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. Exploratory Data Analysis

III. Data Cleaning and Preprocessing

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.

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.

Many kernels posted for this competition doesn't use one hot encoding for many variables which I believe should be treated as such. For example, regionzipid (i.e. zip code) is not exactly a numerical variable and there isn't ordinal relationship between two zip codes. Same goes for other variables such as airconditioningtypeid, architecturalstyletypeid etc. When I try one hot encoding on those variables and use XGB, my scores are worse than not using one hot encoding.

best categorical var. encoding depends both on the model you are using and the variable itself (not the same 3 levels than 10000 levels, what is called "high cardinality"). ordinal (numeric) encoding doesn't normally work in linear models because the model has no mechanism to handle badly ordered features or nonlinear relationships between certain categories

and the target.

V. Model Selection and Training

Linear Regression

R-squared, also called coefficient of determination, indicates the fitting degree of the model to the real data. In one linear regression, r-squared equals the square of Pearson product moment correlation coefficient.

Random Forest The sklearn.ensemble module includes two averaging algorithms based on randomized decision trees: the RandomForest algorithm and the Extra-Trees method. Both algorithms are perturb-and-combine techniques [B1998] specifically designed for trees. This means a diverse set of classifiers is created by introducing randomness in the classifier construction. The prediction of the ensemble is given as the averaged prediction of the individual classifiers. Random Forest has less bias, need to reduce variance. sub models are not weak models. scale cannot be big since it takes a lot of memory. rf is a black box we don't know what's inside.

Neural Network

VI. Model Ensemble

VII. Conclusion