Housing Sale Price Prediction

Piyush Shinde[1], Rahul Raghatate[1], Saurabh Kumar[1]

Abstract
Contrary to the widespread belief that house prices are dependent on the generic factors like number of bedrooms and square area of house, Ames Housing dataset proves that many other factors influence the final price of homes. This dataset contains 79 explanatory variables to describe almost every aspect of the house. Generally house buyers neglect this information. As a result their price estimation is very different from the actual prices. We build a model to predict the prices of residential homes in Ames, Iowa, using advanced regression techniques. This will provide buyers will a rough estimate of what the houses are actually worth. This in turn will help them have better negotiation deals with sellers.

Keywords
RMSE — Lasso — Ridge — XGBoost

[1] *Data Science, School of Informatics and Computing, Indiana University, Bloomington, IN, USA*

Contents

Introduction

Buying a house is a stressful thing.[1] One has to pay huge sums of money and invest many hours and even then there is a persisting concern whether it's a good deal or not. Buyers are generally not aware of factors that influence the house prices. Almost all the houses are described by the total area in square foot, the neighborhood and number of bedrooms. Sometimes houses are even priced at X dollars per square foot. This creates an illusion that house prices are dependent almost solely on the above stated factors.

Most of the houses are bought through real estate agents. People rarely buy directly from the seller, since there are a lot of legal terminology involved and people are unaware of them. Hence real estate agents are trusted with the communication between buyers and sellers as well as laying down a legal contract for the transfer. This just creates a middle man and increases the cost of houses. Therefore the houses are overpriced and a buyer should have a better idea of the actual value of these houses.[2]

There are various tools, like Zillow and Trulia, available online to assist a person with buying houses. These tools provide a price estimation of various houses and are generally free for use. These tools incorporate many factors to estimate the house prices by providing weights to each factor. For example, Zillow creates Zestimate of houses which is "calculated three times a week based on millions of public and user-submitted data points" [3]. The median error rate for these estimates are quite low. The main problem with these tools is that they are heavy on advertisements and they promote real estate agents. Zillow provides paid premium services for real estate agents and this is their main source of income.[4] Estimates of actual house prices will help buyers to have better negotiations with the real estate agents, as the list price of the house and much higher than the actual price. Our prediction model will provide the buyers with these estimates. We use the Ames Housing dataset provided by kaggle.[5]

We first analyze the data and for finding trends in data. We perform dimensionality reduction on the dataset using PCA algorithm and feature_selection module in sklearn package for python. We predict the final house prices using linear regression models like Ridge and Lasso. We also use advanced regression techniques like gradient boosting using XGBoost library in python.

1. Background

Most of the variables were directly related to property sales. Description for these variables can be found here [14]. Approximately 80 variables focus on the quality and quantity

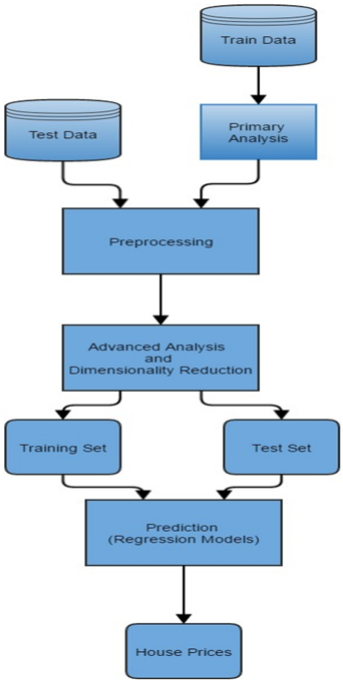

Figure 1. Architecture Diagram

of many physical attributes of the property. For instance lot_size, neighborhood, sq_ft_area, basement which a home buyer would want to know about a potential property.

20 continuous variables relate to various area dimensions for each observation, from the available set of variables. Simple variables like typical lot size and total dwelling square footage can be found on most common home listings, other more specific variables are quantified in the data set. Area measurements on the basement, main living area, and even porches are broken down into individual categories based on quality and type. The large number of continuous variables in this data set lays platform to consider various methods of using and combining the variables. There are few variables quantifying the number of items occurring within the house like the number of kitchens, bedrooms, and bathrooms etc. Thus, this large set of variables provides us with opportunity to construct various models and select best fitting model for predicting housing prices.

2. Algorithm and Methodology

Our approach is to input the Train and Test data, provided by kaggle, and do primary analysis on the Train data. We then do the basic preprocessing on the combined Test and Train data. We perform dimensionality reduction through PCA and feature_selection modules in sklearn package of python. Data is then split into training and test set and fed to various regression models. The final output is the predicted SalePrice for the test data. The following figure shows our model.

2.1 Input

This is a small step which involves taking input *train.csv* and *test.csv* as dataframes. The Train data has 81 columns and 1460 rows. These columns include 79 explanatory variables to describe every aspect of house. Test data is fairly similar to Train data. Test Data has 80 columns and 1459 rows. It has no SalePrice column as it's value is to be predicted by our model.

2.2 Early Analysis

We study the Train data closely, through various graphs, to determine any trends in the data.
The first graph is a distribution of *LotArea*. The below graph is close to a bell curve. This shows that *LotArea* has a normal distribution. We use the seaborn package to plot the graph.

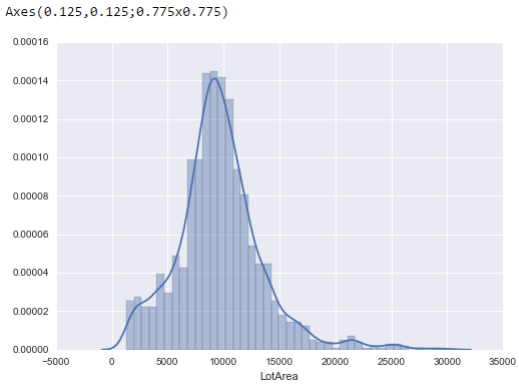

Figure 2. *LotArea* Distribution

The second graph is a distribution of *SalePrice*. The below graph appears to be somewhat right skewed. This suggests that the mean for *SalePrice* is greater than its median.

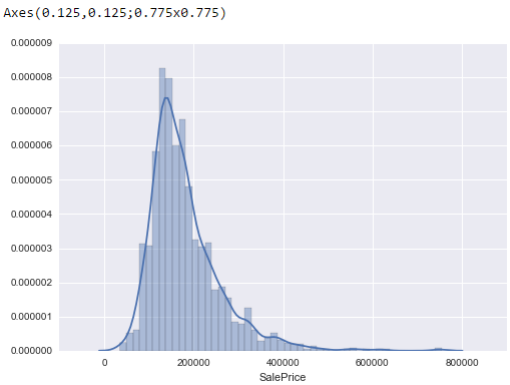

Figure 3. *SalePrice* Distribution

The next graph is a boxplot for *YrSold* and *LotArea*. The below graph shows that median of *LotArea* is almost same for the different years (2006-2010). Years 2006, 2007 and 2010 have many outliers, as can be seen from the graph. Year 2009 has significantly fewer outliers. Year 2007 and 2010 have small inter quartile range (IQR). This signifies data is more consistent and lie closer to median. We have used matplotlib library in python to plot this graph.

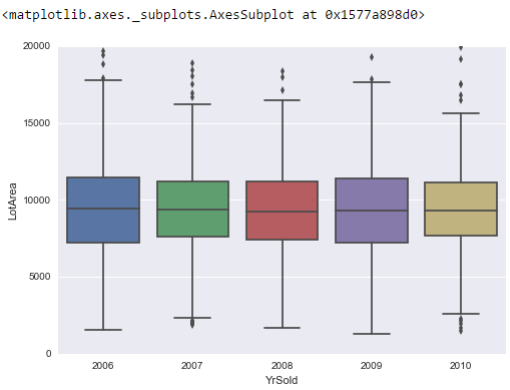

Figure 4. Boxplot for *SalePrice* & *YrSold*

The last graph is a boxplot for *YrSold* and *SalePrice*. The below graph shows that every year (2006-2010) has significant outliers. All the outliers lie on the upper side of the graph. This shows that the outliers have higher *SalePrice*. Every year has similar median which shows that the median *SalePrice* remains unchanged over the years. This suggests that inflation in house prices over the years is negligible. The inter quartile range for 2008 is smaller than the rest. This shows that data is more consistent for year 2008.

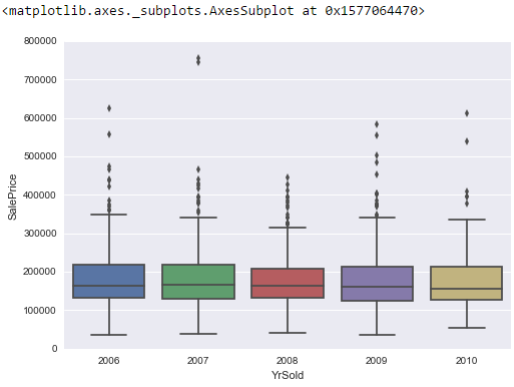

Figure 5. Boxplot for *SalePrice* & *YrSold*

2.3 Preprocessing

The following flow diagram shows the various steps involved in preprocessing step. We concatenate the Test Data and Train Data. This Train Data does not have the *SalePrice* column.

Our model uses linear regression techniques. Input variables with normal distribution is better suited for such techniques. Hence we use log transformation to normalize the data. Firstly we normalize the *SalePrice* using log transformation.

A log transformation of a skewed variable decreases its skewness. If a numeric variable has highly skewed, i.e. skewness > 0.75, we perform log transformation to reduce the skewness. This makes the trend detection in data easier.

We fill the missing values in the data with the respective mean of their column. All the categorical variables are converted to numerical variables using *get_dummies()* function in the pandas library in python. The regression algorithms cannot process categorical variables. Therefore this conversion is needed.

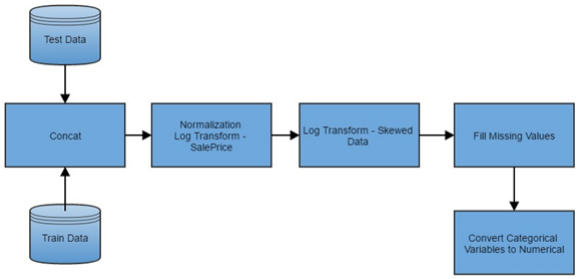

Figure 6. Flow Chart

2.4 Advanced Analysis and Dimension reduction

Dimensionality Reduction is the process of reducing the number of random variables under consideration.[6]

2.4.1 Correlation

(Pearson correlation coefficients) Before dimensionality reduction we calculate the Pearson correlation coefficients between the numeric variables in the dataset. A high correlation coefficient between variables or columns means that the different variables have similar trend and behave similarly over their range. Using this information we can remove few variables before we implement the regression techniques. Fewer dimension helps the regression model perform better and faster.

We use the $corrcoef()$ function in numpy library of python. This function returns Pearson product-moment correlation coefficients.[7] The relationship between the correlation coefficient matrix, R, and the covariance matrix, C, is

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii} * C_{jj}}} \quad (1)$$

We observe that the correlation coefficient between *SalePrice* and other variables is generally very high except for columns like *MiscVal*, *Fireplaces*. This means that we should perform dimensionality reduction on this dataset.

2.4.2 PCA

(sklearn.decompositon.PCA) *"Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space"*.[8]
The PCA algorithm makes uses eigen values and eigen vectors. Eigenvector matrix is used to transform high dimensionality space to low dimensionality space. We use the PCA function in the sklearn.decompositon module of python. It fits the data and projects them into a low dimensionality space.

2.4.3 Feature Selection

(sklearn.feature_selection) There are many ways to perform feature selection. We use the Variance Threshold method. It removes all features whose variance doesn't meet some threshold.[8] Therfore, it automatically removes all zero-variance

features, i.e. features with same sample values throughout the entire range.

As an example, suppose that we have a dataset with boolean features, and we want to remove all features that are either one or zero (on or off) in more than 80% of the samples. Variance of random variables are represented by the formula given below.

$$Var[X] \quad = \quad p(1-p) \tag{2}$$

,where p is the probability.

We choose the probability to be 0.8. Therefore the,

$$Threshold = (.8 * (1 - .8)) \tag{3}$$

We use this value with the $VarianceThreshold()$ function and transform the data.

2.5 Prediction

This is the main part of our model. Here we model the relationship between *SalePrice* (dependent variable) and the explanatory variable (independent variable). Building a linear relationship helps us to predict the values of SalePrice for the test set. Train set is used to build this relationship. We have use three algorithms for prediction, but before that we define a $rmse()$ function to calculate the mean squared error defined by the formula given below[9].

$$MSE \quad = \quad \frac{1}{n}\sum_{i=0}^{n}(\overline{A} - A)^2 \tag{4}$$

$\overline{A}$ is a vector of n predictions and A is a vector of observed values. We take the square root of this value to calculate the root mean square error.

2.5.1 Ridge

This is a regularized linear model. The loss function here is the linear least square function and regularization is given by l2-norm.[10] Regularization reduces variance of estimates. We use Ridge() function from the sklearn.linear module. The regularization strength is α here. We try different values of α and calculate root mean square error (rmse) to compare the results. The different α values are given below -

α's = [0.05, 0.1, 0.3, 1, 3, 5, 10, 15, 30, 50, 75]

We plot a graph for rmse and α to show the result. With too small alpha, the regularization is too strong and the complexities of the data is lost. With large α model begins too overfit. Hence an α value of 10 gives the least rmse, ie rmse = 0.127.

We make the final predictions using Ridge model and store results as $ridge\_pred$.

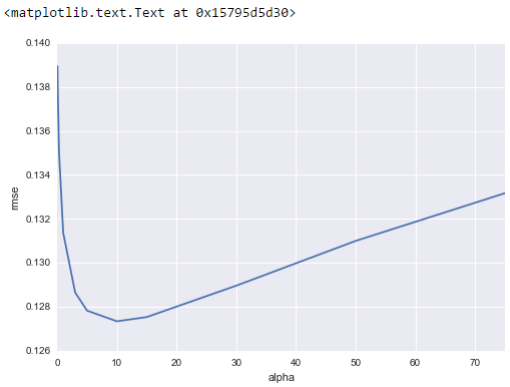

Figure 7. Error vs α

2.5.2 Lasso

"Linear Model trained with L1 prior as regularizer"[11]. We use $LassoCV()$ function from the sklearn.linear module. We do the same thing here. We take a set of α values and calculate the rmse. The default value of α for Lasso is 1.0. The α values taken are given below –

α's = [1, 0.1, 0.001, 0.0005]

The mean of rmse comes to 0.123. Lasso picked 111 variables and eliminated the other 177 variables. We plot a horizontal bar graph to check which is the most important coefficients for Lasso Model. This graph shows that most important feature

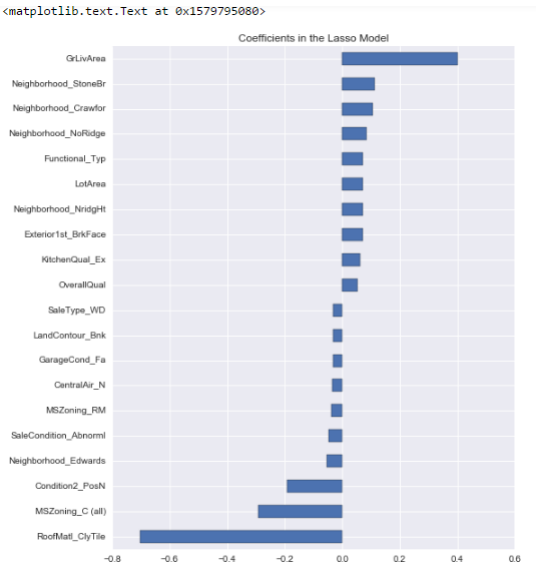

Figure 8. Coefficients in the Lasso Model

in predicting house price is *GrLivArea*. Few features have negative values. We will talk about this in the results section.

2.5.3 Gradient Boosting

"Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees."[12]

We use XGBoost which is extreme gradient boosting. It provides scalable gradient boosting. We use *xgb.cv*() function from the xgboost library in python. We define the maximum depth of decision tree as 2. We calculate the rmse mean for Test and Train set and plot it. We tune XGBoost model with

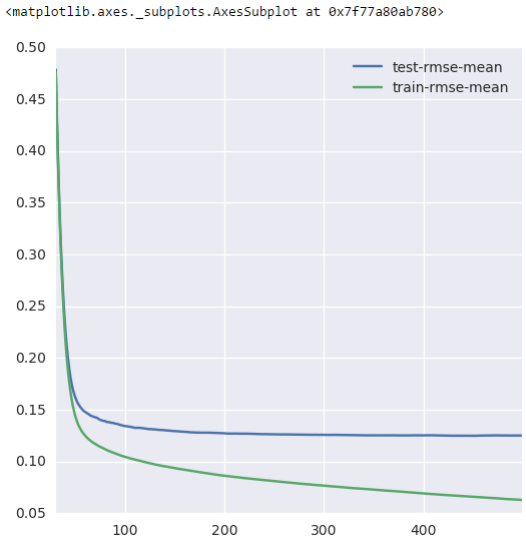

Figure 9. Error mean for Test & Train Sets

the results of the graph and set the *n_estimators* parameter for XGBRegressor to 360. The *max_depth* is chosen as 2.

Finally we make predictions using the Lasso and XGBoost model and store results as *lasso_preds* and *xgb_preds*.
For our final prediction we use the formula given below:

$$preds = 0.8 * lasso\_preds + 0.2 * xgb\_preds \quad (5)$$

3. Experiments and Results

We have made two submissions on kaggle, initial and a final one. In the initial submission, we did not implement the Gradient Boosting and our final predictions were based on the Ridge Predictions *(ridge_preds)* and Lasso Predictions *(lasso_preds)*.
We used the following formula to calculate the final predictions.

$$pred = 0.8 * lasso\_preds + 0.2 * ridge\_preds \quad (6)$$

We received a 0.12097 root mean square logarithmic error *(rmsle)* from kaggle, for our solution.

Individually, Lasso model performed better than Ridge. As we had seen earlier, the min rmse for Ridge was 0.127 and the mean rmse for Lasso was 0.123. We draw a scatter plot between Ridge and Lasso predictions.

The graph is almost linear, except towards the top Ridge model predicts higher house prices. The main reason because Lasso performs better is that it performs feature selection and parameter shrinkage automatically. This is not done by the Ridge algorithm. We saw earlier the result of Lasso Predictions that Lasso picked 111 variables and eliminated the other

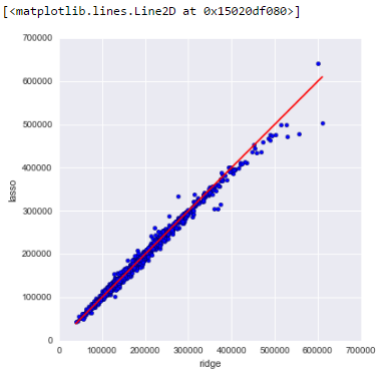

Figure 10. Lasso vs Ridge Predictions

177 variables. Ridge Algorithm cannot zero out coefficients for variables. It either uses all variables or none of them.[13]

We saw earlier that some variables had negative coefficients for the Lasso model. Generally these are the categorical variables which were converted to numeric values. Since they have fixed values, they make less sense than the numeric variables.

As discussed above, for our final submission we implemented Gradient Boosting using XGBoost. Our final predictions were based on Lasso Predictions *(lasso_preds)* and XGBoost Predictions *(xgb_preds)*. We used the following formula to calculate the final predictions.

$$preds = 0.8 * lasso\_preds + 0.2 * xgb\_preds \quad (7)$$

We received a 0.11623 root mean square logarithmic error (rmsle) from kaggle, for our solution. This is lower than the rmsle for our initial method, 0.12097. Hence we stick to the recent solution. We draw a scatter plot between Lasso and XGBoost predictions.

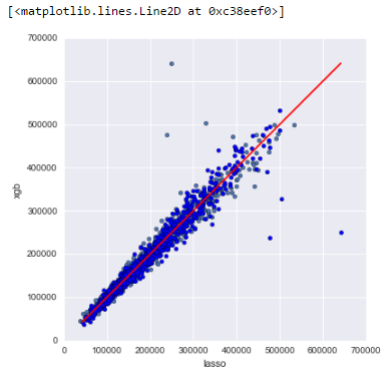

Figure 11. XGBoost vs Lasso Predictions

This graph is fairly linear and dense towards the lower half. XGBoost model involves a lot of tuning. As a future upgrade we can finely tune it to provide better results and even less rmsle.

Currently for the above designed model we have secured a rank of 161 (top 6%) in Kaggle competition.

3.1 Build Specifications

- Processor: Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz
- Memory: 16.0 GB
- Software Platform: Python 3.5
- OS: Windows 10 (64-bit)

4. Conclusion

We predicted the *SalePrice* of the houses for the given Ames Housing dataset using two different methods. The first one included Ridge and Lasso model. The second one had Lasso and XGBoost. Since the second model performed better we the predictions provided by that model. This prediction data will help eventual buyers to have a better knowledge of the property. This will in turn help them to have better negotiation deals with the real estate agents.

5. Future Work

Our model had a low rmsle score, but there is still room for improvement. In a real world scenario, we can use such a model to predict house prices. This model should check for new data, once in a month, and incorporate them to expand the dataset and produce better results.

In Ames Iowa dataset, variables have high correlation. We saw that the Lasso model eliminated few variables in the later stages. We can try out other dimensionality reduction techniques like Univariate Feature Selection and Recursive feature elimination in the initial stages.

We can try out other advanced regression techniques, like Random Forest and Bayesian Ridge Algorithm, for prediction. Since the data is highly correlated, we should also try Elastic Net regression technique.

Acknowledgments

We would like to thank Dr. Dean DeCock, Truman State University for compiling the Ames Iowa dataset that we used in this project. Many thanks to Prof. Mehmet Dalkilic at Indiana University Bloomington for his academic as well as professional guidance. We would also like thank Hasan Kurban and Marcin Malec for guiding us along our project milestones.

References

[1] Press Association (2016). Buying a house 'more stressful than having a child'. http://www.msn.com/en-gb/money/personalfinance/buying-a-house-more-stressful-than-having-a-child/ar-CChG5t

[2] Tara-Nicholle Nelson. 7 Negotiating Tips for Homebuyers. http://www.hgtv.com/design/real-estate/7-negotiating-tips-for-homebuyers

[3] Hal M. Bundrik, CFP (2015). Putting Zillow Zestimates' Accuracy to the Test. https://www.nerdwallet.com/blog/mortgages/putting-zillow-zestimates-accuracy-test/

[4] Amy Fontinelle (2016). Why Zillow Is Free and How It Makes Money (Z). http://www.investopedia.com/articles/personal-finance/110615/why-zillow-free-and-how-it-makes-money.asp

[5] House Prices: Advanced Regression Techniques (2016). https://www.kaggle.com/c/house-prices-advanced-regression-techniques

[6] Dimensionality reduction. https://en.wikipedia.org/wiki/Dimensionality_reduction

[7] numpy.corrcoef. SciPy.org. https://docs.scipy.org/doc/numpy/reference/generated/numpy.corrcoef.html

[8] Removing features with low variance. Feature selection. scikit learn. http://scikit-learn.org/stable/modules/feature_selection.html

[9] Mean squared error. https://en.wikipedia.org/wiki/Mean_squared_error

[10] sklearn.linear_model.Ridge. scikit learn. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

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

[12] Gradient boosting. https://en.wikipedia.org/wiki/Gradient_boosting

[13] Rob Hyndman (2010). When should I use lasso vs ridge? http://stats.stackexchange.com/questions/866/when-should-i-use-lasso-vs-ridge

[14] Ames Iowa: Alternative to the Boston Housing Data Set (2010). http://www.amstat.org/publications/jse/v19n3/decock/DataDocumentation.txt