

Machine Learning Engineer Capstone Project:

Predicting Ames house prices.

House Prices: Advanced Regression Techniques

I. Definition

Project Overview

Predicting the residential house price can guide real estate agents during the decision making and pricing process, therefore assistant their customers to find the best selling price for their houses. Industry leaders also use the prediction of the house price for various purposes. Zillow uses the prediction of the house price to provide guidance to its online seller and buyer. Airbnb leverages the house price prediction to inform their hosts to better pricing their places.

The objective of this project is to predict the final price of a house given 79 explanatory variables that describe almost all aspects of a house in Ames, Iowa. This project is from a Kaggle playground competition (<https://www.kaggle.com/c/house-prices-advanced-regression-techniques>). The data in this project is described in detail by De Cock (2011).

Kaggle is a platform for data science competition. Participants are challenged to build models given the data provided to make predictions and then submit their results to Kaggle. Kaggle usually provides a training dataset and a test dataset and the task of the participants is to build a model based on the training dataset and make predictions on the test dataset. The submitted results will be evaluated by specific evaluation metrics on the test dataset. During the competition, there will be a score calculated based a fraction of the test dataset shown on the Kaggle Leaderboard, which is known as the public score. The final competition score, also called the private score, is based on the complete test dataset. During the model construction process, Kaggle participants usually create a private cross validation score following the competition evaluation metrics on their local machine, which is often called the local score. In this project, I

will mainly focus on the Kaggle's public leaderboard, which means, I will try to improve the performance of the model prediction measured by the Kaggle's public score.

Problem Statement

The Ultimate goal of this project is to train a model based on the given data to predict the house price as accurate as possible evaluated by the Kaggle Leaderboard. The related tasks to this problem include:

- Exploratory data analysis - Understanding the type of features in the dataset, how much missing values of different features, how many outliers are existed and whether specific features are skewed, are crucial in order to create a good model in the end.
- Feature preprocessing - Preprocess the data using the insights obtained from the exploratory data analysis, potentially including feature transformation, data type transformation, outlier detection and imputing missing values.
- Benchmark modeling - Creating a model using a standard technique for the problem in order to set up a benchmark for the future modeling improvement.
- Model improvement - Tuning model parameters to improve the performance of individual model and stacking individual models to provide final submission using ensemble learning technique.

Metrics

The Root Mean Squared Logarithmic Error (RMSLE) will be used to as the evaluation metrics for this project. The RMSLE can be expressed as:

$$\epsilon = \sqrt{\frac{1}{n} \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2}$$

where ϵ is the RMSLE score, n is the total number of observations, p_i is the predicted number of the house price, a_i is the associated actual house price and $\log(x)$ is the natural logarithm of x . The value of RMSLE is higher when the differences between the predicted and actual house prices are larger.

Compared to the most common Root Mean Squared Error (RMSE), RMSLE does not heavily penalize the huge difference between the predicted and actual values. Taking logs means that errors in predicting expensive houses and cheap houses will affect the result equally.

II. Analysis

Data Exploration

The training dataset is mixed with categorical and numerical features as well as some missing values. Specifically, there are 1460 observations, each with 79 features including the sale price of the house. The test dataset has 1461 observations, each with 78 features. Our task is to fill the sale price of the test dataset using the model trained from the training dataset. The histogram of the house price is shown in Figure 1, which shows that it is skewed. Thus, the log transformation needs to be performed in order to normalize it and this is the reason why we need to use the RMSLE as our evaluation metric.

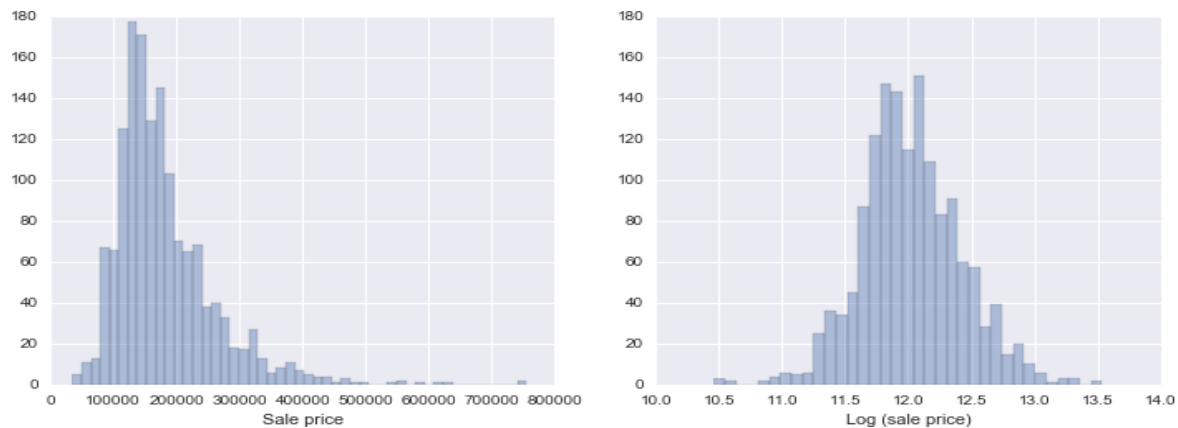


Figure 1: The histograms of the house sale price. (left) Original data; (right) Log transformed data.

Exploratory Visualization

Figure 2 shows the correlation matrix between the numerical variables and the house sale. There are several variables that are strongly correlated with the house price shown in Figure 2. I also explore the top six variables that are highly correlated with the sale price in Figure 3.

OverallQual	0.790982
GrLivArea	0.708624
GarageCars	0.640409
GarageArea	0.623431
TotalBsmtSF	0.613581
1stFlrSF	0.605852
FullBath	0.560664
TotRmsAbvGrd	0.533723
YearBuilt	0.522897
YearRemodAdd	0.507101
GarageYrBlt	0.486362
MasVnrArea	0.477493
.....	

Figure 2: The correlation coefficients between some variables and the sale price

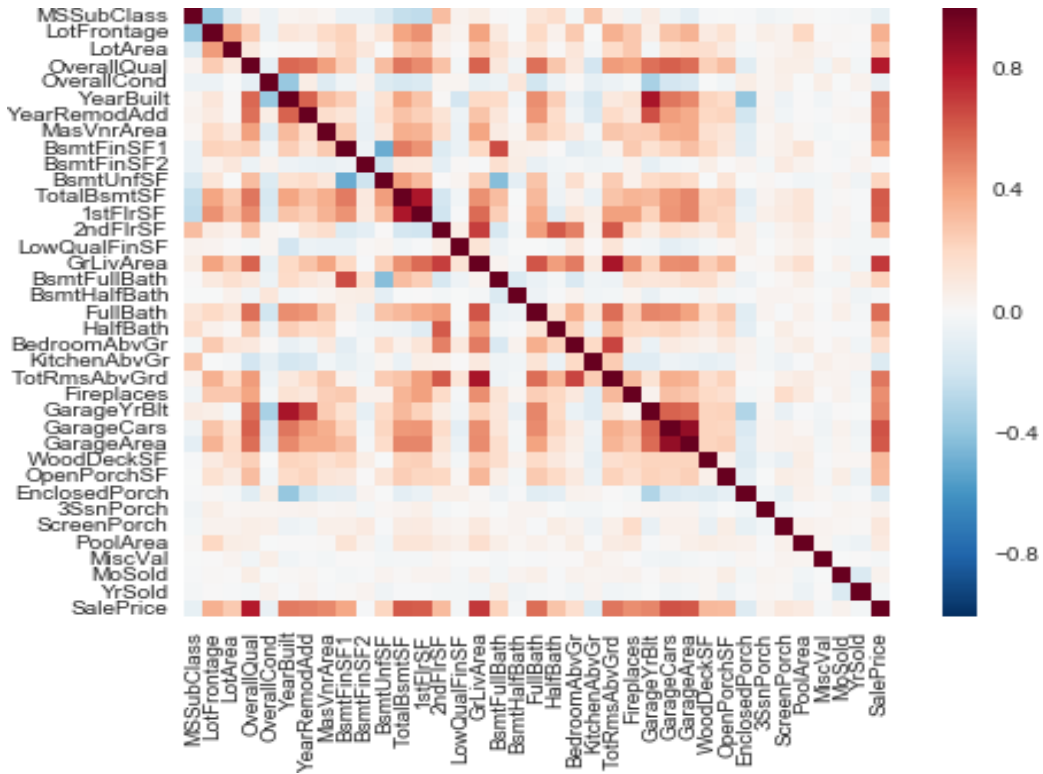


Figure 3: The correlation matrix between different numerical variables.

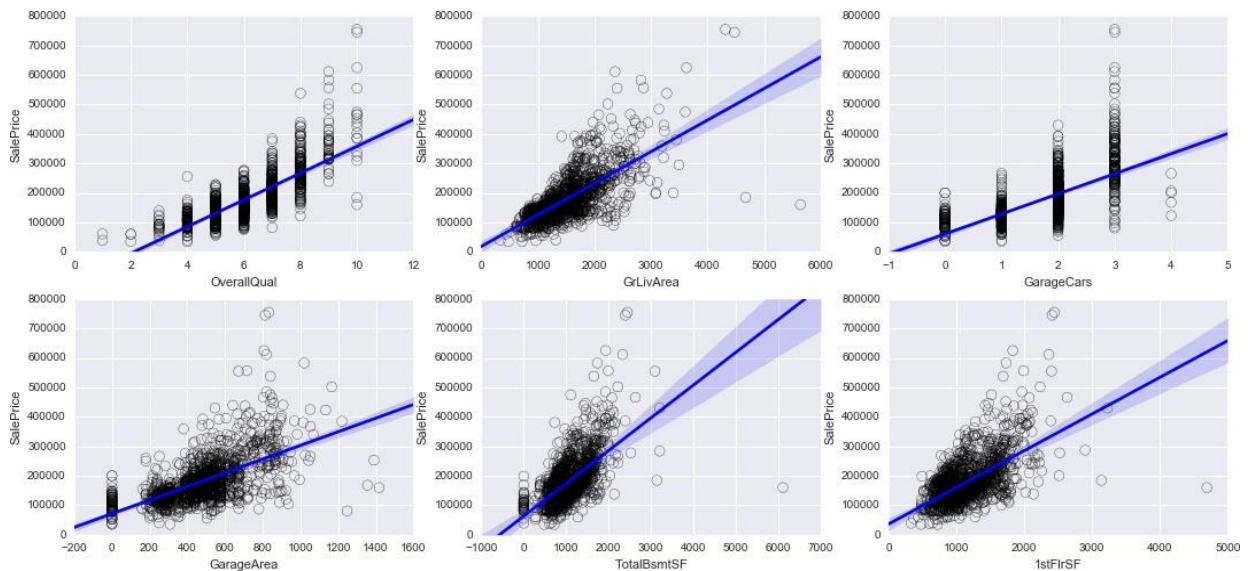


Figure 4: The scatter plot of the top 6 strongly correlated variables with the house price. The regression curve is shown in blue.

To give an example of the categorical variable, the boxplot between the neighborhood and the sale price is shown in Figure 5. We can observe a clear pattern that the neighborhood feature can greatly affect the house price. In order to include all categorical features into our machine learning model, the categorical variables can be transformed into some dummy variables, which takes the value 0 or 1 to indicate the absence or presence of some categorical effect.

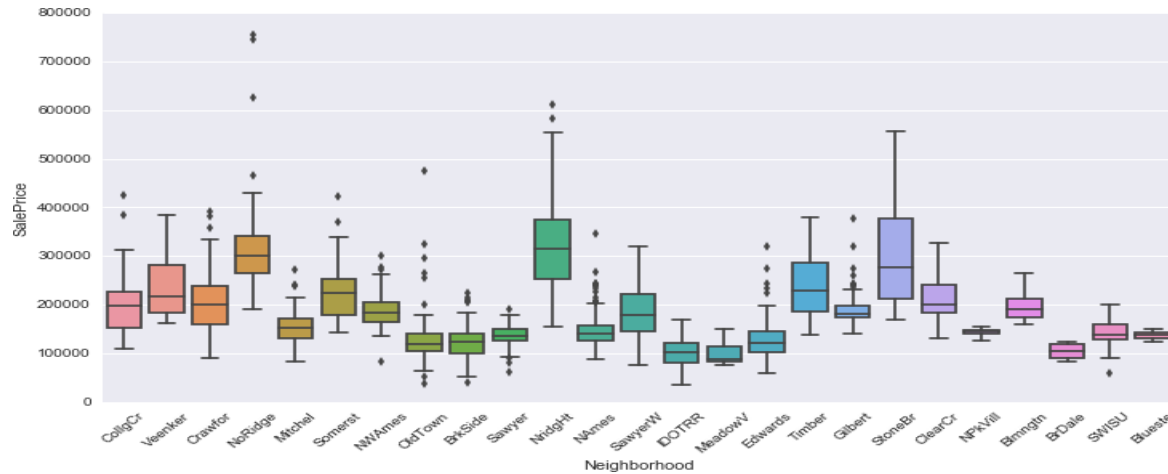


Figure 5: Boxplot of sale price in different neighborhood.

Algorithms and Techniques

Given one of the purposes of this project is to explore advanced regression techniques, I am going to use the ensemble learning technique to achieve better evaluation score. The ensemble learning techniques aim to learn a weighted combination of the base models and it sometimes called a committee method (Murphy, 2012). The technique we implemented in this project is called stacking. Stacking, which stands for “stacked generalization”, was introduced by Wolpert (1992). The data science competition participants use this method extensively in order to win the competition. One well known example is the Netflix prize competition winning solution, BellKor’s Pragmatic Chaos.

The basic idea of stacking is to use another model or “stacker” to combine all previous model predictions in order to reduce the generalization error. An illustration of a 2 level 5 folds stacking approach is shown in Figure 6. First, the training dataset need to be split into 5 folds. Second, we iterate over this 5 folds training dataset. In each iteration, each base model will be trained using 4 folds and predict on the hold out fold. At the same time, each base model also need to provide a prediction on the entire test dataset. After the iteration over all folds, we will have the prediction of the entire training dataset for each model and 5 copies of the prediction of the entire test dataset for each model. Finally, we train second level model, or stacker, using the prediction in the training dataset as new features and use the average of the 5 copies of the test dataset predictions as the test input for the trained model to provide the final prediction.

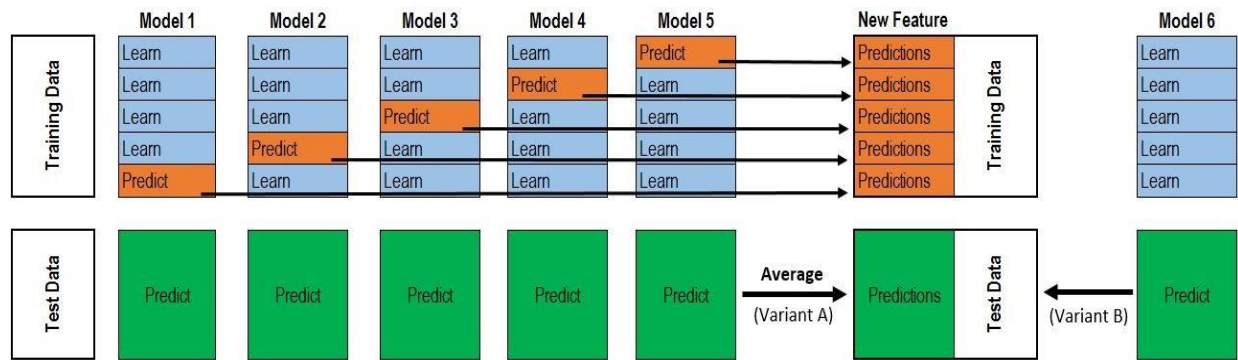


Figure 6: An illustration of stacking approach. (Credit: Kaggle forum)

Benchmark

The Random Forest regression is selected as our benchmark model for this project. The default parameters in scikit-learn package will be used as the benchmark, which earns a local cross validation score of 0.138194 and 0.15702 on the public leaderboard (rank 457 out of 662 teams).

III. Methodology

The overall methodology and modeling approach can be summarized using the flowchart shown in Figure 7. The details of the approach will be discussed in the following subsections.

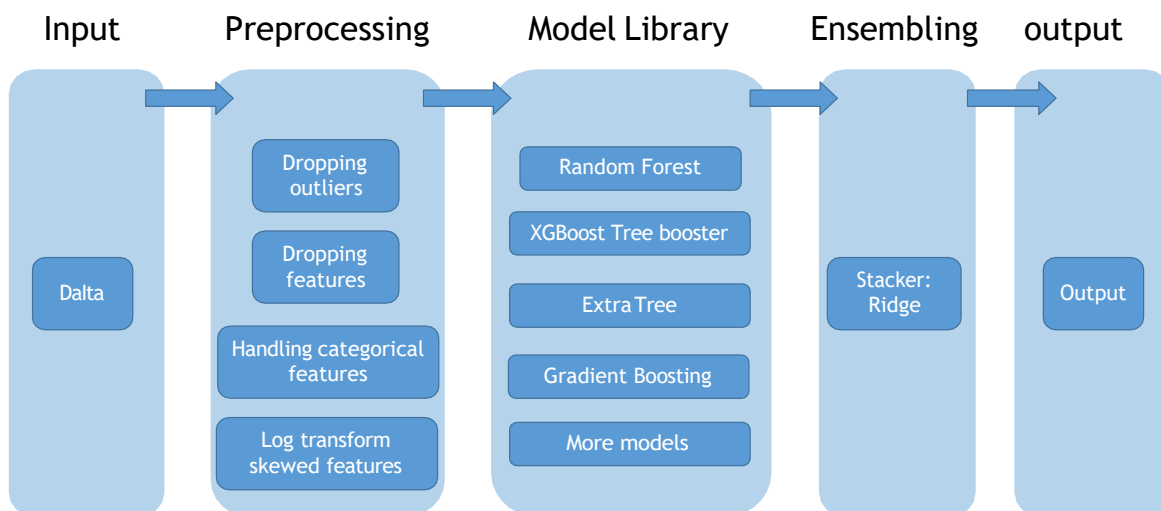


Figure 7: Flowchart of the modeling approach.

Data Preprocessing

- Outliers deletion

Any predictions from the benchmark that differ more than \$12000 than the actual house price are labeled as outliers. The threshold selected here is somewhat arbitrary but this approach actually increases both the local score and the public score a little bit in my explorations. All outliers are deleted from the training dataset for the final submission. This

- Feature selection

The feature selection approach I adopted here is a very simple and naïve one: delete features with high number of missing values. The categorical variables with the largest number of missing values are Alley, FireplaceQu, PoolQC, Fence and MiscFeature. In the data exploration analysis, there are some features that are highly correlated with the sale price. We may treat some of these features as redundant variables and delete them from the training dataset. However, previous investigation shows that this will lead to a small decrease in both the local score and public score. There are many other useful and advanced feature selection techniques, such as the boruta method (<https://m2.icm.edu.pl/boruta/>). A detail discussion of feature selection techniques is beyond the scope of this project.

- Log transform skewed features

All features that have a skewness larger than 0.75 are log transformed.

- Handle categorical features

The categorical features are handled by the pandas `get_dummies` function that converts categorical variables into dummy or indicator variables. See pandas documentation for more details (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html).

Implementation

Before applying the ensemble learning method, I trained and tuned 4 base model including Random Forest regressor, Extra Trees regressor, Gradient Boosting regressor, Extreme Gradient Boosting regressor.

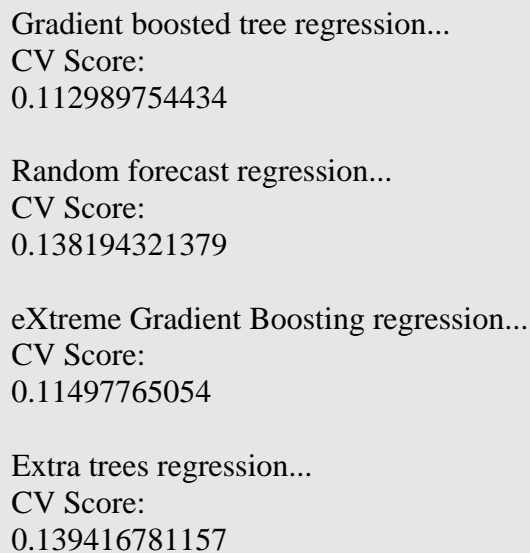
At the beginning of the ensembling implementation, I use only 4 base models to form my model library, including Random Forest regressor, Extra Trees regressor, Gradient Boosting regressor, Extreme Gradient Boosting regressor. And I use the Ridge Regression as my second level model. The idea of the stacking approach is to use the predictions from the first level model as the new features for the second level model and make prediction based on that.

The base models should be as unrelated as possible and this is why we tend to include more models in the ensemble even though they may not perform well. For my final, I use a much larger model library including Random Forest regressor, Extra Trees regressor, Gradient Boosting regressor, Extreme Gradient Boosting regressor, LassoCV regressor, K neighbors regressor, LassoLarsCV regressor, Elastic Net regressor, Support Vector Machine regressor. The Extreme Gradient Boosting regressor is from the XGBoost package, all other techniques are from scikit-learn package. The basic idea of the final model is to use a larger base model library in order to achieve better results. Some of the base models are used only with their scikit-learn default parameter values without fine tuning.

There are two challenging parts during the coding process. The first one is that we need to make sure the feature for the final second level prediction is based on the prediction of the first level base models. Thus, we also need to use the model prediction as the features for the test dataset. The second one is how exactly should we choose the base models. This is a tricky and not fully addressed question in this project since there are so many models to select. But the rule of thumb in my investigation is that I choose the one that improve the local score with the assumption that the local and public score has a good correlation.

Refinement

The refinement is conducted using the grid search technique. For the learning purpose of this project, I only use a few tunable parameters shown in the following (the detailed grid range can be found in the submitted code). Figure 8 and Figure 9 show the local score before and after the grid search. Before the parameter tuning, I just use the scikit-learn default parameter values for each model. After the refinement of the base models, we using stacking technique to ensembling them together to provide the final submission.



```
Gradient boosted tree regression...
CV Score:
0.112989754434

Random forest regression...
CV Score:
0.138194321379

eXtreme Gradient Boosting regression...
CV Score:
0.11497765054

Extra trees regression...
CV Score:
0.139416781157
```

Figure 8: The local cross validation score with default scikit-learn parameter values.

Gradient boosted tree regression...

Best Params:

`{'max_features': 10, 'n_estimators': 500, 'learning_rate': 0.05, 'max_depth': 6, 'subsample': 0.8}`

Best CV Score:

0.111495013076

Random forest regression...

Best Params:

`{'max_features': 25, 'n_estimators': 500, 'max_depth': 11}`

Best CV Score:

0.131998045253

eXtreme Gradient Boosting regression...

Best Params:

`{'n_estimators': 500, 'subsample': 0.8, 'learning_rate': 0.05, 'colsample_bytree': 0.75, 'max_depth': 7}`

Best CV Score:

0.109164143602

Extra trees regression...

Best Params:

`{'max_features': 20, 'n_estimators': 500}`

Best CV Score:

0.139196375677

Figure 9: The local cross validation score for each base model after the grid search refinement.

IV. Results

Model Evaluation and Validation

One interesting observation during the model ensembling process is that increasing the size of the model library can potentially increase both the local validation score and the Kaggle Leaderboard score. In the final submission, a variety of base models are included in the model library with their scikit-learn default parameter values. Using a much larger library enables me to rank top 10 among more than 600 teams at the time when I wrote this project report. I need to acknowledge that the approach is somewhat brute force and a more elegant way to do this is to find the base models to be as uncorrelated as possible. Nevertheless, I will address some of the issues of the model ensembling in the improvement section. Table 1 shows the local validation score and the Kaggle Leaderboard score for different modeling approaches. Figure 10 shows the Kaggle Leaderboard rank for my final submission at the time of preparing for this project.

Model	Local validation
Random forecast	0.13200
Extra Tree	0.13920
Gradient Boosted Tree	0.11150
XGBbooster	0.10916
Stacking 4best	0.10755
Final stacking	0.10069

Table 1: The local validation score and Kaggle public score for different model setup.

Justification

In summary, the ensemble approach in this project greatly improve the performance of the model in terms of the Kaggle Leaderboard score and rank from the benchmark random forecast regression ranked 457/662 to the final model ensemble stacking approach ranked 7/662. The result is quite encouraging and opens a lot more to think, learn and explore.

V. Conclusion

Free-Form Visualization

As my ultimate goal in this project is to build a model as accurate as possible measure by the Kaggle Leaderboard score, I will use the Kaggle Leaderboard score at the different model development stage as my final project visualization.

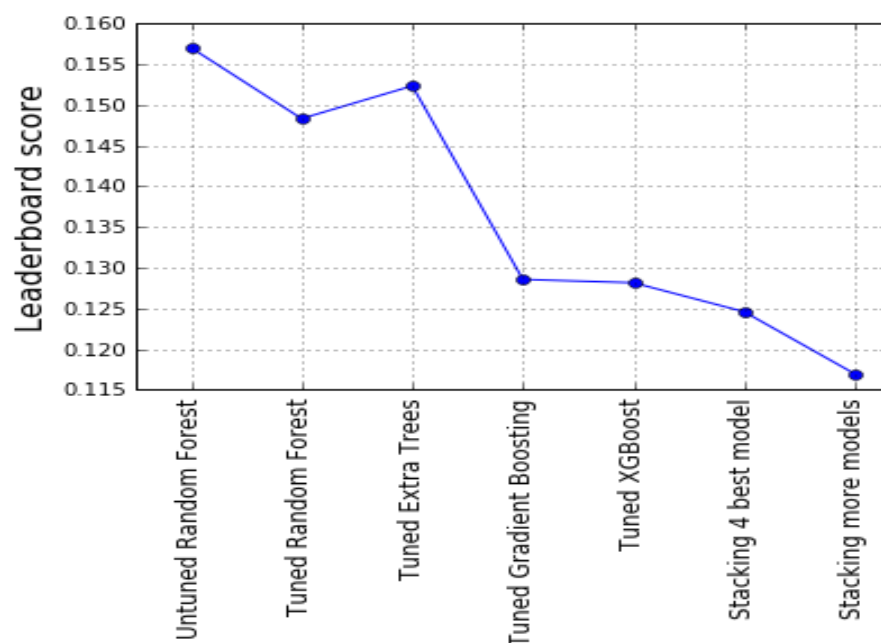


Figure 10: The Kaggle Leaderboard score at the different model development stage

Reflection

For this project, I started with an exploratory data analysis, in which I found that there are some features being skewed, some features are categorical, and some missing values. Several data preprocessing steps were conducted in order to make the data ready for the machine learning

modeling part. For the modeling part, I firstly trained several base models and tuned their parameters. Then, I applied the ensemble learning method to stack all the base models together to provide the final submission. One of difficult parts is data preprocessing since it is hard to explain in a rigorous way of why I did a specific choice to improve the model. There are a lot of exploratory tests and investigations to justify the choice only based on the local cross validation scores. The other challenge is the stacking approach itself since it is more complicated than a single model approach. It is quite easy to use the wrong features for the stacker since it makes prediction based on the predictions of all the base models. The key to success is to write a pseudocode and check it carefully before writing the production code.

Improvement

This project opens several avenues for future work. One part is for data preprocessing, such as more advanced outlier detection techniques, more advanced feature selection techniques, fine tuning of the log transform threshold of the skewed variable and so on. More creative feature engineering will be also extremely useful. Another part is for the regression techniques. My intuition for ensemble learning technique is that a better score may be achieved by increasing the size of the model library. Kaggle competition winners usually train hundreds even thousands of different base models, see Kaggle Otto product classification challenge winning solution for an example (<https://www.kaggle.com/c/otto-group-product-classification-challenge/forums/t/14335/1st-place-winner-solution-gilberto-titericz-stanislaw-semenov>). The dilemma here is a tradeoff between the amount of time and engineering efforts you want to spend and the actual value of the tiny improvement. A very interesting idea to automate this ensembling process is suggested by MLwave (<http://mlwave.com/kaggle-ensembling-guide/>).