Machine Learning Engineer Nanodegree

Capstone Project

Lin Muqing

Jan 13th, 2018

I. Definition

Project Overview

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 US real estate firm that has an in-house house 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.

Related dataset is provided by Kaggle, including data of properties of over 3 million houses, that is a collection of features for each parcel\_id; and Zestimate prediction error data of actual sales, each sale record includes error, sale-date and the parcel\_id involved. There are two versions of these data, properties evaluated at beginning of 2016 and actual sales of 2016, and that for 2017. Be noted that the Zestimate predicted prices are generated by the same model, but with different properties data for 2016 sales and 2017 sales. Training data includes all sales of 2016 months 1 – 9, part of sales of 2016 months 10 – 12; all sales of 2017 months 1 – 8 and part of sales 2017 month 9. Training data release is divided into 2 rounds, the first contains only 2016 data, which is available since the beginning of the competition; the second is 2017 data, which only takes place 2 weeks before deadline. As part of Kaggle rules, there are 2 testing datasets, the public Leaderboard (Public LB), which is used by participants to view their models’ out-of-sample performance before final submission; and the private Leaderboard (Private LB), which is the evaluation dataset to get participants’ final score and ranking. Public LB’s data is not available, but participants can view the performance of models on this dataset multiple times. Private LB’s data is not available neither and participants can only choose 2 models before deadline for submission on this dataset to view the score and use the better one of the two to get final ranking. For Zillow Prize-1, Public LB contains part of sales of 2016 months 10 – 12 and Private LB contains part of sales of 2017 months 10 and 12 and all sales of 2017 month 11. Be noted that competition deadline is 2017 Oct. 17th, so Private LB is a truly out-of-sample evaluation, with no potential leakage.

For this capstone report, I am not going to chronologically record all the stuff I have done for the competition, it was quite a mess and would be confusing. Instead, I will only record several most valuable trials. Regarding dataset, I will directly use the combined version of 2016 and 2017, as it seems trivial to document the fact that model performance gets better when I get more training data as 2017 data gets released.

Problem Statement

This is a classic supervised learning problem, with training data of 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. And it is a regression problem, and target y, the log error, is continuously distributed. 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, also, since we do not know which houses are actually traded in the testing dataset, participants need to make prediction for all of over 3 million houses for both 2016 and 2017 for each month of 10, 11, and 12, and Kaggle will pick up those with actual sales on given year-month combination to evaluate performance.

Metrics

There is a requested model performance evaluation metric in this competition: average Mean Absolute Error between the model predicted log error and true log error, i.e. . It is reasonable to use MAE instead of MSE here, as MSE 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, we 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.

II. Analysis

Data Exploration

We have total of 57 raw features in the provided property data, data exploration is conducted on these features only. Since we are predicting the errors of a fine-tuned model, we don’t really expect any remaining significant linear patterns between features and prediction target. And as mentioned later, since boosted-tree model is used here, we don’t worry about collinearity. So instead of first looking at a corr matrix among features and target, we directly look at each one of the features in the following three angles:

1. Feature vs. log error patterns from local regression. A good feature should see different patterns of mean-log error at different area.
2. Feature vs. abs log error patterns from local regression. Abs log error is not directly related to original problem, but due to heavy tailed log error distribution, we could significantly improve results if we could predict well on the large error area. A potential improvement is to first predict where Zestimate makes a large error and we can build a different model for that.
3. Density: significance of contribution of feature to prediction should consider the sample density in the ‘patterned’ area.

Features are subjectively classified into 4 categories, ranked from 1 to 4:

1. Very good features, with low missing rate and good patterns against log error at high density area.
2. Good features, they have class-1 potential but with higher missing rate or not significant pattern.
3. High cardinal, categorical features with high cardinality.
4. Bad features, either very high missing rate (over 90%) or hardly any pattern.

All the details could be found in data\_explore.html, and summarization of key information could be found in data/features\_info.csv, be noted that a better readable naming is created for each feature and this will be used in the rest part of the project.

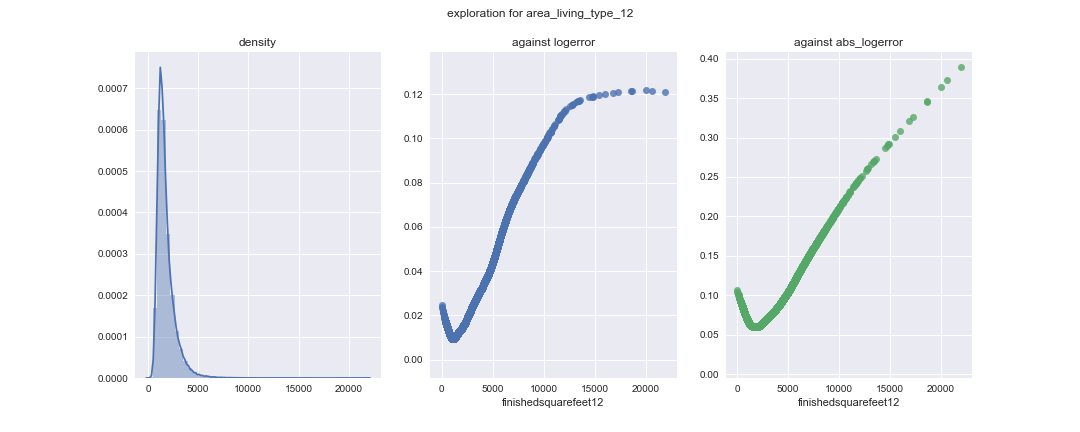
It is worth noting that this hand-labeling of features turns out to be no more efficient than boosted-tree’s feature importance analysis. It helps no more than providing a more concrete idea of what we have in hand. In fact these plots could be misleading in the following 2 aspects:

1. For highly concentrated numerical features, ‘pattern’ across the whole value domain could overshadow the local structures at high-density area that we really care about (e.g. area\_living\_type\_12).
2. Scale of pattern-plot could be dominated by the abnormal-behaving low-density area and difference at high-density area gets visually shrunk (e.g. type\_air\_conditioning).

Exploratory Visualization

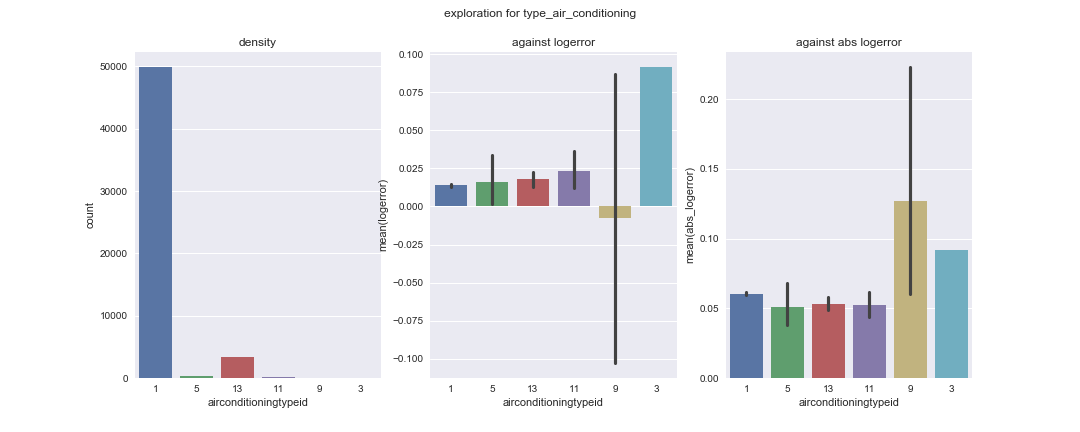
We present visualization of one example feature of each of class 1, 2, and 4. There is no visualization of high-cardinal features.

1. Class-1: area\_living\_type\_12



With low missing rate, area\_living\_type\_12 looks like a good feature.

1. Class-2: type\_air\_conditioning



Missing rate is around 70%, high density area is only type-1 and type-13. There is slight difference between the mean-log-error of the two groups. Type 13 has sample size of around 3k, so tree can make several valid splits in there and potentially capture the pattern.

1. Class-4: area\_living\_type\_15



This is a special feature, besides the over 90% missing rate, it seems the pattern is weak in the high density area. However, later in feature importance analysis, it turns out to be quite useful.

Algorithms and Techniques

I choose to use GBDT (Gradient Boosted Gradient Tree) model family to solve the problem. Tree is non-linear and sufficiently expressive, and boosting mitigates over-fitting. Random Forest is less ideal in the sense that trees are not related to each other, new trees does make use of information of previous ones. AdaBoost does not fit well as its main contribution is to make smart combinations of existing weak predictors, while we are building the model from scratch, another algorithm is needed to first find those predictors. After all, the most direct reason for choosing GBDT is its reputation in the Kaggle community, GBDT model family has been proved well effective in multiple contests.

Idea of GBDT is that, each new predictor fits to gradient of the loss function to previous predictors’ prediction results. The three GBDT mentioned in the report are XGBoost, LightGBM and CatBoost. XGBoost makes use of both first and second derivative of loss function to minimize it, while traditional GBDT uses only first derivative. And there are many other useful features, like including regularization in both size of predictions and tree complexity, please see reference for all details. LightGBM is a computationally improved version of XGBoost in terms of strategic subsampling and taking advantage of sparse features. So it is much faster, but less accurate in training, however turns out to be no worse in prediction with testing data. CatBoost is like XGBoost but advertised for its auto-handling of categorical variables. However, I found its extreme strength with a special logic to fight against biases, details of which will be discussed in model-refinement section.

For my model, LightGBM is chosen against XGBoost for mainly two reasons:

1. XGBoost’s default API does not have MAE as loss function. Although API for customized loss function is provided, without sureness of correct implementation, I would go with LightGBM, where MAE can be directly configured.
2. XGBoost has to handle categorical variables with one-hot encoding. It is less ideal for four reasons that I can think of:
   1. There is no way to directly handle high-cardinal features.
   2. With sub-sampling on features, such setup would give higher weights on categorical features.
   3. For categorical features with more than 4 classes, with one-hot encoding, each split can only look at one class, which loses the big picture of the whole structure.
   4. It expands number of effective columns, which expands data-size (importance here as we have over 3 million rows to predict) and makes feature importance analysis less informative.

Also, the CV framework needs to be carefully designed for following two reasons:

1. Since we cannot choose all models for private LB submission, a ‘best’ model (actually Kaggle allows you to choose two ‘best’ models) has to be chosen without knowing models’ performance in testing dataset.
2. Also, we do not want to overfit to public LB, so number of submission to public LB had better be constrained, and each submission be well supported by local CV.

Because the data for public LB and private LB are collected for a specific time period, seasonality and change model’s predictive power along time needs to be considered. To be specific, public LB only includes 2016 10, 11 and 12 whose training data is noticeably less than the others; private LB only includes 2017 10, 11, and 12 which has no corresponding training data at all. So 3 types of CV are used to evaluate models locally:

1. n\_folds CV on all data, considering no seasonality and regime change.
2. Targeting for public LB, part of data of 2016 7, 8, 9 is held out as validation and the rest is included in training.
3. Targeting for private LB, part of data of 2017 7, 8, 9 is held out as validation, and the rest is not included in training.

Benchmark

In this section, you will need to provide a clearly defined benchmark result or threshold for comparing across performances obtained by your solution. The reasoning behind the benchmark (in the case where it is not an established result) should be discussed. Questions to ask yourself when writing this section:

- \_Has some result or value been provided that acts as a benchmark for measuring performance?\_

- \_Is it clear how this result or value was obtained (whether by data or by hypothesis)?\_

III. Methodology

\_(approx. 3-5 pages)\_

Data Preprocessing

Outlier detection (cleaning rows):

1, 14367791, large logerror, high tax with no bathrooms or bedrooms?? How to identify this type of outlier?

Heuristic feature cleaning, transformation and selection (cleaning columns):

1, missing data imputation: even for high missing rate columns, with prop data, the existing number of samples could be high, if we have ~100k samples with non-missing, we could train imputation model with prop data only (with those low missing rate columns), and do imputation in training data.

Feature Engineering:

1, last traded within last k months?

2, good pattern with latitude and longitude, make better use with them. Visualize, do pseudo 2-D kernel regression. Plot for each grid point avg error within a circle.

3, For too-many-category categorical variables, use LightGBM to do the grouping.

In this section, all of your preprocessing steps will need to be clearly documented, if any were necessary. From the previous section, any of the abnormalities or characteristics that you identified about the dataset will be addressed and corrected here. Questions to ask yourself when writing this section:

- \_If the algorithms chosen require preprocessing steps like feature selection or feature transformations, have they been properly documented?\_

- \_Based on the \*\*Data Exploration\*\* section, if there were abnormalities or characteristics that needed to be addressed, have they been properly corrected?\_

- \_If no preprocessing is needed, has it been made clear why?\_

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

- \_Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?\_

- \_Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?\_

- \_Was there any part of the coding process (e.g., writing complicated functions) that should be documented?\_

Refinement

Tricks ordered by importance:

1, feature selection.

2, feature engineering (grouping categorical features, paired features, num\_ features as categorical, simple nan impute for full\_bathroom.)

3, row selection, remove row outliers.

4, grouping high-group-number categorical features, make them usable.

5, seasonality handling.

5, 2-step modeling, first predict large abs error, then fit / apply two different set of models.

6, missing value imputation by algorithm from property data. (predict missing column with other columns).

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

- \_Has an initial solution been found and clearly reported?\_

- \_Is the process of improvement clearly documented, such as what techniques were used?\_

- \_Are intermediate and final solutions clearly reported as the process is improved?\_

IV. Results

\_(approx. 2-3 pages)\_

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model’s solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

- \_Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?\_

- \_Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?\_

- \_Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?\_

- \_Can results found from the model be trusted?\_

Justification

In this section, your model’s final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

- \_Are the final results found stronger than the benchmark result reported earlier?\_

- \_Have you thoroughly analyzed and discussed the final solution?\_

- \_Is the final solution significant enough to have solved the problem?\_

V. Conclusion

\_(approx. 1-2 pages)\_

Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

- \_Have you visualized a relevant or important quality about the problem, dataset, input data, or results?\_

- \_Is the visualization thoroughly analyzed and discussed?\_

- \_If a plot is provided, are the axes, title, and datum clearly defined?\_

Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

- \_Have you thoroughly summarized the entire process you used for this project?\_

- \_Were there any interesting aspects of the project?\_

- \_Were there any difficult aspects of the project?\_

- \_Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?\_

Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

- \_Are there further improvements that could be made on the algorithms or techniques you used in this project?\_

- \_Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?\_

- \_If you used your final solution as the new benchmark, do you think an even better solution exists?\_

-----------

\*\*Before submitting, ask yourself. . .\*\*

- Does the project report you’ve written follow a well-organized structure similar to that of the project template?

- Is each section (particularly \*\*Analysis\*\* and \*\*Methodology\*\*) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?

- Would the intended audience of your project be able to understand your analysis, methods, and results?

- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?

- Are all the resources used for this project correctly cited and referenced?

- Is the code that implements your solution easily readable and properly commented?

- Does the code execute without error and produce results similar to those reported?