House Prices Analysis with Regression

Pengshuai Shi

April 9, 2017

1 Introduction

For home buyers who just started looking into houses, they probably would not begin with asking for *BsmtQual*(the height of the basement), number of fireplaces or *MasVnrType*(Masonry veneer type) in the right beginning, but by visualizing and analyzing a housing price data set with regression, we will see how each of those attributes of a house could influence its sale price.

2 Data Source

The *Ames Housing dataset*[1] compiled by Dean De Cock is a modernized and expanded alternative version of the often cited *Boston Housing dataset*, it contains 80 columns in total, with 79 being the explanatory variables which includes many dfifferent aspects of an residential house such as *LotArea, MSZoning, BsmtExposure* etc. in Ames, Iowa, and the last one is the sale price in dollars.
In this project, there are 1,460 data points in training set and 1,459 in testing set.

3 Data-Preprocessing

3.1 Missing Data

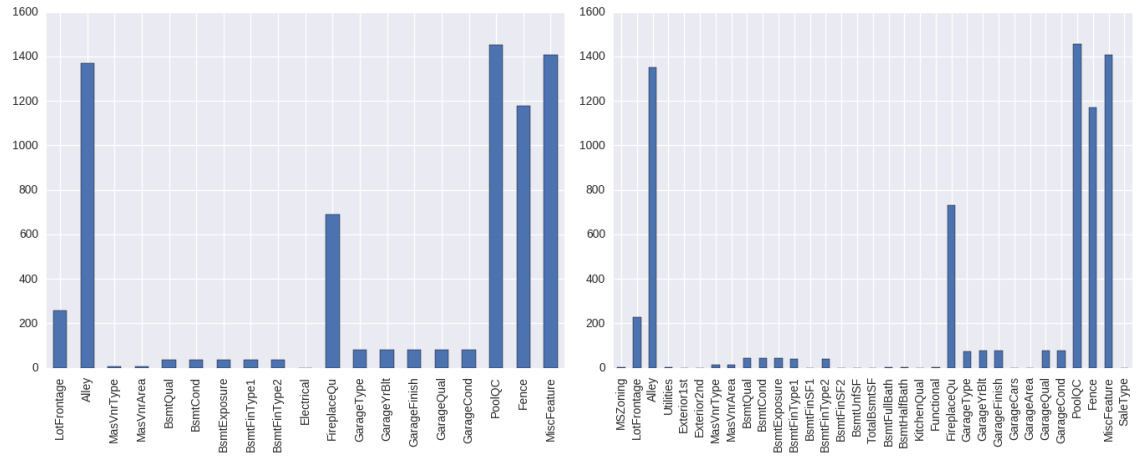
Before we proceed to analyze the data set using regression, we need to make sure it does not have missing values in the data set since regression model expects complete attribute from data. A quick search query shows that there are 1,460 number of rows each contains at least one missing value. Simple deletion would not work here as it would just remove our whole data set. So let us take a look of what attributes are missing and how many are missing in each attribute 1a,

It is also possible that test set could have missing values in an attribute that does not have missing data in training set, therefore we should also check 1b,

Thus we can see that both train and test set has 6 attributes with more than 15% missing, and these six features is discarded - *Fence, LotFrontage, MiscFeature, Alley, FireplaceQu, PoolQC.*

A simple imputation method - replace missing points with mean or median of each column, is used to fill up missing data points in the rest of attributes, although more

[1] `https://ww2.amstat.org/publications/jse/v19n3/decock.pdf`



(a) train (b) test

Figure 1: Numer of missing data points

sophiscated method can be used e.g. MICE(multiple imputation through chained equation), but the end result does not differ by much.

3.2 Outliers

It might also be helpful to look at some attributes and see if there is outliers, this is more relative to the model we are using. The objective function or the loss, which linear regression is trying to minize is very sensitive to outliers(absolute loss is less sensitive comparing to square loss), We can clearly see at the bottom right corner

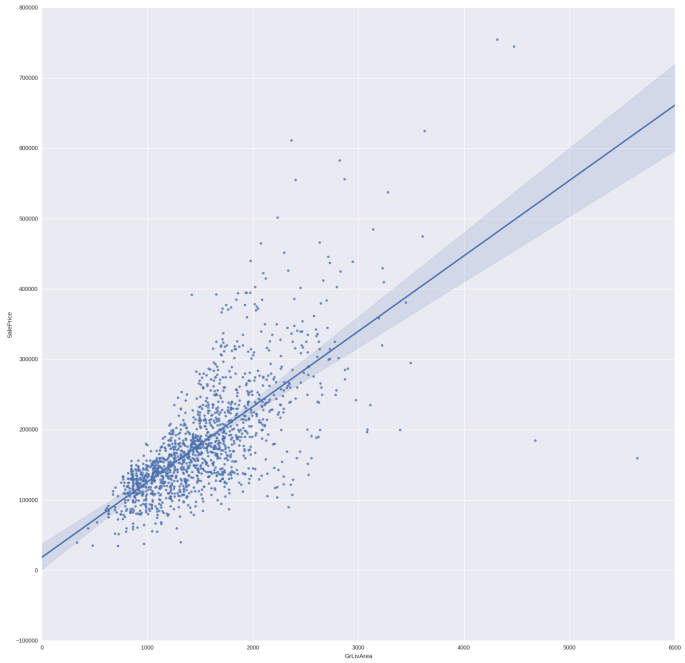


Figure 2: Outliers in GrLiveArea

there are two data points that are likely to be outliers. In general, we want to remove it and see how the model perform, possible reasons we might not want to remove are that: in one dimension(GrLiveArea) they might appear to be away from main stream but it is not the case when we consider the complete feature space, and we

may end up overfitting our train set. Nevertheless, they are removed in this project and we do observe a better performance overall, see the plot of all attributes in Fig 9 Appendix A.

3.3 Skewness

First examine the distribution of the y-value *SalePrice*, so we see that the distri-

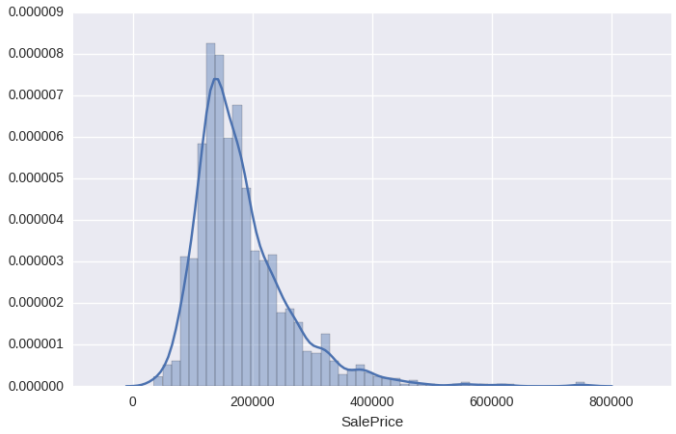


Figure 3: Histogram of SalePrice

bution of y-value is right-skewed, we apply *log1p* transformation on SalePrice, i.e. log(SalePrice + 1), below is the result of the transformation, this might help our

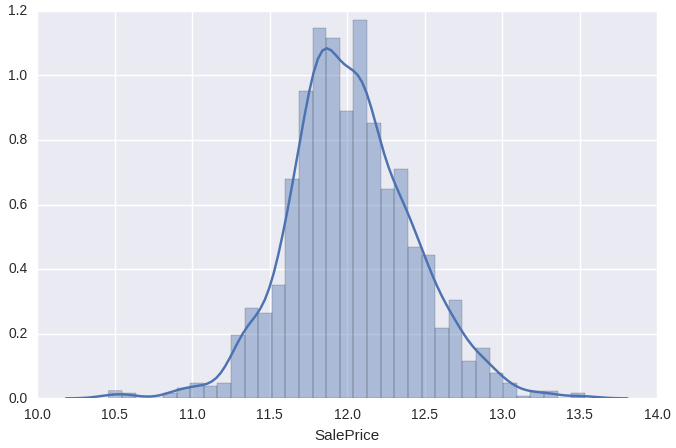


Figure 4: Histogram of SalePrice After *log1p* Transformation

regression (it would not help, for example, tree models) and we apply this transformation to all skewed attributers, see Fig 10 in Appendix B.

3.4 Multicollinearity

Multicollinearity is where two or more features in a regression model being highly correlated, it can make our conclusions from regression less reliable, produce worse result and limit our conclusions.

So we also want to check multicollineary in our attributes, below is a table of correlation between a pair of features in descending order,

```
cor       pair
0.887304  (GarageArea, GarageCars)
0.853926  (Exterior2nd, Exterior1st)
0.851185  (BsmtFinSF2, BsmtFinType2)
0.822983  (TotRmsAbvGrd, GrLivArea)
0.794597  (YearBuilt, GarageYrBlt)
0.783890  (1stFlrSF, TotalBsmtSF)
0.780687  (BsmtFinSF1, BsmtFinType1)
0.680390  (TotRmsAbvGrd, BedroomAbvGr)
0.652879  (GrLivArea, FullBath)
0.636755  (ExterQual, KitchenQual)
```

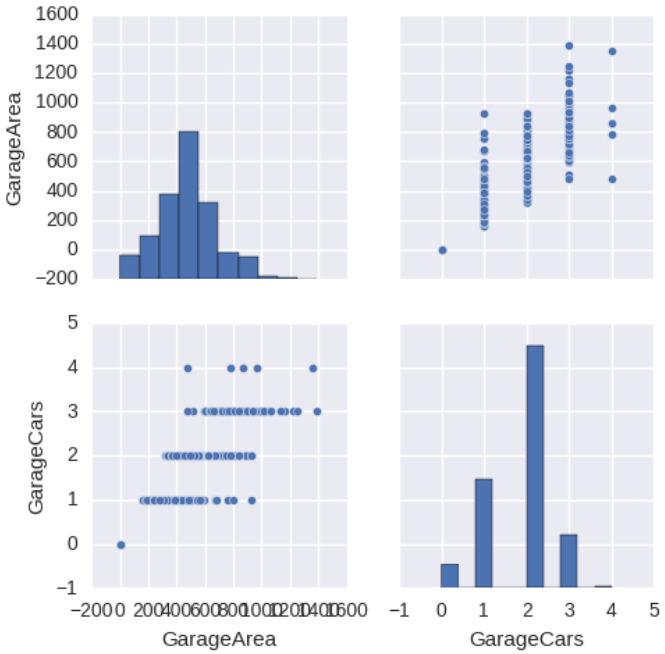


Figure 5: Correlation between GarageArea and GarageCars

The plot is reasonable since in general the number of cars one can park in the garage is highly positive correlated with the garage area. Then we only need to choose one of the two features for regression model.

3.5 Standardization

Substract mean of each column on each data point and division by the standard deviation is applied to train and test data set. The reason we should perform standardization is that, regression is also sensitive to the scales differernce between our features.

For example, *years* and *area*, the former is on the scale around 2,000 but the latter one might around 0.5 scale. This difference would overshadow the effect of *area* in the regression model: $y = a_1 \cdot \text{years} + a_2 \cdot \text{area} + ...$, and $loss = \sum(y - y_{\text{pred}})^2$, the huge value in the square would cover up the value from *area*.

4 Models

4.1 Lasso Regression

Lasso regression is a type of regularized regression. The reason we choose it over ordinary linear regression and ridge regression is that, it is less likely to overfit and this is important in this project since the data set has a relatively large features space thus higher degree of freedom to overfit, the regularization term is less sensitive to outliers and it also comes with feature selection.

This is the objective function Lasso is trying to minimize:

$$\frac{1}{2n_{\text{sample}}} \sum_{i}^{n_{\text{sample}}} ||y - x_i^T w||^2 + \alpha \sum_{j}^{d_{\text{features}}} ||w||_1$$

where $||w||_1$ is the L_1 norm.
The metric we use to evaluate our is r^2 - coefficient of determination,

$$r_2 = \frac{\text{explained variation}}{\text{total variation}}$$

it is a measure of how close the observed data to the fitted regression line.

Linear_model.Lasso[1] from *sklearn* is used to train our model on the 1,460 training set, procedure:

- Perfrom gridsearch by cross-validation[4] to find the best hyper-parameter, in this case - α.

  The gridsearch_cv is performed over 100 different values of α between 0 to 1, where 0 is same as ordinary least square regression, and value closer to 1 corresponds to higher penalty, below is the cross-validation $r2$ result over α in log scale:

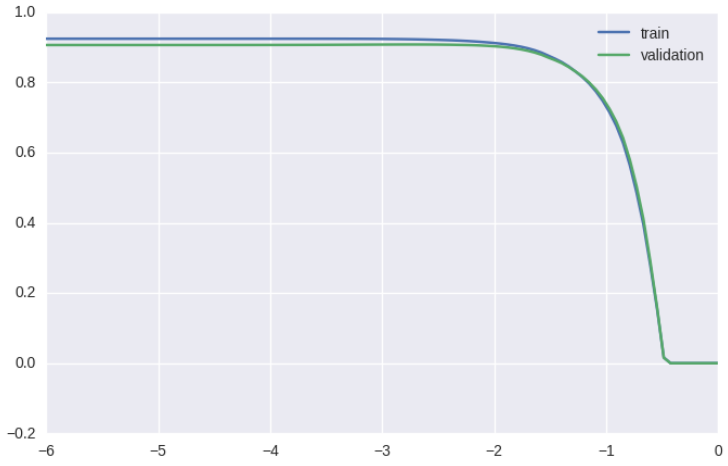


Figure 6: Grid search through cross-validation over α

the best result and parameter is: $r_2 = 0.91078$ and $\alpha = 0.0014$.

Lasso also produces feature importance, i.e. based on the magnitude of coefficient of each feature we can tell which feature is more important to the model:

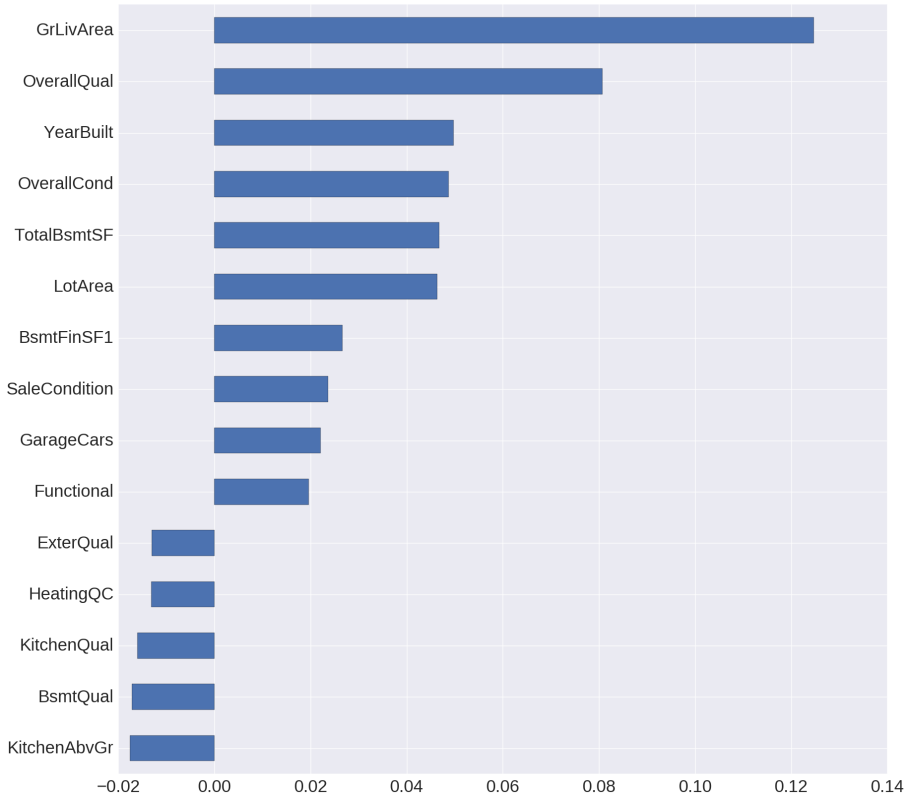


Figure 7: Feature importance

from this we can conclude that,

- GrLivArea: Above grade (ground) living area square feet
- OverallQual: Overall material and finish quality
- YearBuilt: Original construction date
- OverallCond: Overall condition rating
- TotalBsmtSF: Total square feet of basement area
- LotArea: Lot size in square feet

are the top 6 most important features in deciding the *SalePrice* of the house.

- Make predictions on test set, ensemble several predictions over different split of training data to make final prediction, below is the result of each run,

```
[CV 0]
r2: 0.927136691308
RMSLE: 0.119979309487
[CV 1]
r2: 0.92101279528
RMSLE: 0.109265058538
[CV 2]
r2: 0.920464664871
RMSLE: 0.110925238648
[CV 3]
r2: 0.927579929518
```

```
RMSLE: 0.124577910439
[CV 4]
r2: 0.916725315748
RMSLE: 0.122738038905
[CV 5]
r2: 0.921750128229
RMSLE: 0.125474602726
```

where we also include a different evaluation metrics *RMSLE* - root mean squared error in log scale, overall the results are consistent and produce good fit.

4.2 Gradient Boosting

We would like to compare the result from Lasso regression with Gradient Boosting, *gradient boosting* is a technique that fits an additive model in a forward stage-wise manner, in each stage a weak learner is introudced to compensate the shortcomings of previous learners, for example, we start with a base learner called $F(x)$, where it approximately predicts the correct y value,

$$F(x) \approx y$$

we hope to improve the model by adding a function $h(x)$ so that:

$$F(x) + h(x) = y$$

then $h(x)$ can be learned through regression:

$$h(x) = y - F(x)$$

if now the new model $F_1(x) = F(x) + h(x)$ still does not perform very well then we can repeat the addition until satisfied [3].

ensemble.GradientBoostingRegressor[2] in *sklearn* is used to build our model. Similar to the procedures in Lasso, we perform gridsearch_cv to find the best set of parameters, and following is the best r_2 result and the best hyper-parameters:

```
r2 = 0.911430559844
{'max_features': 0.05, 'learning_rate': 0.01, 'max_depth': 3,
 'min_samples_leaf': 3}
```

and we can also compare the important features produced by gradient boosting, see Fig8.

Ensemble the result from model using different resampled data,

```
[CV 0]
r2: 0.936931009404
RMSLE: 0.11085110698
[CV 1]
r2: 0.925904712222
RMSLE: 0.104461990165
[CV 2]
```

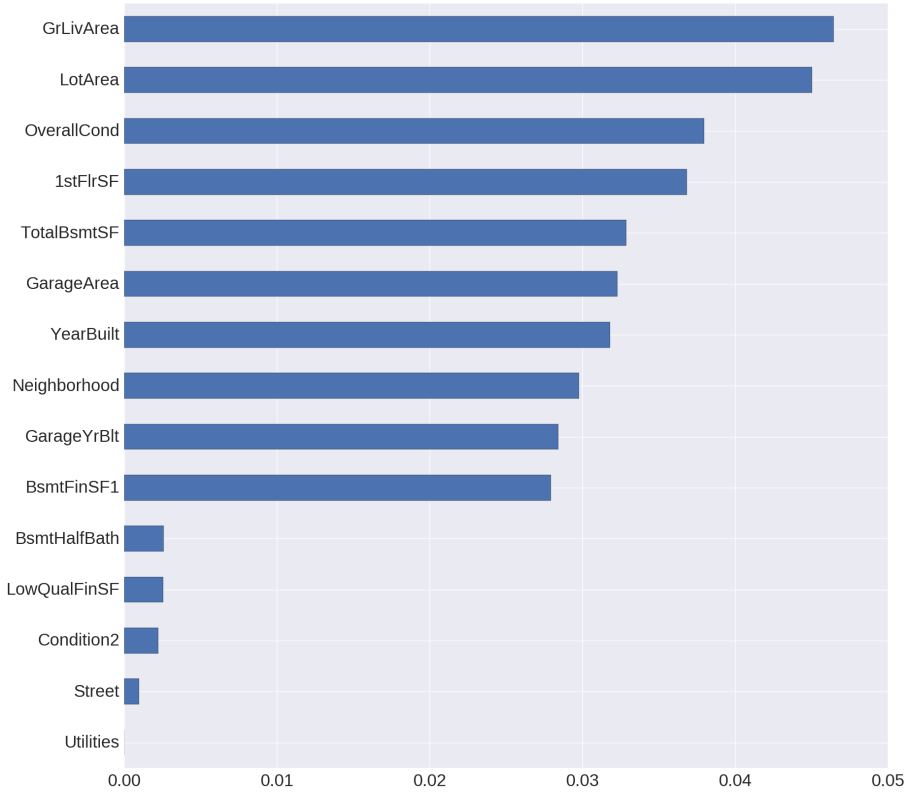


Figure 8: Feature importance in gradient boosting

```
r2: 0.926120655209
RMSLE: 0.106887159554
[CV 3]
r2: 0.940035505163
RMSLE: 0.10932094716
[CV 4]
r2: 0.90448611194
RMSLE: 0.126568657859
[CV 5]
r2: 0.922382972786
RMSLE: 0.123227873843
```

we see that it some times performs better than Lasso but it is less stable, the reason might be the variance and bias trade-off.

The top 6 most importrant features:

- GrLivArea: Above grade (ground) living area square feet
- LotArea: Lot size in square feet
- OverallCond: Overall condition rating
- 1stFlrSF: First Floor square feet
- TotalBsmtSF: Total square feet of basement area
- GarageArea: Size of garage in square feet

5 Conclusions

This housing data set has 6 attributes with more than 15% of missing values, so data imputation is performed. It also contains several right skewed attribute, e.g. *SalePrice* and *log1p* transformation is used to normalize the skewed distribution. Finally, *Lasso regression* and *Gradient boosting* are used to predict *SalePrice* of housing training data. Both of them achieves a good fit to observed data, $r2$ of Lasso is 0.91078 and gradient boosting is 0.91143. They also produce similar top 6 important features to the model, with a few differences, such as *1stFlrSF, GarageArea and YearBuilt*.

References

[1] `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html`

[2] `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html`

[3] `http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf`

[4] `http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

Appendix

A

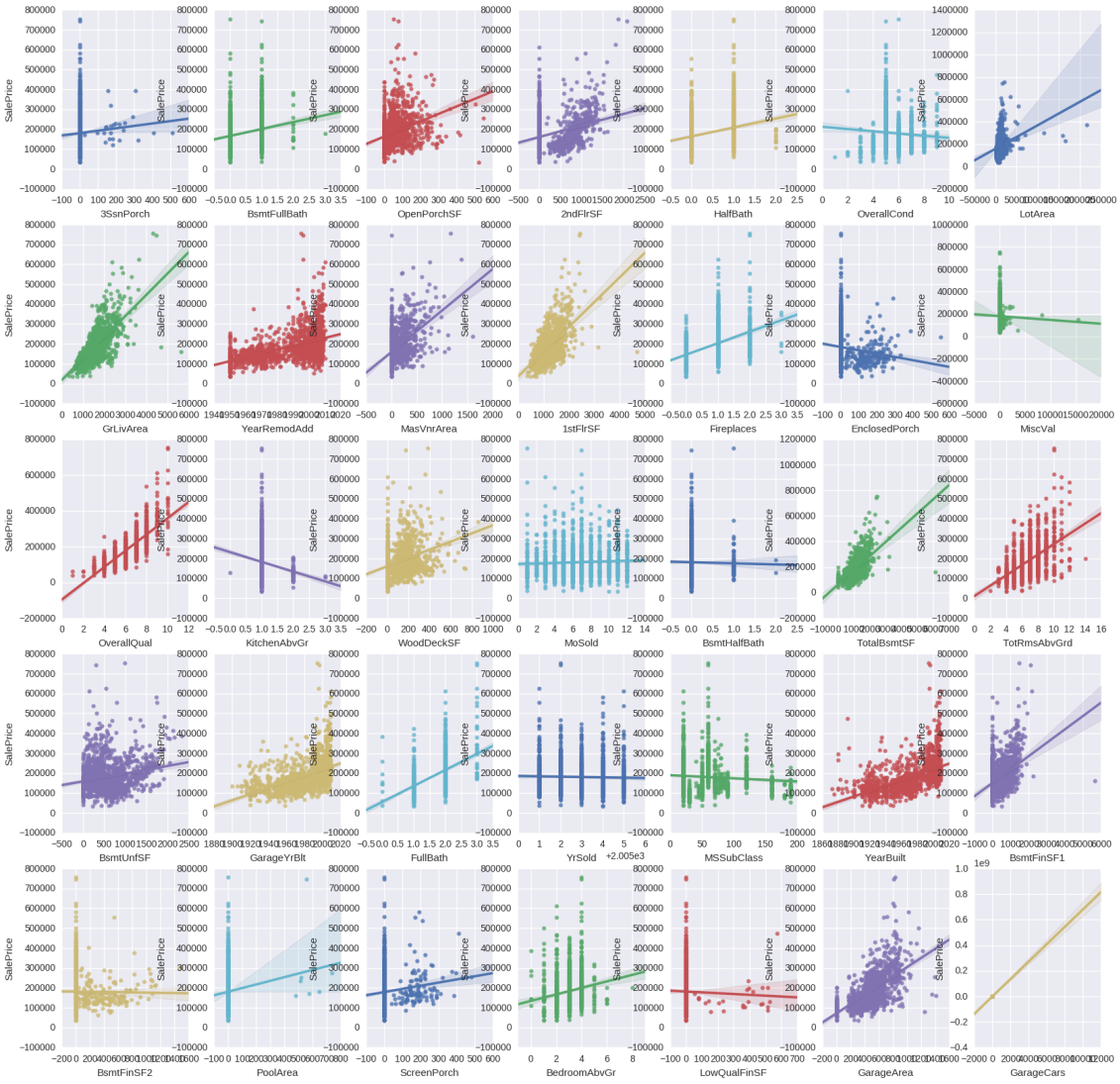


Figure 9: Attributes vs. SalePrice

B

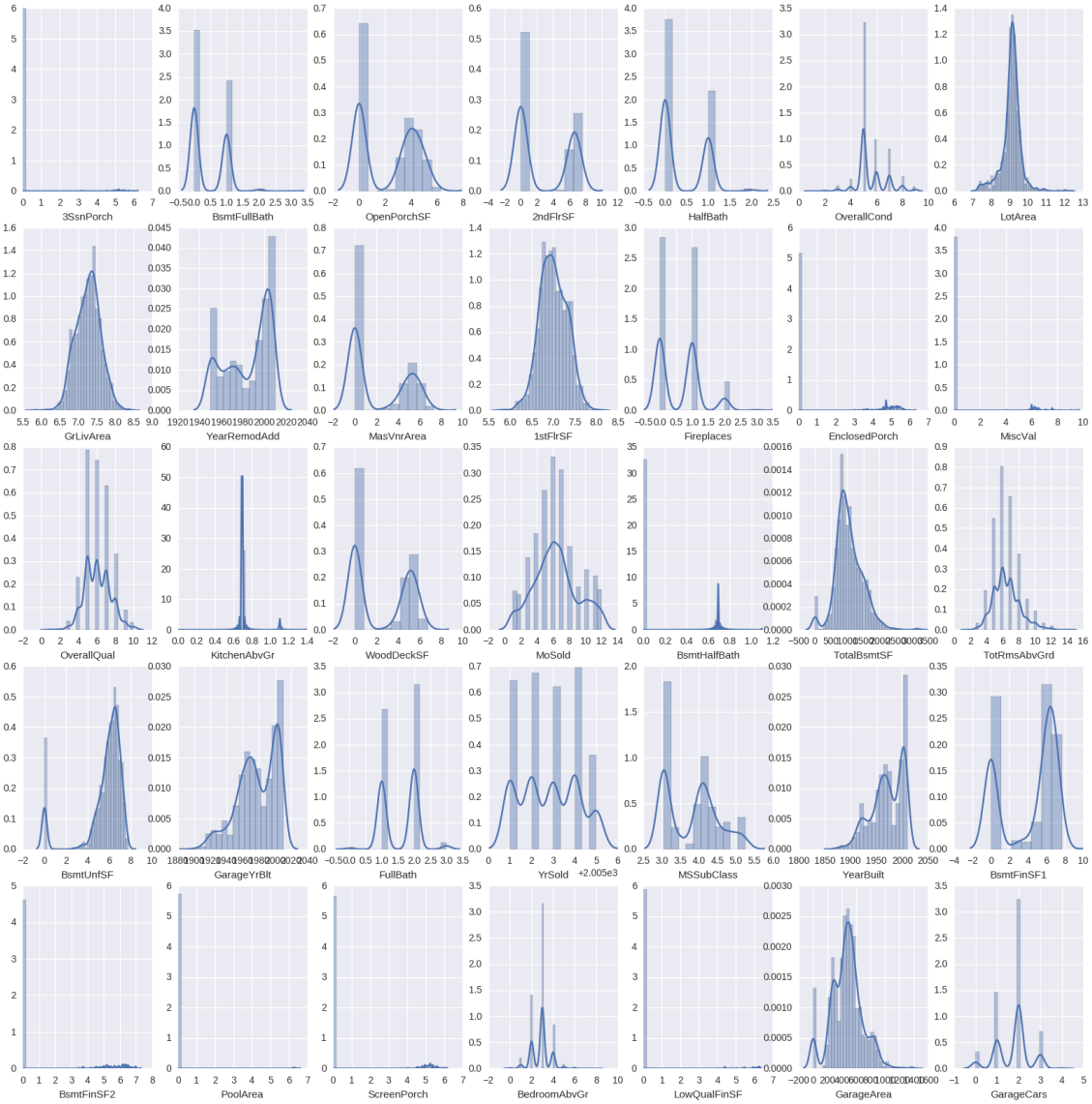


Figure 10: Distribution After *log1p* Transformation