HOUSING PRICES PREDICTIONS BASED ON MACHINE LEARNING ALGORITHMS

Shuhan Zeng
M.A. Economics
Duke University
shuhan.zeng@duke.edu

Xuan Lin
M.S. Economics and Compuation
Duke University
xuan.lin@duke.edu

May 2, 2020

1 Project Introduction

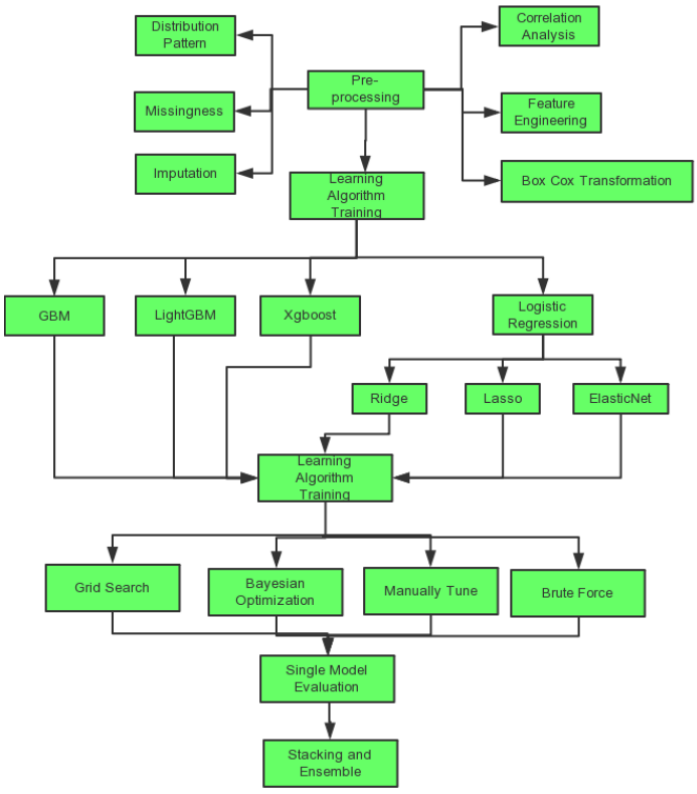


Figure 1: Work Flow

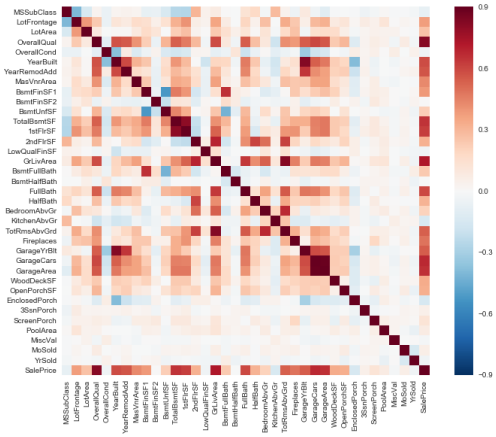
Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. Exquisite decor, spacious yards, proper size, suitable housing orientation, located neighborhoods or nearby environments are all important factors to consider when buying a home. In this Machine Learning project, we use a playground competition's dataset from Kaggle platform to conduct housing price predictions. The Ames Housing dataset was compiled by Dean De Cock for use in data science education. With 79

explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges us to predict the final price of each home. Next we will introduce how we apply our understanding of the real estate industry and combine the appropriate machine learning skills to face this challenge. Before jumping into the detail, a work flow graph that captures the main regression algorithms and other techniques we incorporated into the model is shown above.

2 Data Processing

2.1 Correlation Analysis

After some basic data processing and transformation, an correlation map is plotted to identify the correlation between features and target. Features are separated to numeric and categorical and some highly positive or negative correlations are found between a few variables, especially between “TotalSF”, “1stFlrSF” and “2ndFlrSF”. Such correlations also differ in different number groups. We believe a linear regression model would be suitable option to focus on due to low collinearity between different features.



(a) Correlation Map

	Skew
MiscVal	21.939672
PoolArea	17.688664
LotArea	13.109495
LowQualFinSF	12.084539
3SsnPorch	11.372080
LandSlope	4.973254
KitchenAbvGr	4.300550
BsmtFinSF2	4.144503
EnclosedPorch	4.002344
ScreenPorch	3.945101

(b) Numerical Features Skewness

Figure 2: Data Processing

2.2 Feature Engineering

We first transform some numerical variables that are really categorical and change "OverallCond" into a categorical variable. Then "Year" and "month sold" are transformed into categorical features. We label Encoding some categorical variables that may contain information in their ordering set process columns, apply LabelEncoder to categorical features. After adding total "sqfootage" feature, we got the skew of all numerical features showing below. Based on cutoff value of skewness, any features that have a value bigger than 0.75 are applied with box cox transformation to remove outliners’ impact.

3 Modeling

3.1 Learning Algorithm Training

3.1.1 Ridge

Firstly we can use ridge to get the basic information from the group of data. It is majorly used to prevent overfitting. Since it includes all the features, it is not very useful in case of exorbitantly high number features, say in millions, as it will pose computational challenges. In this model, we should adjust two parameters which called alpha and degree. The best parameter of our results performance is following: alpha: 0.2, degree: 2. The negative mean squared error is -0.0142637151298.

3.1.2 LASSO

Ridge or lasso are forms of regularized linear regressions, which adds tuning parameter to a model to induce smoothness in order to prevent overfitting. Since lasso provides sparse solutions, it is generally the model of choice for modeling cases where the number features are in millions or more. In such a case, getting a sparse solution is of great computational advantage as the features with zero coefficients can simply be ignored. The best parameter of our results performance is following: 'alpha': 0.001. The negative mean squared error of Lasso is -0.0134423581553.

3.1.3 GBM

Apply Gradient Boosting

To improve Lasso, the next step is to apply gradient boosting. At the first place, we set up the gradient boosting regressor with a learning rate of 0.1. Learning rate is considered as shrinkage, which is used for reducing, or shrinking, the impact of each additional fitted base-learner (tree). It reduces the size of incremental steps and thus penalizes the importance of each consecutive iteration. There are other parameters we need to define, including n_estimators, max_depth, min_samples_leaf and min_samples_split.

Table 1: Range of Parameters

Parameters		
Name	Description	Range
n_estimators	number of sequential trees to be modeled	[2000,2500,3000,3500]
max_depth	maximum depth of a tree	[3,4,5,6]
min_samples_leaf	minimum samples required in a terminal node or leaf	[10,11,12,13]
min_samples_split	minimum number of sample for splitting	[9,10,11,12]

Implementing GridSearch within GBM

Since combining four parameters with different values will give various results, we consider to implement GridSearch to find the optimal parameters to employ within the GBM model. As we know, a grid search algorithm must be guided by some performance metric. Here, we evaluate the result by scoring mean squared error in a negative term. The result gives best parameters: min_samples_split: 9, n_estimators: 2000, max_depth: 4, min_samples_leaf: 13 with the negative mean squared error -0.0142.

3.1.4 ElasticNet

After deriving Lasso, GBM and Ridge, we turn to ElasticNet to exploit further potential improvement in performance and prediction accuracy. In this model, we need to look at two parameters: alpha and I1_ratio. Alpha is a constant value that multiples the penalty term. We do not simply employ the default value which is 1.0, instead we utilize grid search method again as what we have done in GBM to discover the optimal value. Similarly, we apply the same procedure for the ElasticNet mixing parameter I1_ratio, whose given range is smaller than 1 and thus defining the penalty term as combination of L1 and L2. The final result gives best parameters choice: {alpha: 0.001, l1_ratio: 0.5}. The negative mean squared error is -0.012953412375.

3.1.5 XGBoost

XGBoost stands for Extreme Gradient Boosting. The XGBoost library implements the gradient boosting decision tree algorithm. Gradient boosting is an approach in which new models are created that predict the residuals or errors of prior models, and then combines to produce a final prediction. It is called gradient boosting because it employs a gradient descent algorithm in order to minimize the loss when adding new models. This approach supports both regression and classification predictive modeling problems. In order to employ the XGBoost model within our own model, we follow the process outlined below.

Tuning parameters

Table 2: Range of Parameters

Parameters		
Name	Description	Range
colsample_bylevel	subsample ratio of columns for each split in each level	[0.6,1.0]
colsample_bytree	subsample ratio of columns when constructing each tree	[0.4,1]
eta	step size shrinkage used in update to prevents overfitting	[0.01,0.1]
gamma	minimum loss reduction required to make a further partition on a leaf node of the tree	[0,1]
max_bin	maximum number of binary	[30,100]
max_depth	maximum depth of a tree	[3,8]
max_leaves	maximum number of leaves in a tree	[10,100]
min_child_weight	minimum sum of instance weight(hessian) needed in a child	[1,10]
n_estimators	number of sequential trees to be modeled	[2000,3000]
reg_alpha	L1 regularization term on weight	[0, 1]
reg_lambda	L2 regularization term on weights	[0, 1]
subsample	Subsample ratio of the training instance	[0.5,1.0]

Bayesian Optimization

After setting the range for each parameter, we used Bayesian algorithms to find the 'best' set of parameters for our model in the range that we had chosen.Bayesian optimization is a methodology employed for the global optimization of noisy blackbox functions. Applied to hyper-parameter optimizations, Bayesian optimization consists of developing a statistical model of the function from hyper-parameter values to the objective evaluated on a validation set.

We finished the first run of bayesian optimization process by now. But a very important step here is changing the range of each parameter, or adding more parameters, to repeat the steps above and try to find a model with greater value. Subsequently, we found no other model with changed parameter value generates a greater result than the 10th step we have done above in the first run.

3.2 Ensemble and Stacking

In order to improve the accuracy of the predictions, we choose to do ensemble and stacking. In this part, we combine the four best performing models in the previous data set to make predictions of housing prices. The four best performing models are lasso, ridge, elastic net and gradient boosting.

3.2.1 Ensemble

An ensemble is itself a supervised learning algorithm, because it can be trained and then used to make predictions. The trained ensemble, therefore, represents a single hypothesis. This hypothesis, however, is not necessarily contained within the hypothesis space of the models from which it is built. Thus, ensembles can be shown to have more flexibility in the functions they can represent. This flexibility can enable them to over-fit the training data more than a single model would.

Empirically, ensembles tend to yield better results when there is a significant diversity among the models. Many ensemble methods, therefore, seek to promote diversity among the models they combine. So our team chose to combine the four best performing models in the previous data set: lasso, ridge, elastic net and gradient boosting.

3.2.2 Stacking

We continue to use a machine learning method. The approach illustrated is called model stacking. The diagram below shows the high-level model stacking architecture, which is composed of two stages. The base-model stage, called Level 0, is composed of three modeling algorithms: gradient boosting, elastic net, kernel ridge. The meta-model stage, Level 1, is composed of a single lasso model. The predictions from Level 1 are viewed as the final predictions.

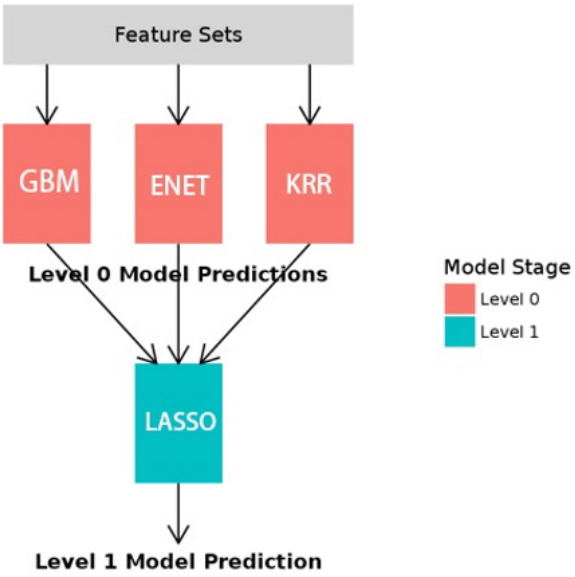


Figure 3: Stacking Model

4 Results

Table 3: Root-mean-square deviation

Fold	Range
Fold 1	0.138736
Fold 2	0.143393
Fold 3	0.112714
Fold 4	0.154527
Fold 5	0.145655
Full	0.139705

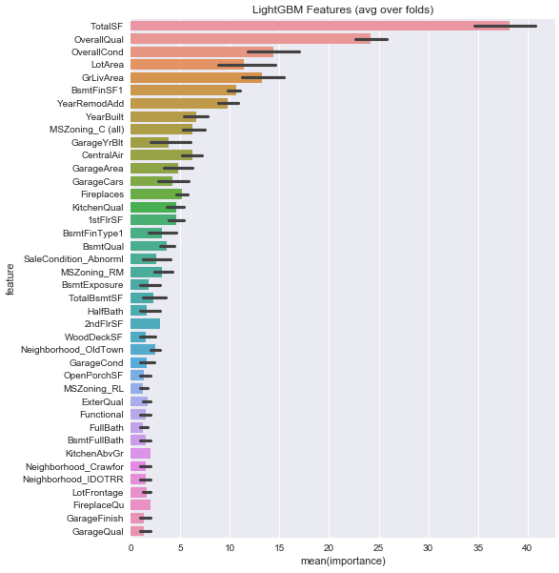


Figure 4: Main Results

Based on our stacking model that incorporate four main algorithms and Lasso regression, the prediction results is pretty good. Buy running K-fold regression, the full root mean square deviation is 0.139705, surpassing the performance of other models proposed by others on the Kaggle platform. Another important aspect of our prediction is to elaborate the feature importance of respective variables. It is found that the total square footage and overall quality are the two main factors that determine the housing prices in general, which concludes our project.