Zillow's Home Value Prediction

Lei Zhang, Wenyu Chen, Yuheng Cai

April 15, 2018

Contributions:

- Lei Zhang: Exploration(Univariate Analysis), Neural Network
- Wenyu Chen: Summary, Boosting tree model
- Yuheng Cai: Exploration(Missing values), Ordinary Least Square Regression

1. Introduction

In 2007, Zillow launched the famous Zestimate, the free proprietary automated valuation system, which is as much censured as praised. Some people think Zestimate is a leading example of impactful machine learning, which helps Zillow become one of the largest, most trusted marketplaces for real estate information in the United States.[1] Some insiders in the real estate industry believed Zillow is not in real estate. It is a business website, which was established to lure eyeballs who want to buy or sell homes and, in turn, to sell advertising to real estate professionals, like local agents.

Zestimate analyzes data from 110M U.S. homes and the benchmark algorithms of Zestimate are based on 7.5 million statistical and machine learning models, as Zillow claimed.[2] In this project, we try to predict the $log(error)$ by implementing Neural Network, Boosting tree, and Ordinary Least Square Regression(OLS) with Principle Component Analysis(PCA). Our objective mainly focuses on comparing the efficiency of algorithms, finding important features for predicting $log(error)$, and pointing out the future direction for improving data collection and analysis.

2. Data Resource

The data set is provided by "Zillow Price" competition on Kaggle. It contains a full list of real estate properties in three counties (Los Angeles, Orange and Ventura, California) data for 2017(released on 10/2/2017). The response is $log(error)$, which is defined as the difference between log(Zestimate) and log(actual sale price). There are 58 features, which are substantial redundant. These features covers information about physical attributes(location, lot size, number of bathroom, type of air conditioning etc), tax assessments(property tax information, actual property taxes paid, tax assessors' records, etc), and prior and current transactions.

3. Data Exploration

3.1 log(error)

We first look at the distribution of our target $log(error)$:

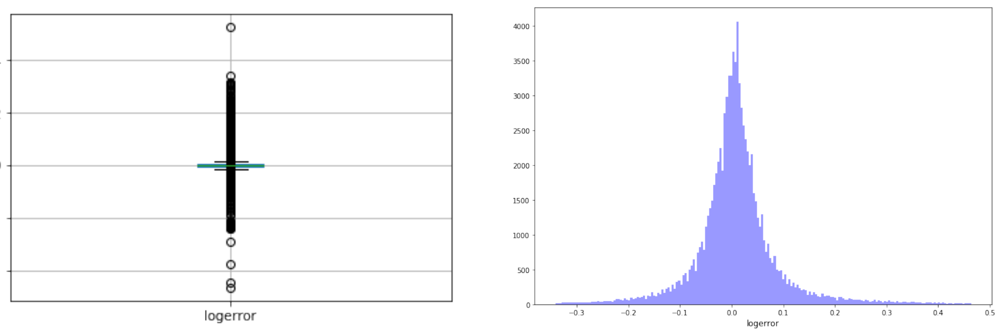

Figure 1: log(error) distribution

The left figure shows that there are some outliers on the both side of the data. After removing those outliers, the distribution which contains 98% of the data is shown on the right. We can see that $log(error)$ follows a normal distribution with a small standard deviation of 0.0833. Besides, the mean of $log(error)$ is slightly larger than zero, which means Zillow tends to slightly overestimate houses in those areas.

3.2 Features

Since there are a lot of features in the data set, to have a better understanding of their roles in our models, we plot a feature importance graph by boosting tree as shown in figure 2. Figure 3 discovers the correlation among top 20 important features. The result shows that many features are correlated, so we decide to apply neural network and boosting tree which are robust to collinearity and implement PCA in OLS.

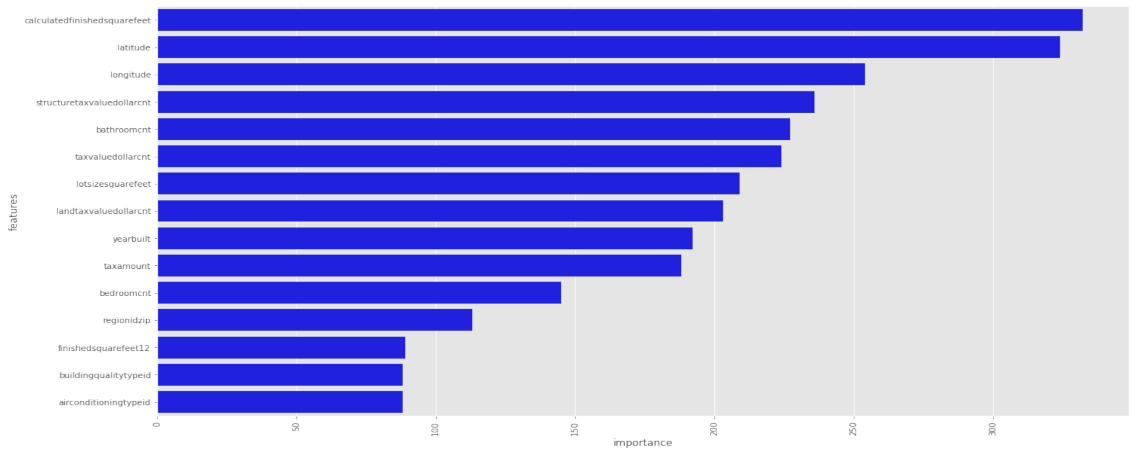

Figure 2: feature importance

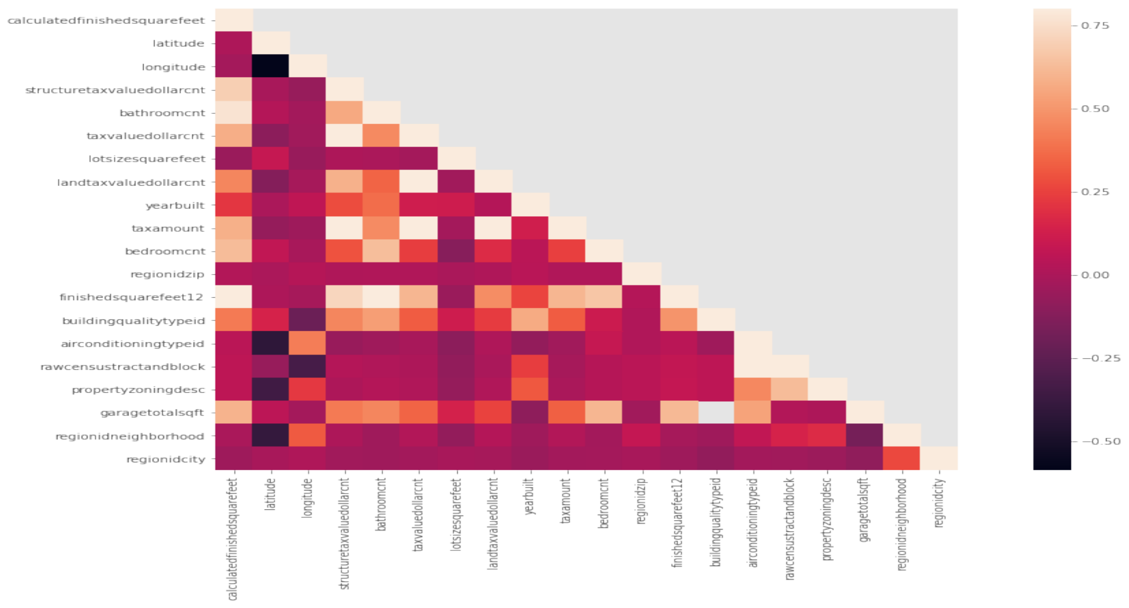

Figure 3: important features' correlation

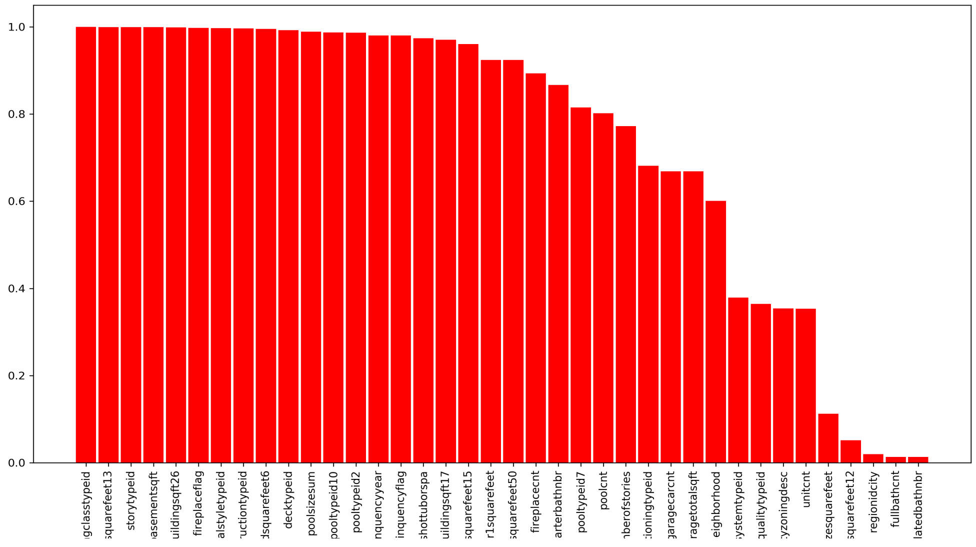

Figure 4: missing values percentage

3.3 Missing Values

By counting missing values for each feature, we found most feature contain missing values. Figure 4 shows the percentage of missing values for features, which have at least 1% of missing values in the record. However, after further exploration, we capture that the missing values of some features can be filled with zero. Most of these features describe the physical attributes of the property, for example, the feature $poolcnt$ measures the number of pools on the lot and it only has value 1 and nan. Therefore, we assume that nan actually means there are no pools on the lot and we shall fill these missing values with zero.

Besides, some features have multiple categories according to the data dictionary. However, when we check their values in data set, we find that only one category is in the record and the rest values in those features are missing. For example, there are 5 types of building classes according to the data dictionary, but only one type appears in the feature $buildingclasstypeid$. Those features contain little information and we cannot make inference about those missing values. As a result, we choose to delete these features. Table 1 shows how we handle these features.

features	description	options
heatingorsystemtypeid	Type of home heating system	fill missing value with zero
poolcnt	Number of pools on the lot	fill missing value with zero
pooltypeid10	Spa or Hot Tub	fill missing value with zero
pooltypeid2	Pool with Spa/Hot Tub	fill missing value with zero
pooltypeid7	Pool without hot tub	fill missing value with zero
poolsizesum	Total square footage of all pools on property	fill missing value with zero
basementsqft	Finished living area below or partially below ground level	fill missing value with zero
fireplacecnt	Number of fireplaces in a home	fill missing value with zero
fullbathcnt	Number of full bathrooms present in home	fill missing value with zero
threequarterbathnbr	Number of 3/4 bathrooms in house	fill missing value with zero
yardbuildingsqft17	Patio in yard	fill missing value with zero
yardbuildingsqft26	Storage shed/building in yard	fill missing value with zero
buildingclasstypeid	Overall assessment of condition of the building(only has 4 and nan)	delete feature
decktypeid	Type of deck present on parcel(only has 0.66 and nan)	delete feature
storytypeid	Type of floors in a multi-story house(only has 7 and nan)	delete feature

Table 1: handle missing values

3.4 Univariate Analysis

We apply univariate analysis to all important features, and select 3 features to comment some interesting facts.

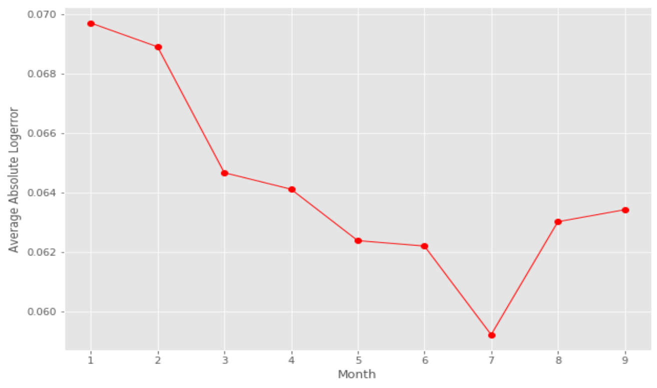

Figure 5: $log(error)$ vs transaction date

Figure 5 describes how does absolute value of $log(error)$ change with time. We choose absolute values here since a small absolute value of $log(error)$ means the Zestimate prediction are close to the sale price. One thing we have noticed is that the mean of $log(error)$ among May to August are lower than the one in other months. It could be caused by seasonal effect, or it can be a drawback in Zestimate's benchmark algorithms, which worth the effort for future research.

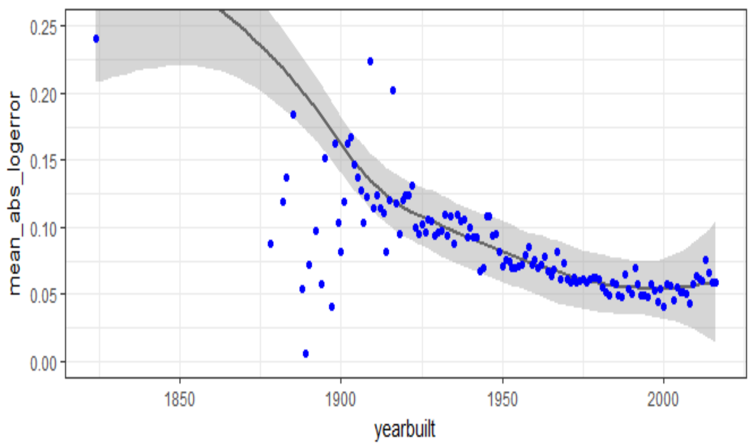

Figure 6: build year vs abs(log(error))

Figure 6 shows the relationship between absolute value of $log(error)$ and $yearbuilt$. There is a trend which indicates older buildings have larger error.

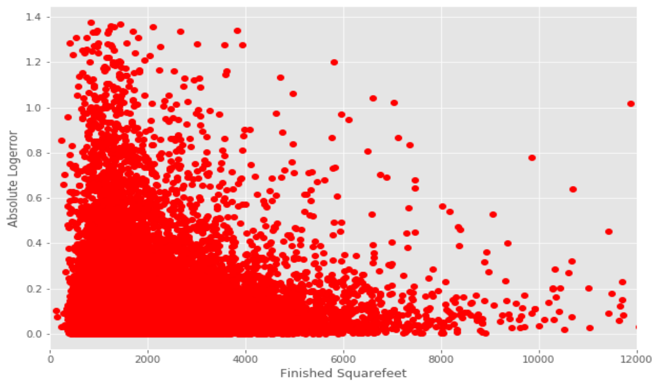

Figure 7: finished square feet vs abs(log(error))

Figure 7 shows the relationship between absolute value of $log(error)$ and $calculatedfinishedsquarefeet$. It turns out that lager houses are more easily to accurately estimate by Zillow. Our interpretation is when consumers sacrifice total living area, the feature importance ranking will varied from different groups, for example, school location is more important for parents than others, and Zillow does not capture these changes well, which causes the higher variance when estimating small houses.

4. Experiment Results

4.1 Ordinary Least Square Regression

4.1.1 Introduction

Compared with Neural Network and XGBoost, OLS has the least complexity. Even though OLS can only learn a linear combination of input features, while Neural Network can learn any functions of input features including non-linear functions and XGBoost has to learn all the tree sequentially, we do not need to worry about the risk of overfitting. Therefore, it is meaningful to include OLS into our toolkit to compare with other models.

4.1.2 Data Processing

After the general feature engineering method stated before, we make some further modification for OLS:

1. Instead of imputing the missing values of continuous variable with 0, we impute the mean values into missing values of $buildingqualitytypeid$, $lotsizesquarefeet$, $finishedsquarefeet12$, $fullbathcnt$, $calculatedbathnbr$, $calculatedfinishedsquarefeet$. That is because we believe that regression is not as robust as the other methods, which requires more careful choice of the value.
2. Split $transactiondate$ into 3 columns: date, month and quarter.
3. Turn all categorical features except $propertyzoningdesc$ and $transactiondate$ into dummy features. The reason to exclude $propertyzoningdesc$ is because it has more than 2000 unique values. Additionally, there are some other features which contains the location-relative information as $propertyzoningdesc$ do.

4.1.3 Principle Component Analysis

After getting the dummy variables, we have 794 features in total. In order to handle the serious collinearity among these features, we decide to implement PCA to the dataset. First, figure 8 is the scree plot and we find 'elbow' at PC3. However, the result of RMSE is not as favorable as we expected.

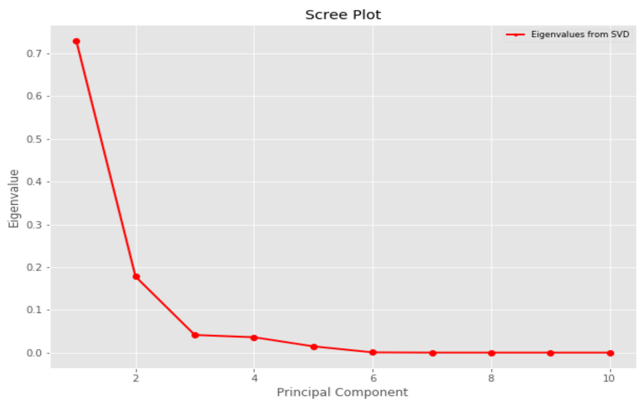

Figure 8: scree plot

Therefore, we make a plot of RMSE versus the number of principle components and find the lowest CV-error at PC75 which is 0.1752096 as shown in figure 9. The reason that RMSE goes up is it includes some noise.

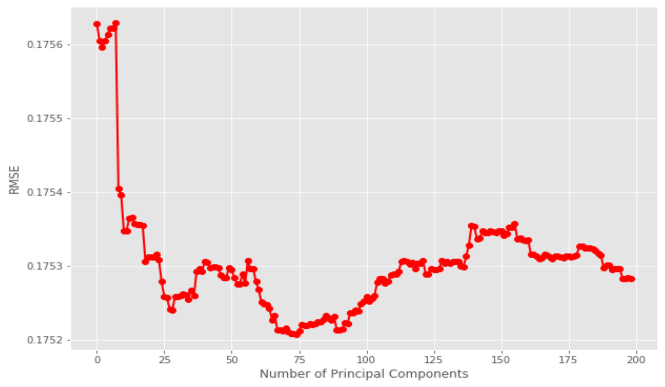

Figure 9: RMSE versus the number of PCs

4.1.4 Result

Our test error is 0.1753167. Figure 10 shows the prediction result and the true value for all point. We find that the prediction result always has smaller absolute value than the real value, which indicate that there is still a large room for improvement.

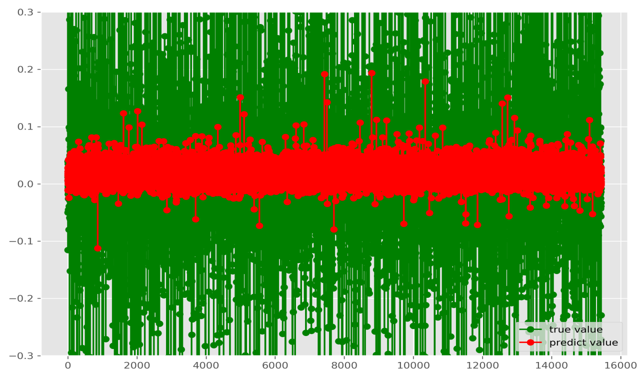

Figure 10: True Value vs Predicted Value

4.2 Neural Network

4.2.1 Introduction

The supervised neural network is developed from a set of known features and the ultimate goal of the model is to estimate the predicted output values with the observed under the 'supervision' of the training process. It has tremendous advantages over other methodologies in discovering complex nonlinear relationships regardless of dependency among explanatory variables. We choose to implement a multilayer feed-forward network to predict the $log(error)$.

4.2.2 Model Selection

One of the challenges for implementing neural networks is to capture the optimal architectures of the network, and we have to make the trade-off between complexity and the computational burden.
Moreover, the literature suggested us to use Adam as the solver.[3][4] Table 2 shows the RMSE of 12 different trials. The lower RMSE that the architecture has represents a stronger predictive power. From the table the logistic activation function with 5 hidden layers and higher number of nodes in each layer has the lowest RMSE.

activation function	relu	identity	logistic	tanh
(100,80,50,30)	0.3453608	0.3494683	0.1706238	0.1720931
(200,160,100,60)	0.1800953	0.1842591	0.1708820	0.1707759
(200,160,70,35,20)	0.1745636	0.1758827	0.1704358	0.1722257

Table 2: RMSE of Neural Network models

Figure 11 confirmed our finding from the table. After the literature research and the result from the trials with cross validation, we decided to choose the logistic function as the activation function, and the architecture that has 5 hidden layers with 200,160,70,35,20 nodes in each layer was finalized for future analysis.

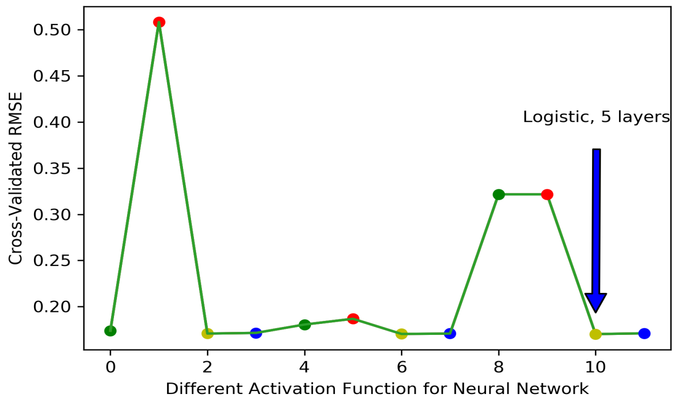

Figure 11: Mean of RMSE Neural Networks with cross validation

4.2.3 Analysis

One thing we noticed is that 5 hidden layers slightly improve the results, while increased the number of nodes on each layer significantly reduced the RMSE. This can be a data point for future research in supervised neural networks, when they face the trade-off between layers and nodes.

More than 98 percents of test data have $log(error)$ ranged in (-0.4, 0.4). Figure 12 shows the plot for predicted values versus the true values based on the test data set. Our test error is 0.1769576.

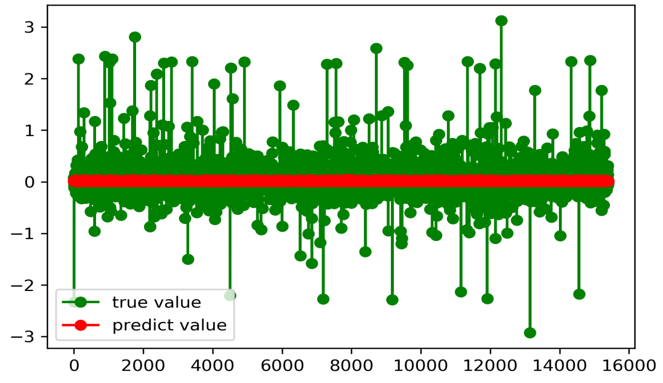

Figure 12: True Value vs Predicted Value

4.3 Boosting tree

4.3.1 Introduction

In this section, we will use boosting tree model to predict the $log(error)$. Boosting tree is an ensemble model, which use decision tree as its basic model and learns the residuals of previous models at each iteration.[5] Since we are trying to reduce bias at each iteration, boosting tree is easy to be overfitting. Therefore, it is important to control the model complexity.
Here we choose XGBoost model, which is an improved boosting tree model and widely used by data scientists to achieve state-of-the-art results on many machine learning problems.[6]
Generally speaking, XGBoost has following improvements compared with traditional boosting tree model. Firstly, XGBoost has a regularization term in its loss function to control the complexity of the model in case of overfitting. The loss function is:

$$L = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k),\ \Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_j \|\omega_j\|^2$$

where $l(\hat{y}, y)$ is the traditional loss function, which measures the difference between prediction value and true value, k is the number of subtree, T is the number of leaf and ω_j is the score on each leaf(the scores are similar to the Gini indexes, which decides whether we should split a node or not). Secondly, column subsampling is used to further prevent overfitting. This technique derived from random forest model and also works well in boosting tree model. Besides, XGBoost model introduces a lot of techniques to speed up the computation.

4.3.2 Data Processing

XGBoost is robust for dealing with missing values. When split a feature with missing values, the algorithm will learn the best direction for missing values. However, as we mentioned before, some of the missing values should be 0, we will fill them with zero to prevent unnecessary computation. Besides, since boosting tree will split its nodes greedily, unimportant features will not affect our model as long as we control the depth of each tree. Therefore, we decide to remain all left features even some of them may contribute little.

4.3.3 Cross Validation

In this part, we choose a 10-folds cross-validation and RMSE for error evaluation to find the optimal parameters in our model.

parameter	meaning	value
eta	learning rate,shrinks the feature weights	0.03
num_boost_round	number of trees in the model	349
max_depth	max depth of each tree	5
$subsample$	subsample ratio of the training instance	0.5
min_child_weight	the minimum sum of weights required in a child	0.8
λ	the regularization term on weights	1.0

Table 3: parameters

Firstly, we set eta=0.1 and num_boost_round=500 to find the optimal values of all other parameters. In each experiment, the num_boost_round should be large enough to find the lowest CV-error(Figure 13 shows the CV-error should decrease first and then increase). Lower eta requires larger num_boost_round. After fixed all these parameters, we set eta=0.03 and num_boost_round=1000 to increase modeling accuracy. In the end, we choose num_boost_round=349 as shown in the figure 13. Using the parameters in table 4.3.3 in our final model, the test error is 0.176406.

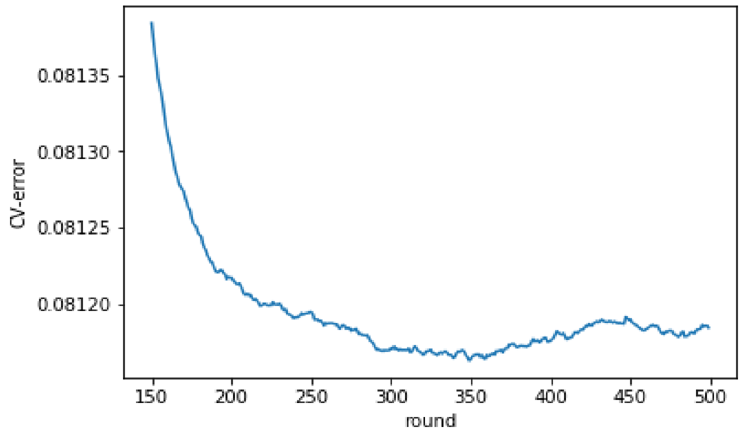

Figure 13: CV-error vs round

4.3.4 Analysis

Here are top 20 important features in our model, compare with the percentage of missing value we have shown before, it is obviously that the features with less missing value tend to be more important.

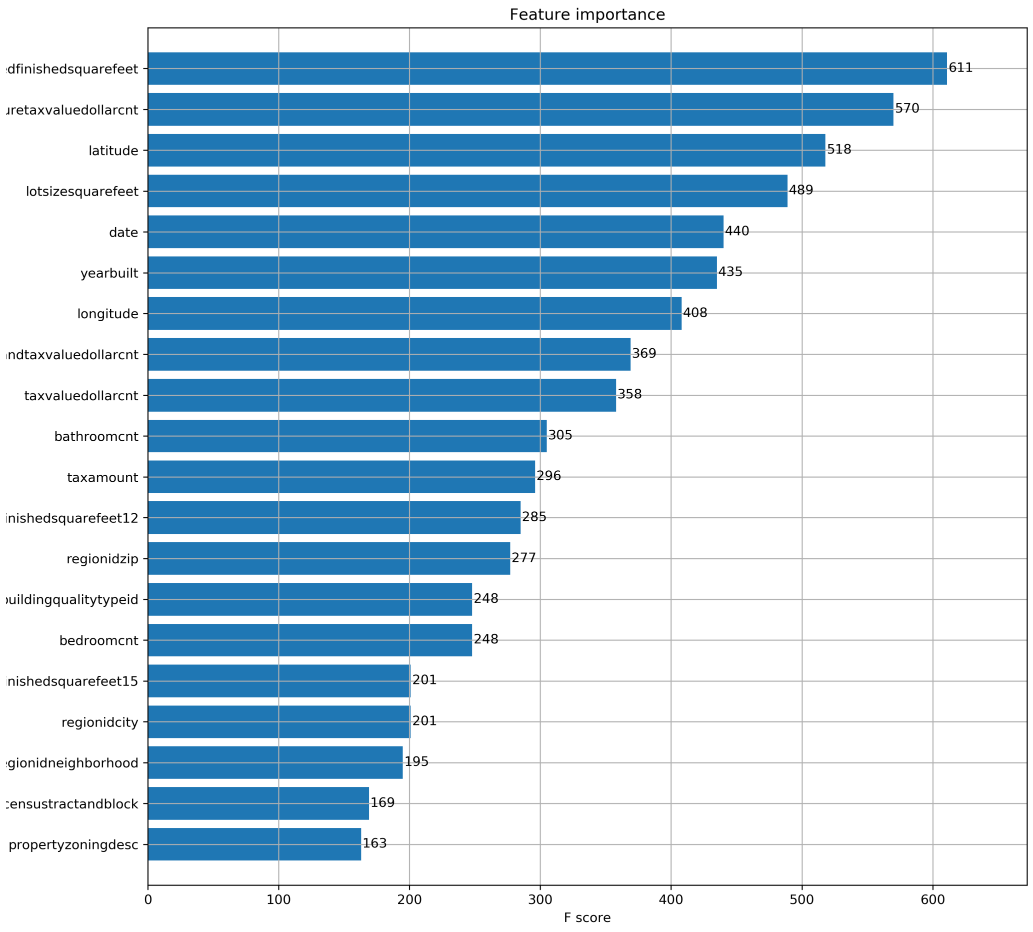

Figure 14: feature importance

5. Summary

From aforementioned distribution of $log(error)$, it has small standard deviation, which was confirmed by our range of predictions. Since property physical attributes on Zillow are obtained from open resources, which contain out-of-date information that are hard to be validated, predicting $log(error)$ based on this data set which has tremendous amount of missing value, noises and redundant features is challenging. We found that feature's importance and the number of missing value in each feature are strongly related, specifically, important features in our study are features with few missing values, so those unimportant features could become important, once we have enough data. We believe the quality of data set impacts three models we chose,and the test error among 3 models are so closed that we cannot fairly judge the analytical power of different models.

For further research, there are still so many drawbacks that are waiting for not only Zestimate, also for all artificial intelligence to overcome. Zillow cannot reflect how a buyer will feel when he or she enters the home. Zillow's data cannot substitute the function of a local real estate agent, because it cannot tell buyers whether the workmanship is skillful, or of the materials used are superior, or any other number of factors local agents and appraisers use when they know the neighborhood and have inspected the home in person. If Zillow can improve the data quality in such areas and modify algorithms so that it can measure some subjective features and neighborhood geographic, the $log(error)$ can be further reduced.

References

[1] https://www.zillow.com/zestimate/#faq-6

[2] https://www.kaggle.com/c/zillow-prize-1

[3] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[4] Dreiseitl, Stephan, and Lucila Ohno-Machado. "Logistic regression and artificial neural network classification models: a methodology review." Journal of biomedical informatics 35.5-6 (2002): 352-359.

[5] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.

[6] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016.