Predicting Housing Prices in Ames, Iowa

Cao Jilin
caojilin@berkeley.edu

May 13, 2021

1 Introduction

The goal of this project is to use a linear model[6] to predict the housing prices in Ames, Iowa. Ideally, the same methods applied in this project can be extended to other similar housing data. The dataset, collected by Bart de Cock in 2011[5], covers house prices in Ames, Iowa from the period of 2006–2010. It is considerably larger than the famous Boston housing dataset of Harrison and Rubinfeld (1978)[2], containing both more examples and more features. If someone would like to buy a house in Ames, the following questions might be interesting to them: What is the predicted sale price of a 2B2B house with 1500 square feet living area? or Which is the more important variable in predicting sale price, garage area, or open porch area? In this project, we want to build a linear model to answer this type of question and we are also interested in which variables are most important in terms of predicting housing price. The model is particularly useful for both customers and housing agencies. Customers can quickly estimate how much they need to pay for their ideal house. Housing agencies can also modify the house to increase the housing value if they can. For instance, the Remodel date is an important predictor, houses can sell more if they are remodeled recently.

2 Data Description

The Boston Housing Data has only 506 observations, 14 variables, and was collected in 1978. It is pretty outdated nowadays. Ames data has 2589 observations(between 2006-2010), 341 observations(2010) and 82 variables. Having more data and variables gives us more flexibility to choose the model and analyze the data. The detailed documentation of this data set can be found here: http://jse.amstat.org/v19n3/decock/DataDocumentation.txt. We use 2589 observations(between 2006-2010) as train data and 341 observations(2010) as test data.

2.1 Data Preprocessing

2.1.1 Missing Value

The data set has a lot of missing values.

- 94% data of "Alley" are missing.

- 47% data of "FireplaceQu" are missing.
- 99.5% data of "PoolQC" are missing.
- 81% data of "Fence" are missing.
- 96% data of "MiscFeature" are missing.

Since these variables have too many missing values, they will be ignored during the analysis. For variables that have a small portion of missing values, we will impute by either mean or mode. Specifically, for categorical variables, we will use mode and for numerical variables, we will use mean. Figure 1 has more detail.

2.2 Dummify categorical variabels

To deal with discrete values. This includes features such as "MSZoning". Multi-class categorical variables are transformed into one-hot encoding dummy variables. For instance, "MSZoning" assumes the values "RL" and "RM". Dropping the "MSZoning" feature, two new indicator features "MSZoning_RL" and "MSZoning_RM" are created with values being either 0 or 1. According to one-hot encoding, if the original value of "MSZoning" is "RL", then "MSZoning_RL" is 1, and "MSZoning_RM" is 0. Note that after we dummify the categorical variables, we have 332 features now.

2.3 Nomalize skewed numerical variables

From Figure 2, we can see our numerical variables have a wide variety of data types and units. To put all features into same scale, we standardize the data by re-scaling features to zero mean and unit variance:

$$x \leftarrow \frac{x - \mu}{\sigma}$$

, where μ and denote mean and standard deviation, respectively. Intuitively, we standardize the data for two reasons. First, it proves convenient for optimization. Second, because we do not know which features will be relevant, we do not want to penalize coefficients assigned to one feature more than on any other.

```
Bsmt Cond           0.027304
Bsmt Qual           0.027304
BsmtFin Type 1      0.027304
Mas Vnr Area        0.007850
Mas Vnr Type        0.007850
Bsmt Half Bath      0.000683
Bsmt Full Bath      0.000683
BsmtFin SF 1        0.000341
Electrical          0.000341
Garage Cars         0.000341
Garage Area         0.000341
Total Bsmt SF       0.000341
Bsmt Unf SF         0.000341
BsmtFin SF 2        0.000341
```

(a)

```
Pool QC             0.995563
Misc Feature        0.963823
Alley               0.932423
SalePrice           0.883618
Fence               0.804778
Fireplace Qu        0.485324
Lot Frontage        0.167235
Garage Cond         0.054266
Garage Qual         0.054266
Garage Finish       0.054266
Garage Yr Blt       0.054266
Garage Type         0.053584
Bsmt Exposure       0.028328
BsmtFin Type 2      0.027645
```

(b)

Figure 1: percentage of missing values

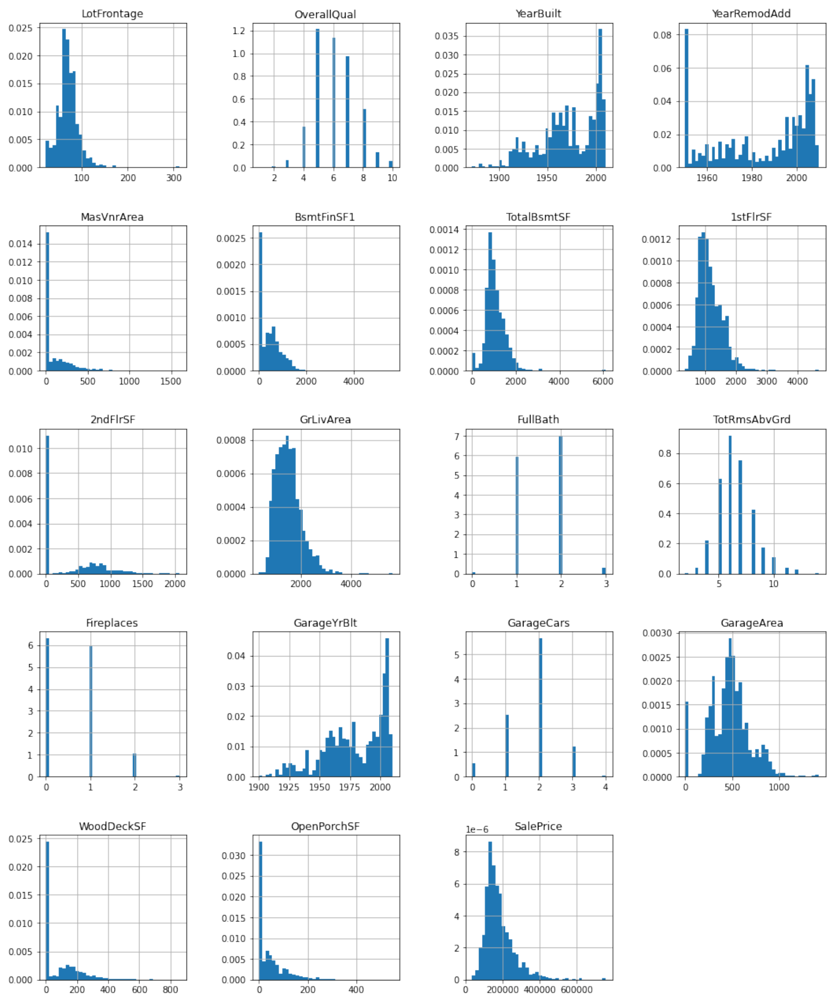


Figure 2: distribution of numerical variables

3 EDA

3.1 Numerical variables

After we clean up the data, the next step is to explore our data. First, we plotted the heatmap which shows the pair-wise correlations among all features. We can see the correlations between different variables. We are most curious about which explanatory variables are most related to the target variable "SalePrice". As the correlation heatmap shows, "SalePrice" is more related to "OverallQual", "GrLivArea" and "GarageCars" etc.

Figure 4 shows the scatter plots of these highly correlated variables. The positive relationship is pretty clear. For example,

- The larger the GrLivArea(Above grade (ground) living area square feet) the higher the SalePrice.
- The larger the TotalBsmtSF(Total square feet of basement area), the higher the SalePrice.
- The better the Overall Qual(Overall material and finish quality), the higher the SalePrice.
- The large the GarageArea(Size of garage in square feet), the higher the SalePrice.

3.2 Categorical variables

The relationship between SalePrice and categorical variables is also informative. For example, in Figure 5, FV(Floating Village Residential) is the most expensive, and Agriculture zoning housing is the least expensive. Figure 6 shows the average housing prices in different neighborhoods. Neighborhoods such as "Stone Brook" and "Northridge" are most expensive, while "Briardale" and "Meadow Village" are the cheapest. We are interested in these plots because we want to select variables that have large variations instead of small variations. If the variance is low or close to zero, then a feature is approximately constant and will not improve the performance of the model.[1]

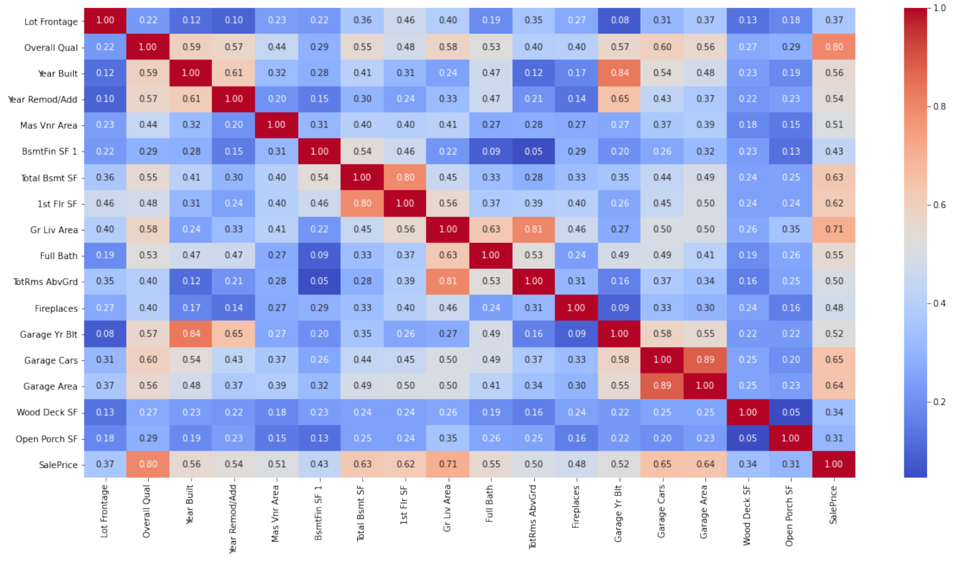


Figure 3: heatmap

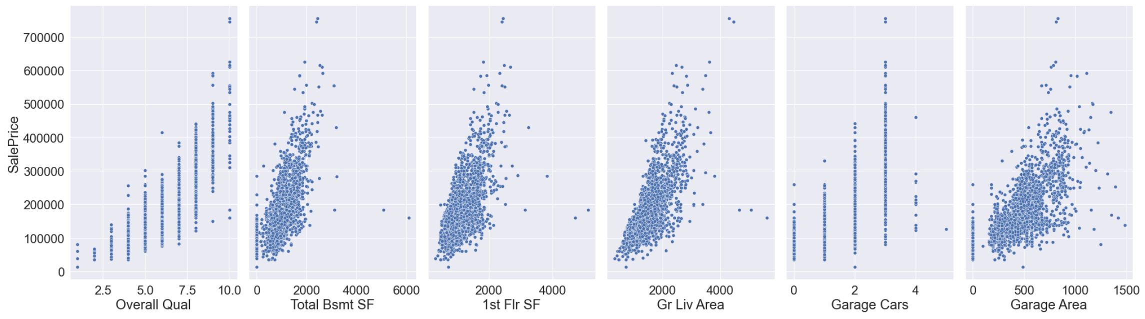


Figure 4: scatter plot

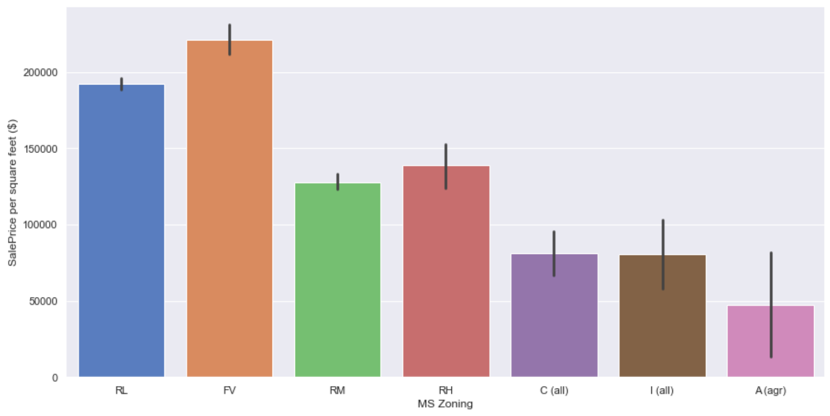


Figure 5: bar plot

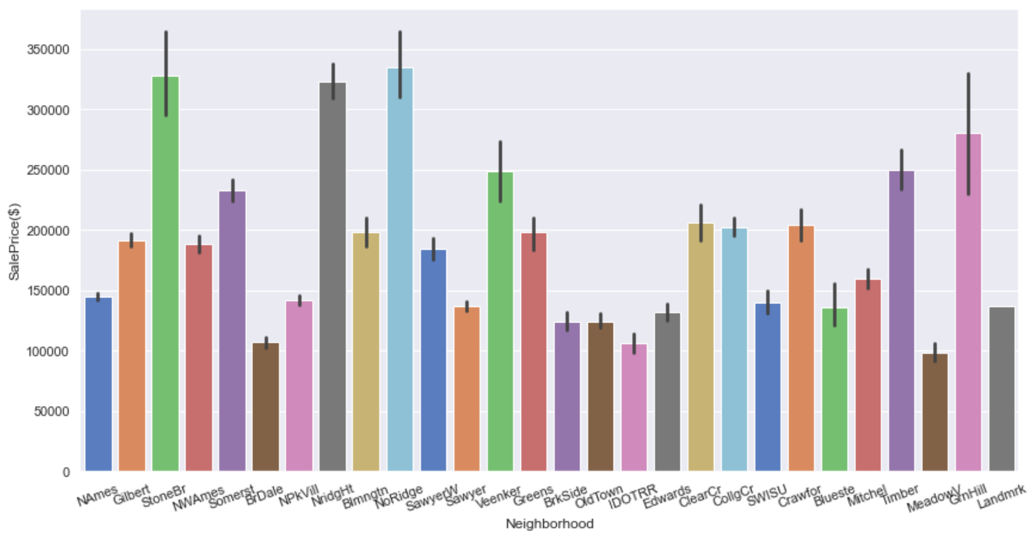


Figure 6: bar plot

4 Methods

With house prices, we care more about relative quantities more than absolute quantities. That is we tend to care more about relative error $\frac{y-\hat{y}}{y}$ than the absolute error $y - \hat{y}$. For instance, if our prediction is off by \$100,000 when estimating the price of a house in Riverside, California, where the value of a typical house is \$350,000, then we are probably doing a horrible job. On the other hand, if we make an error by this amount in Los Altos Hills, California, this might represent a stunningly accurate prediction (there, the median house price exceeds 4 million dollars).
One way to address this problem is to measure the discrepancy in the logarithm of the price estimates. After all, a small value δ for $|\log y - \log \hat{y}| \le \delta$ translates into $e^{-\delta} \le \frac{\hat{y}}{y} \le e^{\delta}$. This leads to the following root-mean-squared-error between the logarithm of the predicted price and the logarithm of the label price[8]:

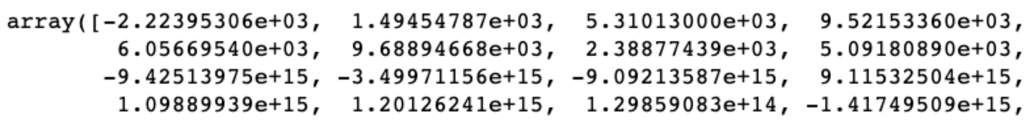
$$log\_rmse(y, \hat{y}) = \sqrt{\frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (\log_e(1 + y_i) - \log_e(1 + \hat{y}_i))^2}$$

4.1 OLS

OLS minimizes residual sum of squares:

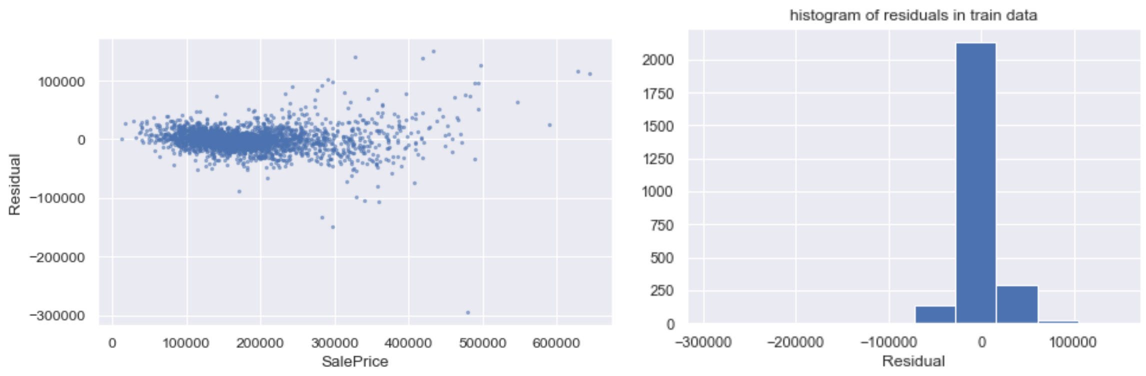
$$\min_w ||Xw - y||_2^2$$

We first fit OLS with all 343 features(including dummy variables). The train data gives $R^2 = 0.936$, while test data gives $R^2 = -4.58 \times 10^{18}$. The out-of-sample data(test data) generates a very large negative R^2. This indicates the



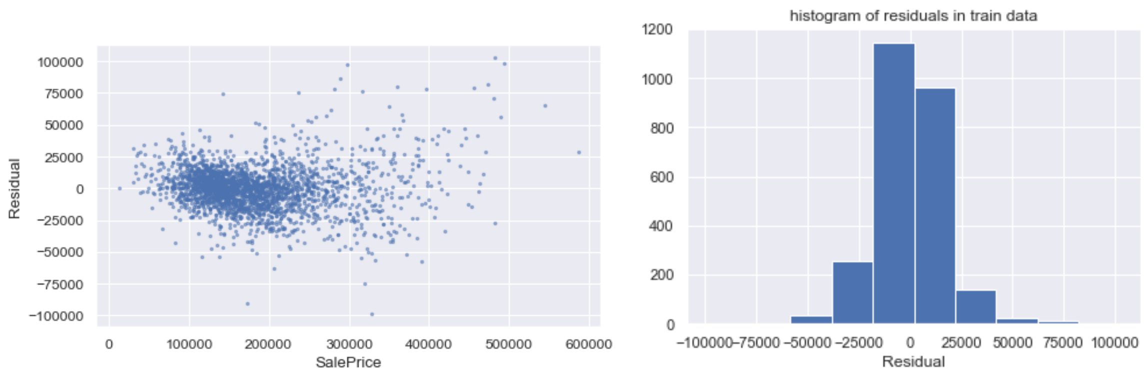
```
array([-2.22395306e+03,  1.49454787e+03,  5.31013000e+03,  9.52153360e+03,
        6.05669540e+03,  9.68894668e+03,  2.38877439e+03,  5.09180890e+03,
       -9.42513975e+15, -3.49971156e+15, -9.09213587e+15,  9.11532504e+15,
        1.09889939e+15,  1.20126241e+15,  1.29859083e+14, -1.41749509e+15,
```

Figure 7: the first 16 coefficients of the OLS model



(a) (b)

Figure 8: Residual plots of OLS in train data



(a) (b)

Figure 9: Residual plots of OLS in train data after removing outliers

model is over-fitting[4]. Examining the coefficients, we see that the absolute value of all the coefficients is very large.

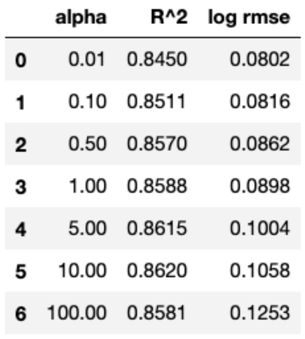
Figure 8(a) shows that there are a couple of outlier data points. These are the points that are far away from the y=0 axis. Some residual is greater than $\pm\$100,000$. After removing these outliers and re-fit the OLS, the model gives $R^2 = 0.951$ in train data and $R^2 = -8.82 \times 10^{16}$ in test data. Although the residual plots look better, the model still gives very large coefficients. This indicates that we need to constraint the coefficients and avoid over-fitting.

4.2 Ridge

Ridge regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients. The ridge coefficients minimize a penalized residual sum of squares:

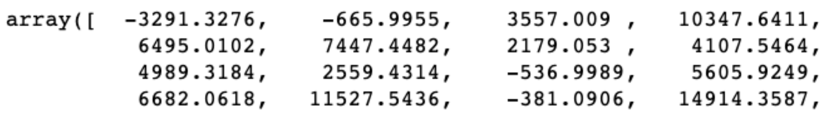
$$\min_{w} ||Xw - y||_2^2 + \alpha||w||_2^2$$

The complexity parameter α controls the amount of shrinkage: the larger the value of α, the greater the amount of shrinkage, and thus the coefficients become more robust to collinearity[9].
The α is chosen by 5-fold cross-validation from a wide range : [0.01, 0.1, 0.5, 1, 5, 10, 100].



	alpha	R^2	log rmse
0	0.01	0.8450	0.0802
1	0.10	0.8511	0.0816
2	0.50	0.8570	0.0862
3	1.00	0.8588	0.0898
4	5.00	0.8615	0.1004
5	10.00	0.8620	0.1058
6	100.00	0.8581	0.1253

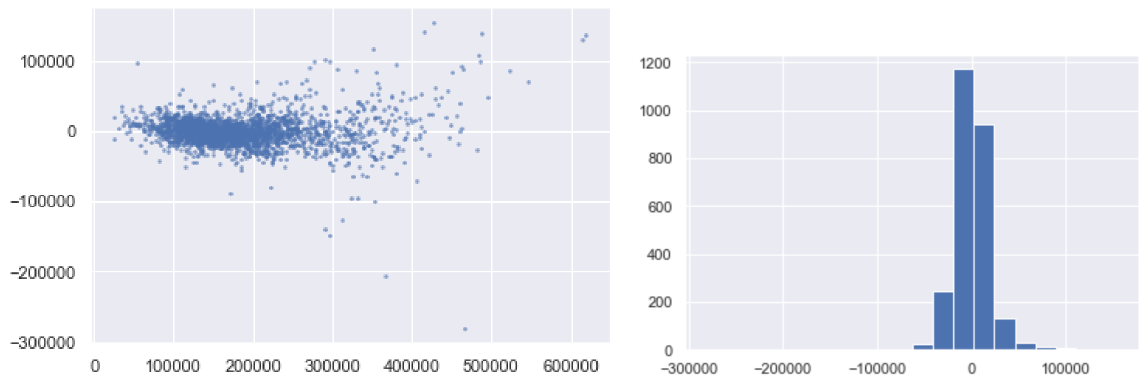
Figure 10: cross-validation for selecting alpha



```
array([  -3291.3276,    -665.9955,    3557.009 ,   10347.6411,
          6495.0102,    7447.4482,    2179.053 ,    4107.5464,
          4989.3184,    2559.4314,    -536.9989,    5605.9249,
          6682.0618,   11527.5436,    -381.0906,   14914.3587,
```

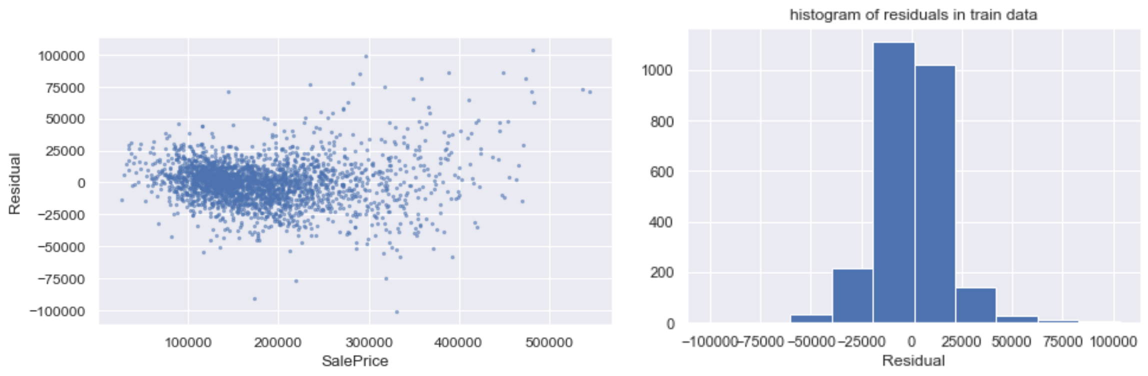
Figure 11: the first 16 coefficients of the ridge model

There aren't a lot of differences between these α, but $\alpha = 1$ gives the best combination of R^2 and log rmse. Then we fit the model with all train data, we get $R^2 = 0.927$, $log\_rmse = 0.115$ in train data, and $R^2 = 0.898$, $log\_rmse = 0.172$ in test data. This time, R^2 in test data is no longer negative. The coefficients are much smaller than OLS's coefficients(Figure 11). This proves that regularization is indeed useful. We also plotted the residual plots as we did for OLS.



(a) (b)

Figure 12: Residual plots of Ridge in train data



(a) (b)

Figure 13: Residual plots of Ridge in train data after removing outliers

There are still some outliers. In total, there are 13 residuals over $\pm 100,000$. I removed these points whose and then refitted the ridge regression. The residual plots look more centered around the y=0 axis, and the histogram of residuals looks more like Gaussian. Figure 14 plotted the coefficients with the largest absolute values. This plot can directly show which variables are most related to the target variable ‘SalePrice‘. For example ”Roof Mati_ClyTile”(Roof Material made of Clay or Tile) has the most negative coefficient, which indicates that a house whose roof is made of this kind of material will usually sell much less. The R^2 and log rmse improved as well after removing these outliers.

Summary of Ridge		
	R^2	log rmse
Train Data	0.927	0.115
Test Data	0.898	0.172
model re-fitted after removing outliers		
Train Data	0.949	0.105
Test Data	0.911	0.159

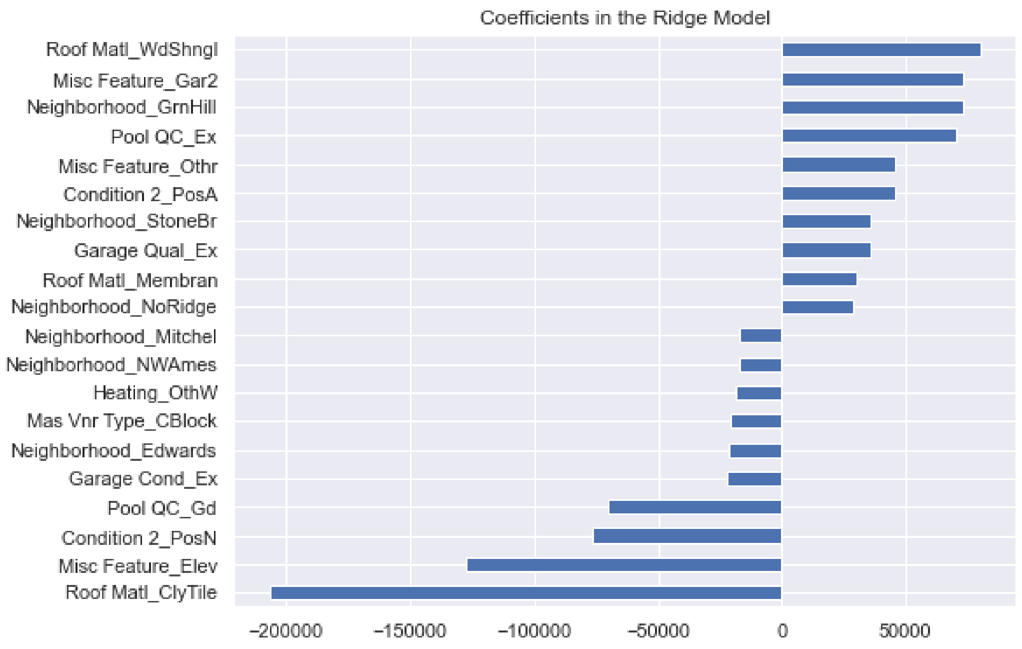


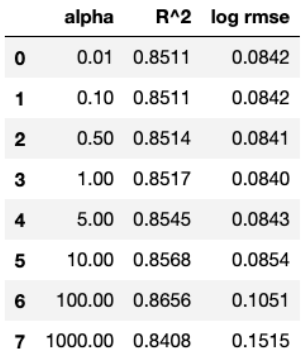
Figure 14: Coefficients in the Ridge Model

4.3 Lasso

The Lasso is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer non-zero coefficients, effectively reducing the number of features upon which the given solution is dependent.[7] Mathematically, it consists of a linear model with an added regularization term. The objective function to minimize is:

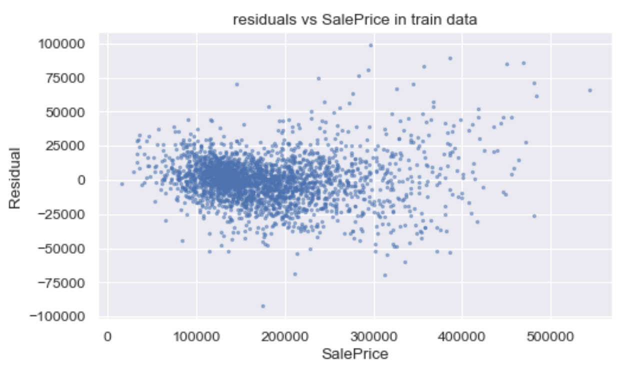
$$\min_{w} \frac{1}{2n_{samples}} ||Xw - y||_2^2 + \alpha ||w||_1$$

Since we have a large number of features(332 features), Lasso can effectively set many of the coefficients into zeros. This can greatly make the model simpler and make the inference easier. Same as Ridge, we use cross-validation to select alpha. When $alpha = 1$, the model gives 271 non-zero coefficients, when $alpha = 10$. the model gives 218 non-zero coefficients. We select $alpha = 10$ as it gives the best combination of R^2 and log rmse. The residual plots are drawn for both train data and test data. From the test data, we can see most of prediction errors are within \$50,000. There are some irregular data points whose predictions are off by \$100,000. This is unavoidable as in real life sometimes abnormal sales do happen. From the summary table, we can see that Lasso is slighter better than Ridge in this data. The variable importance plot (Figure 18) is also different from Ridge. Although ”Roof Mati_ClyTile” is still the most negative coefficient, many features that are selected are different from Ridge.

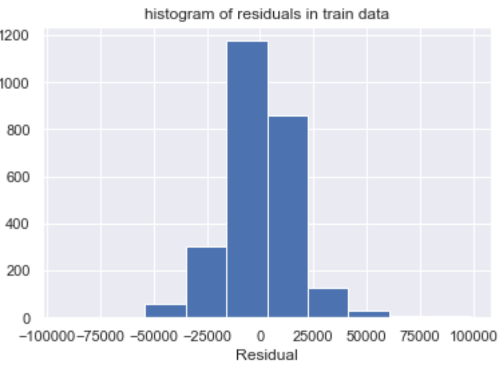


	alpha	R^2	log rmse
0	0.01	0.8511	0.0842
1	0.10	0.8511	0.0842
2	0.50	0.8514	0.0841
3	1.00	0.8517	0.0840
4	5.00	0.8545	0.0843
5	10.00	0.8568	0.0854
6	100.00	0.8656	0.1051
7	1000.00	0.8408	0.1515

Figure 15: cross-validation for selecting alpha

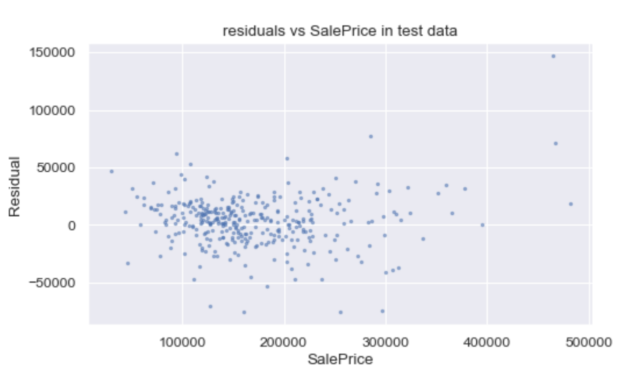


(a)

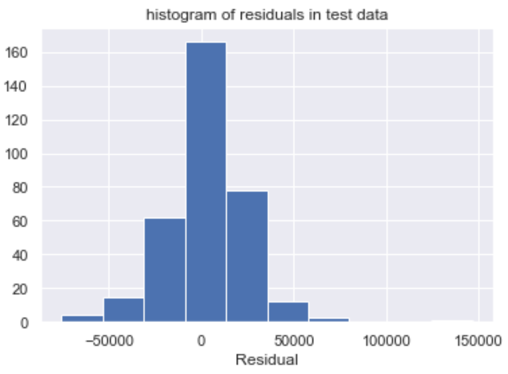


(b)

Figure 16: Residual plots of Lasso in train data



(a)



(b)

Figure 17: Residual plots of Lasso in test data

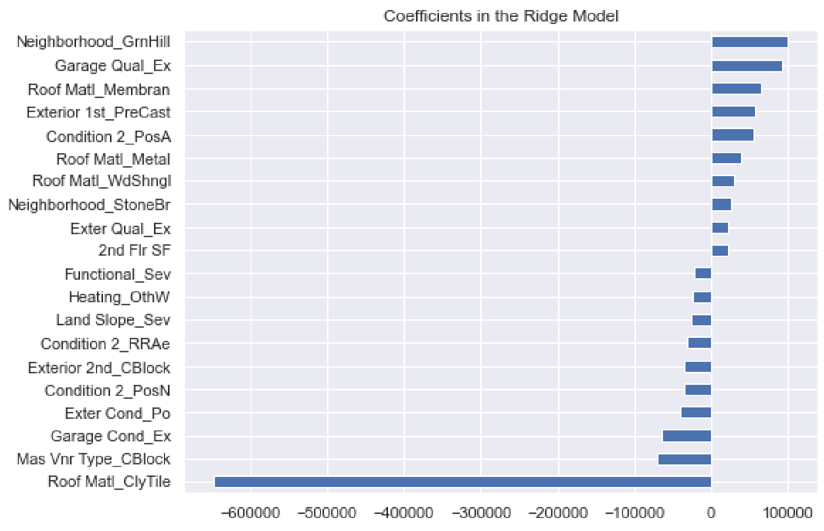


Figure 18: Coefficients in the Lasso Model

Summary of Lasso		
	R^2	log rmse
Train Data	0.950	0.103
Test Data	0.909	0.168

5 Non-Linear model

I used a non-linear model to compare with linear models. I fitted a random forest model(with 100 trees) using the same data. The detail of selecting parameters is ignored. The summary table follows:

Summary of Random Forest		
	R^2	log rmse
Train Data	0.986	0.054
Test Data	0.896	0.164

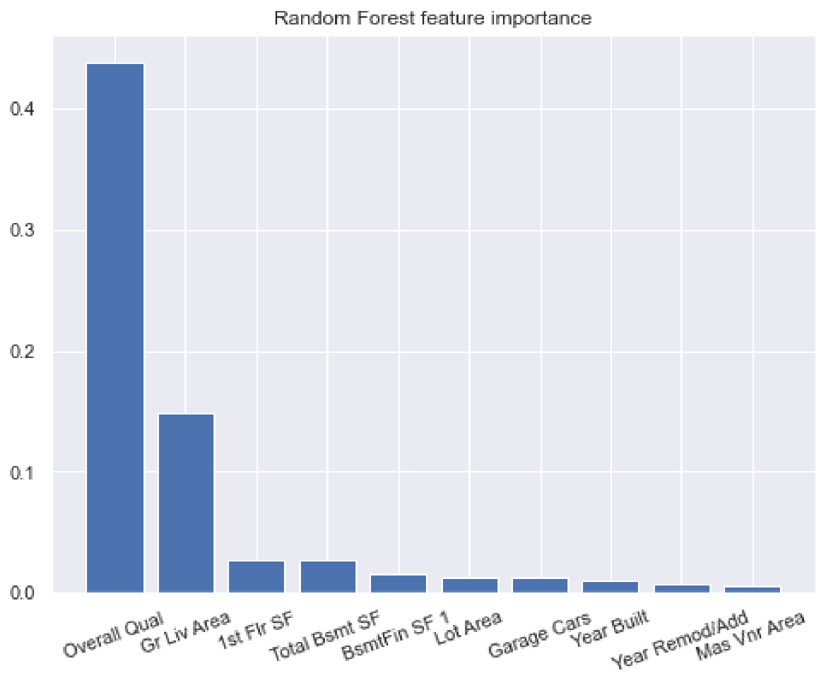


Figure 19: Random Forest feature importance

In terms of fitting train data, the random forest is better than linear models in this data set. The random forest has R^2 as high as 0.986 and log rmse as low as 0.054. However, in terms of generalizing to test data, random forest is not better than Ridge and Lasso. The top 10 important features[3] is shown in the bar plot(Figure 19).

6 Conclusion

Without relying on human feature engineering, both Lasso and Ridge generate very good performance. In this data set, the coefficients of OLS exploded, while Ridge and Lasso constraint the coefficients to be small. Ridge and Lasso are very useful when we have lots of features and when we don't have enough domain knowledge to do feature engineering. Generally speaking, when features are good, simple models can also perform as well as complex models. In many cases, we don't have very clean features. Ridge and Lasso have been proven very useful in this real-world data set and the advantage is that they are linear models. With linear models, we can interpret the coefficients very easily. Certainly, non-linear models would usually perform better than linear models in terms of prediction. They usually lost interpretability. This is a trade-off. Future work on this project is to try other non-linear models if we mainly care about prediction accuracy.

References

[1] Alvinctk. *Feature Selection - features with low variance*. URL: `https://community.dataquest.io/t/feature-selection-features-with-low-variance/2418`.

[2] Luc Anselin. *Boston Housing Data*. URL: `http://www.dm.unibo.it/~simoncin/boston_metadata.html`.

[3] L. Breiman and A. Cutler. *Random Forests*. URL: `http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`.

[4] William Chiu. *When-is-R-squared-negative*. URL: `https://www.quora.com/When-is-R-squared-negative`.

[5] Truman State University Dean De Cock. *Journal of Statistics Education Volume 19, Number 3(2011), Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project*. URL: `http://jse.amstat.org/v19n3/decock.pdf`.

[6] Peng Ding. *Linear Model and Extensions.*

[7] Sklearn. *Lasso Regression.* URL: `https://scikit-learn.org/stable/modules/linear_model.html#ridge-regression`.

[8] Sklearn. *Mean squared logarithmic error.* URL: `https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-log-error`.

[9] Sklearn. *Ridge Regression.* URL: `https://scikit-learn.org/stable/modules/linear_model.html#ridge-regression`.