Zestimate: Zillow's Home Value Prediction

Armin Moridi
UC Riverside
amori009@ucr.edu

1 Introduction

This project targets the biggest Kaggle challenge so far with 1 million dollar grand prize. Zillow is one of the biggest companies in real estate world and it has become a price reference for anyone who plans to buy a house with their machine-learning model that estimates each houses' value based on various features. In this competition, participants will develop an algorithm that predicts future sales price of homes and the goal is to improve Zestimate residual error. In other words, we need to minimize the mean absolute error between the predicted log error and the actual log error:

(1) logerror = log(Zestimate) – log(SalePrice)

For the final result, we need to calculate this log error for each property in 6 different timepoints: October 2016 (201610), November 2016 (201611), December 2016 (201612), October 2017 (201710), November 2017 (201711), and December 2017 (201712)

This dataset has more than 1.4 GB size consisting of 58 features that describes house in various aspects. Based on the job description, in this report, I mainly focus on two parts: 1. Preprocessing and cleansing the data 2. Machine-learning techniques for building this model

2 Experimental Set up

I used Python in Jupiter Notebook for my coding environment. For preparing the data, I mainly used pandas, numpy, seaborn, matplotlib, Scipy, sklearn, and lightgbm libraries.

2.1 Dataset

There are four csv files:

1. train_2016_v2.csv: The Training spreadsheet contains the log error and transaction dates for 90,275 homes sold during 2016.
2. properties_2016.csv: The Property dataset contains 58 different features/details on almost 3 Million homes. Unlike the Training Data that has only homes that have been sold in 2016, this dataset features information on all homes. A lot of information are missing in this dataset, which means it needs a huge pruning and changing the data formats.
3. sample_submission.csv: This is a sample submission file in the correct format.
4. zillow_data_dictionary.xlsx: This is the explanation file of all data fields.

3 Preprocessing the Data

First, I will check the missing data in columns and drop ones that cannot be filled and will increase the error:

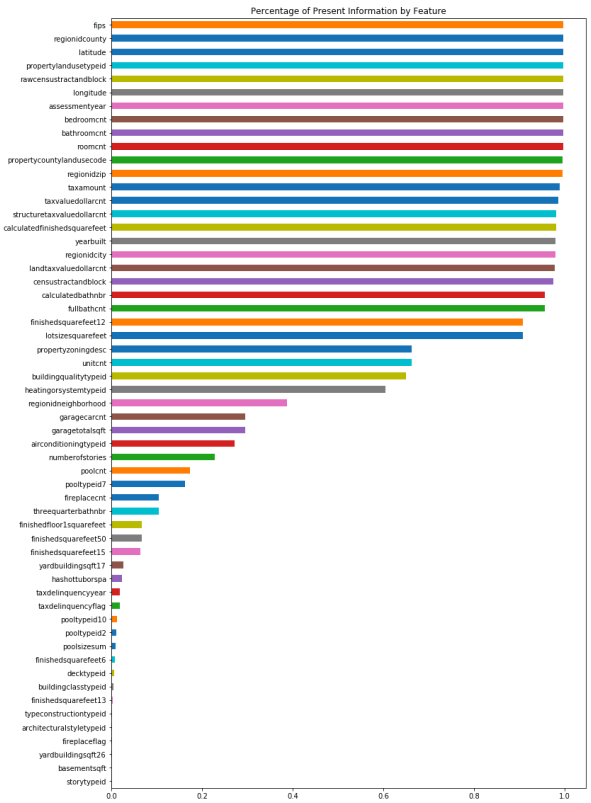

Fig. 1

As seen in Fig. 1, most of the columns have missing data (NaN value) instead of having 0 or false. On the other hands, some columns are related to each other and we can fill missing data based on their information. To avoid dropping possible valuable columns, I will correct the data structure in the first step.

3.1 Filling the Missing Data

There is not a general solution that works for all the columns. So, I need to manipulate each one individually and I divided them into 4 categories:

1. These columns have NaN value that we replace them with 0. (e.g. 'poolcnt' is number of pools in a lot NaN means there is no pool. So, I will replace all these values with 0.): 'pooltypeid2' , 'pooltypeid7', 'poolcnt', 'basement-sqft', 'yardbuildingsqft26', 'yardbuildingsqft17', 'airconditioningtypeid', 'heatingorsystemtypeid', 'unitcnt', 'landtaxvaluedollarcnt', 'structuretaxvaluedollarcnt'

2. These columns have True value instead of numeric values and we replace their NaN that stands for False with 0 and True cells with 1: 'hashottuborspa', 'taxdelinquencyflag', 'decktypeid', 'numberofstories'

3. The value of these columns can be determined by other columns (e.g. 'poolsizesum' is total square feet of pool in the property. We fill NaN values of this column based on Their 'poolcnt' column which is either 0 that we insert 0 for this column as well or they have at least one pool that we replace their pool size with median of other homes with pool.) or it combination of this case with case 2: 'poolsizesum', 'fireplacecnt', 'fireplaceflag', 'garagecarcnt', 'garage-totalsqft', 'calculatedfinishedsquarefeet', 'finishedsqu-arefeet15', 'finishedsquarefeet50', 'calculatedbathnbr', 'buildingqualitytypeid', 'propertyzoningdesc', 'lotsizesquarefeet', 'taxvaluedollarcnt', 'yearbuilt', 'fips', 'propertylandusetypeid', 'latitude', 'longitude', 'rawcensustractandblock', 'assessmentyear', 'bedroomcnt', 'bathroomcnt', 'roomcnt', 'propertycountylandusecode', 'regionidzip'

4. In this case, we have columns that have repeated information and we can simply drop them due to redundancy avoidance (e.g. 'pooltypeid10' tells us exact information same as 'hashottuborspa') or their data cannot help on improving the model like years: 'pooltypeid10', 'taxdelinquencyyear', 'storytypeid', 'architecturalstyletypeid', 'typeconstructiontypeid', 'finishedsquarefeet13', 'buildingclasstypeid', 'finishedsquarefeet6', 'finishedsquarefeet12', 'finishedfloor1squarefeet', 'threequarterbathnbr', 'fullbathcnt', 'regionidneighborhood', 'censustractandblock', 'regionidcounty'

Rare Case: To replace 'taxamount' with means, we need to define new column by dividing 'taxamount' by 'taxvaluedollarcnt' and replacing the NaN values with the average. Then we can drop 'taxamount' column.

We have cleaned our messy property dataset and reduced 58 columns to 42 ones that all have meaningful values.

3.2 Examing the Data

We know that these properties are located in three Los Angeles areas. So, the plot below examines 'latitude' and 'longitude' columns data quality by drawing Geographical Maps:

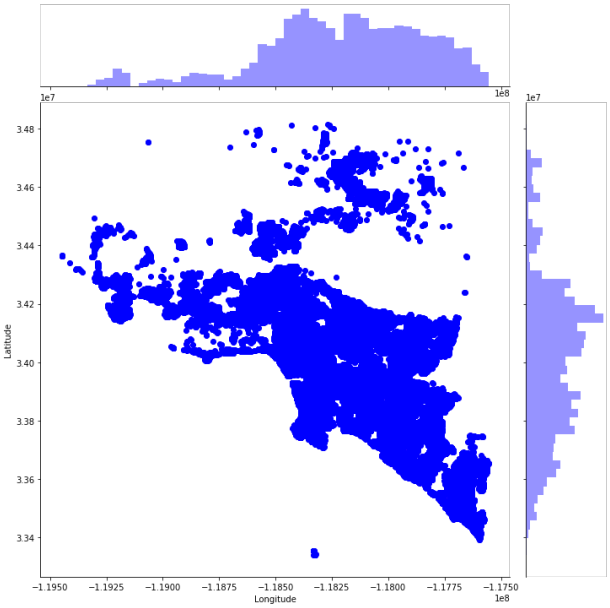

Fig. 2

In the next step, we need to examine train_2016_v2 spreadsheet. The plot below shows number of sales based on each month:

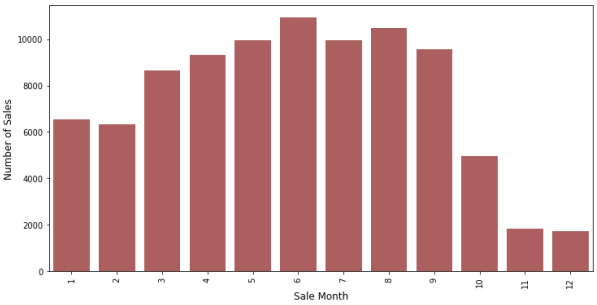

Fig. 3

This plot shows that sales will change seasonly with the most values between March and September and considerable drop-off between October and February.

According to Fig. 4, we can see that log error will increase between October and February period which

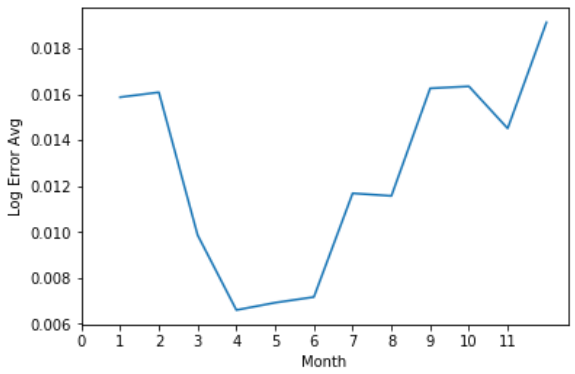

Fig. 4

number of sales has decreased drastically. Speculating, one might hypothesize that selling prices during these months might be a bit more sparse than other months. Imagine someone wants to sell his house in December. Since, there would be less demands in the market, the seller may be willing to accept an offer below market rather than hold onto the home for a few more months. So, I think that this increase in log errors are because of overvaluing home prices.

In the last step, we will merge (left join) property and train spreadsheets together on 'parcelid' that is homes' unique identifier in both tables. Finally, We will have 44 columns and 90275 properties.

4 Machine Learning Approach

In Fig. 5, I examined all the possible correlations between each feature and the log error column. We cannot see string correlation between any of them. Table below analyzes shows features with greater correlation values:

	feature_col	corr_values
1	structuretaxvaluedollarcnt	0.023141
2	bedroomcnt	0.025467
3	bathroomcnt	0.027889
4	calculatedbathnbr	0.028788
5	finishedsquarefeet50	0.036174
6	finishedsquarefeet15	0.038305
7	calculatedfinishedsquarefeet	0.038305

Table 1

As seen in the heatmap Fig. 6, important variables have high correlations and we can build our regression model based on these features.

We need to find a fast algorithm to handle all these big

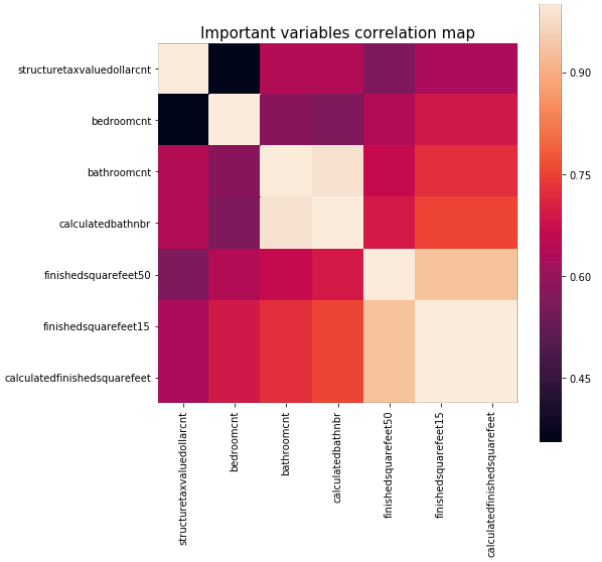

Fig. 6

data and for this problem I will use Gradient Boosting algorithm from Ligh GBM library that is very fast and can minimize the mean absolute error metric:

(2) $$MAE = \sum |y_i - y_i^p|$$

Boosting is an ensemble technique in which the predictors are not made independently, but sequentially. My predictors are regressors to handle residuals log error and minimize them. Our feature list has 39 columns by dropping 'parcelid', 'logerror', 'transaction-date', 'propertyzoningdesc', and 'propertycounty-land-usecode' columns and label column just has 'logerror' column. Train data consists of 90% of data and the rest is test data. Then, we assign required parameter (Fig. 7) for the model and we train the model based on that. Finally, we predict the model and find the MAE. We have achieved 0.065318 for mean absolute error that is a huge improvement in this dataset. Last but not least, I have written all the residual log errors in a csv file as the kaggle request for the competition.

```
params = {}
params['learning_rate'] = 0.002
params['boosting_type'] = 'gbdt' #gradient boosting
params['objective'] = 'regression'
params['metric'] = 'mae'
params['sub_feature'] = 0.5
params['num_leaves'] = 60
params['min_data'] = 500
params['min_hessian'] = 1 #avoid overfitting
```

Fig. 7

To avoid any memory problems, I have retrieved all the the resources of variables by using garbage collector library. Fig. 8 shows summarized machine learning code for training and predictions the model:

```
x_train = merged_data.drop(['parcelid', 'logerror', 'transactiondate',
                            'propertyzoningdesc',
                            'propertycountylandusecode'], axis=1)
y_train = merged_data['logerror'].values
split = 81000
x_train, y_train, x_valid, y_valid = x_train[:split], y_train[:split],
x_train[split:], y_train[split:]
d_train = lgb.Dataset(x_train, label=y_train)
d_valid = lgb.Dataset(x_valid, label=y_valid)
watchlist = [d_valid]
clf = lgb.train(params, d_train, 500, watchlist)
y_pred=clf.predict(x_valid)
```

Fig. 8

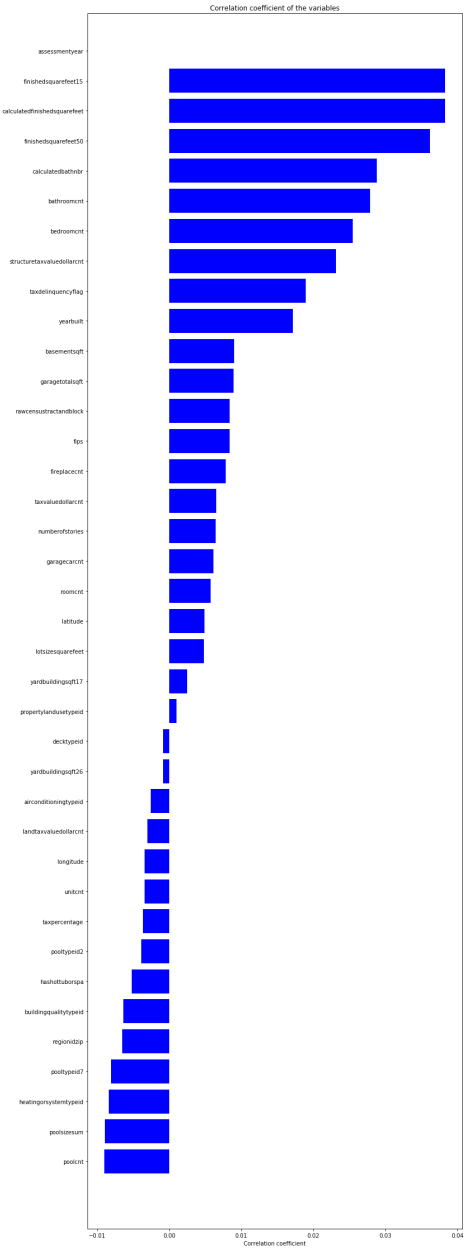
Correlation coefficient of the variables
assessmentyear
finishedsquarefeet15
calculatedfinishedsquarefeet
finishedsquarefeet50
calculatedbathnbr
bathroomcnt
bedroomcnt
structuretaxvaluedollarcnt
taxdelinquencyflag
yearbuilt
basementsqft
garagetotalsqft
rawcensustractandblock
fips
fireplacecnt
taxvaluedollarcnt
numberofstories
garagecarcnt
roomcnt
latitude
lotsizesquarefeet
yardbuildingsqft17
propertylanduisetypeid
decktypeid
yardbuildingsqft26
airconditioningtypeid
landtaxvaluedollarcnt
longitude
unitcnt
taxpercentage
pooltypeid2
hashottuborspa
buildingqualitytypeid
regionidzip
pooltypeid7
heatingorsystemtypeid
poolsizesum
poolcnt
−0.01
0.00
0.01
0.02
0.03
0.04
Correlation coefficient

Fig. 5