Modeling Log-error with Zillow Housing Data

Jiali Cheng[1], Sicheng Zhang[1]
[1]Graduate School of Engineering, Northeastern University, Boston, USA

***Abstract*—In this paper, we are aiming at fitting the log-error of Zillow housing price in 2016 and 2017 using Multi-Linear Regression, Random Forest and Neural Network based on Mean Squared Error and Mean Average Percent Error. First we do an in-depth Exploratory Data Analysis, which gave us a brief view of what the data looks like. Then we clean the data and select 9 features out of a total of 53. Then we throw the training set into Multi-Linear Regression, Random Forest and Neural Network. It turns out that Random Forest gets the highest accuracy with reasonable computation time. Then we build this model on AWS Machine Learning and access it with REST API.**

***Keywords*—Log-error, Multi-Linear Regression, Random Forest, Neural Network, MSE, MAPE**

I. Introduction

Zillow is an online real estate renting and selling platform, who keeps records of thousands of houses and estimate their selling price. In this paper, we use the Zillow housing data from 2016 and 2017 to estimate the log-error, which is defined by Zillow as log(Zestimate) - log(sale price).

II. EDA and Preprocessing

In exploratory data analysis, we take a detailed look at our data and do some preprocessing on the data for future use.

First, we ingest the 4 files. We use describe() to see the description of a data frame, including max value, min value, mean value and median value of each feature. And we add a column indicating the transaction year to each property file. For each file, we figure out the type of the values inside. And we take a look at the missing values of the features. Then we look at some specific columns like 'fips', which only have 3 possible values. This may be a useless feature due to a low variance, since it may not contain much information. Also, we plot some specific columns that we thought may be a part of the model, like square feet and log error, the target.

After that, we do some preprocessing. Within each year, we combine the "train" file and the "property" file on the intersection of the "parcelid", since the "train" file contains the houses that have been truly sold, while there are many other house information in the "property" file that is useless because of no transaction made. So we use the intersection. Later, we concatenate the two files and drop duplicate transactions according to "parcelid", since we only consider the first transaction (see the data description on Kaggle.com). Now we make this data frame the training set, which is ready for cleaning.

The target is to fit the logerror. But instead of fitting log error directly, we assume that the sale price is always 1000, since we cannot fit an error but only estimate the price. So according to the formula on Kaggle, the estimate price is transformed into exp(logerror)*10000

III. Data Cleaning

There are different types of missing values and we need to adopt different methods accordingly so that we clean the data most logistically.
Basically by a value is "missing", we mean one of the following:
i) a value that is simply missing.
ii) a value that implies something wrong or impossible, such as month of 13, which makes no sense.
iii) a value missing as another occasion.

In this project, we fill the missing data based on the statistical feature of them, correct outliers and do some useful transformation. And we follow the principle of correcting data with less time and effort.

First we calculate the missing-to-all ratio of every feature. We found that some features have more than 70 percent of data missing. It is such a high missing ratio that we delete these features. A house may have a lot of features and some of them are too specific and may be of less use. Moreover, we have little clue from which we can fill these features. And filling them would cost too much time. Hence, simply deleting the features with more than 70 percent missing ratio is a good choice. There are also some houses who only have logerre and transaction data but have no any other feature. So they are also of no use and dropped.

Then come the rest of the features. For categorical data, we fill the missing part with the mode value, which is the most possible value. And for numerical data, we fill the missing ones with the mean value, which is the most possible value.

Later, we plot some specific features out to find out the outliers. And we correct them with also the mean. We also transform longitude and latitude into logistic values and the transaction data into month, since there is already a column indicating transaction year.

IV. Feature Engineering

Feature selection is a process where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested. According to many data scientist, "Data and its features would decide the upper bound of machine learning. While algorithm and model can only get closer to that upper bound." This may reflect the importance of feature engineering.

Having too many irrelevant features in your data can decrease the accuracy of the models. Three benefits of performing feature selection before modeling your data are:

i) Reducing Overfittingness: Less redundant data means less opportunity to make decisions based on noise.

ii) Improving Accuracy: Less misleading data means modeling accuracy improves.

iii) Reducing Training Time: Less data means that algorithms train faster.

In this project, we basically select useful features out of the 58 features given. In order to do feature engineering, we take a careful look at our data set. There are numerical data, such as square feet, and categorical data, such as many type ID. There also exist some features that can be regarded as both numerical and categorical, such as bedroom number. We follow the rule of selecting fewer features in less time. Therefore, we treat features that is both numerical and categorical as numerical, which save our time.

We conduct feature selection in 3 parts: Selecting based on the information of an individual feature, selecting based on the feature importance on the target and selecting based on the correlation between two features. And for categorical features, we do a numerical encoding so that they can be regarded as numerical features.

First we select based on the internal variance of a feature. A feature with very low variance would result in less information. This is always used for categorical data since there are limited possible values. Here we do a numerical encoding for the categorical data. And we only encode them within 10 possible values. A possible value with higher likelihood can get an individual number, while values with low likelihood get another number collectively. Then we set the variance threshold as 0.5 and drop the features below the threshold.

Then we select based on the importance of a feature on the target. That is to say, we calculate how informative a feature is to a the target. We use two ways: the pure mathematical correlation between features and target and the xgboost. Within each method, we take some features with high influence on the target and find both the intersection and the union of the two parts as two potential training sets for future use.

At last, we calculate the co-variance between the features within the intersection and the union. Because features with high co-variance means they may convey information that is partly same and overlapping. Using both may result in a waste of time and overfitting. In this project, we set the co-variance threshold 0.5. Features whose co-variance is less than 0.5 can be both used in the model. Only one of the two features can be used if the co-variance is greater than 0.5. In this scenario, we take the feature who has more influence on the target in step 2.

Finally, we reach to the training sets, the intersection that has 8 features and the union that has 11 features.