CENTRAL UNIVERSITY OF FINANCE AND ECONOMICS
BEIJING, CHINA

House Prices: Advanced Regression Techniques Project Report

Yiqiao Wang

This project report is based on the competition "House Prices: Advanced Regression Techniques"[1] on Kaggle, which is the one of the most famous places to do data science and machine learning projects. Through the complete process of applying various mathematical and programming tools to solve practical problems, I hope to have a better understanding of data science, and further, improve the ability of solving business problems with a "data" perspective in the future.

1 PROBLEM DESCRIPTION

House, as one of the most significant assets in a person's life, has attracted a great amount of economists to conduct research based on it. Ask a home buyer to describe his dream house, he probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. There are much more latent factors influences price negotiations than the number of bedrooms or a white-picket fence.

This competition provides us with Ames Housing Dataset [2], which includes 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa. In this project, I will predict the price of each home based on these variables to explore important factors that influence house prices.

2 DATA PROCESSING

2.1 DATA DESCRIPTION

The Ames Housing Dataset was compiled by Dean De Cock for use in data science education. It's an incredible alternative for data scientists looking for a modernized and expanded version

of the often cited Boston Housing dataset.

In this dataset, there are 1460 training samples and 1459 test samples, which include 79 explanatory variables and training samples are also provided with true sale prices as labels. These explanatory variables consist of 36 numerical variables and 43 categorical variables, which cover every aspect of house information. However, if we take a careful look at these data, we can find there are missing values in some variables.

To effectively use training data and give reasonable predictions on test data, we need to first process data to make them clean and effective as inputs of various models. There are some basic things we can do:

- Analyze the target variable and conduct necessary processing
- Process missing data (remove relevant variables or fill missing values)
- Analyze explanatory variables and conduct necessary processing

Next, I will describe specific processing approaches according to the order of above steps.

2.2 Target Variable Analysis

The basic statistics of the target variable (i.e."SalePrice") are as follows:

Table 1: Statistics of SalePrice

mean	std	min	25%	50%	75%	max
180921.19	79442.50	34900.00	129975.00	163000.00	214000.00	755000.00

The range of the target variable is large, from 34900 to 755000. The 50% quantile (median) is greater than the mean, which indicates great values occupy the majority of all data. To observe the target variable more clearly, I draw the histogram and Q-Q (quantile-quantile) plot as Figure 1.

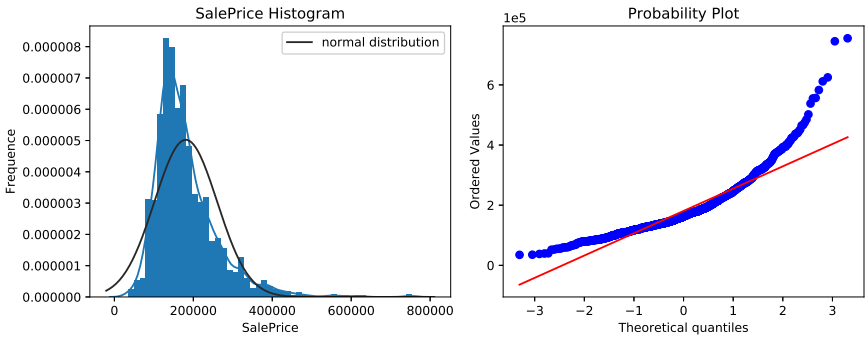


Figure 1: SalePrice histogram and Q-Q plot

It can be seen that the distribution of "SalePrice" deviates from the normal distribution and has an obvious positive skewness.

As regression models usually love normally distributed data, we need to apply a transformation on it. In case of positive skewness, a simple way to solve this problem is the log transformation. After log transformation, the histogram and Q-Q plot of "SalePrice" are as Figure 2.

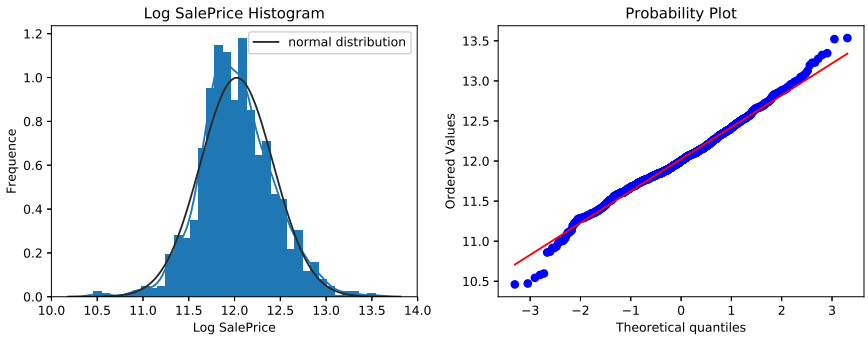


Figure 2: log-SalePrice histogram and Q-Q plot

It can be seen that the log transformation works well. The skew is corrected and the target variable appears more normally distributed.

2.3 Processing Missing Data

As mentioned before, there are some missing data which need to be processed to make subsequent regression models work correctly. This subsection will describe the approach to dealing with missing data in detail.

2.3.1 Overall condition of missing data

First, I take a look at the overall condition of missing data. There are 34 features (explanatory variables) having missing values in total. The top 20 of them are shown in Figure 3.

Looking at the first several features with most missing values: PoolQC (Pool quality), MiscFeature (Miscellaneous feature not covered in other categories), Alley (Type of alley access to property) and Fence (Fence quality). It is reasonable that they contain a mass of missing values because many houses probably don't have pools, alleys, fences and other miscellaneous features.

2.3.2 Imputing missing values

Then, I impute missing values of each feature according to the corresponding data description. Considering the characteristics of missing values, relevant features can be divided into following categories:

- For the feature where "NA" means "No X", fill in None. These features contain: "PoolQC", "MiscFeature", "GarageType", "Alley", "Fence", "FireplaceQu", "MSSubClass", "MasVnrType", "GarageFinish", "GarageQual", "GarageCond", "BsmtQual", "BsmtCond", "BsmtExposure", "BsmtFinType1", "BsmtFinType2".

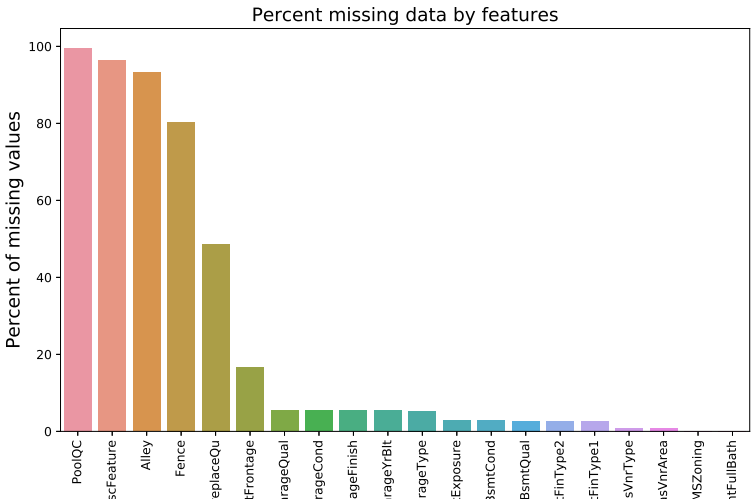


Figure 3: Top 10 variables with most missing data

- For the feature which has only few "NA" values, fill in the most frequent values of this feature. These features contain: "Electrical", "KitchenQual", "Exterior1st", "Exterior2nd", "SaleType", "MSZoning".
- For garage relevant features ("GarageYrBlt", "GarageArea" and "GarageCars"), fill missing data with 0, because it is most possible that there is no car. Similarly, I also impute missing values with 0 in basement relevant features ("BsmtFinSF1", "BsmtFinSF2", "BsmtUnfSF","TotalBsmtSF", "BsmtFullBath", "BsmtHalfBath") and "MasVnrArea".
- Other features:
  - "LotFrontage": Linear feet of street connected to property. Fill in missing values with the median "LotFrontage" of the neighborhood.
  - "Utilities": Type of utilities available. As all records are "AllPub", except for one "NoSeWa" and 2 "NA", and "NoSeWa" only appears in the training set, this feature won't help in predicting, so we can safely remove this feature.
  - "Functional": Home functionality. Since the data description says "Assume typical unless deductions are warranted", we should fill in missing values with "Typ".

After above processing, there is no remain missing values. Since the data are clean enough, what we need to consider next is how to process explanatory variables. Next two subsections will introduce different approaches to process categorical and numerical data separately.

2.4 Processing categorical data

For processing categorical data, there are two things that we should take care of: 1. There may exist some numerical variables that are actually categorical, which don't contain ordering information. If we view them as numerical variables wrongly, it will bring noises to the subsequent regression model. 2. Categorical data can't be directly used by the model. We need to encode them with different ways according to the characteristics of these variables.

2.4.1 NUMERICAL VARIABLES TRANSFORMATION

After looking into data descriptions in detail, I find only "MSSubClass" (Identifies the type of dwelling involved in the sale) is the numerical variable but actually the categorical variable. Besides, I'm open to how to process date information, such as "YrSold", "MoSold", "GarageYrBlt". In this project, I just view them as numerical variables. Maybe a time series analysis method could be applied to extract more information from these variables.

2.4.2 LABEL ENCODING AND ONE-HOT ENCODING

There are two ways to transform categorical variables to numerical variables. For variables with ordering information, such as the quality or condition scores of the houses, we'd better use label encoding, which keeps the ordering information. For other variables in which there are no "good" or "bad" properties, we can simply one-hot encode them.

In label encoding, I use 1-d vector to represent one categorical variable, of which the greater value means the better property. Such variables include "ExterQual", "ExterCond", "BsmtQual", "BsmtCond", "HeatingQC", "KitchenQual", "FireplaceQu", "GarageQual", "GarageCond" and "PoolQC".

In one-hot encoding, I use N-d vector to represent one variable, where N is the number of this variable's possible properties. If the rank of this variable's property is i, then $N_i = 1$, $N_j = 0$ $(j \neq i)$. Such variables include "BsmtExposure","BsmtFinType1", "BsmtFinType2", "GarageFinish", "Fence", "Functional", "LandSlope", "LotShape" and "PavedDrive".

After label encoding, there are 54 numerical features and 24 categorical features. After one-hot encoding for remaining categorical features, there are 185 features in total.

2.5 PROCESSING NUMERICAL DATA

The next step after processing categorical data is analyzing the numerical data. I'd like to do some preliminary observations about which features have most important influence on the sale price according to the correlation matrix. Besides, visualization is a good way to analyze data, verify observations from the correlation matrix and remove outliers if needed.

2.5.1 CORRELATION MATRIX

The correlation matrix of top 10 variables that have highest correlation coefficient with the sale price is shown in Figure 4.

It can be seen that the rates of the house quality from different aspects, such as "OverallQual", "ExterQual", "KitchenQual", "BsmtQual" have a significant influence on house prices. The sizes of the above ground living area, the garage area and the basement area also play an important role in determining house prices. What's more, people seem to like to pay additional attention on the garage condition, such as the interior finish condition of the garage.

2.5.2 VISUALIZATION

I visualize the first two most important factors, "OverallQual" and "GrLivArea", to observe their relationships with the sale price clearly. "GrLiveArea" is a continuous variable, so I show the scatter plot for it (Figure 5(a)). While "OverallQual" is a discrete variable, I show the box plot for it (Figure 5(b)).

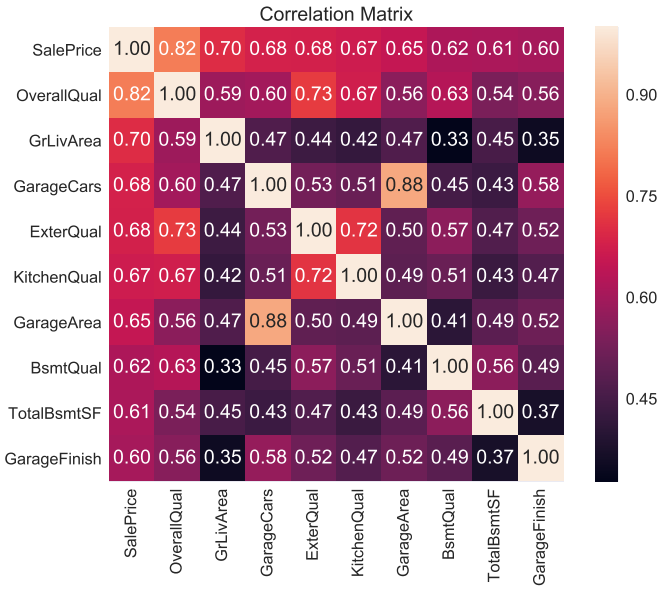
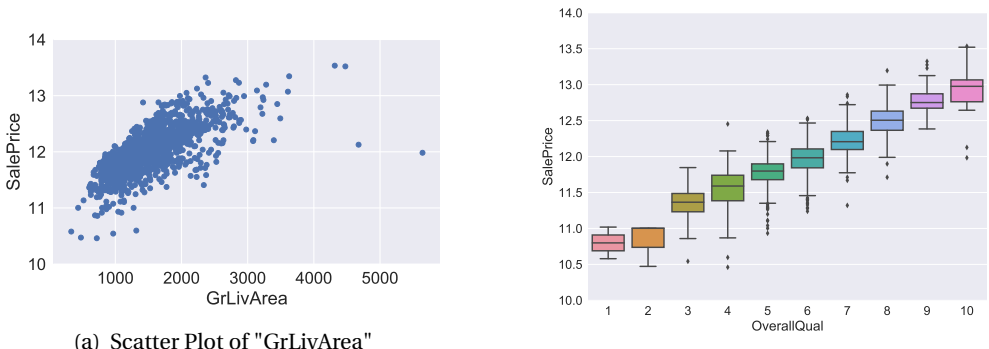


Figure 4: Correlation Matrix of Explanation Variables and Sale Price (Top 10)

We can see that the "GrLivArea" and "OverallQual" both show a strong positive correlation with the sale price, which conforms previous results shown in the correlation matrix.

2.5.3 Outliers

Another thing we need to pay special attention on is the outliers. In statistics, an outlier is an observation point that is distant from other observations. Outliers may cause serious problems in training subsequent regression models. However, if we delete outliers with a radical strategy, it may also influence the robustness of our model, since it is possible that there are also outliers in the test set. In this dataset, there seems to be 2 extreme outliers on the bottom right of Figure 5(a): large houses that are sold really cheap. Here, I follow the suggestion of the author



(a) Scatter Plot of "GrLivArea"

(b) Box Plot of "OverallQual"

Figure 5: Examples of scatter plot for continuous variable and box plot for discrete variable

of this dataset, to remove "any houses with more than 4000 square feet" from the training set.

2.5.4 SKEWED FEATURES

Besides the target variable, there are also skewed numerical features. I also apply log transformation on highly skewed features (|skewness| > 1.0) as described in 2.2. Take the "GrLivArea" as example (Figure 6), this trick shows great performance again, as Figure 7 shows.

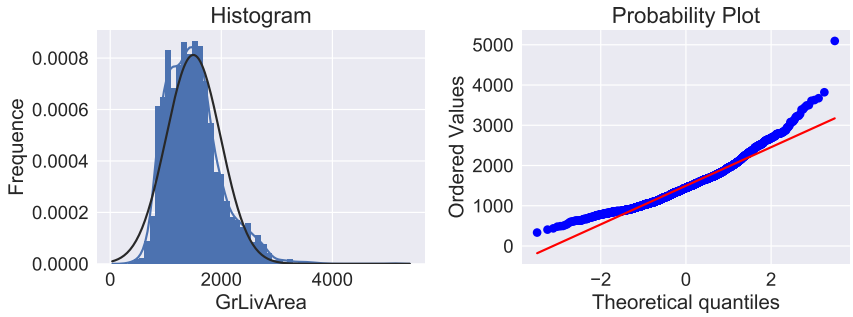


Figure 6: "GrLivArea" Histogram and Q-Q Plot

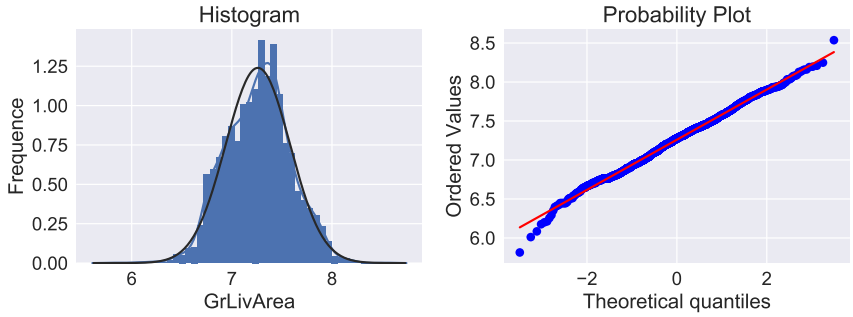


Figure 7: log-"GrLivArea" Histogram and Q-Q Plot

3 METHOD

In this project, I test several regression methods, including linear regression (standard Linear Regression, Ridge, Lasso) and ensemble learning regression (Random Forest, Gradient Boosting), and also test two model ensembling techniques (averaging and stacking) to predict house prices. In this section, I will describe these methods in brief.

3.1 LINEAR REGRESSION MODEL

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and explanatory variables. Formally, given a data set $\{y_i, x_{i1}, \cdots, x_{ip}\}_{i=1}^{N}$, where y denotes the target variable, $\mathbf{x}$ denotes p explanatory variables, the linear regression model

assumes the relationship between y and $\mathbf{x}$ is linear and there is a disturbance term ϵ added to this relationship as noise. Thus, the model takes the form as Eq. (1).

$$y = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i = \mathbf{x}_i^{\mathbf{T}} \beta + \epsilon_i, \quad i = 1, 2, \ldots, N \tag{1}$$

These N equations are often stacked together and written in matrix notation as $\mathbf{y} = \mathbf{X}\beta + \epsilon$.

STANDARD LINEAR REGRESSION The linear regression model is often fitted using the *least square error* approach, or from the perspective of statistics, to minimize the sum of squared residuals. The objective of it can be written as:

$$\min_{\beta} \{\frac{1}{N} \sum_{i=1}^{N} (y_i - \mathbf{x}_i^{\mathbf{T}} \beta)^2\} \tag{2}$$

Since in many cases, the optimization problem we aim to solve is an ill-posed problem, i.e. the solution is not unique, there is also a regularization term in the objective of linear regression model.

RIDGE REGRESSION Tikhonov regularization is the most commonly used method of regularization, which is also known as ridge regression in statistics or weight decay in machine learning. It has the form like $\| \Gamma \mathbf{w} \|_2^2$. The Tikhonov matrix Γ is often chosen as a multiple of the identity matrix ($\Gamma = \alpha \mathbf{I}$), and it is also known as L_2 regularization. The final objective of ridge regression is:

$$\min_{\beta} \{\frac{1}{N} \sum_{i=1}^{N} (y_i - \mathbf{x}_i^{\mathbf{T}} \beta)^2 + \alpha \| \beta \|_2^2\} \tag{3}$$

where α is a scalar controlling the weight of regularization, $\| \cdot \|$ denotes the L_2-norm.

LASSO REGRESSION Lasso (Least Absolute Shrinkage and Selection Operator) is another common regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of regression models. It can select only a subset of the provided covariates rather than all for use in the final model. Compared to ridge regression which improves the prediction accuracy by shrinking large regression coefficients to reduce overfitting, lasso regression also performs covariate selection to make the model more interpretable. It is achieved by adding a L_1 -norm regularization term to the objective, forcing certain regression coefficients to be set to zero, which produces a simpler model. The final objective of lasso regression is:

$$\min_{\beta} \{\frac{1}{N} \sum_{i=1}^{N} (y_i - \mathbf{x}_i^{\mathbf{T}} \beta)^2 + \alpha \| \beta \|_1\} \tag{4}$$

3.2 ENSEMBLE LEARNING REGRESSION MODEL

RANDOM FOREST Another important branch of regression methods is through ensemble learning. Random forest is one typical ensemble learning method, which constructs a multitude of decision trees (weak prediction models) and output the mean prediction of the individual trees.

GRADIENT BOOSTING Gradient boosting shares similar ideas with random forest, but it builds the model in a stage-wise fashion and allows optimization of an arbitrary differentiable loss function. The main idea of gradient boosting is that boosting algorithms can be viewed as iterative functional gradient descent algorithms, which iteratively chooses a weak hypothesis that points in the negative gradient direction. For the regression task, our goal is to learn a model $\hat{F}(x)$ to minimize the expected value of loss function $L(y,F(x)) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$, where y_i is the real value and $\hat{y}_i$ is the value predicted by our model. The gradient boosting method seeks $\hat{F}(x)$ in the form of a weighted sum of functions $h_i x$ from some class $\mathscr{H}$ called base (weak) learners:

$$\hat{F}(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + \text{const} \tag{5}$$

The gradient boosting algorithm starts the estimation from a constant function $F_0(x)$ and incrementally expands it in a greedy fashion:

$$\begin{aligned} F_0(x) &= \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma) \\ F_m(x) &= F_{m-1}(x) + \arg\min_{h_m \in \mathscr{H}} [\sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + h_m(x_i))] \end{aligned} \tag{6}$$

In practice, we can use the gradient descent algorithm to solve the minimization problem in Eq. (6). The update equations are like Eq. (7).

$$\begin{aligned} F_m(x) &= F_{m-1}(x) - \gamma_m \sum_{i=1}^{N} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i)) \\ \gamma_m &= \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) - \gamma \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))) \end{aligned} \tag{7}$$

In this project, I tested several different implementations of gradient boosting. They include the implementation from scikit-learn [3], DMLC's XGBoost [4], and Microsoft's LightGBM [5].

3.3 MODEL ENSEMBLING

Model ensembling is a powerful technique to reduce the generalization error and increase accuracy on a variety of machine learning tasks [6]. In this project, I tested two model ensembling methods: **averaging** and **stacking**.

AVERAGING. The simplest way to ensemble models is to average the prediction scores of individual models. Average predictions often reduces overfitting, which improve the model's generalization ability to unseen data.

STACKING. Stacked generalization was introduced by Wolpert in 1992 [7]. The basic idea behind stacked generalization is to train a pool of basic classifiers and then use another classifier to combine their predictions to achieve the goal of reducing generalization errors. The overall procedure of it can be summarized as follows:

1. Split the training set to n folds

2. Set one of them as test (holdout) set, the others as training set.
3. Train several base models on the training set.
4. Test these base models on the holdout set.
5. Return to step 2 until each fold has been used as the holdout set once.
6. Use predictions of base models as inputs and the target variable as outputs to train a meta-model. The prediction of the meta-model is used as the final result.

4 EXPERIMENTS

In the experiment section, I first test the influence of two important factors about data processing, which include outliers elimination and data scaling. Then, according to the methods described above, I will analyze the results of linear regression model and ensemble learning regression model. Different model ensembling approaches are explored and analyzed at last. The code is published [1].

4.1 INFLUENCE OF DATA PROCESSING

Outliers and data pre-processing may have a non-negligible influence for training the regression model. I trained standard linear regression models with different outliers and scaling processing methods. The Root-Mean-Squared-Error (RMSE) on test set are reported in Table 2 and Table 3. The smaller RMSE indicate the better model. We can see that whether to eliminate outliers has a considerable influence on the model's performance. After removing only 4 outliers, the RMSE can be improved from 0.1455 to 0.1218.

As for the data scaling, I test three methods: no scaling, standard scaling, which remove the mean and scales to unit variance), and robust scaling, which removes the median and scales the data according to the quantile range (the range between the 25th quantile and the 75th quantile). It can be seen that robust scaling is a better data pre-processing approach for this problem. It can sligtly improve the RMSE. On the contrary, improper scaling such as standard scaling may bring down the regression model's performance.

Table 2: Influence of outliers elimination

Model	RMSE
Linear with outliers	0.1455
Linear without outliers	**0.1218**

4.2 REGRESSION RESULTS

The regression results of linear models and ensemble learning models are shown in Table 4. We can see that Lasso regression performs best in linear regression models and Graident Boosting (from sklearn package) performs best in ensemble learning models.

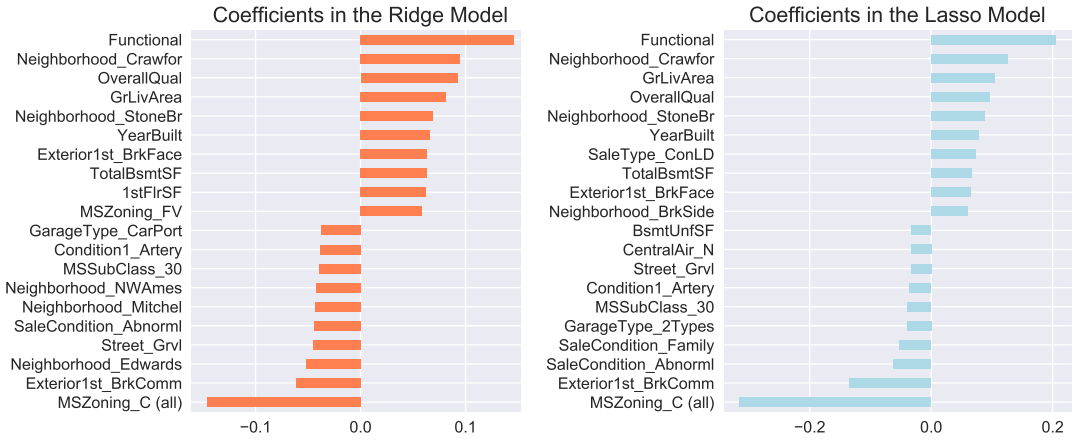
[1] `https://github.com/yqwang717/kaggle_house_price`

Table 3: Influence of data scaling

Model	RMSE	Model	RMSE
Ridge	0.1109	Lasso	**0.1099**
Ridge(standard)	0.1187	Lasso(standard)	0.1161
Ridge(robust)	**0.1108**	Lasso(robust)	**0.1099**

As the regularization of model's weights, Ridge and Lasso both show robustness to outliers, which lead to a better RMSE score compared to standard linear regression model. Lasso shows stronger feature selection ability, which eliminates 120 features, while Ridge only eliminates 5 features. The better RMSE score of Lasso indicates such feature selection is effective. Top 10 and last 10 coefficients of Ridge and Lasso model are show in Fig. 8. They are consistent with the aforementioned correlation matrix analysis. Such as "OverallQual" and "GrLivArea" numerical features have large regression coefficients.

Among ensembling models, standard Graident Boosting has the best performance. The overall performance of ensemble learning regression models is worse than linear regression models. It is possible because there are much more hyper-parameters could be adjusted and current parameters may not be optimal in this task.



(a) Ridge coefficients

(b) Lasso coefficients

Figure 8: Top 10 and last 10 coefficients in Ridge regression model and Lasso regression model

4.3 Model Ensembling Results

I test two model ensembling methods, averaging and stacking, and their results are show in Table. 5. It can be seen that these modeling ensembling techniques indeed improve the model's performance further. The combination of Lasso and Gradient Boosting, two best methods in linear regression models and ensemble learning regression models separately, shows even better performance when only using a simple averaging technique. For stacking, it

Table 4: Regression results

Method	Model	RMSE
Linear Regression	Linear	0.1218
	Ridge	0.1108
	Lasso	**0.1099**
Ensemble Learning	Random Forest	0.1371
	Gradient Boosting	**0.1135**
	XGBoost	0.1152
	LightGBM	0.1150

seems that more base models will bring better results and the stacking technique is slightly better than averaging.

It is worth noticing another interesting phenomenon. If we use the result of Lasso as a feature selection criterion and apply it as a step of data processing. Regression models could obtain better prediction results, which can be seen in Table 6. After feature selection, Ridge performs best in all linear models and model ensembling shows further improvement.

Table 5: Model ensembling results

Stacking Method	Model	RMSE
Averaging	(Lasso, GBoost)	**0.1073**
	(Lasso, XGBoost)	0.1089
	(Ridge, Lasso, GBoost)	0.1074
Stacking	(Ridge, Lasso) + Linear	0.1097
	(Lasso, GBoost) + Linear	0.1069
	(Lasso, GBoost, XGBoost) + Linear	0.1068
	(Ridge, Lasso, GBoost, XGBoost) + Linear	**0.1067**

Table 6: Regression results after feature selection with Lasso

Method	Model	RMSE
Linear Regression	Ridge	0.1081
Averaging	(Lasso, GBoost)	0.1059
Stacking	(Ridge, Lasso, GBoost, XGBoost) + Linear	**0.1056**

5 Conclusion

Through this project, I exercise my programming skills and have gained a valuable experience of applying various mathematical and programming tools to solve a practical problem. My final result achieves a top 20% score in the leaderboard of this Kaggle competition. There is still improvement space for this result, such as applying more complex feature selection and feature engineering techniques. More date science knowledge and experience need to be accumulated through practice.

Reference

[1] Kaggle Competition. House prices: Advanced regression techniques. `https://www.kaggle.com/c/house-prices-advanced-regression-techniques`.

[2] Dean De Cock. Ames, iowa: Alternative to the boston housing data as an end of semester regression project. *Journal of Statistics Education*, Volume 19, 2013.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[5] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

[6] Armando Segnini Hendrik Jacob van Veen, Le Nguyen The Dat. Kaggle ensembling guide. `https://mlwave.com/kaggle-ensembling-guide/`, 2015. [accessed 2018 Feb 6].

[7] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.