Instructor: Shuo Zhang, Ph.D., Columbia University

Contact: shuozhang1985@gmail.com

Kaggle Competition: House Pricing

1. Introduction

Upon commencement, we had some experience of R, Python, and machine learning basics. During the process, we gained knowledge of various different machine learning algorithms, from supervised or unsupervised to reinforcement learning. Now it's time to apply them to solving real problems and testing what we have learned. Fortunately, we discovered an ongoing and fascinating competition on kaggle.com, a practical competition whose goal is to predict house prices in Ames, Iowa using data covering the different features of the houses collected in 2010. There are 79 explanatory features describing every aspect of residential homes in Ames, Iowa. We considered this competition approachable because detailed explanations of the features had been provided in full to the participants. It was an invaluable opportunity to both practice and hone our skills and knowledge of advanced machine learning, such as creative feature engineering and advanced regression techniques like random forest and gradient boosting.

In four weeks, we have performed EDA, feature engineering, ensembling, stacking, and have constantly fine tuned our parameters. Our own codes are score-oriented for the sake of further improving the prediction score. Currently, and after several submissions of our machine learning project, the best model place in the top XX% out of more than xxx teams on the Kaggle leader board.

1. Workflow

data preprocessing, machine learning models(lasso,kernel ridge, elsaticnet,gradient boosting), ensemble and stacking, prediciton and submission

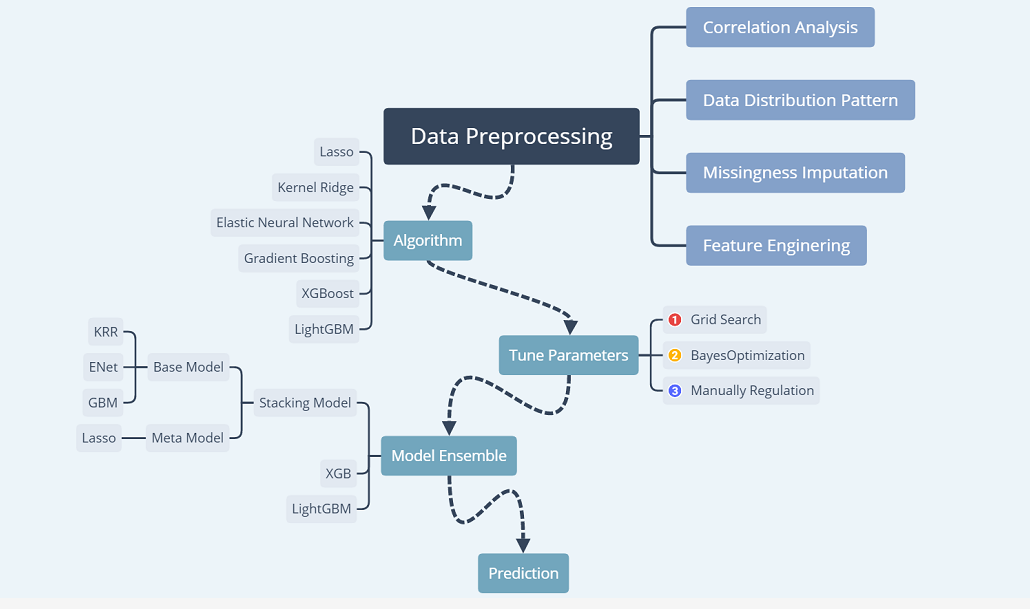


Figure1. Workflow of the program

3. Pre-Processing

3.1 Data Distribution Pattern

Our aim is to predict the house price (abbreviated as HP), thus it’s important to determine the distribution of the HP of the sample.

As shown in the first map below, the shaded blue area represents the HP density histogram plot, and the blue and black lines represent the HP density line and fitted HP density line respectively. It’s obvious that the distribution of the HP is not a normal one, which may cause considerable errors in our predictions. We can also infer this from the second map, which compares the theoretical quantile value with the real quantile value. We can see that most of the points are beyond the boundary, meaning that the distribution of HP is non-normal.

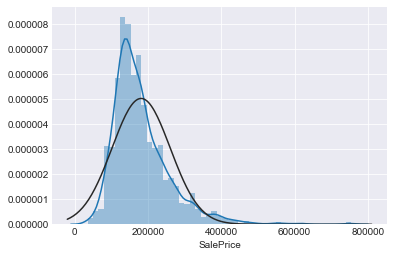


Figure2. Original Distribution of the Sale Price

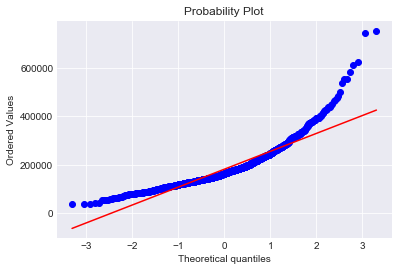


Figure3. Original Probability Plot

This non-normal distribution shows a trailing trend, namely that most of the samples are clustering around the lesser values. Therefore, we can employ the cox-box method to handle this problem.

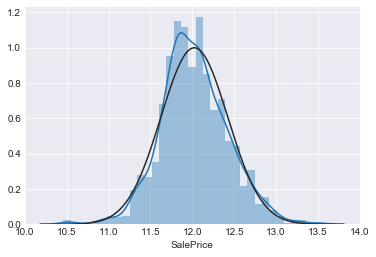
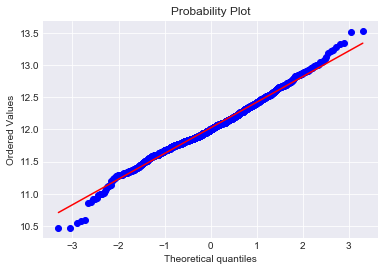


Figure4. Distribution of the Sale Price after Normalization



Original Figure 5. Probability Plot after Normalization

We can observe that the samples are close to the normal distribution after using the cox-box method.

3.2 Missing Values

The next important process is dealing with missing values. In the code, it is easy to see that 30 columns contained missing values. We dealt with the missing values as the following table shows.

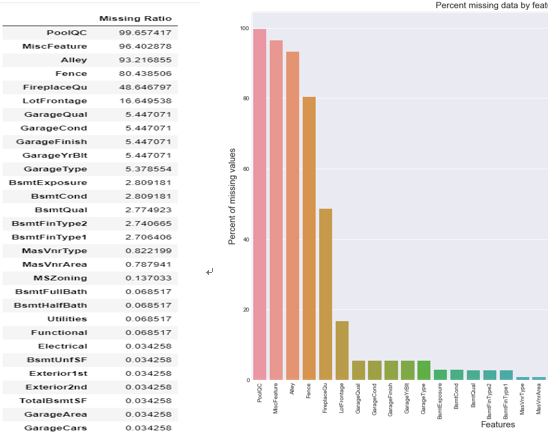


Figure6. Missing Values

Table1. Imputation of the Missing Values

|  |  |
| --- | --- |
| Method | Feature |
| None | PoolQC, Alley, Fence, FireplaceQu, MiscFeature, GarageType, GarageFinish, GarageQual, GarageCond, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, MasVnrType, Functional |
| 0 | GarageYrBlt, GarageArea, GarageCars, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, BsmtFullBath, BsmtHalfBath, MasVnrArea |
| Most Frequent Class | MSZoning, Electrical, KitchenQual, Exterior1st |
| Median Replacement | LotFrontage |
| Drop | Utilities |

For the “None” group, which includes features such as ‘PoolQC’, ‘MiscFeature’, ‘Alley’, ‘Fence’, ‘FireplaceQu’, etc, missing values perhaps occur because most of the houses don’t have these facilities, which may cause ‘missing’. It is therefore safe to fill the NA with “None”.

Secondly, for those houses with no basement, garage, or wall cladding, this may also cause missing values to occur, as can be seen in the features 'GarageYrBlt, 'BsmtFinSF1', 'BsmtFinSF2', etc. So we fix them with the value 0.

By observing the feature ‘MSZoning’, 'Electrical', 'KitchenQual', 'Exterior1st', 'SaleType', 'Exterior2nd', we can observe that all of them frequently show a value which is not determined by the sample’s other features. Therefore it’s a good idea to fill these missing values with the most frequent value within the sample.

The ‘LotFrontage’ usually shows a definite value in a definite area, thus we can simply fill in this feature’s missing value with a median value of this area which is grouped by ‘Neighborhood’.

For the feature ‘Utilities’, all records are "AllPub", except for one "NoSeWa" and 2 NA . Since the house with 'NoSewa' is in the training set, this feature won't benefit our predictive model. We can then safely remove it for that very reason.

We have officially filled in all the missing values.

3.3 Correlation Analysis

In order to analyze the correlation between each feature and the housing price, we plotted the correlation heat map of these features. We discovered that some numerical features, which are correlate highly with the price. In particular, there are strong positive correlations between “TotaLBsmtSF” and “1stFlrSF”, as well as “GarageAreas” and “GarageCars”. Such information is useful in determining the correlation of features and offering evidences beneficial to performing imputations. We also know that multi-collinearity may make it more difficult to make inferences about the relationships between our independent and dependent variables.

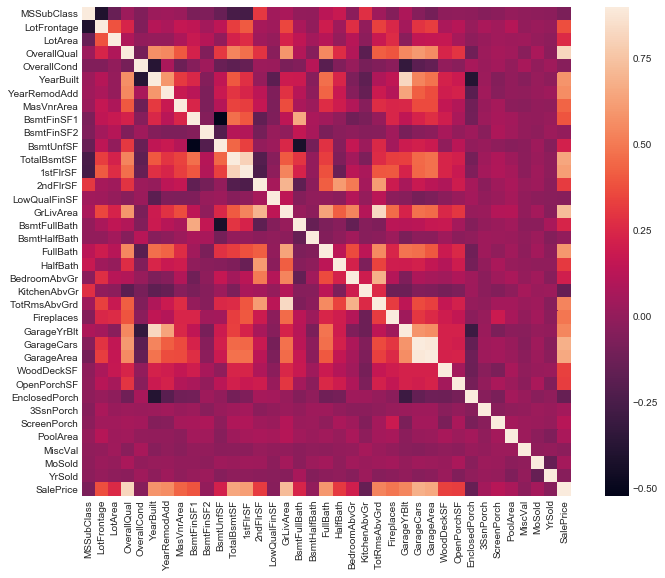


Figure7. Correlation Before Imputation

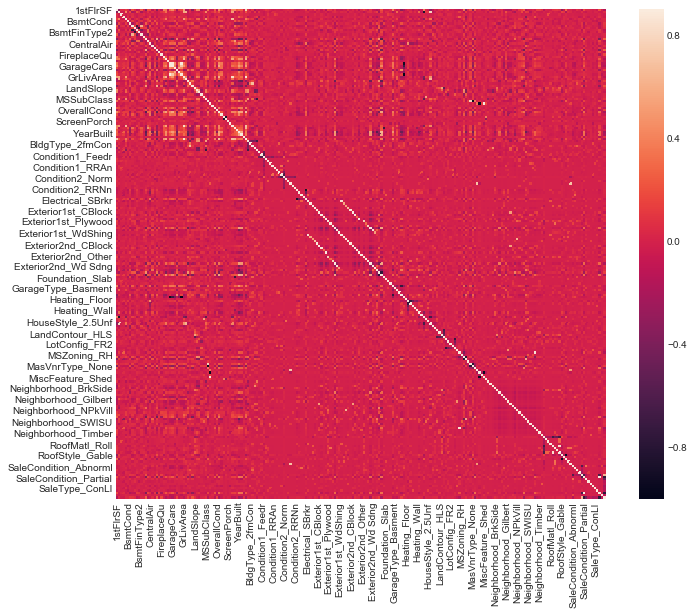


Figure8. Correlation After Imputation

3.4 Feature Engineering

3.4.1 TRANSFORMING THE DATA TYPE

In the algorithm to be applied next, the data type should be legal, which means that we should transform some of this data.

In one method, we transferred some numeric features to categorical types including 'MSSubClass', 'OverallCond', ‘'YrSold', 'MoSold'.

Another way, which allows for categorical features with three or more classes, is to employ the method of ‘LabelEncoder’ to transform them, which includes 'FireplaceQu', 'BsmtQual', 'BsmtCond', 'GarageQual', 'GarageCond', 'ExterQual', 'ExterCond','HeatingQC', 'PoolQC', 'KitchenQual', 'BsmtFinType1', 'BsmtFinType2', 'Functional', 'Fence', 'BsmtExposure', 'GarageFinish', 'LandSlope','LotShape', 'PavedDrive', 'Street', 'Alley', 'CentralAir', 'MSSubClass', 'OverallCond', 'YrSold', 'MoSold'. At last we implement ‘One-Hot-Encoding’ in order to make dummy variables out of all the categorical data which will have better effects on the algorithms.

3.4.2 ADDING A NEW FEATURE

We know that the mean house price for different parts of the house is the same. In this way, we can combine the features 'TotalBsmtSF', '1stFlrSF', and '2ndFlrSF' into a new feature, namely 'TotalSF'.

3.4.3 FEATURE SKEWNESS

In most of the algorithms, data with Gauss distribution can minimize error. Thus, we need to discover the skewing feature and then correct it by calculating the relative asymmetry of the data.

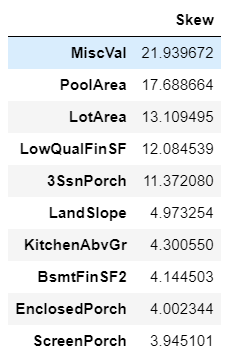
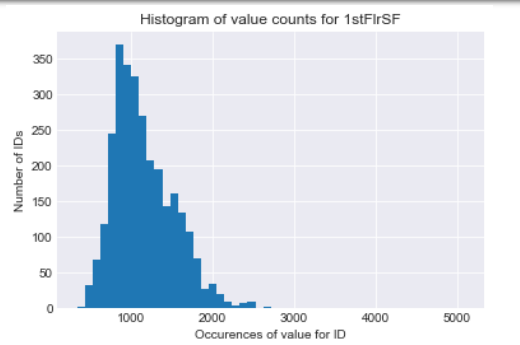


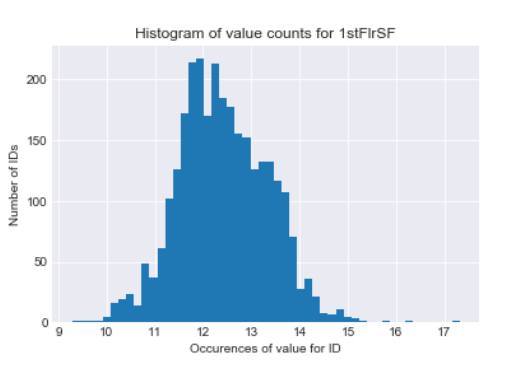
Figure9. Skewness of the Feature

We select the top 10 skewing features, all of which display high rates of skewness. For all of the numerical data within the data set, we set 0.75 as the threshold of the skewness. In this condition, we come to find that we have 59 features in total to be corrected. By histograming the density map of the features, we can employ box-cox as our method to correct the skewing (LAM = 0.15).



Before

Figure10. Skewness of 1stFirSF Before Imputation



After

Figure11. Skewness of 1stFirSF after Imputation

1. Models

4.1 Elastic Net

In [statistics](https://en.wikipedia.org/wiki/Statistics), and in particular the fitting of [linear](https://en.wikipedia.org/wiki/Linear_regression) or [logistic regression](https://en.wikipedia.org/wiki/Logistic_regression) models, the elastic net is a [regularized](https://en.wikipedia.org/wiki/Regularization_(mathematics)) regression method that [linearly combines](https://en.wikipedia.org/wiki/Linear_combination) the [lasso](https://en.wikipedia.org/wiki/Lasso_(statistics)) and [ridge](https://en.wikipedia.org/wiki/Tikhonov_regularization) methods.

There are two parameters we need to tune for: alpha, I1\_ratio.

Alpha: the constant that multiplies the penalty terms

I1\_ratio: the ElasticNet mixing parameter, with 0 <= l1\_ratio <= 1. For l1\_ratio = 0, the penalty is an L2 penalty. For l1\_ratio = 1 it is an L1 penalty. For 0 < l1\_ratio < 1, the penalty is a combination of L1 and L2.

Based on the process, we decided: Alpha[0.001,0.1], I1\_ratio[1,2]

We then added the new parameters “random\_state”, the seed of the pseudo random number generator that selects a random feature to update.

We used the MSE in order to judge the score and to discover the most ideal parameters.

table: tuning parameter, range, best value.

train score

4.3GBM

The next step is to apply gradient boosting. There are 6 tuning parameters: n\_estimators, learning\_rate, max\_depth, max\_features, min\_samples\_leaf, min\_samples\_split.

Table3. GBM

|  |  |
| --- | --- |
| parameter | note |
| n\_estimators | The number of sequential trees to be modeled. |
| learning\_rate | This determines the impact of each tree on the final outcome. GBM works by starting with an initial estimate which is updated using the output of each tree. The learning parameter controls the magnitude of this change in the estimates. |
| max\_depth | The maximum depth of a tree. |
| max\_features | The number of features to consider while searching for the best split. |
| min\_samples\_leaf | Defines the minimum samples (or observations) required by a terminal node or leaf |
| min\_samples\_split | Defines the minimum number of samples (or observations) which are required by a node to be considered for splitting. |

There are two types of parameter that we need to tune here, tree based and boosting parameters. We knew that there were no optimum values for learning rates as low values always work better, given that we trained on a sufficient number of trees. While GBM is robust enough to prevent over-fitting within an increased number of trees, a high number for a particular learning rate can still lead to over-fitting. As we reduced the learning rate and increased trees, the computation took a long time to run and produce results.

In such a case, we adopted the following approach. Initially, we used 0.05, however it did not produce a favorable result. Thus, our second attempt saw us choosing the default value of 0.1. Meanwhile, we lowered the learning rate and increased the estimators proportionally in order to produce more robust models. Additionally, it is worth mentioning that you should pay careful attention to the order of tuning variables. For example, the parameters that exert greater impacts on the result should be tuned firstly, such as max\_depth and min\_samples\_split, and then you could tune min\_samples\_leaf and max\_features.

4.4 KernelRidge

Let’s have a look at our liner machine learning model, Kernel Ridge Regression (KRR). KRR combines ridge regression (linear least squares with l2-norm regularization) with the kernel trick. It thus learns and adapts a linear function in the space induced by the respective kernel and the data. For non-linear kernels, this corresponds to a non-linear function within the original space. In our case, there were four parameters, including alpha, kernel, degree and coef0. The table below is their definitions.

Table 4. KernelRidge

|  |  |
| --- | --- |
| parameter | note |
| alpha | Small positive values of alpha improve the conditioning of the problem and reduce the variance of the estimates. Alpha corresponds to (2\*C)^-1 in other linear models such as LogisticRegression or LinearSVC. If an array is passed, penalties are assumed to be specific to the targets. Hence they must correspond in number. |
| kernel | Kernel mapping is used internally. A callable should accept two arguments and the keyword arguments passed to this object as kernel\_params, and should return a floating point number. |
| degree | Degree of the polynomial kernel. Ignored by other kernels. |
| coef0 | Zero coefficient for polynomial and sigmoid kernels. Ignored by other kernels. |

4.6 Lasso

Lasso is a powerful technique generally used for creating parsimonious models in the presence of a large number of features. The loss function is based on the ‘Least squares’. Lasso adds the L1 penalty in order to penalize a large number of coefficients. What’s more, it can also minimize the coefficient to 0 when certain features fail to work within the model.

We used the grid search method to determine the parameter’s alpha value.

Lasso is a linear method, and therefore it can be used to enhance the generalization of the model in the sections of the samples where the price is highly related to the area of the house. But the actual model is so much more complicated than that, and thus we chose to use Lasso as the meta model for the stacking model, and the implementation has been a success, ultimately producing very accurate predictions.

4.7 Summary

Table6. Summary of all Models

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Description | Pros | Cons |
| Lasso | * Lasso regression is a powerful techniques generally used for creating parsimonious models in the presence of a ‘large’ number of features | * Adds a regularization term which restrains the efficiency brought about by the increased data size. * The regularization term can minimize to zero in order to totally move the non-relative features. | * Sensitive to the outliers, which might cause over-fitting. |
| Elastic Net | * In statistics, and in the fitting of linear or logistic regression models in particular, the elastic net is a regularized regression method that linearly combines the L1 and L2 penalties of the lasso and ridge methods | * Enforces sparsity * No limitation on the number of selected variable * Encourages a grouping effect in the presence of highly correlated predictors | * Native elastic net suffers from double shrinkage |
| KRR | * Kernel ridge regression (KRR) combines ridge regression (linear least squares with l2-norm regularization) with the kernel trick. It thus learns a linear function within the space induced by the respective kernel and the data. For non-linear kernels, this corresponds to a non-linear function within the original space. | * Strong theoretical guarantees. * Generalization to outputs in the single matrix inversion * Uses kernels | * Solution not sparse. * Training time for large matrices is a low-rank approximations of kernel matrix, |
| GBM | * Produces a prediction model in the form of an ensemble of decision trees * Builds the model in a stage-wise fashion * Generalizes results by allowing for the optimization of an arbitrary differentiable loss function | * High-performing * Uses all features * Lower possibility of over-fitting with more trees | * Susceptible to outliers. * Lack of interpretability and higher complexity. * Harder to tune the parameters than other models. * Slow training speed. |
| LightGBM | * An advanced implementation of gradient boosting. * Uses a more regularized formalization to control over-fitting which gives a better performance within the model. | * Faster training speed and higher efficiency * Lower memory usage * Better accuracy * Parallel and GPU learning supported * Capable of handling large-scale data | * Easily over-fitted |
| XGBOOST | * An advanced implementation of gradient boosting. * Uses a more regularized formalization to control over-fitting which produces a better performance within the model. | * Uses regularization to reduce over-fitting. * Support parallel processing. * Allows for splits up to the max\_depth specified and then starts pruning the tree backwards and removes splits beyond which there is no positive gain. * Built-in cross validation. | * Susceptible to outliers. * Lack of interpretability and higher complexity. * Harder to tune the parameters than other models. * Slow training speed. |

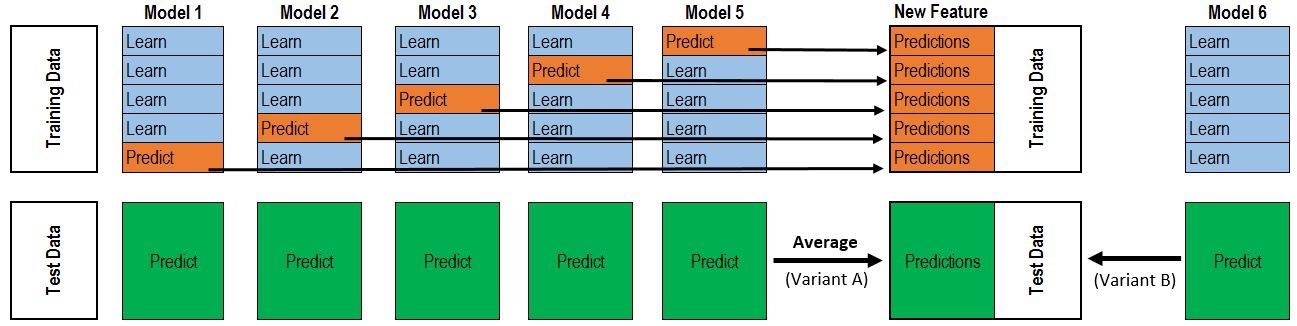
1. Averaging Based Ensemble Methods and Stacked Ensemble Methods

The implementation of only one model cannot independently produce the best score, so we decided to ensemble all the models in order to figure out the best way to predict the outcomes.

5.1Simple averaging model

Within the simple averaging method, for every sample of test datasets, the average predictions are calculated. This method often reduces over-fitting and creates a smoother regression model. The scoring method is MSE, and it turn out that the more linear the regression, the better the score is. Ultimately, the best score was produced by the combined efforts of lasso, elastic net, KRR, and GBoost

5.2 Stacked:

Figure12. Stacked Method

As you can see from the two layers of stacking within the figure above, the first layer has five models and the second has only one. The role of the first layer model is to train an Rn×m feature matrix for inputting the second layer model training, where n is the number of training data rows and m is the number of first layer models. We used lasso for the second model and the other models for the first layer. If the first layer has less than five models, we can ensemble the left model by changing the weight. We employed the following equation: “ensemble = stacked\_pred\*0.70 + averaged\_pred\*0.2 +xgb\_pred\*0.1”, and after careful processing, the results were as follows: the first layer is made up of KRR, elastic net, and GBoost, and the second layer is lasso. In this way, the best score is achieved.

1. Final score