Machine Learning Nanodegree Capstone Proposal

Robert Burrus
Sept 21, 2017

Domain Background
This proposal is for a submission into Zillow's Home Value Prediction competition on Kaggle. Accurate home valuations are extremely important during the purchase of a home, and for monitoring home valuation over time. Eleven years ago, Zillow created an automated home valuation estimate called Zestimate. This estimate is calculated from public and user-submitted data, taking into account special features, location, and market conditions. Today, Zestimates are calculated for more than 100 million homes nationwide. The median margin of error of Zestimates has dropped from 14% (11 years ago) to 5%, establishing Zillow as one of the largest and most trusted marketplaces for real estate information in the US.

US housing prices have been steady over much of the last 40 years, but there have been occasional declines. A lot of research is dedicated to modeling these changes in the housing market over time, as seen here, here, and here. Zillow has been researching this modeling challenge for years, and they discuss many insights on their blog. For example, this post discusses the data quality challenges with public records. Like many machine learning problems, acquiring quality data is the first priority. Zillow builds their home database from a range of sources including county records, tax data, home sale listings, home rental listings, and mortgage information. Issues of data completeness, data latency, and data interpretation are crucial problems to solve before applying machine learning algorithms. While Zillow does not disclose the implementation details of their algorithm for obvious reasons, they do offer many insights into Zestimate evaluation and accuracy, and occasionally discuss some of Zestimate's sub-models.

Zillow is now turning to Kaggle to help push the accuracy of the Zestimate even further.

Problem Statement
Zillow has split their Kaggle competition into 2 stages: (1) Predict the error between Zestimate and actual sale price, and (2) Build a home valuation algorithm from the ground up, using external data source to help engineer new features. The 2nd stage of the contest is to be comprised of the top 100 entries from the 1st stage. **My capstone proposal is in regards to the first stage of the competition.** The task is to *predict* the log-error between the Zestimate and the actual sale price, given all of the features of the home. In other words, this is a regression task to anticipate when the Zestimate will be more and less accurate. The task is *not* to predict the home valuation itself, but to predict how good the Zestimate home valuation will be.

Datasets and Inputs
Zillow is providing a list of over 2.9 million real estate properties in 3 counties (Los Angeles, Orange, and Ventura, California) in 2016. Each property has 50+ data fields, though not all properties have data for all fields. A small subset (about 90,000) of those properties were actually sold in 2016, and the transaction date and ground truth log error are also provided for the properties. The transaction / log-error data is from Jan 1 – Oct 15 2016, plus some from Oct 15 – Dec 31, 2016. The remaining Oct 15 – Dec 31 2016 data is unavailable, and will be used as the test data for the public Kaggle leaderboard. On October 2, 2017, Zillow will release additional property / transaction data from Jan 1 – Sep 15, 2017. The final test data will be data from Oct – Dec 2017. This is the data that the final Kaggle leaderboard will be based on.

Because of the timeline of the Kaggle competition, I will evaluate my solution for this project on the Oct 15 – Dec 31 2016 data that Kaggle withheld and is using to rank the public leaderboard. This means I will train on the 2016 data provided, and use the Kaggle public leaderboard test set for my final model evaluation.

As mentioned above, the property data has 50+ fields. This includes things like build year, various area measurements, number of bathrooms / garages / bedrooms / fireplaces, location information, tax information, architectural style, etc. Often in Kaggle competitions and machine learning projects, more data is better. I will investigate supplementing this data from outside sources, though I do not currently have that data in hand.

Evaluation Metrics
The Kaggle competition submissions are evaluated on Mean Absolute Error (MAE) between the predicted log error and the actual log error (actual logerror = log(Zestimate) – log(SalePrice)).

Benchmark
This is a unique project in that the task is to evaluate a proprietary algorithm, making predictions about it's performance, not trying to beat. Thus, a benchmark is somewhat difficult to define. One possible benchmark is the median MAE score on the Kaggle leaderboard. Out of ~3,000 contestants, the current median MAE is 0.0646644, while 1st place is 0.0639094. Another, perhaps more useful, benchmark is to train an out-of-the-box regression algorithm on the project data and compare it's performance with my final solution. I will set a benchmark with a Random Forest regressor, and then try to engineer a solution to beat it.

Solution Statement
My solution to this project will be a regression algorithm, or ensemble of regression algorithms, which produces a MAE better than the benchmark model, and hopefully ranks highly on the Kaggle leaderboard. Working towards a solution will require a lot of data exploration, pre-processing, and hyper parameter tuning (discussed in Project Design). I plan to explore the effectiveness of feature scaling, normalization, outlier removal, and principle component analysis, as well as test multiple regression algorithms: XGBoost, Stacknet.

Project Design

The following workflow represents the basic steps of my approach to this project. As often happens in machine learning projects, we return to data exploration and data preprocessing again and again while exploring different algorithms or fine tuning hyper parameters. Many of these techniques will only be applied if found to be an improvement over not doing them.

1. Data Exploration. Explore data through visualizations to understand how each feature is related to the others and to the log-error. Consider the relevance of each feature.
    a. What data is missing? Explore different techniques of handling missing data: Input missing value with data set mean vs regression to predict it's value. Consider removing a piece of data if many important data fields are missing.
    b. What are the data types of the property fields?
    c. What is the shape of the distribution of the log-errors for the training data? What is the mean? Are there outliers?
    d. What are the correlations between property fields and log error?
    e. How does the log error change with time?
    f. Determine feature relevance with decision tree regressor
2. Data preprocessing. Based on the data exploration, consider applying the following techniques:
    a. Feature scaling – explore log scaling, batch normalization
    b. Outlier removal. There are many definitions of an outlier. My definition will be determined based on data exploration, and trying different cutoffs.
    c. Explore feature transformation – Principle Component Analysis. This step would only be applied after running the model without any feature transformation.
    d. Cross validation
3. Algorithm exploration. Try various ML algorithms and analyze model performance
    a. XGBoost
    b. StackNet
    c. Explore ensemble of multiple algorithms and take average (or weighted average) of the log-errors. If I'm able to find 2+ well performing models, I will try averaging the outputs. Different models may predict well on different data, and averaging could increase predictive power over individual models.
4. Fine tuning of algorithm
    a. Grid search of hyper parameters