

Final Report: Kaggle House Prices Prediction Challenge

Team Name: **Saquibvai**

Contestants Name:

- Md Nazmus Saquib (1640CSE00476)
- Khaled Mahmud (1640CSE00477)
- Emon khan Toha (1640CSE00522)

Kaggle link- kaggle.com/saquibvai

Git link- github.com/saquibvai/saquibvai

1 Project Goal

This competition is hosted by data analysis club IIT Palakkad. The challenge is to predict the price of each house given some information related to houses. Goal of competition is to make us familiar with environment of kaggle and basics of regression, and our goal was to learn and do our best in this contest.

Problem Statement

This playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence. With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, we have to predict the final price of each home.

2 Data Description and Preprocessing

Training dataset contains 1460 training samples and 1459 testing samples. Each sample is represented by 81 features in the training dataset and 80 features in the test dataset.

We used a function to simplify the analysis of general characteristics of the data.

Inspired on the *str* function of R, this function returns the *types*, *counts*, *distinct*, count *nulls*, missing ratio and unique values of each field/feature.

If the study involve some supervised learning, this function can return the study of the correlation, for this we just need provide the dependent variable to the *pred* parameter.

3 Feature Selection

The LightGBM model the importance is calculated from, if 'split', result contains numbers of times the feature is used in a model, if 'gain', result contains total gains of splits which use the feature.

On the XGBoost model the importance is calculated by:

- 'weight': the number of times a feature is used to split the data across all trees.
- 'gain': the average gain across all splits the feature is used in.
- 'cover': the average coverage across all splits the feature is used in.
- 'total_gain': the total gain across all splits the feature is used in.
- 'total_cover': the total coverage across all splits the feature is used in.

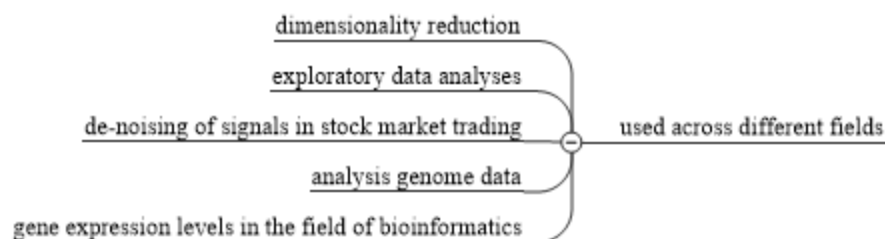
Feature Selection into the Pipeline

Since we have a very different selection of features selection methods, from the results it may be interesting keeping only the removal of collinear and multicollinear, and can decide with we must have the pre polynomials and apply PCA or not. We can still improve the results through hyper parameterization and cross-validation.

4 Compressing Data via Dimensionality Reduction

PCA

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. If there are n observations with p variables, then the number of distinct principal components is $\min(n-1, p)$. This transformation is defined in such a way that the first principal component has the largest possible variance, and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.



5 Modeling Methods

First, we started to look at different approaches to implement linear regression models, and use hyper parametrization, cross validation and compare the results between different errors measures.

Model Hyper Parametrization

Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2.$$

Root-Mean-Square Error (RMSE)

$$\text{RMSE}(\hat{\theta}) = \sqrt{\text{MSE}(\hat{\theta})} = \sqrt{E((\hat{\theta} - \theta)^2)}.$$

Mean Absolute Error (MAE)

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n} = \frac{\sum_{i=1}^n |e_i|}{n}.$$

Residuals Plots

The plot of differences or vertical distances between the actual and predicted values. Commonly used graphical analysis for diagnosing regression models to detect nonlinearity and outliers, and to check if the errors are randomly distributed.

- Since $\text{Residual} = \text{Observed} - \text{Predicted}$ **positive values** for the residual (on the y-axis) mean the **prediction was too low**, and **negative values** mean the **prediction was too high**; 0 means the guess was exactly correct.
- *They're pretty symmetrically distributed, tending to cluster towards the middle of the plot.*

- Detect outliers, which are represented by the points with a large deviation from the centerline.
- *They're clustered around the lower single digits of the y-axis (e.g., 0.5 or 1.5, not 30 or 150).*
- If we see patterns in a residual plot, it means that our model is unable to capture some explanatory information.
- *Non-constant error variance shows up on a residuals vs. fits (or predictor) plot in any of the following ways:*
- The plot has a "fanning" effect. That is, the residuals are close to 0 for small x values and are more spread out for large x values.
- The plot has a "funneling" effect. That is, the residuals are spread out for small x values and close to 0 for large x values.
- Or, the spread of the residuals in the residuals vs. fits plot varies in some complex fashion.

Lasso (Least Absolute Shrinkage and Selection Operator)

Lasso is able to achieve both of these goals by forcing the sum of the absolute value of the regression coefficients to be less than a fixed value, which depending on the regularization strength, certain weights can become zero, which makes the Lasso also useful as a supervised feature selection technique, by effectively choosing a simpler model that does not include those coefficients. However, a limitation of the Lasso is that it selects at most n variables if $m > n$.

This idea is similar to ridge regression, in which the sum of the squares of the coefficients is forced to be less than a fixed value, though in the case of ridge regression, this only shrinks the size of the coefficients, it does not set any of them to zero.

The optimization objective for Lasso is: $(1 / (2 * n_samples)) * ||y - Xw||^2_2 + \alpha * ||w||_1$

Technically the Lasso model is optimizing the same objective function as the Elastic Net with `l1_ratio=1.0`, no L2 penalty.

As

	Scorer	Index	BestScore	BestScoreStd	MeanScore	MeanScoreStd
0	MEA	51	0.080	0.003	0.153	0.006
0	R2	51	92.364	0.223	69.284	0.743
0	RMSE	51	0.116	0.026	0.247	0.065

After removing some outliers the result was impressive, shown below:

	Scorer	Index	BestScore	BestScoreStd	MeanScore	MeanScoreStd
0	MEA	244	14582.133	854.804	14647.674	858.145
0	R2	242	92.600	0.579	92.537	0.526
0	RMSE	268	21504.641	7071.982	21600.052	7013.087

6 Modeling: We used multiple regressions and got the best score from the models hyper parameterization. Used regressions are given below:

- XGB Regressor
- Hub
- BayR
- SGDR
- ELA
- LR
- ORT
- PassR
- Lasso
- GBR

	Name	BestScore	BestScoreStd
0	XGBRegressor	19959.976	6373.877
0	Hub	21315.975	7462.560
0	BayR	21390.979	6975.660
0	SGDR	21393.908	7005.896
0	ELA	21395.037	7174.726
0	LR	21396.016	7012.141
0	ORT	21396.016	7012.141
0	PassR	21424.972	7066.795
0	lasso	21504.641	7071.982
0	GBR	23838.287	9177.712

6 Stacking the Models

After modeling we stacked the models and averaged base model score.

```
Recive 187 features...
Select 109 features
Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=4)]: Done   5 out of   5 | elapsed:    7.2s finished

Recive 187 features...
Select 109 features
Fitting 5 folds for each of 1 candidates, totalling 5 fits


[Parallel(n_jobs=4)]: Done   5 out of   5 | elapsed:    0.5s finished

Recive 187 features...
Select 109 features
Fitting 5 folds for each of 4 candidates, totalling 20 fits

[Parallel(n_jobs=4)]: Done  20 out of  20 | elapsed:    1.5s finished

Select 109 features
Select 109 features
Select 109 features
Select 109 features
Select 109 features
Select 109 features
RMSLE score on the train data: 18933.2952
Select 109 features
Select 109 features
Select 109 features
Accuracy score: 94.314833%
```

Result & Discussion: And here's a list of our submission given below:

2 submissions for saquibvai		Sort by	Most recent ▼
All Successful Selected			
Submission and Description		Public Score	Use for Final Score
sample_submission.csv 16 days ago by saquibvai add submission details		0.40890	<input checked="" type="checkbox"/>
sample_submission.csv 17 days ago by saquibvai add submission details 		0.40890	<input type="checkbox"/>
No more submissions to show			