Machine Learning Engineer Nanodegree

Capstone Proposal

Lin Muqing
Sept 9th, 2017

Domain Background

I choose the Kaggle "Zillow Prize: Zillow's Home Value Prediction" round 1 competition (https://www.kaggle.com/c/zillow-prize-1#evaluation) as my capstone project. Zillow is a real estate firm that has an in-house home value prediction model Zestimate. In the first round of the competition, participants need to be able to build up a model to use given features to predict the log error between the actual transaction price and the Zestimate valuation.

Problem Statement

This is a classic supervised learning problem, with properties of each transaction, including properties of parcel being transacted and time of transaction, as input x and log difference between Zestimate and actual transaction price, i.e. log error = log(Zestimate) – log(SalePrice), as output y. Be noted that while each parcel has a fixed features specification, it could be traded many times and each time the transaction value could be different. So, for each parcel, more than one prediction of log error is needed, for different time periods (using month as time period definition), taking into consideration of seasonality effect and regime change. In the first round of the competition, we need to predict log error for 201610, 201611, 201612 and 201710, 201711 and 201712. Data for 2017 prediction will not be released until Oct, which is after close of Udacity MLND program. So for capstone, I will only use the three months of 2016 as target y.

Datasets and Inputs

As provided by the competition, we have two datasets. One contains properties of ~3 million parcels, and one contains log errors and transaction dates of ~90 thousands actual transactions happened in 2016. One thing to note is that since part of 10, 11 and 12 months data are held back for testing, in training data, we have imbalanced-sample situation with less data in those 3 months than others.

For each parcel, there are 58 properties, I will only list several that are most obviously related to parcel valuation: room count, size of the basement, size of the finished living area, assessed total property tax. For actual transactions as training data, each transaction is featured by ID of the parcel being transacted, transaction date and log error of that transaction. One special handling for a transaction is that, when more than one transaction occurs for certain parcel in one month, the first transaction with reasonable sale price will be used as official record for the parcel on given month with other records ignored.

Solution Statement

The solution is quite straight forward for a supervised learning problem. The general workflow would be to use raw training data for both training and cross validation, with training data for parameter optimization and validation data for hyper parameter tuning, and testing data for final evaluation. The process should be composed of data preprocessing, feature engineering, base model selection and boosting, details being described in project design.

Benchmark Model

Since the in-house Zestimate has been used and improved in practice for years, the log error itself is already noise-like. A naïve benchmark would be to simply predict all the log errors as 0 for all transaction months. Any valid attempt should at least be better than that, and it would be interesting to see how much better it could be. However, a more responsible benchmark model should take some effort. So I would use xgboost as official benchmark model, with simple filtering of raw properties (by NaN ratio), no consideration of seasonality effect or feature engineering. So I would have 2 benchmark models, a naïve one with all zero prediction and an official one using direct xgboost. Both should help understand the extent of improvement along the model iteration process.

Evaluation Metrics

As requested by the competition, model performance is evaluated by average Mean Absolute Error between the model predicted log error and true log error on the held-out test set, i.e. $\frac{1}{n}\sum_1^n |\text{model log error} - \text{true log error}|$. For a regression problem, either MSE or MAE is fine. Here, for this problem, I agree MAE is more suitable. Using MSE as evaluation metric implicitly gives more weight on samples that have larger absolute error to predict the logerror. Here the logerror distribution is highly heavy tailed, and very likely we will do badly on those extreme values than others. So if using MSE will try to improve prediction on those large logerror samples while sacrificing accuracy on others, i.e. for parcel value prediction, it will sacrifice overall accuracy when trying to do better on those we do very bad before, which I believe would not be a preferred solution for business. Using MAE means one unit error reduction in those with large errors is equally valuable to us as in those with small errors.

Project Design

- First of all, some data exploration should be conducted.
  - Parcel property data set:
    - NaN ratio of each feature (all parcels).
    - NaN ratio of each parcel (parcels in training data set only).
    - Data type of each feature.
    - Actual information represented by each feature.
    - Frequency of each category of categorical variables.

- Transaction data set:
  - Distribution of log errors.
  - Number of samples falling into each month.
  - Cases of more-than-one-trade for each parcel in each month.
- Correlation: how each feature is linearly related to log error?

- Then based on data exploration results, do necessary data-preprocessing:
  - Exclude features with too high NaN ratio for prediction.
  - Exclude parcels with too high NaN ratio for learning (weighting features by heuristically determined importance might be necessary).
  - Perform necessary data type transformation (pandas will read in number-like string as float.)
  - Perform necessary information separation, for example, feature “census tract and block” actually includes more than one pieces of information.
  - Merge transaction samples with same parcel ID in same month.
  - And possibly missing data imputation.
- feature engineering:
  - Several potential angles:
    - Feature cleaning: xgboost takes in only numerical features, so all categorical features should be one-hot-encoded. Some categorical features include many groups, e.g. “Heating Or System Type ID” has 22 categories. Potentially some categories could be merged and results in smaller final feature set.
    - Direct new feature construction, e.g. taking log of assessed tax or sum of room counts.
    - New features from unsupervised learning:
      - PCA to draw critical information from highly correlated features, e.g. several types of square feet.
      - Clustering to create categorical labels from numerical features, e.g. latitude and longitude.
  - For each newly created feature, its contribution can be first roughly evaluated by simple-tree feature importance analysis, and scrutinized from performance on testing data set.
- Model fitting:
  - In Solution Statement section, I described it as separated steps as base model selection and boosting. Actually some model itself is a combination of two, e.g. Random Forest and xgboost.
  - When using a shallow-model based algo, I think feature quality is much more important than algorithm. As long as the algorithm has some basic components, e.g. can model non-linearity, can handle both numerical and categorical inputs, can incorporate regularization and can be boosted. Xgboost meets all above criterions and wraps them all in one box, so I would use xgboost as the model for all trials.
  - Model fitting process would be standard: fitting using provided package and hyper parameter tuning using cross validation.
- Some other tricks: maybe above process would make it a valid capstone submission, but to stand out in the competition, something tricky worth tried:

- Seasonality:
  - It worth investigating:
    - Does the seasonality effect exists? We only have one year of data.
    - How to represent the seasonality effect? Simply different intersections?
  - Moreover, if seasonality does worth modeling, sample size imbalance might need to be considered.
- Discriminate large and small log errors:
  - Log errors is already noise like, but heavy tailed, meaning, if pattern can be captured in those large log error samples, results can be significantly improved.
  - Then some questions worth being answered:
    - How to define large-error samples?
    - Is the number of those samples big enough to be modeled?
    - Can we extract a pattern to predict whether a sample would be or large error or not properties? If so we can build a specific model for it.