# Zillow's Home Value log error Prediction

Yangfan Xu

# Definition

## 1.1 Project Overview

The housing market has always been a topic of attention. Zillow is an online real estate database company. The Zestimate® home value is Zillow's estimated market value for an individual home and is calculated for about 100 million homes nationwide. Zillow provide a full list of real estate properties in three counties data. This project would predict log error of home value. The target is improving estimating home value.

Jiaoyang Wu has researched housing price prediction using support vector regression[1]. Aaron NG has applied machine learning on housing price prediction[2]. He finally choose Gaussian Process regression.

In dataset, there are four files. The first one is features file, which is full list of real estate properties in three counties (Los Angeles, Orange and Ventura, California) data with all features. The second one is some properties' transactions in 2016. The third one is some properties' transactions from 1/1/2017 to 9/25/2017. Every transaction include date, property ID and log error. The third one explains features. The fourth file explain features.

## 1.2 Problem Statement

The solution model will predict the log error between Zestimate and the actual sale price. Predicting log error would help to improve the Zestimate.

The log error is defined as:

$$\boldsymbol{logerror} = \boldsymbol{log}(\boldsymbol{Zestimate}) - \boldsymbol{log}(\boldsymbol{SalePrice})$$

This project would use machine learning technique to build model, predict the log error.

Algorithm is core of model. From above statement. The output of our model would be labeled data. This means the algorithm should be a supervised one. It is continuous, regression is suitable for it. In additional, the features file shows us that characteristic of inputs is classification. We can declare that the problem is regression and classification problem. The solution model need an algorithm which could solve this kind of problem.

## 1.3 Metrics

This project use Mean Absolute Error for Evaluation which between the predicted log error and the actual log error. It be used to quantify the performance of both the benchmark model and the solution model.

Mean absolute error (MAE) is a measure of difference between two continuous variables. Assume X and Y are variables of paired observations that express the same phenomenon.

The Mean Absolute Error is given by:

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

In this case, the MAE should be:

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |logerror\_pre_{\ i} - logerror\_actual_{\ i}|}{n}$$

the smaller the MAE, the better the model fits your data.

Every property with equally weight exist in model. For $(logerror\_pre_{\ i} - logerror\_actual_{\ i})$, there are positive values and negative values. Before taking sum, It must take absolute value. The Mean Absolute Error provides average log error of properties in model. The smaller average log error show that the prediction of model is closer to actual log error. The best model is which could get smallest Mean Absolute Error.

# Analysis

## 2.1 Data Exploration

In this project, dataset comes from Zillow. There are three files.

The first one is named ‘property’, which is full list of real estate properties in three counties (Los Angeles, Orange and Ventura, California) data with all features. It includes 2985217 properties. Every property includes 58 features. The size of file is 619M.

There are 25 features, which include valid values more than 2,500,000.

| Feature, Amount | Feature, Amount | Feature, Amount |
|---|---|---|
| bathroomcnt 2982260 | bedroomcnt 2982272 | calculatedbathnbr 2868061 |
| calculatedfinishedsquarefeet 2940120 | finishedsquarefeet12 2720786 | fips 2982285 |
| fullbathcnt 2868061 | latitude 2982285 | longitude 2982285 |
| lotsizesquarefeet 2712511 | propertycountylandusecode 2982218 | propertylandusetypeid 2982285 |
| rawcensustractandblock 2982285 | regionidcity 2923089 | regionidcounty 2982285 |
| regionidzip 2972503 | roomcnt 2982248 | yearbuilt 2937384 |
| structuretaxvaluedollarcnt 2938753 | taxvaluedollarcnt 2950951 | assessmentyear 2982284 |
| landtaxvaluedollarcnt 2925291 | taxamount 2962465 | censustractandblock 2910232 |

There are 25 features, which include valid values more than 2,500,000. In these 25 feature, 'parcelid' is Unique identifier for parcels. ‘propertylandusetypeid’ is county land code.

The second file ‘train_2016’ is 90363 properties’ transactions in 2016. The third file ‘train_2017’ is 77613 properties’ transactions from 1/12017 to 9/25/2017. Every transaction includes date, property ID and log error. Fig2 and Fig3 are scatters which show log error in plot. Obviously, A lot of outliers need to remove in processing.

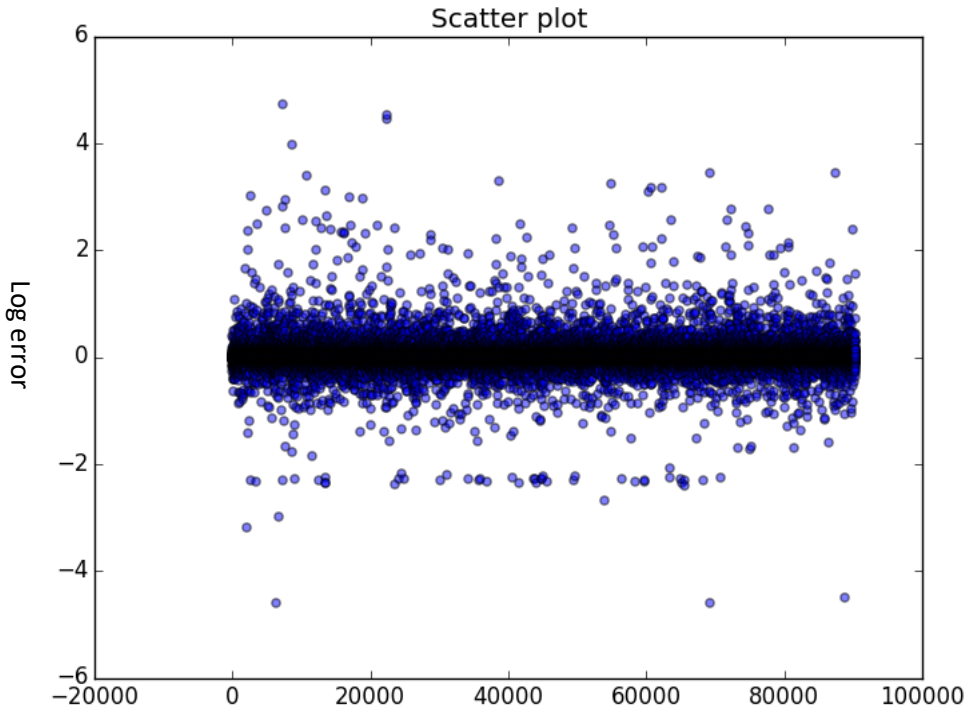



Fig 2

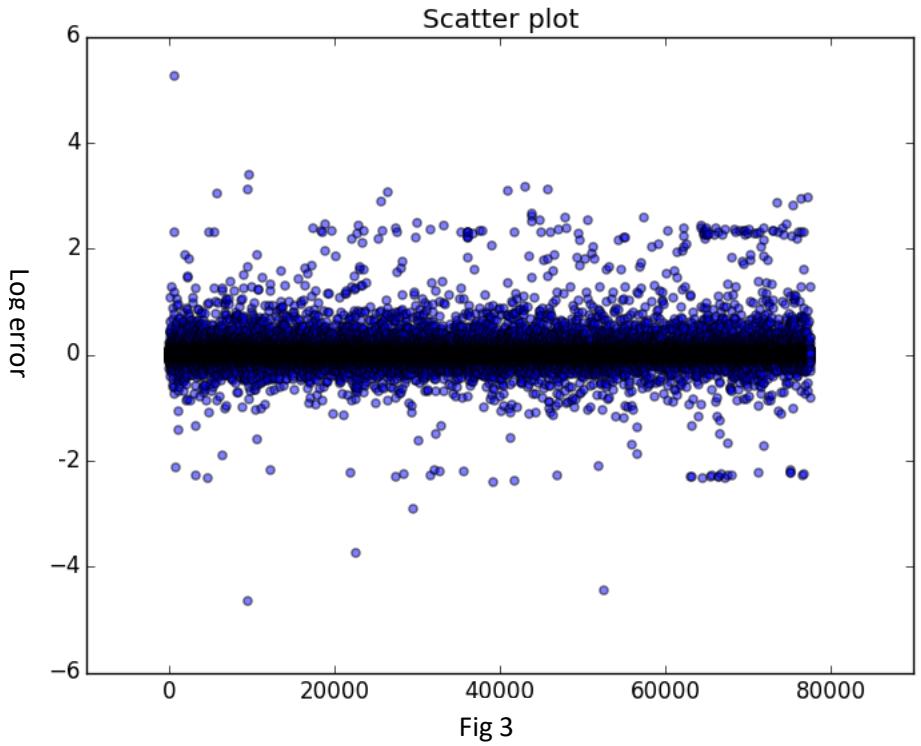


Fig 3

The last one explains property's 58 features and is named 'data_dictionary'. It can help to understanding features.

# 2.2 Algorithms and Techniques

## 2.2.1 Missing values

From above data exploration, there are 25 features, which include valid values more than 2,500,000. Another features don't have enough values to predict. The project would select the 25 features to training. These 25 features also have some missing values. In sklearn, The Imputer class provides basic strategies for imputing missing values, either using the mean, the median or the most frequent value of the row or column in which the missing values are located. The project could use Imputer to replacing missing values. Setting strategy to "median", it would calculate the median of column and replace missing values by the median.

## 2.2.2 Dimensionality reduction

For features, if two features are correlate, We could condense them into one new feature. The project would to use principal components analysis which could reduce the number of features. 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 (or sometimes, principal modes of variation). The number of principal components is less than or equal to the smaller of the number of original variables or the number of observations.

### 2.2.3 Training

Algorithm would be core of solution model. According above statement, The transactions file shows actual log error. The output of our model would be labeled data. This means the model's algorithm should be a supervised one. It is continuous, regression is suitable for it. In additional, the features file shows us that characteristic of some inputs is classification. We can declare that the problem is regression and classification problem.

[3]Gradient boosting is a machine learning technique for regression and classification problems. In sklearn, GradientBoostingRegressor is available. Gradient Boosting Regressors (GBR) are ensemble decision tree regressor models. It is suitable for regression and classification problem. The project plan to try it.

Gradient boosting involves three elements. They are loss function, weak learner and additive model. GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function[3].

1> For example: You are given$(x_1, y_1)$, $(x_2, y_2)$, ……..,$(x_n, y_n)$, get F(x) using decision tree algorithm. Decision trees are used as the weak learner in gradient boosting.
2> The loss function used depends on the type of problem being solved. Because this case is a regression, We would use least squares $(y_i - F(x_i))^2$. $(y_i-F(x_i))$ are parts that existing model F cannot do well.
3> Because those parts that existing model F cannot do well, additive model 'h' should be added. The role of h is to compensate the shortcoming of existing model F. If the new model F + h is still not satisfactory, we can add another tree. Adding additive model in order to reduce loss.

The following training parameters can be tuned to optimize the regressor:

- Learning_rate
- n_estimators
- max_depth
- min_samples_leaf

Parameters was tuned by GridsearchCV which exhaustively considers all parameter combination.

### 2.2.4 Benchmark model

The different of solution model and benchmark model is training algorithms. In benchmark model, it would to use DecisionTreeRegressor. Regression tree could solve regression and classification problem. The predicted outcome of regression tree can be considered a real number. Comparing benchmark model and the solution model would be based on Mean Absolute Error.

The following training parameters are same with solution model and can be tuned to optimize this regressor:

- max_depth
- min_samples_leaf

# Methodology

## 3.1 Data Preprocessing

The preprocessing done in the 'Prepare data' consists of the following steps:

1. Selection features with enough valid values

select features which number's values over 2,500,000

2. Remove log error's outlier

The fig2 and fig3 show logerror's outlier in data exploration. The project removed these outlier which is bigger than 0.4 and smaller than -0.4. The property could be doing training which includes logerror values between 0.4 and -0.4.

3. Create a new feature 'month'

For home value's log error, sale date is a factor. The transaction date would be considered. Then, according to the transaction in file 'train_2016', The project got month value from transaction date. This value would be as value of feature month. For transactions in 2017. Each

month's values which comes from the transaction date, it would plus 12 before assigning feature month.

4. Set feature 'parcelid' to index of data frame.

Feature 'parcelid' is the ID number of property. it should not be in training.

5. Replace NaN-value with median

creates a new *instance* of the Imputer and assigns this object to the 'imp'. For this instance, setting 'strategy' as 'median'. Calling 'fit_transform' method get data which has ready imputed NaN value.

6. Scaling feature's values

creates a new *instance* of the MinMaxScaler and assigns this object to the 'scaler'. Calling 'fit_transform' method to scaling data.

## 3.2 Implementation

The main stage of implementation process is regressor training. During this stage, the regressor was trained on the preprocessed training data using Pipeline. The Pipeline includes a instance for principal component analysis and a gradient boosting regressor. The parameter of these two instance was tuning using by GridsearchCV.

- PCA

PCA would reduce the number of features. Because higher dimension lead to bigger variance, We want to reduce dimension. Assign PCA () to 'pca'. Parameter 'components' should be a integer. We would set it latter.

Code:

```
pca = decomposition.PCA()
```

- gradient boosting regressor

assign a gradient boosting regressor instance to ‘reg_GB’.

Code:

```
reg_GB= ensemble.GradientBoostingRegressor()
```

- Pipeline

The purpose of Pipeline is assembling several steps. It make the training precessing more convenience. We need set ’steps’. In this case, It has two steps. The first one is principal component analysis, and second one is training by gradient boosting regressor.

Code:

```
pipe = Pipeline(steps=[('pca', pca),('GradientBoostingRegressor', reg_GB)])
```

- Parameters for ‘pca’ and ‘’ 'GradientBoostingRegressor'

They are 24 features. We try to set ‘n_component’ to [21, 23]. If the best (the best could be return by GridsearCV) is 21, We would try smaller one. If the best is 23. I think 23 could be trusted.

For ‘GradientBoostingRegressor’,

1) Parameter ‘n_estimators’ is assigned [150, 300]. Because its default value is 100 and a large number usually results in better performance, we decide assign bigger number.

2) Parameter ‘learning_rate’ is assigned [0.05, 0.01] which is smaller than default value(0.1). The reason is that we ‘n_estimators’ is bigger than default, which means more boosting steps. We want to keep balance between ‘n_estimators’ and ‘learning_rate’.

3) Parameter’max_depth’ means maximum depth of the individual regression estimators. We want to make bigger value in order to get more possibility. The default value is ‘3’. Here ‘max_depth’ is assigned [4, 6]. If the best is ‘6’. We would to try bigger value.

Code:

```
n_components = [21, 23]
n_estimators = [150, 300]
learning_rate = [0.05 ,0.01]
max_depth = [4, 6]
min_samples_leaf =[100, 200]
```

- GridsearchCV

The project create estimator of GridsearchCV. Exhaustive search over specified parameter values for an estimator. This is faster techinique of turning parameters.

Put pipe in GridsearchCV and Assign parameters to ‘param_grid’.

code:

```
estimator = GridSearchCV( pipe, param_grid = dict(pca__n_components=n_components,
                          GradientBoostingRegressor__n_estimators = n_estimators,
                          GradientBoostingRegressor__learning_rate = learning_rate,
                          GradientBoostingRegressor__max_depth = max_depth,
                          GradientBoostingRegressor__min_samples_leaf = min_samples_leaf
                          ))
```

## 3.3 Refinement

Based on above parameter, The MAE value on testing set is 0.0519630820599

Best parameters:

'pca__n_components': 23, 'GradientBoostingRegressor__n_estimators': 300,
'GradientBoostingRegressor__max_depth':4,
'GradientBoostingRegressor__min_samples_leaf': 200,
'GradientBoostingRegressor__learning_rate': 0.05

We try to remove more outlier which value of log error is bigger than 0.33 or smaller than -0.33. The MAE value on testing set is 0.0496774807937. The best parameters is same. The MAE on testing set decrease 4.4%.

# Results

The final architecture and algorithm were chosen which given us smaller mean absolute error between prediction log error and actual log error. The final model list below.

- ❖ Selection features with over 2,500,000 valid values

- Remove log error's outlier which value is bigger than 0.33 or smaller than -0.33
- Create a new feature 'month'
- Replace NaN-value with median
- Scaling feature's values using MinMaxScaler
- principal component analysis, Number of components is 23
- gradient boosting regressor ( n_estimators: 300, max_depth: 4, min_samples_leaf: 200, learning_rate: 0.05)

For this model, we can get

training set MAE value: 0.0491506319324

testing set MAE value: 0.0496774807937

The MAE of testing set is close to MAE of training set, which indicate good robustness of this model.

For predicting future log error, feature 'month' need be set value. The project plan to predict October's, November's and December's log error in 2017. For October, feature 'month' would be 22. For November, feature 'month' would be 23. For December, feature 'month' would be 24.

There are ten predictions below.

| | Parceled | Oct | Nov | Dec |
|---|---|---|---|---|
| 0 | 10754147 | 0.008694 | 0.008694 | 0.008694 |
| 1 | 10759547 | 0.01193 | 0.01193 | 0.01193 |
| 2 | 10843547 | 0.007708 | 0.007708 | 0.007708 |
| 3 | 10859147 | 0.020093 | 0.020093 | 0.020093 |
| 4 | 10879947 | 0.016866 | 0.016866 | 0.016866 |
| 5 | 10898347 | 0.019953 | 0.019953 | 0.019953 |
| 6 | 10933547 | -0.01465 | -0.01465 | -0.01465 |

| | | | | |
|---|---|---|---|---|
| 7 | 10940747 | 0.025929 | 0.025929 | 0.025929 |
| 8 | 10954547 | 0.021731 | 0.021731 | 0.021731 |
| 9 | 10976347 | 0.022062 | 0.022062 | 0.022062 |

Depend on parameters what have be choose before, the MAE value of benchmark model is 0. 0500034335105. It's slightly bigger than MAE value of GradientBoostingRegressor. It means that GradientBoostingRegressor is better than DecisionTreeRegressor in this case.

# Conclusion

## 5.1 Free-Form Visualization

Fig 4 shows amount  of non – NaN value of features. It indicate that some feature without enough information. For example: 'fireplaceflag'.

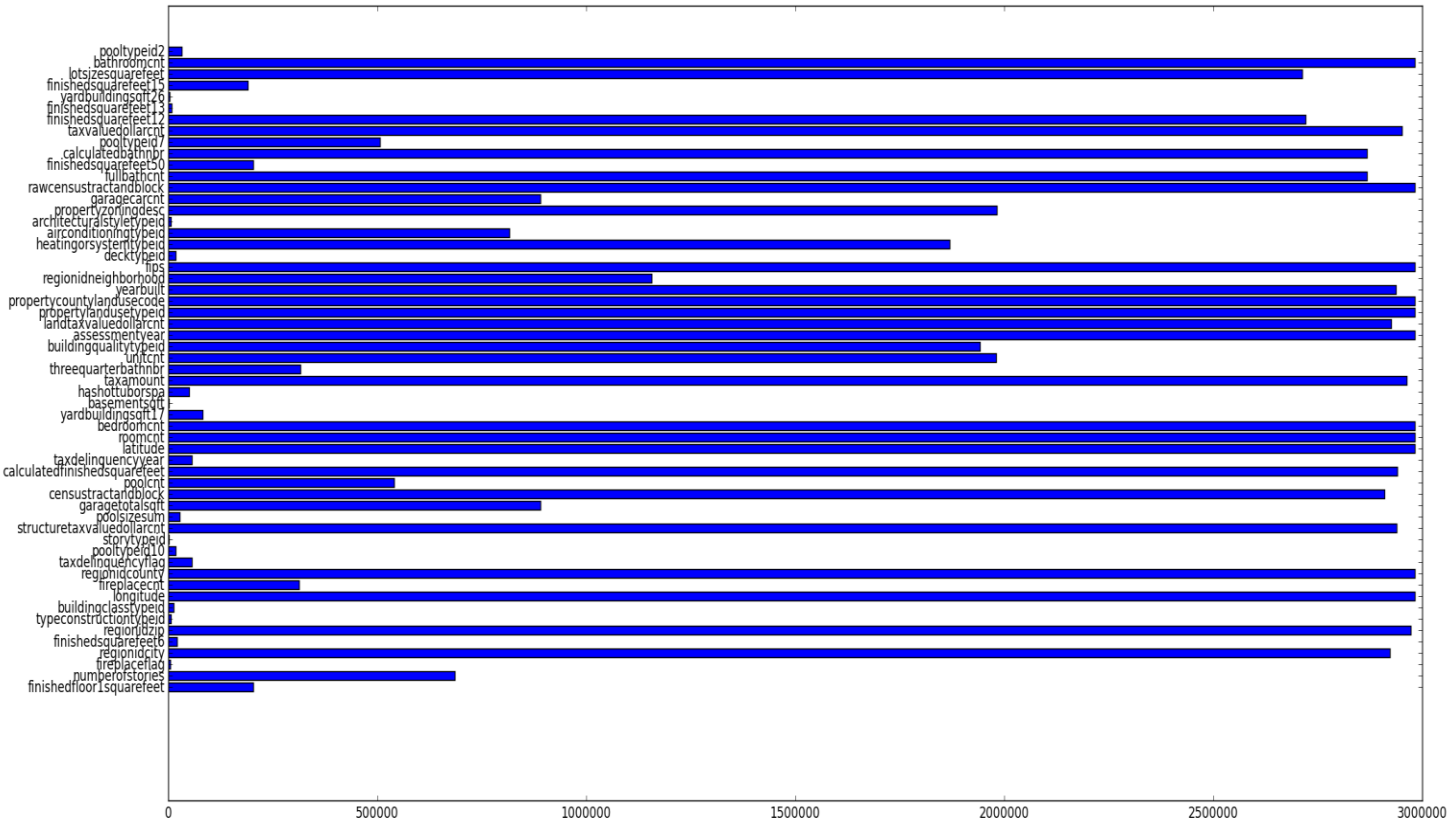



Fig4  Amount of non-NaN values

There are two figure that shows log error after remove values (> 0.33 or < -0.33). Compare with Fig 2 and Fig 3, obviously, a lot of outlier has been removed.

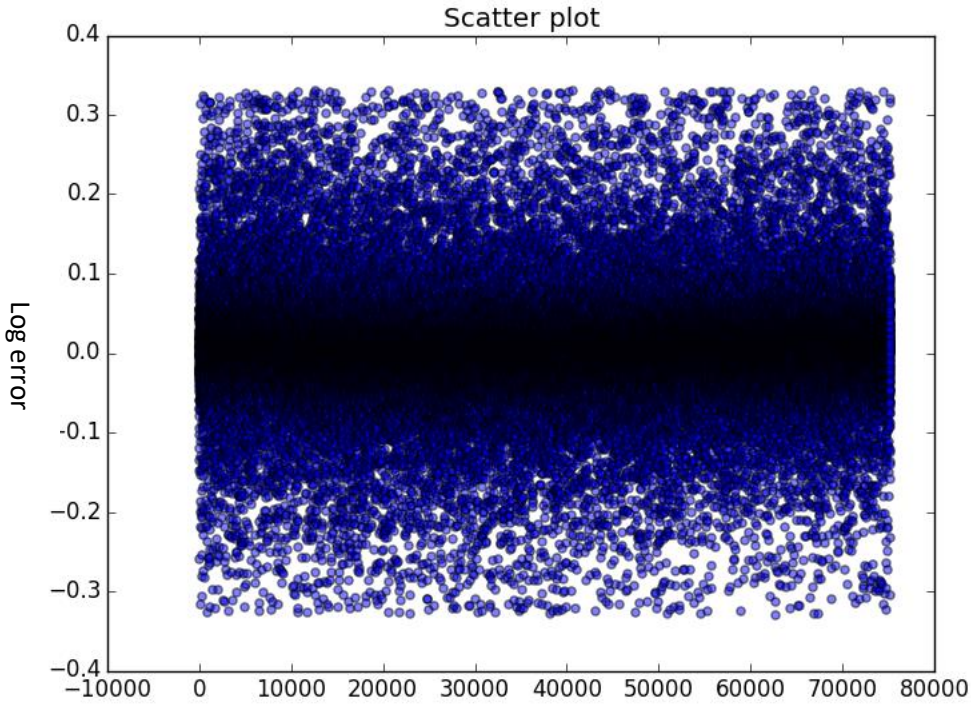



Fig 5

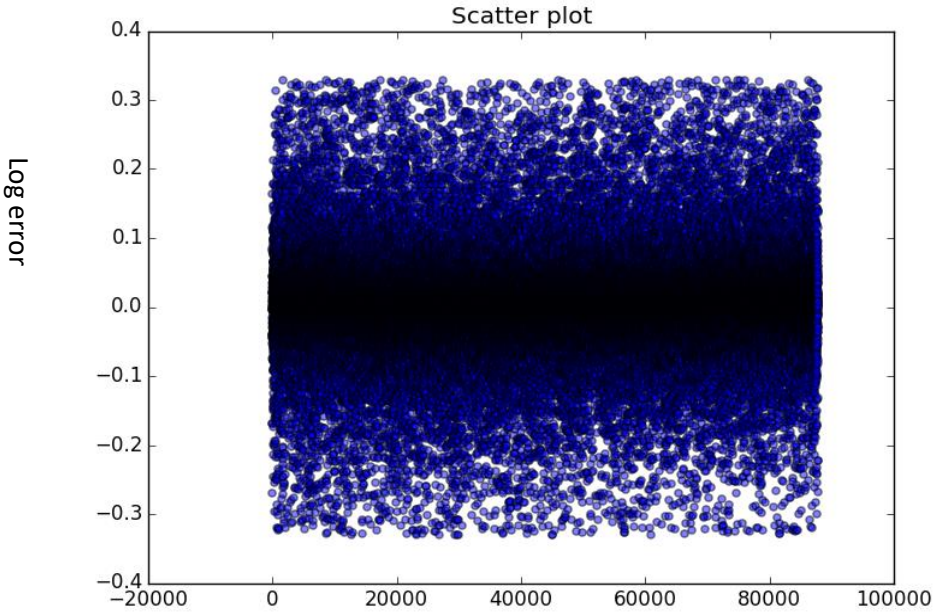
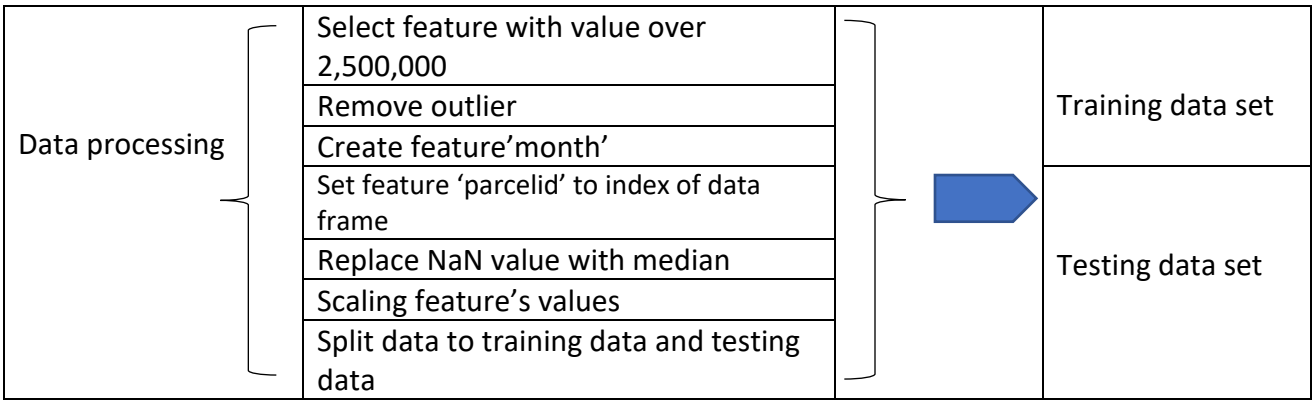
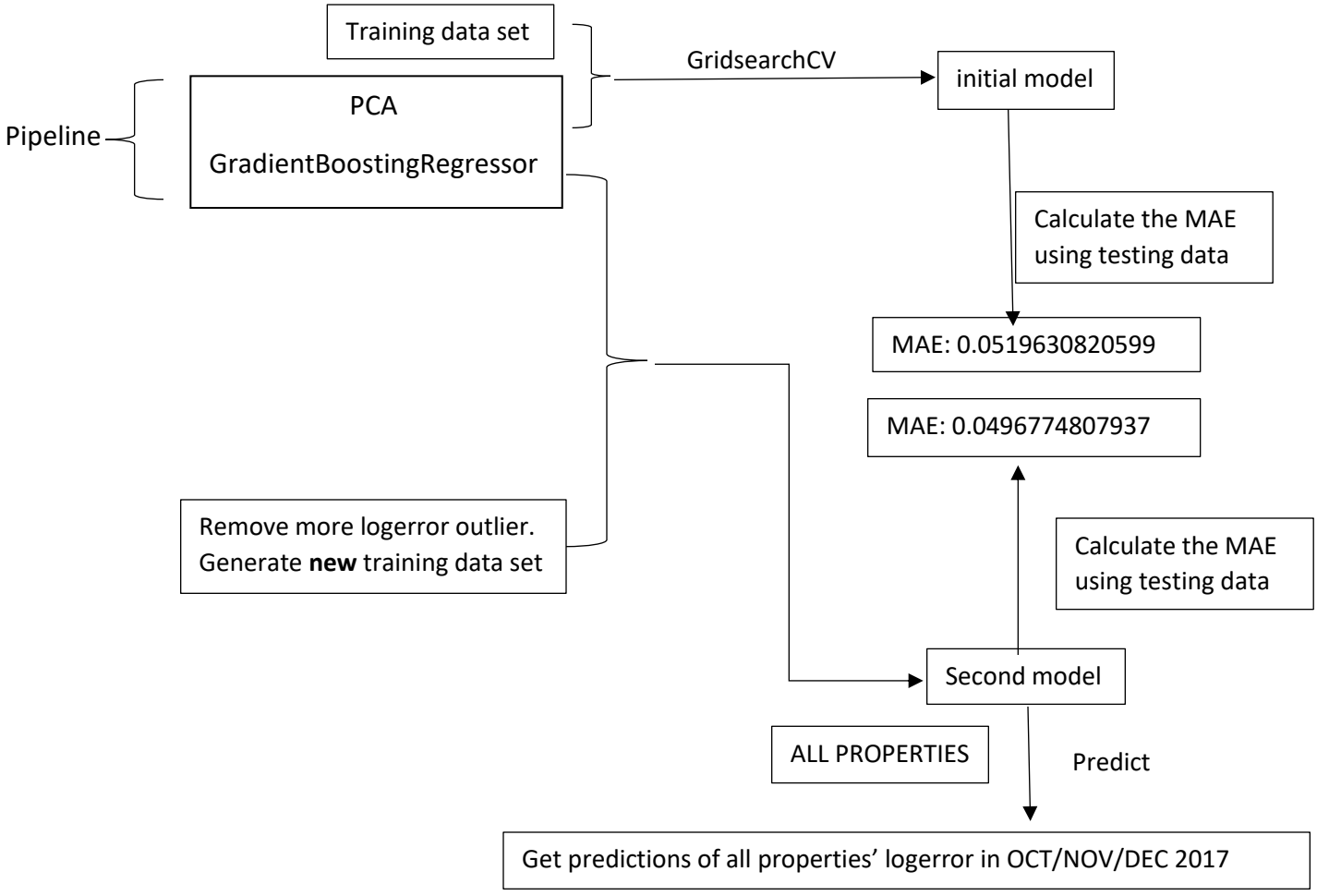



Fig 6

# 5.2 Reflection

The project have four main stages. There are data processing, training, evaluation and prediction.



The model have trained gradient boosting regressor algorithm of properties with 163038 properties. The MAE of test data is 0.0496774807937. We predict future log error using this model.

## 5.3 Improvement

Even though the results are promising. Another data could be used in training. It's interest rate. It's a common belief in real estate that house prices are correlated to interest rates. The idea, beloved by homebuyers, is that if mortgage rates rise, prices of homes for sale must fall because otherwise those homes will become less affordable. In this year, interest rate has been risen twice. It's not constant. We plan to consider it in future work.

In addition, we could add cross-validation splitting strategy. There is a parameter 'cv' in GridSearchCV . Parameter 'cv' is cross-validation generator. It would help model more better.

# Reference

[1] Jiaoyang Wu. "Housing price prediction using support vector regression" http://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1540&context=etd_projects (May,2017)

[2]Aaron NG."Machine learning for a london housing price prediction mobile application"

http://www.doc.ic.ac.uk/~mpd37/theses/2015_beng_aaron-ng.pdf (2015)

[3]

http://scikitlearn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html