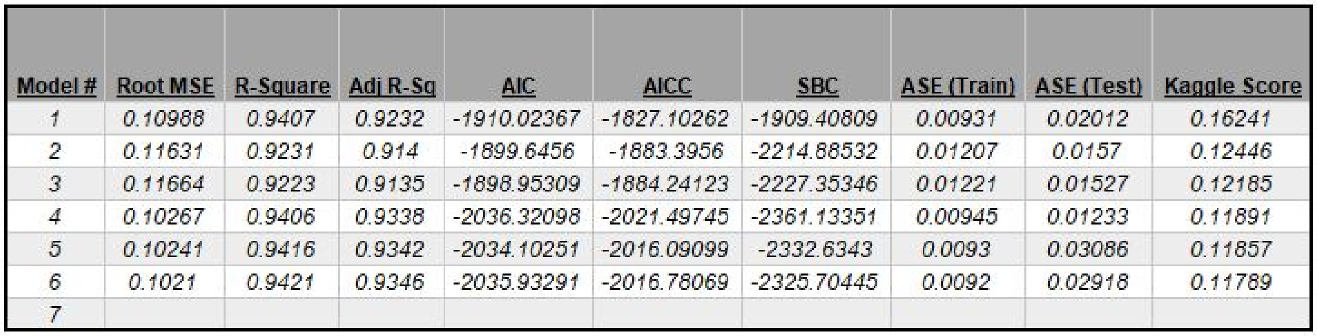
1. GrLivArea*FullBath
2. LotArea*Foundation

The Model Performance Statistics table is updated with Model 5, which includes the two interactions listed here.



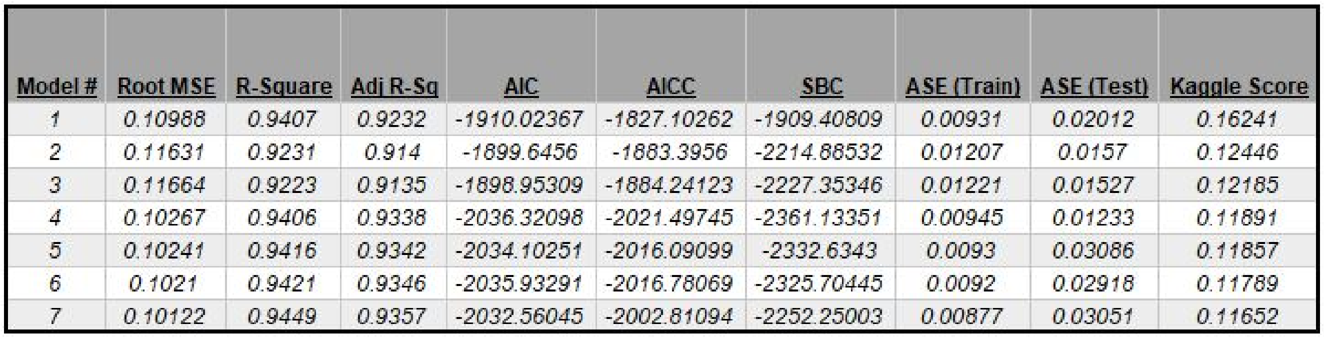
Model #	Root MSE	R-Square	Adj R-Sq	AIC	AICC	SBC	ASE (Train)	ASE (Test)	Kaggle Score
1	0.10988	0.9407	0.9232	-1910.02367	-1827.10262	-1909.40809	0.00931	0.02012	0.16241
2	0.11631	0.9231	0.914	-1899.6456	-1883.3956	-2214.88532	0.01207	0.0157	0.12446
3	0.11664	0.9223	0.9135	-1898.95309	-1884.24123	-2227.35346	0.01221	0.01527	0.12185
4	0.10267	0.9406	0.9338	-2036.32098	-2021.49745	-2361.13351	0.00945	0.01233	0.11891
5	0.10241	0.9416	0.9342	-2034.10251	-2016.09099	-2332.6343	0.0093	0.03086	0.11857
6									
7									

AFter this, two indicator terms (`idxhasNoFB1` and `idxhasNoWD`) were included to allow increased flexibility in the parameterizations of the `BsmtFinSF1*idxhasFB1` and `WoodDeckSF*idxhasWD` interactions that were in Models 1-3. This essentially allowed for homes without these features to be fit to a separate parameter so that the polynomial fits represent homes that actually have the feature.



Model #	Root MSE	R-Square	Adj R-Sq	AIC	AICC	SBC	ASE (Train)	ASE (Test)	Kaggle Score
1	0.10988	0.9407	0.9232	-1910.02367	-1827.10262	-1909.40809	0.00931	0.02012	0.16241
2	0.11631	0.9231	0.914	-1899.6456	-1883.3956	-2214.88532	0.01207	0.0157	0.12446
3	0.11664	0.9223	0.9135	-1898.95309	-1884.24123	-2227.35346	0.01221	0.01527	0.12185
4	0.10267	0.9406	0.9338	-2036.32098	-2021.49745	-2361.13351	0.00945	0.01233	0.11891
5	0.10241	0.9416	0.9342	-2034.10251	-2016.09099	-2332.6343	0.0093	0.03086	0.11857
6	0.1021	0.9421	0.9346	-2035.93291	-2016.78069	-2325.70445	0.0092	0.02918	0.11789
7									

In the 7th model iteration, YearBuilt was converted into a piecewise interaction with an old vs new indicator so that we can allow for differing polynomial regression slopes based on whether homes are built after 2000. An additional continuous variable for 2nd floor square footage was added, and a few variables had their factor levels rearranged for optimal fit.

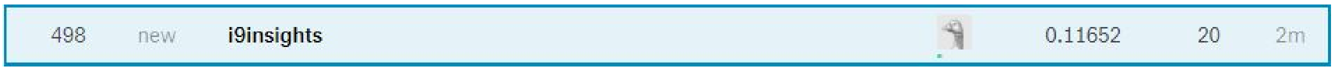


Model #	Root MSE	R-Square	Adj R-Sq	AIC	AICC	SBC	ASE (Train)	ASE (Test)	Kaggle Score
1	0.10988	0.9407	0.9232	-1910.02367	-1827.10262	-1909.40809	0.00931	0.02012	0.16241
2	0.11631	0.9231	0.914	-1899.6456	-1883.3956	-2214.88532	0.01207	0.0157	0.12446
3	0.11664	0.9223	0.9135	-1898.95309	-1884.24123	-2227.35346	0.01221	0.01527	0.12185
4	0.10267	0.9406	0.9338	-2036.32098	-2021.49745	-2361.13351	0.00945	0.01233	0.11891
5	0.10241	0.9416	0.9342	-2034.10251	-2016.09099	-2332.6343	0.0093	0.03086	0.11857
6	0.1021	0.9421	0.9346	-2035.93291	-2016.78069	-2325.70445	0.0092	0.02918	0.11789
7	0.10122	0.9449	0.9357	-2032.56045	-2002.81094	-2252.25003	0.00877	0.03051	0.11652

Top Models

Models 4, 3, and 2 had the top test ASE scores, and models 7, 6, and 5 had the top kaggle scores.

The figure below is a screenshot of the top Kaggle score received. The R code in appendix C and SAS code in appendix D can recreate the CSV file submitted to Kaggle. The CSV file itself is also located in the appendix folder of this repo.


498 new i9insights 0.11652 20 2m

Additional Improvements

The predictive capabilities utilized here did not make use of any clustering analysis that may have provided improved model fits. For example, rather than using 2000 as a "flex" point for the polynomial term YearBuilt, we could have optimized the flex point based on some sort of variability optimization between homes built before 2000 and those built after 2000.

Another consideration is the assumption of independence in the context of home sale prices. It can be argued that sale price does reflect the sale prices of neighboring homes. These neighboring home sale prices are included in every home appraisal and give every shopper a benchmark price to compare the home they are looking at with. It is hard to imagine this results in a truly independent set of home sale prices in the area. Perhaps some analysis can be done to examine correlations in home sale prices and reflect this in the standard errors of the sale price regression model.

Conclusion

In the battle of the property value tropes, it turns out that the old adage of "location, location, location" is right afterall. We compared models that focused on location, curb appeal, and the interior construction of a property, and found that on all measures, the location centric model was most predictive. This is useful for the parties in a real estate transaction because they know that playing up the location of a property can lead to a higher sale price. On the flipside, if you are looking for a home, it is good to know that you can likely get a good deal on a otherwise very nice property if you are williing to live outside of the premier neighborhoods.

Appendix: Code for all analyses

Appendix A: interp_models.R

The below commented code provides the steps taken in order to create, analyze, and select our models focused on interpretability.

```
# Load libraries
# If you don't have one, you will have to run install.packages('library')
library(dplyr)
library(purrr)
library(broom)

train <- read.csv("data/train1_clean.csv")

# Select variables related to the location of a property
location <- train %>%
  select(MSZoning,
         LotFrontage,
         LotArea,
         Neighborhood,
         Condition1,
         SaleCondition,
         SalePrice)
# Select variables related to the external appearance of a property
outside <- train %>%
  select(LotConfig,
         BldgType,
         HouseStyle,
         RoofStyle,
         Exterior1st,
         Exterior2nd,
         MasVnrType,
         MasVnrArea,
         ExterQual,
         ExterCond,
         PavedDrive,
         SaleCondition,
         SalePrice)

# Select variables related to the internal
inside <- train %>%
  select(Foundation,
         BsmtFinType1,
         BsmtFinType2,
         Heating,
         HeatingQC,
         CentralAir,
         Electrical,
         Fireplaces,
         SaleCondition,
         SalePrice)

# Combine our 3 datasets into a list for easy functional mapping
model_dfs <- list(location, outside, inside)

# Creat a helper function that will generate a model formula
# and run a standard OLS regression of SalePrice against all
# variables for each of our 3 datasets
model_fit <- function(x) {
  model_formula <- formula("SalePrice ~ .")
  lm(model_formula, data = x)
}

# Map our model fitting function above to all 3 datasets
models <- map(model_dfs, model_fit) %>%
  set_names(c("location", "outside", "inside"))

# Add NAmes as a level because it shows up in the test
# data and not the training data
#models$location$xlevels$Neighborhood <-
#  c(models$location$xlevels$Neighborhood, "NAmes")

# Map the broom::tidy function across our models to get parameter
# estimates in a tidy data frame
model_params <- map(models, tidy)
```

```
# Map the broom::glance function across our models to
# get model diagnostics in a tidy data frame
model_diags <- map(models, glance)

# After looking at the below, we can see that location seems to
# be the best model, will continue analysis with that model
bind_cols(
  data.frame(ModelName = c("location", "inside", "outside")),
  bind_rows(model_diags)
)
# Give selected model its own variable for easy reference
loc_lm <- models$location

# Check Assumptions
# Constant variance looks good, one suspicious point but not too bad
#plot(loc_lm$residuals)

# Symmetric mostly normal distribution, assumption ok
#hist(loc_lm$residuals)

# Read in the test data and get rid of the SalePrice column
# which we added to facilitate prediction in SAS
test <- read.csv("data/test_clean.csv")
test_x <- test[, -which(names(test) == "SalePrice")]

# Map the predict function aross our 3 models to generate
# predictions for submitting to kaggle
preds <- list()
preds <- map(models, predict, newdata = test_x)
```

Appendix B: cleaner_funs.R

The below code provides the functions we used for cleaning up the data set.

```
#Libraries

#Functions
factor.Adjust = function(data,adj,narep=FALSE) {
  for (i in seq(nrow(adj))) {
    x = as.character(unlist(unname(adj[i,])))
    feature = x[1]
    replace = x[2]
    with = x[3]

    if (!narep) {

      if (is.factor(data[,feature])) {
        feature.NewLevels = gsub(replace,with,levels(data[,feature]),fixed=TRUE)
        levels(data[,feature]) = feature.NewLevels
      } else
        data[data[,feature]==as.numeric(replace),feature] = as.numeric(with)
    } else {
      my.Class = class(data[,feature])
      data[is.na(data[,feature]),feature] = with
      class(data[,feature]) = my.Class

    }
  }
  return(data)
}

indicator.Add = function(data,indices) {
  for (i in seq(length(indices))) {
    new.Col = names(indices)[i]
    data[,new.Col] = 0
    data[indices[[i]],new.Col] = 1
  }
  return(data)
}

impute.bySampling = function(data) {
  data.Complete = data[!is.na(data)]
  na.Indices = which(is.na(data))

  sample.Size = length(na.Indices)
  data[na.Indices] = sample(data.Complete,size=sample.Size,replace=TRUE)
  return(data)
}

find.Inconsistencies = function(data1, data2) {
  lvl.Check = list()
  for (i in seq(length(data1))) {
    print(paste(colnames(data1)[i],sum(!(data1[,i] %in% data2[,i]))))
  }
  return(lvl.Check)
}
```

Appendix C: cleaner_script.R

The below code provides the script that calls the above cleaner functions, and does various other data cleaning.

```
# Import data

train = read.csv('data/train1.csv')
test = read.csv('data/test.csv')

#Drop unwanted columns
train_drop = c('LandContour','Utilities','LandSlope','Condition2',
                 'SaleType','Street','RoofMatl','X3SsnPorch', 'Functio.1l')

test_drop = c('LandContour','Utilities','LandSlope','Condition2',
                 'SaleType','Street','RoofMatl','X3SsnPorch','PoolQC', 'Fence',
                 'MiscFeature', 'Alley', 'Functional')

train.Clean = train[, !(names(train) %in% train_drop)]
test.Clean = test[, !(names(test) %in% test_drop)]
```

```
#Fix column names
names(train.Clean)[names(train.Clean)=='X1stFlrSF'] = 'FirstFlrSF'
names(train.Clean)[names(train.Clean)=='X2ndFlrSF'] = 'SecFlrSF'
names(train.Clean)[names(train.Clean)=='Kitche.1bvGr'] = 'KitchenAbvGr'
names(train.Clean)[names(train.Clean)=='Functio.1l'] = 'Functional'
names(train.Clean)[names(train.Clean)=='ScreenPorch'] = 'ScreenPorchSF'

names(test.Clean)[names(test.Clean)=='X1stFlrSF'] = 'FirstFlrSF'
names(test.Clean)[names(test.Clean)=='X2ndFlrSF'] = 'SecFlrSF'
names(test.Clean)[names(test.Clean)=='Functio.1l'] = 'Functional'
names(test.Clean)[names(test.Clean)=='ScreenPorch'] = 'ScreenPorchSF'

#Create None factor level for FireplaceQu to represent the NA values
levels(test.Clean$FireplaceQu) = c(levels(test.Clean$FireplaceQu),'None')

#Identify source of missing data and reassign values
#The custom function factor.adjust will handle both categorical and continuous
#continuous variables

level.Fix =  rbind.data.frame(
  c('MSZoning','C (all)','C'),
  c('LotFrontage','-1','0'),
  c('Neighborhood','-1mes','NAmes'),
  c('MasVnrType','-1','None'),
  c('MasVnrArea','-1','0'),
  c('BsmtQual','-1','None'),
  c('BsmtCond','-1','None'),
  c('BsmtExposure','-1','None'),
  c('BsmtFinType1','-1','None'),
  c('BsmtFinType2','-1','None'),
  c('Electrical','-1','SBrkr'),
  c('FireplaceQu','-1','None'),
  c('GarageType','-1','None'),
  c('GarageYrBlt','-1','0'),
  c('GarageFinish','-1','None'),
  c('GarageQual','-1','None'),
  c('GarageCond','-1','None'),
  c('LotFrontage','313','150'),
  c('LotFrontage','174','150'),
  c('LotFrontage','200','150'),
  c('LotFrontage','168','150'),
  c('LotFrontage','182','150'),
  c('LotFrontage','160','150'),
  c('LotFrontage','152','150'),
  c('LotFrontage','153','150')
)

level.Fix2 =  rbind.data.frame(
  c('MSZoning','C (all)','C')
)

#Assign missing values to existing factor levels
```

```
level.Fix3 = rbind.data.frame(

  c('MSZoning','NA','RL'),
  c('Exterior1st','NA','VinylSd'),
  c('Exterior2nd','NA','VinylSd'),
  c('MasVnrType','NA','None'),
  c('MasVnrArea','NA','0'),
  c('BsmtQual','NA','TA'),
  c('BsmtCond','NA','TA'),
  c('BsmtExposure','NA','No'),
  c('BsmtFinSF1','NA','0'),
  c('BsmtFinSF2','NA','0'),
  c('BsmtFinType2','NA','Unf'),
  c('BsmtUnfSF','NA','0'),
  c('TotalBsmtSF','NA','0'),
  c('BsmtHalfBath','NA','0'),
  c('KitchenQual','NA','TA'),
  c('FireplaceQu','NA','None'),
  c('GarageCars','NA','2'),
  c('GarageQual','NA','TA'),
  c('GarageCond','NA','TA')
)

names(level.Fix) = c('Feature','Replace','With')

train.Clean = factor.Adjust(data=train.Clean,adj=level.Fix)
test.Clean = factor.Adjust(data=test.Clean,adj=level.Fix2)
test.Clean = factor.Adjust(data=test.Clean,adj=level.Fix3,narep=TRUE)


#New features based on Age
train.Clean$HomeAge = train.Clean$YrSold - train.Clean$YearBuilt
train.Clean$RemodAge = train.Clean$YrSold - train.Clean$YearRemodAdd

test.Clean$HomeAge = test.Clean$YrSold - test.Clean$YearBuilt
test.Clean$RemodAge = test.Clean$YrSold - test.Clean$YearRemodAdd

#Create a predicted SalePrice for the train
test.Clean$SalePrice = -1

#Indicator variables which are required for correct parameterization of
#continuous variables which have no value for certain homes
#e.g. 81 homes have no garage, so we use hasGarage x GarageYrBlt
#instead of dropping GarageYrBlt because of the homes with value 0

new.Indicators.Indices =
  list(
    'idxhasG' = which(train.Clean$GarageYrBlt != 0),
    'idxhasMV' = which(train.Clean$MasVnrArea != 0),
    'idxhasFB1' = which(train.Clean$BsmtFinSF1 != 0),
    'idxhasFB2' = which(train.Clean$BsmtFinSF2 != 0),
    'idxhasB' = which(train.Clean$TotalBsmtSF != 0),
    'idxhasSF' = which(train.Clean$SecFlrSF != 0),
    'idxhasPool' = which(train.Clean$PoolArea != 0),
    'idxhasLF' = which(train.Clean$LotFrontage != 0),
    'idxhasLQF' = which(train.Clean$LowQualFinSF != 0),
    'idxhasWD' = which(train.Clean$WoodDeckSF != 0),
    'idxhasOP' = which(train.Clean$OpenPorchSF != 0),
    'idxhasEP' = which(train.Clean$EnclosedPorch != 0),
    'idxhasSP' = which(train.Clean$ScreenPorchSF != 0),
    'idxhasMV' = which(train.Clean$MiscVal != 0),
    'idxisOld' = which(train.Clean$YearBuilt < 2000),
    'idxhasRem' = which(train.Clean$YearRemodAdd != train.Clean$YearBuilt),
    'idxisNew' = which(train.Clean$YearBuilt >= 2000)
  )

new.Indicators.Indices2 =
  list(
    'idxhasG' = which(test.Clean$GarageYrBlt != 0),
    'idxhasMV' = which(test.Clean$MasVnrArea != 0),
    'idxhasFB1' = which(test.Clean$BsmtFinSF1 != 0),
    'idxhasFB2' = which(test.Clean$BsmtFinSF2 != 0),
    'idxhasB' = which(test.Clean$TotalBsmtSF != 0),
    'idxhasSF' = which(test.Clean$SecFlrSF != 0),
    'idxhasPool' = which(test.Clean$PoolArea != 0),
    'idxhasLF' = which(test.Clean$LotFrontage != 0),
    'idxhasLQF' = which(test.Clean$LowQualFinSF != 0),
    'idxhasWD' = which(test.Clean$WoodDeckSF != 0),
    'idxhasOP' = which(test.Clean$OpenPorchSF != 0),
    'idxhasEP' = which(test.Clean$EnclosedPorch != 0),
    'idxhasSP' = which(test.Clean$ScreenPorchSF != 0),
    'idxhasMV' = which(test.Clean$MiscVal != 0),
    'idxisOld' = which(test.Clean$YearBuilt < 2000),
    'idxhasRem' = which(test.Clean$YearRemodAdd != test.Clean$YearBuilt),
    'idxisNew' = which(test.Clean$YearBuilt >= 2000)
  )

train.Clean = indicator.Add(data=train.Clean,indices = new.Indicators.Indices)
test.Clean = indicator.Add(data=test.Clean,indices = new.Indicators.Indices2)


#Imputing by sampling
test.Clean$LotFrontage = impute.bySampling(data=test.Clean$LotFrontage)
test.Clean$BsmtFinType1 = impute.bySampling(data=test.Clean$BsmtFinType1)
test.Clean$BsmtFullBath = impute.bySampling(data=test.Clean$BsmtFullBath)
test.Clean$GarageType = impute.bySampling(data=test.Clean$GarageType)
test.Clean$GarageYrBlt = impute.bySampling(data=test.Clean$GarageYrBlt)
test.Clean$GarageFinish = impute.bySampling(data=test.Clean$GarageFinish)
test.Clean$GarageArea = impute.bySampling(data=test.Clean$GarageArea)

#Variable Transformations

log.Transform = c('LotArea', 'MasVnrArea', 'BsmtFinSF1',
                  'TotalBsmtSF', 'FirstFlrSF', 'SecFlrSF', 'GrLivArea',
                  'WoodDeckSF', 'OpenPorchSF', 'MiscVal', 'SalePrice')

log.Transform2 = c('LotArea', 'MasVnrArea', 'BsmtFinSF1',
                   'TotalBsmtSF', 'FirstFlrSF', 'SecFlrSF', 'GrLivArea',
                    'WoodDeckSF', 'OpenPorchSF', 'MiscVal')

#ln(.001) ~ -7
train.Clean[,log.Transform] = log(train.Clean[,log.Transform] + .0001)
test.Clean[,log.Transform2] = log(test.Clean[,log.Transform2] + .0001)

#Drop outlier records - irregular lot homes with unusually large lot space
outlier.idx = c(31,969,524,1299,899,1063,711,496,89)
train.Clean = train.Clean[-outlier.idx,]

#Merge training and test files, including '-1' for empty salePrices for SAS to predict
merge.Clean = rbind.data.frame(train.Clean,test.Clean)

write.csv(train.Clean,'data/train1_clean.csv',row.names=FALSE)
write.csv(test.Clean,'data/test_clean.csv',row.names=FALSE)
write.csv(merge.Clean,'data/merge_clean.csv',row.names=FALSE)

#
# pdf('plots.pdf')
# feature.Levels = lapply(X=train.Clean,table)
# for (i in seq(feature.Levels)) {
# f.num = i
# barplot(feature.Levels[[f.num]],main=names(feature.Levels[f.num]))
# plot(x=train.Clean[,names(feature.Levels[f.num])],y=train.Clean$SalePrice,
#      xlab=names(feature.Levels[f.num]),col='black')
# points(x=train.Clean[outlier.idx,names(feature.Levels[f.num])],y=train.Clean[outlier.idx,'SalePrice']
#      xlab=names(feature.Levels[f.num]),col='red')
# text(x=train.Clean[outlier.idx,names(feature.Levels[f.num])],y=train.Clean[outlier.idx,'SalePrice'],
#      labels = row.names(train.Clean[outlier.idx,]), pos = 1)
# }
# dev.off()
# fit = lm(log(SalePrice) ~ train.Clean[,names(feature.Levels[f.num])],data=train.Clean)
#
```

Appendix C: sas_modeling.txt

The below code provides the SAS code used for generating and evaluating our SAS regressions used for finding the most predictive model.

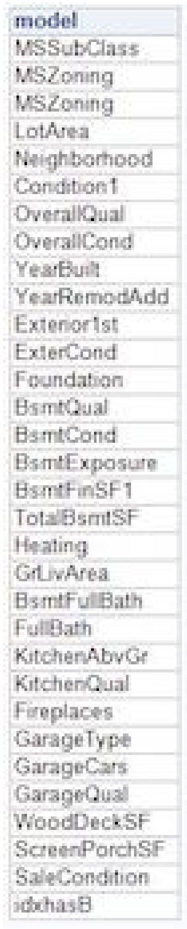
```
proc import datafile="C:\Users\jat05\Downloads\MSDS_6372\Project 1\UNIT 6 Project Folder\merge_clean.cs
     dbms=csv out=work.saleprice replace;
     delimiter=',';
     getnames=yes;
     guessingrows=32000;
run;

PROC SQL NOPRINT;
    UPDATE saleprice
    SET SalePrice = .
    WHERE SalePrice = -1;

PROC SQL;
CREATE TABLE work.saleprice_dm1 AS
SELECT
```

```
Id,
CASE WHEN MSSubClass = 20 THEN 20
    WHEN MSSubClass in (30,40) THEN 30
    WHEN MSSubClass in (45,50) THEN 50
    WHEN MSSubClass = 60 THEN 60
    WHEN MSSubClass = 70 THEN 70
    WHEN MSSubClass in (75,80,85,90) THEN 90
    WHEN MSSubClass = 120 THEN 120
    WHEN MSSubClass in (150,160,180,190) THEN 190
    END as MSSubClass,
CASE WHEN MSZoning not in ('RM','C') THEN 'Other'
    ELSE MSZoning END as MSZoning,
LotFrontage,
LotArea,
CASE WHEN LotShape <> 'Reg' THEN 'Other'
    ELSE LotShape END as LotShape,
LotConfig,
CASE WHEN Neighborhood not in ('BrkSide','Crawfor','Edwards','NoRidge','NridgHt','Somerst','StoneBr') T
    ELSE Neighborhood END as Neighborhood,
CASE WHEN Condition1 = 'Norm' THEN 'Norm'
    ELSE Condition1 END as Condition1,
BldgType,
HouseStyle,
CASE WHEN OverallQual in (1,2,3) THEN 3
    ELSE OverallQual END as OverallQual,
CASE WHEN OverallCond in (1,2,3) THEN 3
    WHEN OverallCond = 4 THEN 4
    WHEN OverallCond = 5 THEN 5
    WHEN OverallCond = 6 THEN 6
    ELSE 7 END as OverallCond,
YearBuilt,
YearRemodAdd,
RoofStyle,
CASE WHEN Exterior1st not in ('BrkFace','AsbShng') then 'Other'
    ELSE Exterior1st END as Exterior1st,
Exterior2nd,
MasVnrType,
MasVnrArea,
CASE WHEN ExterQual <> 'TA' THEN 'Other'
    ELSE ExterQual END as ExterQual,
CASE WHEN ExterCond <> 'Fa' THEN 'Other'
    ELSE ExterCond END as ExterCond,
CASE WHEN Foundation in ('Wood','BrkTil') THEN 'BrkTil'
    ELSE Foundation END as Foundation,
CASE WHEN BsmtQual <> 'Ex' THEN 'Other'
    ELSE BsmtQual END as BsmtQual,
CASE WHEN BsmtCond not in ('Gd','TA') THEN 'Other'
    ELSE 'TA' END as BsmtCond,
CASE WHEN BsmtExposure <> 'Gd' THEN 'Other'
    ELSE BsmtExposure END as BsmtExposure,
BsmtFinType1,
BsmtFinSF1,
BsmtFinType2,
BsmtFinSF2,
BsmtUnfSF,
TotalBsmtSF,
CASE WHEN Heating <> 'Grav' THEN 'Other'
    ELSE Heating END as Heating,
CASE WHEN HeatingQC <> 'Ex' THEN 'Other'
    ELSE HeatingQC END as HeatingQC,
CentralAir,
Electrical,
FirstFlrSF,
SecFlrSF,
LowQualFinSF,
GrLivArea,
CASE WHEN BsmtFullBath > 0 then 1
    ELSE BsmtFullBath END as BsmtFullBath,
BsmtHalfBath,
CASE WHEN FullBath in (0,1) THEN 1
    WHEN FullBath in (3,4) then 3
    ELSE FullBath END as FullBath,
CASE WHEN HalfBath = 2 THEN 1
    ELSE HalfBath END as HalfBath,
BedroomAbvGr,
CASE WHEN KitchenAbvGr <> 2 THEN 1
    ELSE KitchenAbvGr END as KitchenAbvGr,
CASE WHEN KitchenQual in ('Fa','TA', 'Gd') THEN 'TA'
    ELSE KitchenQual END as KitchenQual,
CASE WHEN TotRmsAbvGrd > 10 THEN 10
    ELSE TotRmsAbvGrd END as TotRmsAbvGrd,
CASE WHEN Fireplaces > 2 THEN 2
    ELSE Fireplaces END as Fireplaces,
FireplaceQu,
CASE WHEN GarageType not in ('Detchd','Attchd') THEN 'Other'
    ELSE GarageType END as GarageType,
GarageYrBlt,
GarageFinish,
CASE WHEN GarageCars > 3 THEN 3
    WHEN GarageCars = 0 THEN 1
    ELSE GarageCars END as GarageCars,
GarageArea,
CASE WHEN GarageQual in ('None','Po') then 'Low'
    ELSE GarageQual END as GarageQual,
GarageCond,
PavedDrive,
WoodDeckSF,
OpenPorchSF,
EnclosedPorch,
ScreenPorchSF,
PoolArea,
MiscVal,
MoSold,
YrSold,
CASE WHEN SaleCondition not in ('Abnorml', 'Partial') THEN 'Other'
    ELSE SaleCondition END as SaleCondition,
SalePrice,
HomeAge,
RemodAge,
idxhasG,
idxhasMV,
idxhasFB1,
CASE WHEN idxhasFB1 = 1 THEN 0 ELSE 1 END as idxhasNoFB1,
idxhasFB2,
CASE WHEN idxhasFB2 = 1 THEN 0 ELSE 1 END as idxhasNoFB2,
idxhasB,
CASE WHEN idxhasB = 1 THEN 0 ELSE 1 END as idxhasNoB,
idxhasSF,
idxhasPool,
idxhasLF,
CASE WHEN idxhasLF = 1 THEN 0 ELSE 1 END as idxhasNoLF,
idxhasLQF,
idxhasWD,
CASE WHEN idxhasWD = 1 THEN 0 ELSE 1 END as idxhasNoWD,
idxhasOP,
idxhasEP,
idxhasSP,
CASE WHEN idxhasSP = 1 THEN 0 ELSE 1 END as idxhasNoSP,
idxisOld,
idxhasRem,
idxisNew
FROM work.saleprice;
RUN;

DATA work.modelterms;
length model$ 50 class$ 25;
input model class;
datalines;
BsmtCond BsmtCond
BsmtExposure BsmtExposure
BsmtFinSF1*idxhasFB1 continuous
BsmtFullBath BsmtFullBath
BsmtQual BsmtQual
CentralAir CentralAir
Condition1 Condition1
Exterior1st Exterior1st
ExterQual ExterQual
Fireplaces Fireplaces
FirstFlrSF continuous
Foundation Foundation
FullBath FullBath
GarageCars continuous
GarageType GarageType
GrLivArea continuous
HeatingQC HeatingQC
KitchenAbvGr KitchenAbvGr
KitchenQual KitchenQual
LotArea continuous
MSSubClass MSSubClass
MSZoning MSZoning
Neighborhood Neighborhood
OverallCond OverallCond
```

```
OverallQual continuous
SaleCondition SaleCondition
ScreenPorchSF*idxhasSP continuous
TotalBsmtSF*idxhasB continuous
WoodDeckSF*idxhasWD continuous
YearRemodAdd continuous
ExterCond ExterCond
GarageQual GarageQual
GrLivArea*FullBath continuous
LotArea*Foundation continuous
idxhasNoFB1 continuous
idxhasNoWD continuous
YearBuilt*idxisOld continuous
YearBuilt*idxisNew continuous
idxisNew continuous
;
RUN;


PROC SQL;
    SELECT model
    INTO :classList separated by ' '
    FROM work.modelterms
    WHERE class <> 'continuous';

PROC SQL;
    SELECT model
    INTO :modelList separated by ' '
    FROM work.modelterms;

proc GLM DATA=work.saleprice_dm1;
        class &classList;
        model SalePrice = &modelList / clparm solution;
        output out=work.estPrice predicted=SalePrice2 residual=Resid;
run;
proc GLMSELECT DATA=work.saleprice_dm1 seed=12345;
        class &classList;
        partition fraction(test=.6);
        model SalePrice = &modelList / SELECTION=stepwise(include=39 stop=39);
        *modelaverage nsamples=500 REFIT;
        *output out=work.estPriceMA predicted=SalePrice2;
run;

PROC SQL;
    CREATE TABLE work.estprice_kaggle AS (
    SELECT Id, exp(SalePrice2) as SalePrice
    FROM work.estPrice
    WHERE Id > 1460);
RUN;
```

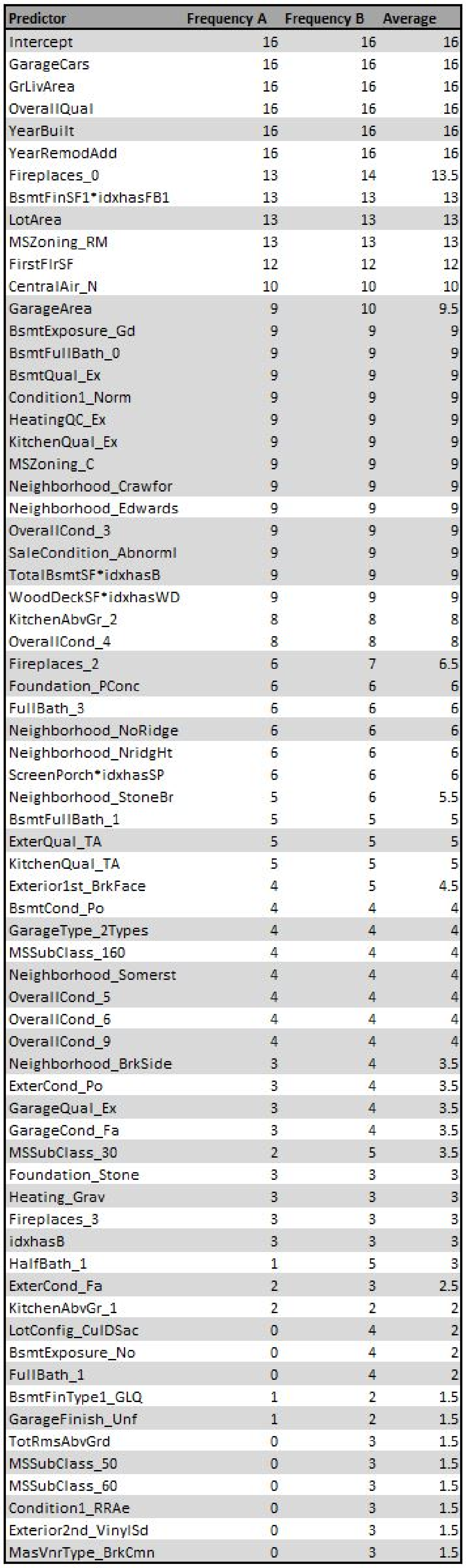
Appendix D: Extra Figures

Figure 1: Predictors used in assumption validation model

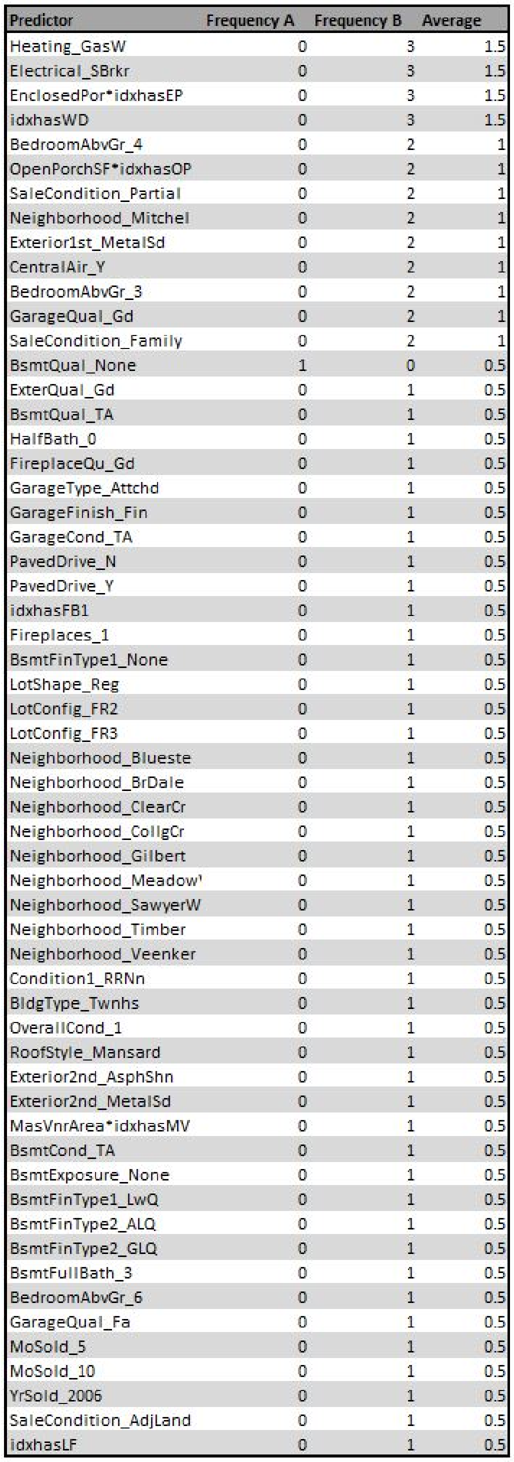


model
MSSubClass
MSZoning
MSZoning
LotArea
Neighborhood
Condition1
OverallQual
OverallCond
YearBuilt
YearRemodAdd
Exterior1st
ExterCond
Foundation
BsmtQual
BsmtCond
BsmtExposure
BsmtFinSF1
TotalBsmtSF
Heating
GrLivArea
BsmtFullBath
FullBath
KitchenAbvGr
KitchenQual
Fireplaces
GarageType
GarageCars
GarageQual
WoodDeckSF
ScreenPorchSF
SaleCondition
idxhasB

Figures 2 and 3: Model Predictors and Associated Scores

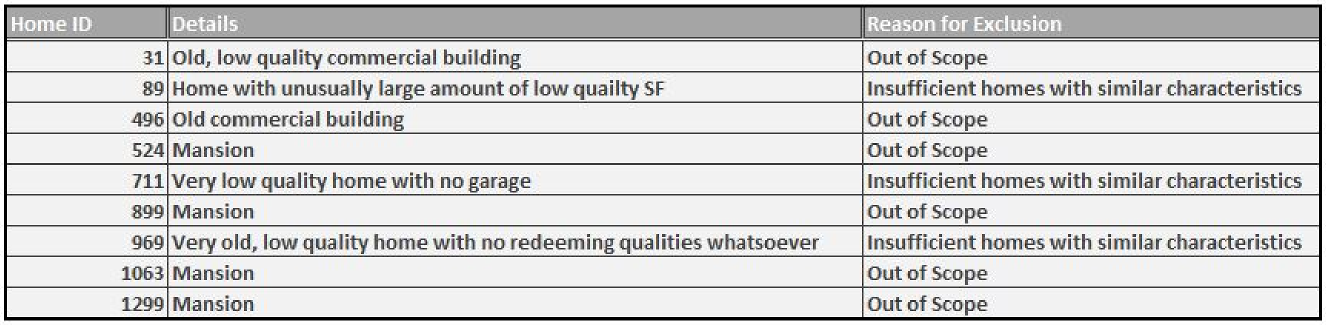


Predictor	Frequency A	Frequency B	Average
Intercept	16	16	16
GarageCars	16	16	16
GrLivArea	16	16	16
OverallQual	16	16	16
YearBuilt	16	16	16
YearRemodAdd	16	16	16
Fireplaces_0	13	14	13.5
BsmtFinSF1*idxhasFB1	13	13	13
LotArea	13	13	13
MSZoning_RM	13	13	13
FirstFlrSF	12	12	12
CentralAir_N	10	10	10
GarageArea	9	10	9.5
BsmtExposure_Gd	9	9	9
BsmtFullBath_0	9	9	9
BsmtQual_Ex	9	9	9
Condition1_Norm	9	9	9
HeatingQC_Ex	9	9	9
KitchenQual_Ex	9	9	9
MSZoning_C	9	9	9
Neighborhood_Crawfor	9	9	9
Neighborhood_Edwards	9	9	9
OverallCond_3	9	9	9
SaleCondition_Abnorml	9	9	9
TotalBsmtSF*idxhasB	9	9	9
WoodDeckSF*idxhasWD	9	9	9
KitchenAbvGr_2	8	8	8
OverallCond_4	8	8	8
Fireplaces_2	6	7	6.5
Foundation_PConc	6	6	6
FullBath_3	6	6	6
Neighborhood_NoRidge	6	6	6
Neighborhood_NridgHt	6	6	6
ScreenPorch*idxhasSP	6	6	6
Neighborhood_StoneBr	5	6	5.5
BsmtFullBath_1	5	5	5
ExterQual_TA	5	5	5
KitchenQual_TA	5	5	5
Exterior1st_BrkFace	4	5	4.5
BsmtCond_Po	4	4	4
GarageType_2Types	4	4	4
MSSubClass_160	4	4	4
Neighborhood_Somerst	4	4	4
OverallCond_5	4	4	4
OverallCond_6	4	4	4
OverallCond_9	4	4	4
Neighborhood_BrkSide	3	4	3.5
ExterCond_Po	3	4	3.5
GarageQual_Ex	3	4	3.5
GarageCond_Fa	3	4	3.5
MSSubClass_30	2	5	3.5
Foundation_Stone	3	3	3
Heating_Grav	3	3	3
Fireplaces_3	3	3	3
idxhasB	3	3	3
HalfBath_1	1	5	3
ExterCond_Fa	2	3	2.5
KitchenAbvGr_1	2	2	2
LotConfig_CulDSac	0	4	2
BsmtExposure_No	0	4	2
FullBath_1	0	4	2
BsmtFinType1_GLQ	1	2	1.5
GarageFinish_Unf	1	2	1.5
TotRmsAbvGrd	0	3	1.5
MSSubClass_50	0	3	1.5
MSSubClass_60	0	3	1.5
Condition1_RRAe	0	3	1.5
Exterior2nd_VinylSd	0	3	1.5
MasVnrType_BrkCmn	0	3	1.5



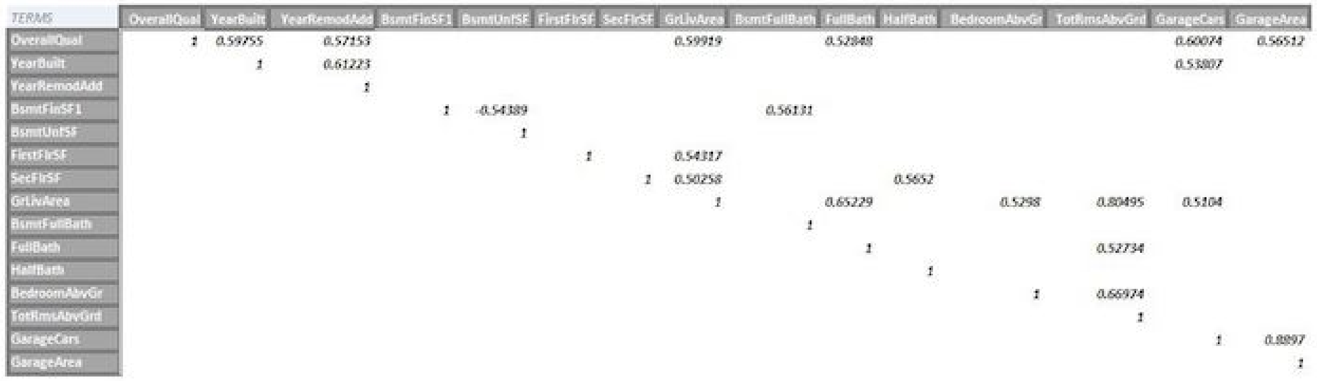
Predictor	Frequency A	Frequency B	Average
Heating_GasW	0	3	1.5
Electrical_SBrkr	0	3	1.5
EnclosedPor*idxhasEP	0	3	1.5
idxhasWD	0	3	1.5
BedroomAbvGr_4	0	2	1
OpenPorchSF*idxhasOP	0	2	1
SaleCondition_Partial	0	2	1
Neighborhood_Mitchel	0	2	1
Exterior1st_MetalSd	0	2	1
CentralAir_Y	0	2	1
BedroomAbvGr_3	0	2	1
GarageQual_Gd	0	2	1
SaleCondition_Family	0	2	1
BsmtQual_None	1	0	0.5
ExterQual_Gd	0	1	0.5
BsmtQual_TA	0	1	0.5
HalfBath_0	0	1	0.5
FireplaceQu_Gd	0	1	0.5
GarageType_Attchd	0	1	0.5
GarageFinish_Fin	0	1	0.5
GarageCond_TA	0	1	0.5
PavedDrive_N	0	1	0.5
PavedDrive_Y	0	1	0.5
idxhasFB1	0	1	0.5
Fireplaces_1	0	1	0.5
BsmtFinType1_None	0	1	0.5
LotShape_Reg	0	1	0.5
LotConfig_FR2	0	1	0.5
LotConfig_FR3	0	1	0.5
Neighborhood_Blueste	0	1	0.5
Neighborhood_BrDale	0	1	0.5
Neighborhood_ClearCr	0	1	0.5
Neighborhood_CollgCr	0	1	0.5
Neighborhood_Gilbert	0	1	0.5
Neighborhood_Meadow'	0	1	0.5
Neighborhood_SawyerW	0	1	0.5
Neighborhood_Timber	0	1	0.5
Neighborhood_Veenker	0	1	0.5
Condition1_RRNn	0	1	0.5
BldgType_Twnhs	0	1	0.5
OverallCond_1	0	1	0.5
RoofStyle_Mansard	0	1	0.5
Exterior2nd_AsphShn	0	1	0.5
Exterior2nd_MetalSd	0	1	0.5
MasVnrArea*idxhasMV	0	1	0.5
BsmtCond_TA	0	1	0.5
BsmtExposure_None	0	1	0.5
BsmtFinType1_LwQ	0	1	0.5
BsmtFinType2_ALQ	0	1	0.5
BsmtFinType2_GLQ	0	1	0.5
BsmtFullBath_3	0	1	0.5
BedroomAbvGr_6	0	1	0.5
GarageQual_Fa	0	1	0.5
MoSold_5	0	1	0.5
MoSold_10	0	1	0.5
YrSold_2006	0	1	0.5
SaleCondition_AdjLand	0	1	0.5
idxhasLF	0	1	0.5

Figure 4: Outliers Removed and Explanation



Home ID	Details	Reason for Exclusion
31	Old, low quality commercial building	Out of Scope
89	Home with unusually large amount of low quailty SF	Insufficient homes with similar characteristics
496	Old commercial building	Out of Scope
524	Mansion	Out of Scope
711	Very low quality home with no garage	Insufficient homes with similar characteristics
899	Mansion	Out of Scope
969	Very old, low quality home with no redeeming qualities whatsoever	Insufficient homes with similar characteristics
1063	Mansion	Out of Scope
1299	Mansion	Out of Scope

Figure 5: Linear Correlations in Training Data Sample



TERMS	OverallQual	YearBuilt	YearRemodAdd	BsmtFinSF1	BsmtUnfSF	FirstFlrSF	SecFlrSF	GrLivArea	BsmtFullBath	FullBath	HalfBath	BedroomAbvGr	TotRmsAbvGrd	GarageCars	GarageArea
OverallQual	1	0.59755	0.57153					0.59919		0.52848				0.60074	0.56512
YearBuilt		1	0.61223											0.53807	
YearRemodAdd			1												
BsmtFinSF1				1	-0.54389				0.56131						
BsmtUnfSF					1										
FirstFlrSF						1		0.54317							
SecFlrSF							1	0.50258			0.5652				
GrLivArea								1		0.65229		0.5298	0.80495	0.5104	
BsmtFullBath									1						
FullBath										1			0.52734		
HalfBath											1				
BedroomAbvGr												1	0.66974		
TotRmsAbvGrd													1		
GarageCars														1	0.8897
GarageArea															1