Final Project CSYE 7374 Report
Prediction of the Zillow's Home Value

Group_4 Member

Peichun Tseng, Pinho Wang

Key Words

XGBoost Regression, Linear Regression, Decision Trees, Random Forest, Gradient Boosted Regressor

Background

House is usually the most important and expensive thing being purchased in a person's life. However, people can only know the estimated price rather than the market price until they purchased it.

Goals

In this project, we used the Zillow housing dataset in Kaggle dataset and we develop different algorithms that can make prediction about the market sale prices with their estimated prices. We will utilize the parallel techniques to reduce the computing time of training process. Finally, the result will be compared in the conclusion.

Methodology

We measure the model accuracy by Mean Absolution Error (MAE) between the predicted log error and the actual log error. The definition of both MAE and log error are as below:

The MAE

$$\boldsymbol{MAE = \frac{\sum_{i=1}^{n}|yi - xi|}{n}}$$

The log error

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

Analysis

Data Exploration

In Zillow housing dataset, it provides a full list of real estate properties in three counties (Los Angeles, Orange, and Ventura, California) data in 2016. In this dataset, there are two separately files, one, **properties_2016.csv**, is the mainly csv file which has 2,985,217 of rows data and 58 different fields. The other, **train_2016_v2.csv**, is the corresponded log error value, as the labels.

As we explored the dataset, we found some fields have almost 90% of null value, Fig. 1, for these fields, we replaced the null value as its mean value for the following analysis.

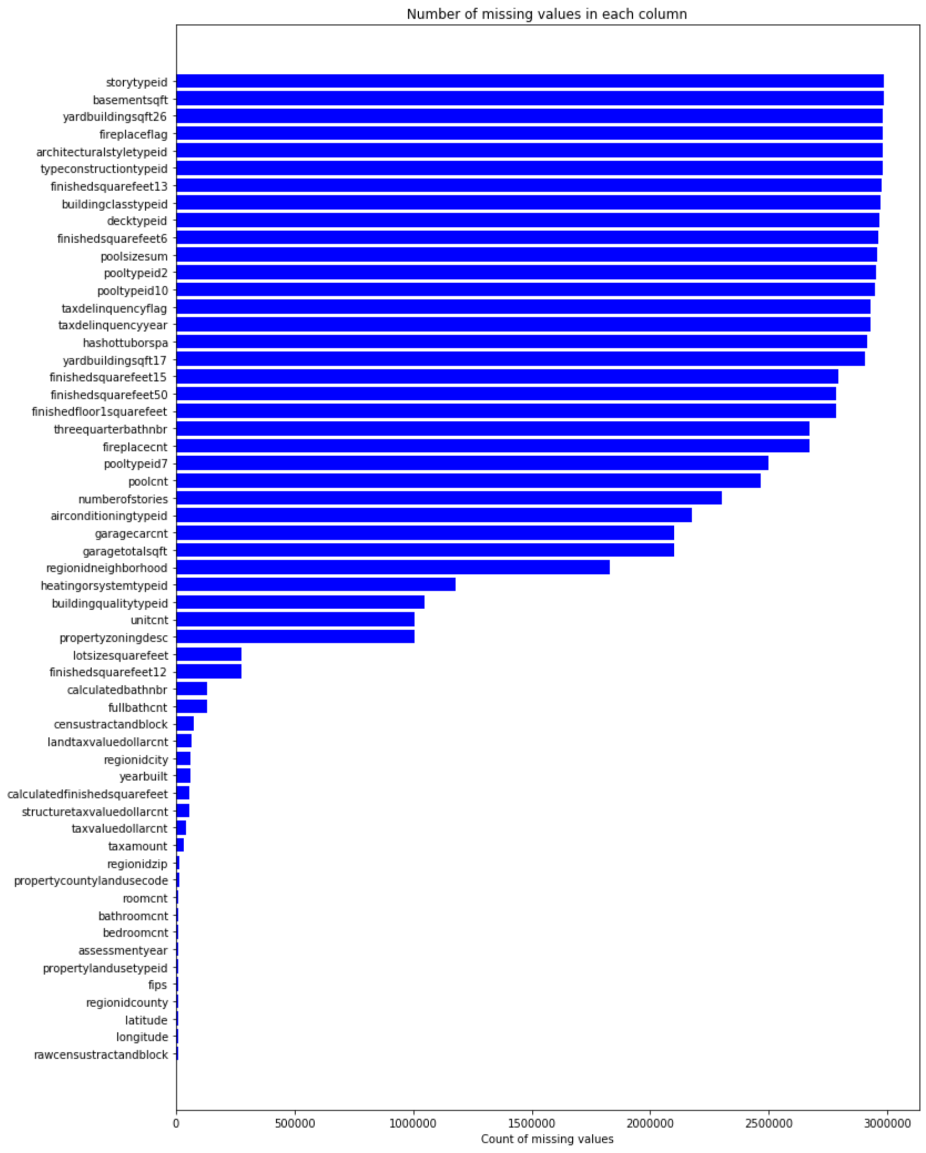


Fig. 1 Null value of the fields

Target important variable for this project is "logerror" field. So let us do some analysis on this field first. In the train_2016_v2.csv file, we found some rows have unusual log error as Fig. 2. We trimmed the outliers of these data as Fig. 3.

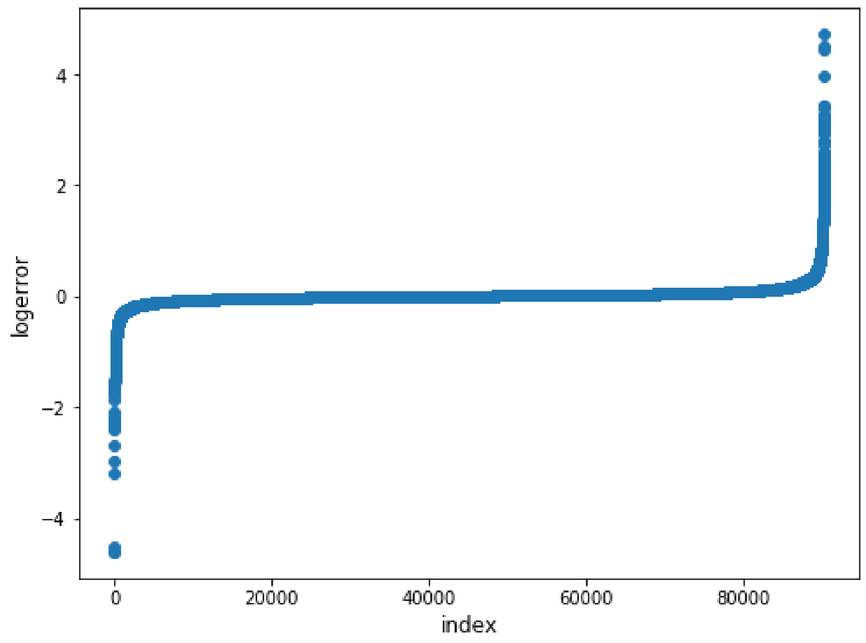


Fig. 2 Log error for dataset

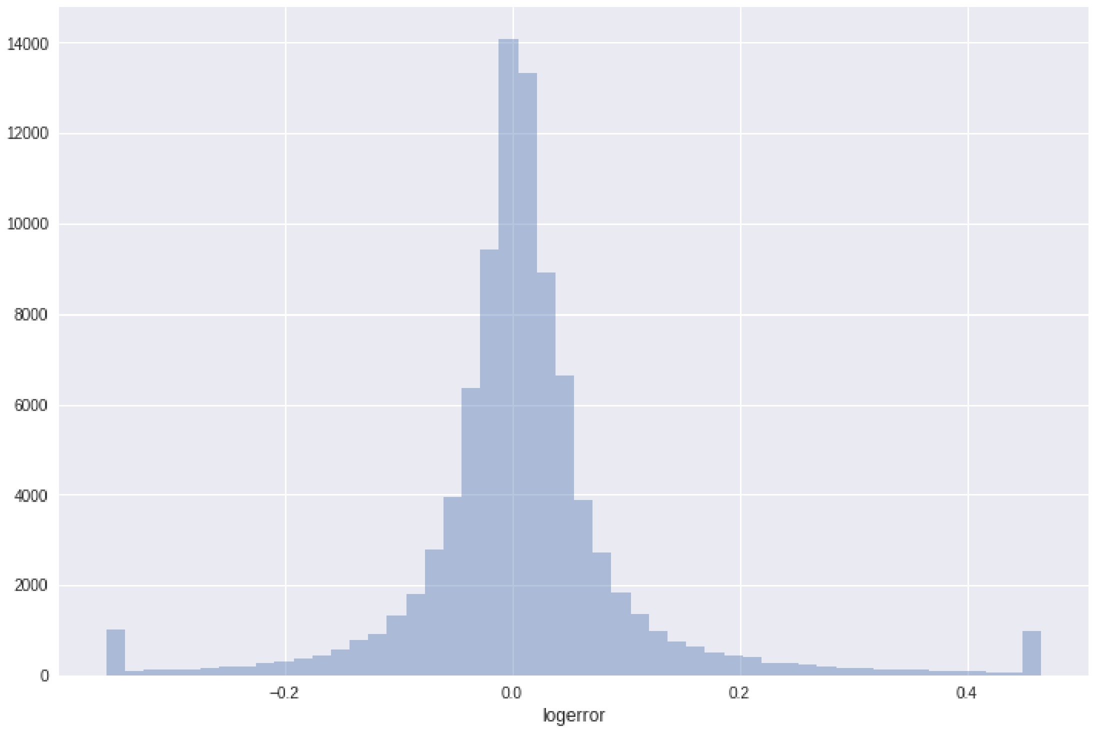


Fig. 3 Distribution of log error

As we have 58 fields, it’s not wise to use all the fields because it may cause the overfitting problem. So, we chose the float data type and calculate the correlation coefficient with the target variable, as Fig. 4.

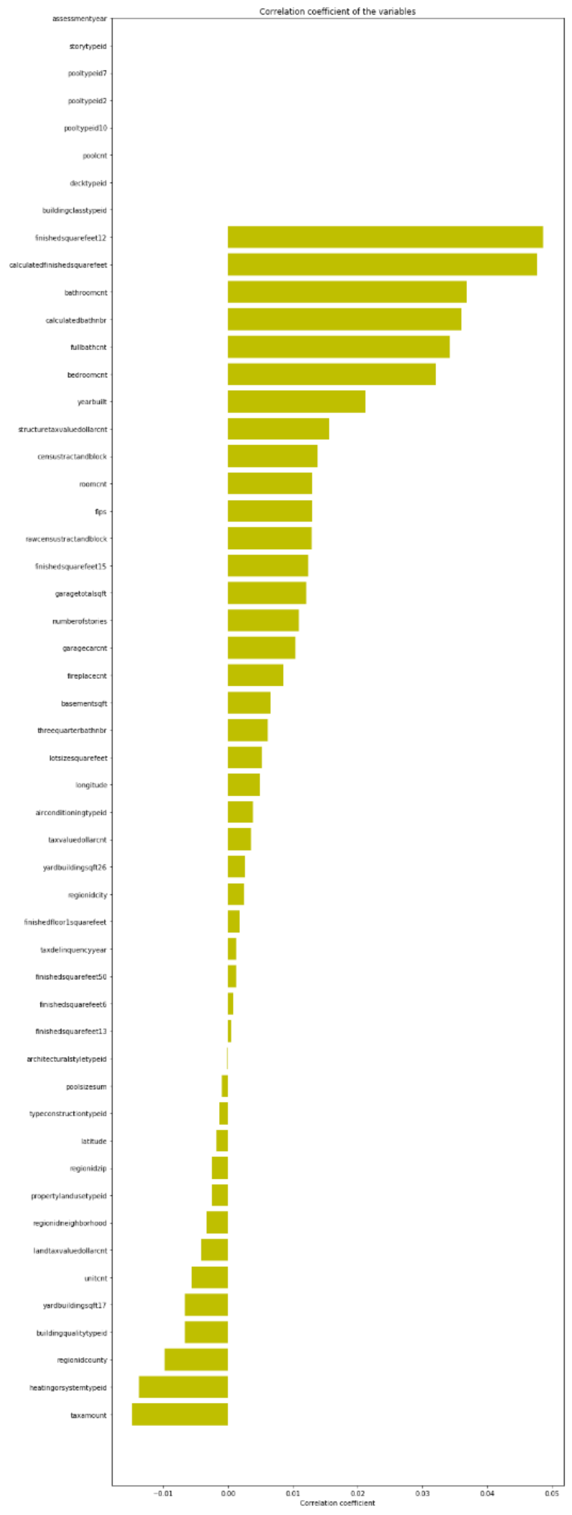
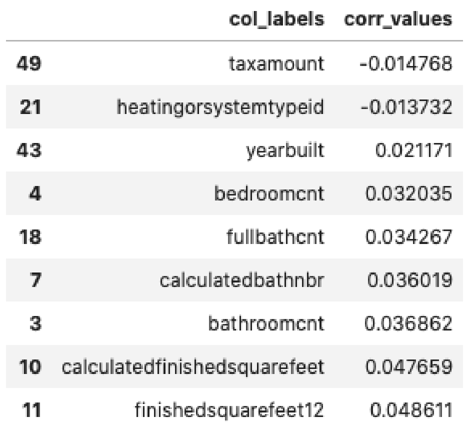


Fig. 4 Correlation between Variables and Target Variable

We chose the top 9 fields with high correlation values like Fig. 5.



	col_labels	corr_values
49	taxamount	-0.014768
21	heatingorsystemtypeid	-0.013732
43	yearbuilt	0.021171
4	bedroomcnt	0.032035
18	fullbathcnt	0.034267
7	calculatedbathnbr	0.036019
3	bathroomcnt	0.036862
10	calculatedfinishedsquarefeet	0.047659
11	finishedsquarefeet12	0.048611

Fig. 5

And investigated top 9 fields' correlation between each other, show as Fig. 6.

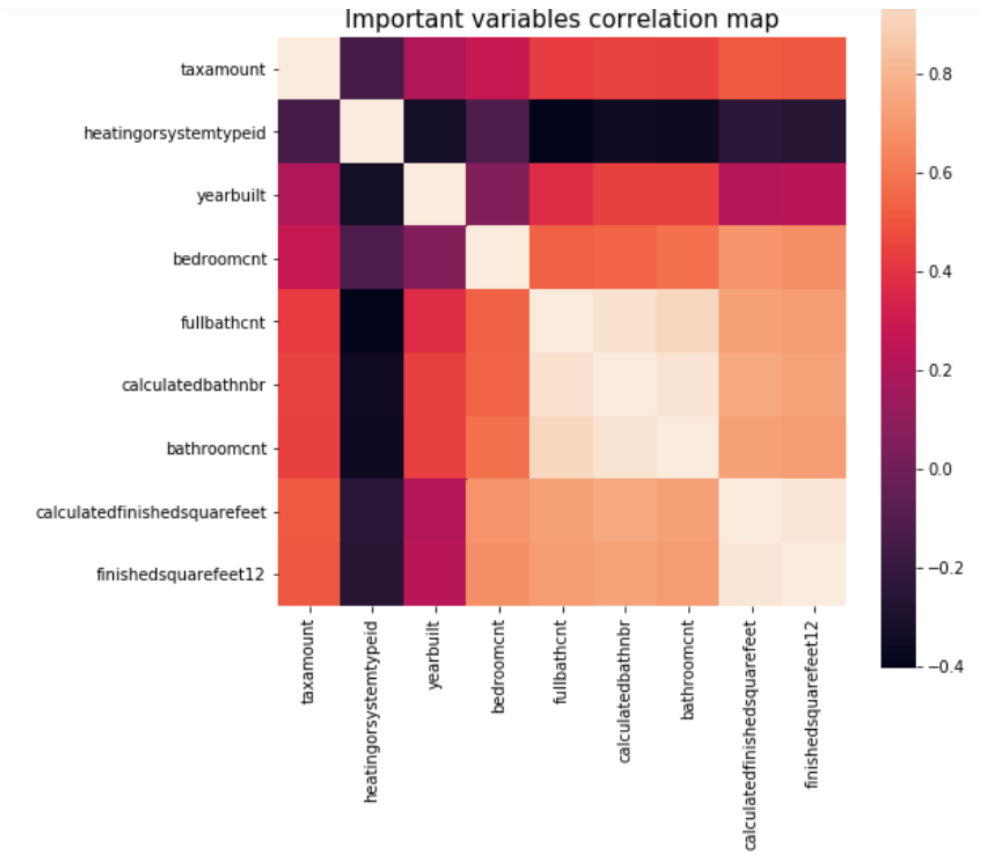


Fig. 6 Important Variables Correlation Map

The important variables themselves are very highly correlated, so that, we investigated few of them. The distribution of target variable with variables, **finishedsquarefeet12** and **calculatedfinishedsquarefeet**, are plotted as Fig. 7(a) and Fig 7(b), respectively.

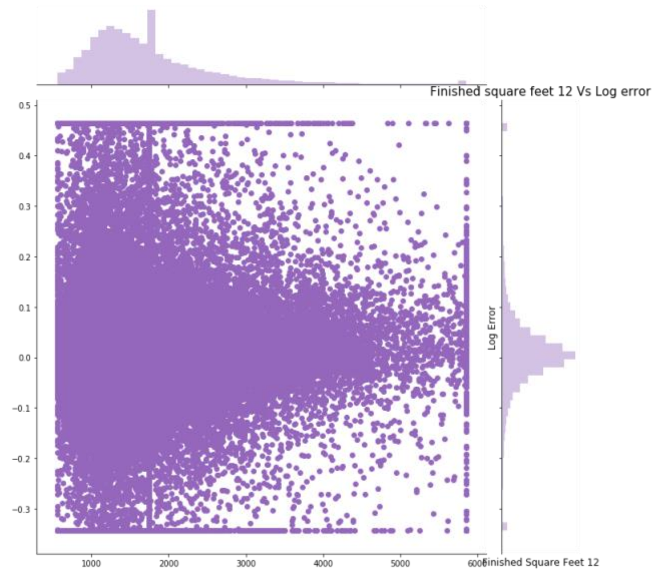


Fig. 7(a) Distribution of finished squarefeet 12.

It seems the range of logerror narrows down with increase in finished square feet 12 variable. Probably larger houses are easy to predict?

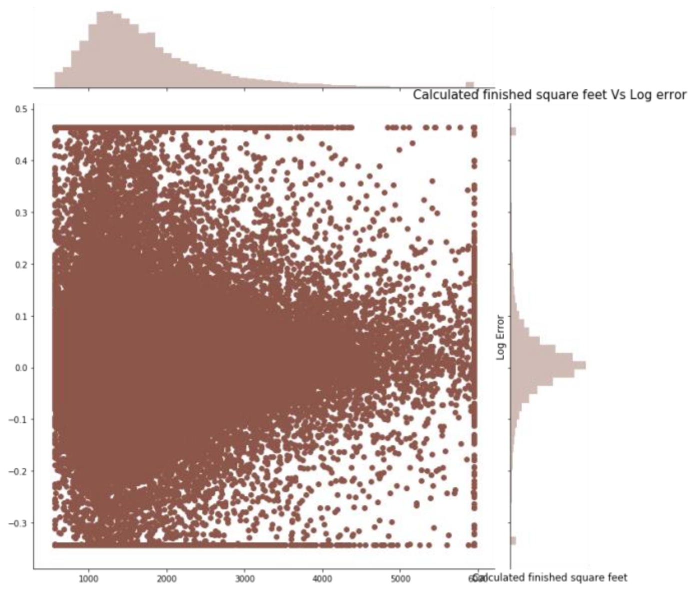


Fig. 7(b) Distribution of calculated finished squarefeet.

For the variables with non-float value, such as number of bedroom, number of bathroom, we calculated their number. As for number of bathroom and bedroom, the frequency is shown as Fig. 8(a) and Fig. 8(b), respectively. It turns out most of the houses have 2 bathrooms and 3 bedrooms.

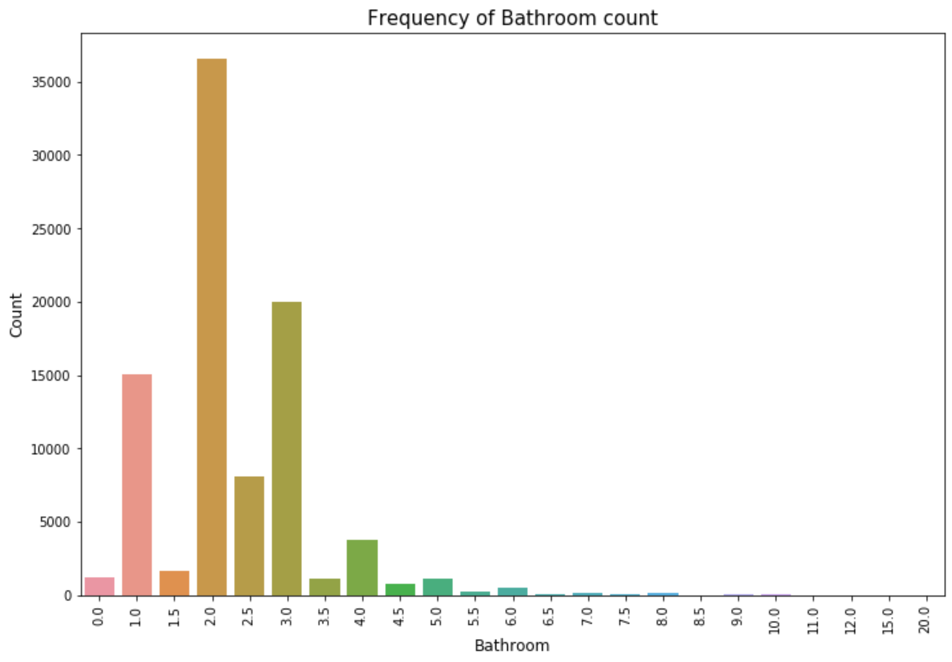


Fig. 8 (a) Number of Bathroom

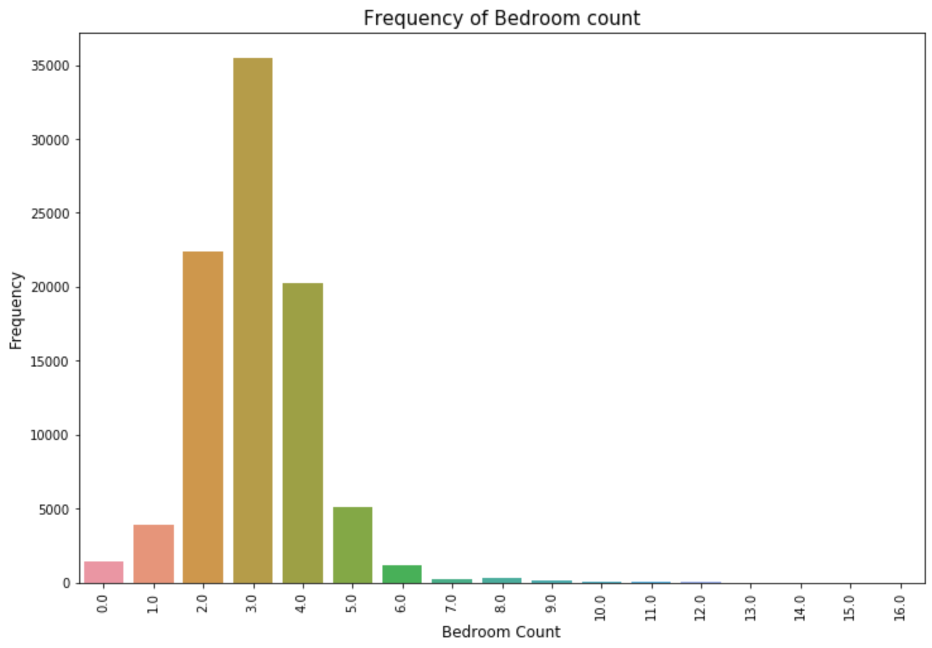
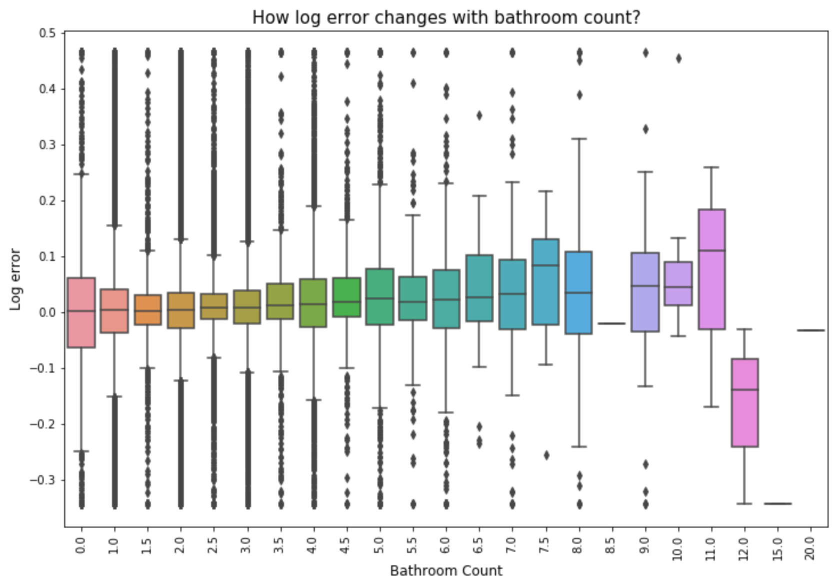


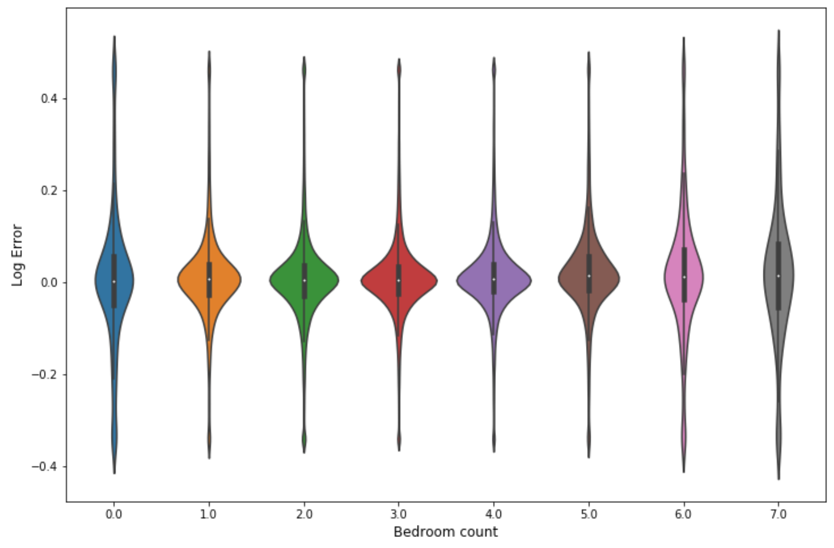
Fig. 8 (b) Number of Bedroom

Then, we investigated the relation of these two variables between target variable. We found the mean value of bathroom, 2.27, has a most significant change of log error, shown as Fig. 9(a). As for the bedroom, the mean value is 3.03, shown as Fig. 9(b). We used these mean value to replace the null values in these two fields.

Fig. 9



(a) The Distribution of Log Error for Bathroom



(b) The Distribution of Log Error for Bedroom

There are the location fields in our dataset, it's important to know whether the locations of the houses are making sense, we plotted the locations by their latitude and longitude as Fig. 10. Obviously, most of the locations are located around the downtown of LA, just few located in eastern side, which makes sense because there are two airports nearby eastern of LA, which may have less houses for selling.

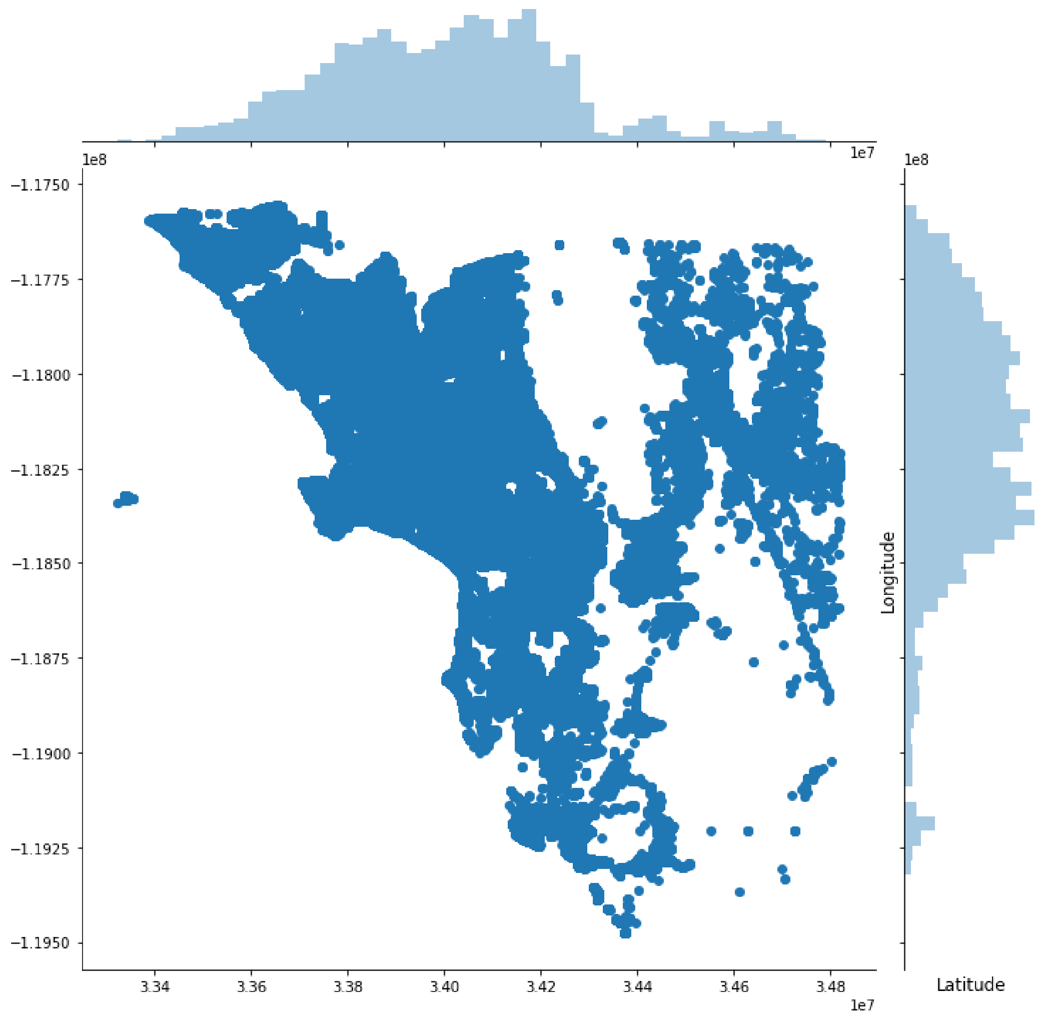


Fig. 10 The Location of Houses in Dataset

It turned out most of these fields have a non-linear relation with the target variable. So, we used couples of models to approach the higher accuracy and furthermore, using parallel techniques to accelerate the training process.

Feature Engineering

As we have lots of null values in our dataset, it is a great way to do feature engineering in order to avoid underfitting problem. We have about 90,275 rows in train but we have about 2,985,217 rows in properties file. So we merge the two files and then carry out our analysis. We also added three new features via our original fields.

First, we inserted a new field called, **New_LivingAreaProp**, it's a proportion of living area, which we used calculatedfinishedsquarefeet divided by lotsizesquarefeet. It means the numbers of lots for the house.

Secondly, we inserted new fields called, **New_zip_count**, which is the count of its related zip area.

Third, we inserted new fields called, **New_city_count**, like the previous one, it is the count of its city.

After we finished the data exploration and feature engineering, we can start to build our model, which is shown in next section.

Algorithms and Techniques

We chose five different machine learning algorithms and implemented each of them into our problem. The experiment and the result will be shown in next section. In this section, we briefly introduced the methodologies we used as below:

Linear Regression

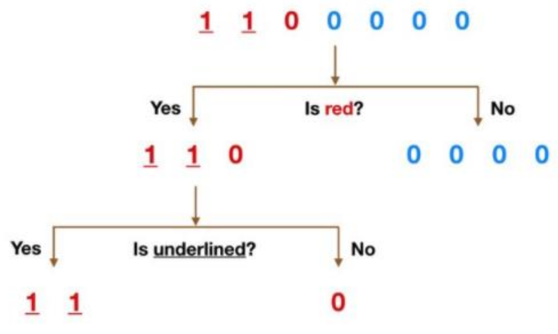
Although our problem may not have linear relation with the target variable, we first tried the most basic algorithm, linear regression, to see the output in the first place.

In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression.

Decision Tree

We tried to use a non-linear model to approach our problem, the first simplest model is Decision Trees.

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes).



Simple Decision Tree Example

Obviously in real life our data will not be this clean but the logic that a decision tree employs remains the same.

Random Forest

In order to improve our decision tree model, we can use ensemble learning method to make our output more reliable, we chose Random Forest to improve.

The random forest is a classification algorithm consisting of many decision trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

The prerequisites for random forest to perform well are:

1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.
2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

Gradient Boosted Regressor

Instead of randomly choose the subset of features, we use boosting to alter the evaluation criteria based on feedback from the previous results, furthermore, the gradient descent algorithm is imposed on it, which known as Gradient Boosting.

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

Boosting algorithms feature:

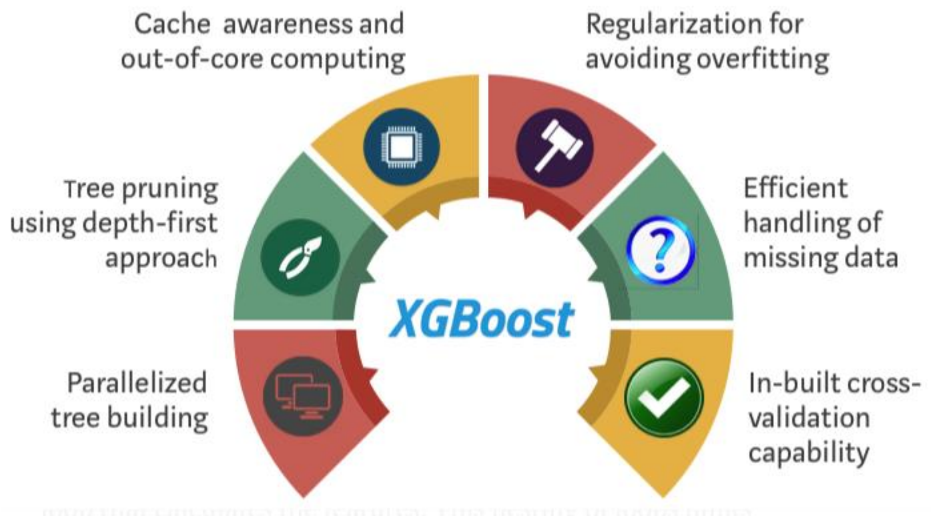
1. All observations in the dataset are initially given equal weights.
2. A machine learning model is trained on this dataset.
3. The model from step 2 is added to the ensemble with an error associated with it that is the sum of the weights for the misclassifed observations.
4. Misclassified observations are given greater weights.
5. Steps 2–4 are repeated for a number of epochs.

This method is named gradient boosting as it uses a gradient descent algorithm to minimise loss when adding models to the ensemble. Subsequent models are fit to pseudo-residuals instead of adjusting weights. Stochastic gradient descent (as used by XGBoost) adds randomness by sampling observations and features in each stage, which is similar to the random forest algorithm except the sampling is done without replacement.

XGBoost Regression

Finally, we used so called 'Extreme Gradient Boosting', XGBoost, to evaluate our result, which has optimized the Gradient Boosting in many ways.

XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners (CARTs generally) using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.


Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient handling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

XGBoost is a perfect combination of software and hardware optimization techniques to yield superior results using less computing resources in the shortest amount of time.

Experiments & Results

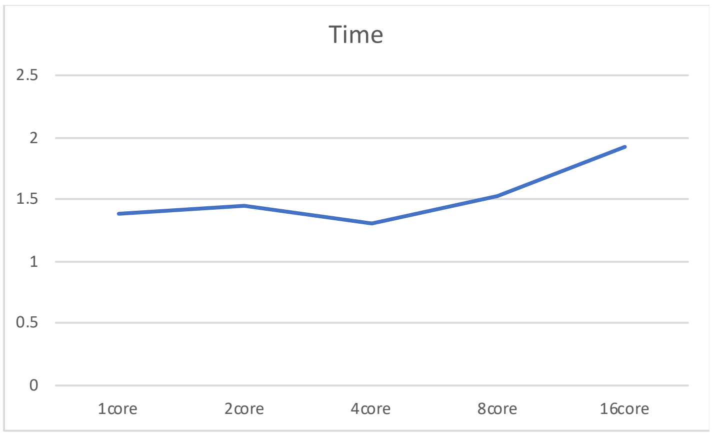
In this section, we investigated the five algorithms in last section and evaluated the loss function of each of them. We utilized the Mean Absolution Error (MAE) as the metric to evaluate the model. Furthermore, the cross validation will also be involved. We used K-fold cross validation and set K equals to 10 for each model. The final score of each Regression is mean value of total MAE.

In this section, we also investigated the effect of using parallel computing. We will use 1, 2, 4, 8, 16 threads in the training process and the result will also be shown.

Linear Regression

MAE mean score: -0.05338599489034875

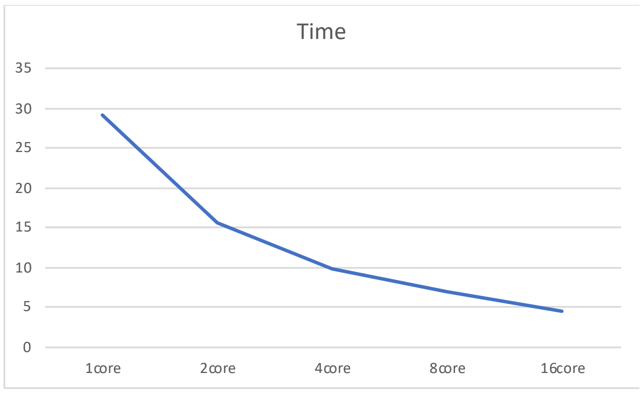
Linear Regression	Time (sec.)
1 process	1.382
2 process	1.449
4 process	**1.300**
8 process	1.523
16 process	1.930



Decision Tree

MAE mean score: -0.08412201256527355

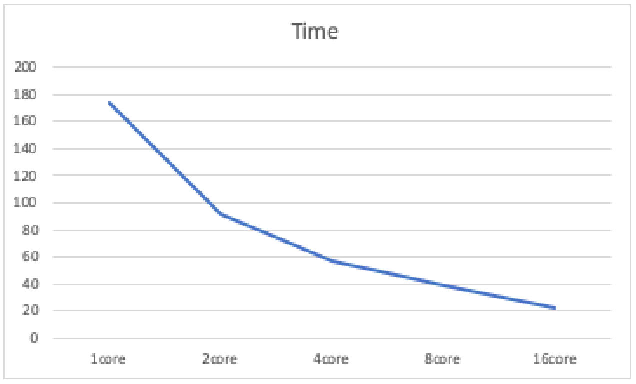
Decision Trees	Time (sec.)
1 process	29.049
2 process	15.654
4 process	9.780
8 process	7.037
16 process	**4.552**



Random Forest

MAE mean score: -0.059771056928513554

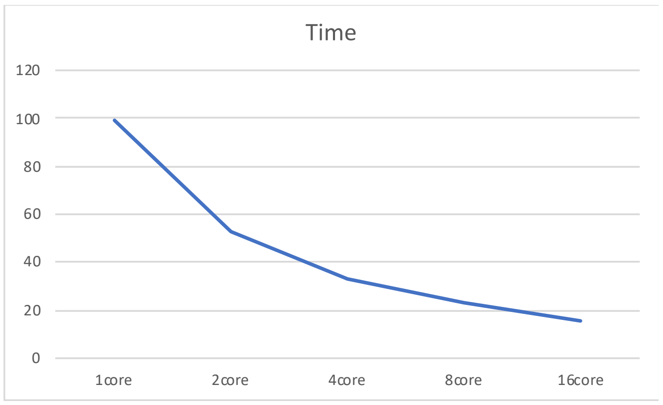
Random Forest	Time (sec.)
1 process	173.308
2 process	91.090
4 process	56.696
8 process	38.349
16 process	**21.711**



Gradient Boosted Regressor

MAE mean score: -0.05308919729188812

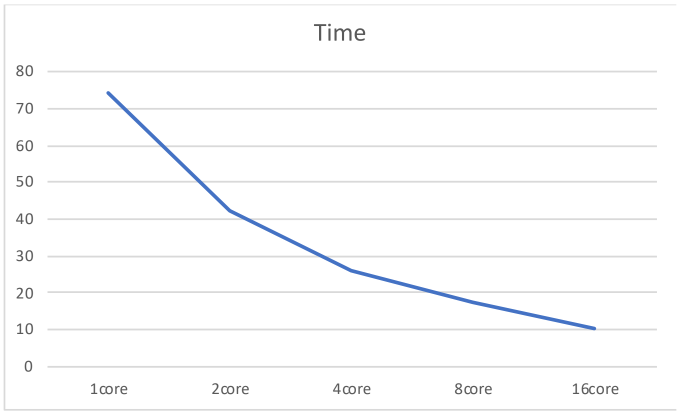
Gradient Boosted Regressor	Time (sec.)
1 process	98.920
2 process	52.391
4 process	33.000
8 process	23.311
16 process	**15.502**



XGBoost Regression

MAE mean score: -0.053039984032511714

XGB Regression	Time (sec.)
1 process	74.457
2 process	42.144
4 process	26.207
8 process	17.229
16 process	**10.347**



Results

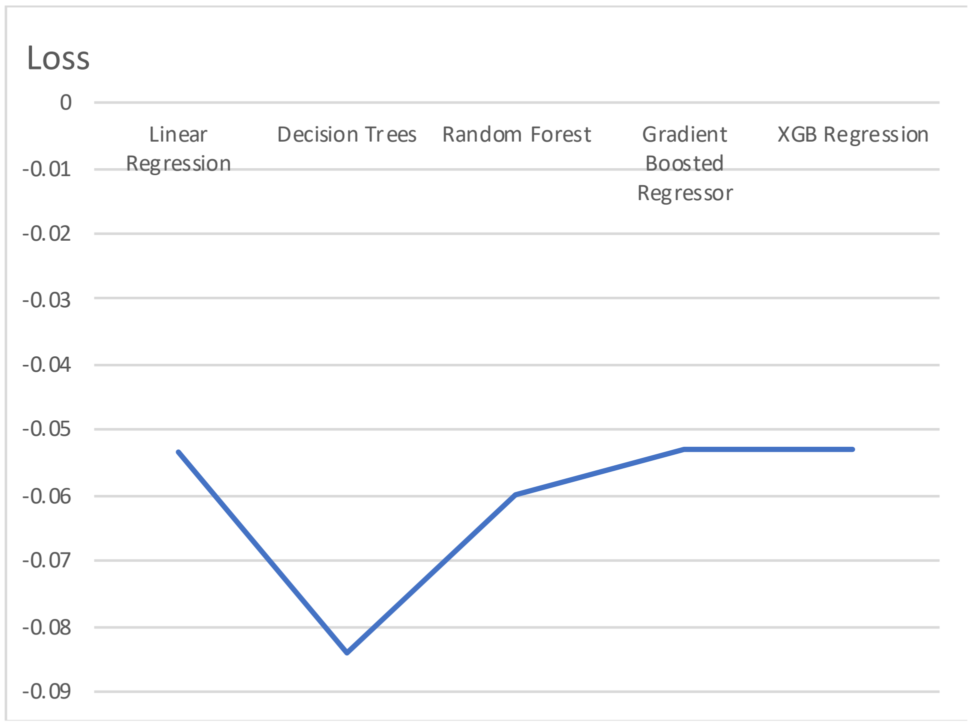
Table 1 and Table 2 are the computing time and the loss without implying the parallel process. In Table 1, we can easily say that the native linear regression used the less time compare with other ensemble and boosting methods. Although spending less time, the performance of linear regression is not so well. For these five algorithms, XGB Regression has lowest loss, which means best performance.

Table 1.

	Time (sec.)
Linear Regression	**1.382**
Decision Trees	29.049
Random Forest	173.308
Gradient Boosted Regressor	98.920
XGB Regression	74.457

Table 2.

	Loss
Linear Regression	-0.05338599489034875
Decision Trees	-0.08412201256527355
Random Forest	-0.059771056928513554
Gradient Boosted Regressor	-0.05308919729188812
XGB Regression	**-0.053039984032511714**



XGBoost performance

There are some advantages that make XGBoost perform well:

1. Parallelization: XGBoost approaches the process of sequential tree building using parallelized implementation.
2. Regularization: It penalizes more complex models through both LASSO (L1) and Ridge (L2) regularization to prevent overfitting.
3. Weighted Quantile Sketch: XGBoost employs the distributed weighted Quantile Sketch algorithm to effectively find the optimal split points among weighted datasets.

Conclusion

We carefully select the proper features of the model by investigating the correlation and the significant. By training the regression model, we need to process data first and then choose features.

First, we merge two file by parceid and find there are a lot of NaN values in the dataset. So we drop some columns that NaN values is greater than 70% and add new features by merging some features which are related with others like New_LivingAreaProp, New_zip_count and New_city_count.

The result still have too much feature, and then we get the correlation with the target variable. We choose the features with high correlation values become a new dataset for training and analyze their correlation with each. Obviously, there are some features have high correlation value with others, we trim these features from the training dataset.

We use cross_val_score function for each regression and get the MAE score. The Score calculation by the mean value of total sum of |y-y'| which y represent predicted value and y' stand for real result. In these module, we can see that XGB Regression has highest score we also train each module of different processors.

In the XGB Regression, we use cross validation method to train the model and set the CV argument by 10, but it always cost much time for training model. When the processor is 16, the time will be reduced 86% from only one processor so this is a big advantage when we use multiple processors.

References & Sources

[1]. Data source : https://www.kaggle.com/c/zillow-prize-1/data

[2]. Decision tree : https://en.wikipedia.org/wiki/Decision_tree

[3]. Random Forest : https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[4]. Gradient Boosted Regressor : https://en.wikipedia.org/wiki/Gradient_boosting

[5]. XGBoost Regression: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[6]. Linear Regression : https://www.statsmodels.org/stable/regression.html

[7]. neg_mean_squared_error : https://scikitlearn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html#sklearn.metrics.mean_squared_error

[8]. Decision tree sample : https://towardsdatascience.com/understanding-random-forest-58381e0602d2