Signate - Student Cup 2019 - 7th solution

Shou-Jen, Chang (Angus Chang)

November 8, 2019

Special thanks to Signate and Mynavi for holding this competition for students, and congrats to all the top teams. In this documentation, I'll briefly talk about my solution and some thought about this competition.

About me

Currently first year of Master in Tokyo Tech. For further information, please see my Kaggle Profile.

Summary

My final score (Public: 12161.67399 / Private: 13073.62281) is an ensemble of 7 models. I used Lightgbm for all my modeling since it is relatively fast compared to Xgboost and Catboost. For the validation scheme, I just simply use 7-Fold cross-validation since there's some overlap between train and test set (same loaction appeared both in train and test). My local CV is always aligned with public leaderboard, the only worry is that the std between folds is too big. However, in consequence it didn't produce any bad effects on me.

In hindsight, forgetting to explore more on "location" probably have been my biggest mistake in this competition, the validation approach itself is fine but after I finished reading other competitior's twitter, I just discovered that I can further utilize this information to explore the leak, such as using other column to check whether the samples are in the same building.

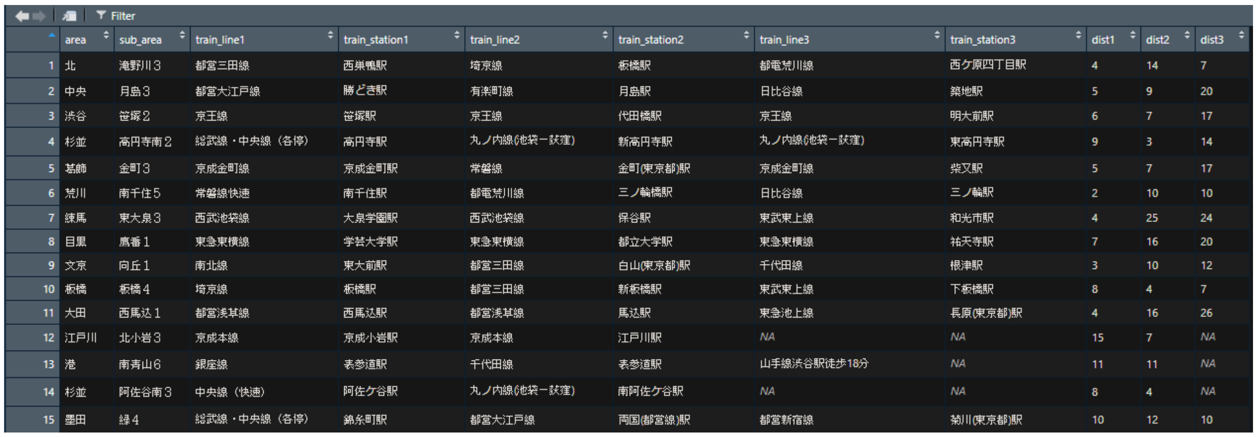
Hyperparameter

Shallow trees. I use this setting for all the modeling.

```r
lgb_param <- list(boosting_type = 'gbdt',
                  objective = "regression_l2" ,
                  boost_from_average = 'false',
                  metric = "rmse",
                  learning_rate = 0.05,
                  num_leaves = 10,
                  #  min_gain_to_split = 0.01,
                  feature_fraction = 0.3,
                  #  feature_fraction_seed = 777777,
                  bagging_freq = 1,
                  bagging_fraction = 0.7,
                  min_sum_hessian_in_leaf = 5,
                  #  min_data_in_leaf = 50,
                  lambda_l1 = 0,
                  lambda_l2 = 0
                  )
```

Preprocessing

Since data is quite dirty, I spend lot of time on seperating them into individual features. For the features like "location" or "access", I simply split them by blank or specific words. About another kinds of features like "bath/toilet", "kitchen" or "signal", I transformed them into a kind of one-hot encoded form, if the house(sample) have specific kind of attachment, then marked as 1, if not, then marked as 0. Besides, data cleanging is also important, actually not only do some features having error, there's also a sample that having WRONG TARGET and may biased your prediction, I'll talk about this later.



	area	sub_area	train_line1	train_station1	train_line2	train_station2	train_line3	train_station3	dist1	dist2	dist3
1	北	滝野川３	都営三田線	西巣鴨駅	埼京線	板橋駅	都電荒川線	西ケ原四丁目駅	4	14	7
2	中央	月島３	都営大江戸線	勝どき駅	有楽町線	月島駅	日比谷線	築地駅	5	9	20
3	渋谷	笹塚２	京王線	笹塚駅	京王線	代田橋駅	京王線	明大前駅	6	7	17
4	杉並	高円寺南２	総武線・中央線（各停）	高円寺駅	丸ノ内線(池袋－荻窪)	新高円寺駅	丸ノ内線(池袋－荻窪)	東高円寺駅	9	3	14
5	葛飾	金町３	京成金町線	京成金町駅	常磐線	金町(東京都)駅	京成金町線	柴又駅	5	7	17
6	荒川	南千住５	常磐線快速	南千住駅	都電荒川線	三ノ輪橋駅	日比谷線	三ノ輪駅	2	10	10
7	練馬	東大泉３	西武池袋線	大泉学園駅	西武池袋線	保谷駅	東武東上線	和光市駅	4	25	24
8	目黒	鷹番１	東急東横線	学芸大学駅	東急東横線	都立大学駅	東急東横線	祐天寺駅	7	16	20
9	文京	向丘１	南北線	東大前駅	都営三田線	白山(東京都)駅	千代田線	根津駅	3	10	12
10	板橋	板橋４	埼京線	板橋駅	都営三田線	新板橋駅	東武東上線	下板橋駅	8	4	7
11	大田	西馬込１	都営浅草線	西馬込駅	都営浅草線	馬込駅	東急池上線	長原(東京都)駅	4	16	26
12	江戸川	北小岩３	京成本線	京成小岩駅	京成本線	江戸川駅	NA	NA	15	7	NA
13	港	南青山６	銀座線	表参道駅	千代田線	表参道駅	山手線渋谷駅徒歩18分	NA	11	11	NA
14	杉並	阿佐谷南３	中央線（快速）	阿佐ケ谷駅	丸ノ内線(池袋－荻窪)	南阿佐ケ谷駅	NA	NA	8	4	NA
15	墨田	緑４	総武線・中央線（各停）	錦糸町駅	都営大江戸線	両国(都営線)駅	都営新宿線	菊川(東京都)駅	10	12	10

Figure 1: “location” and “access” to categorical features

Feature Engineering

My FE is more like a brute force solution, I made over 4000 dense features by 2 kinds of “group method”, which are doing aggregations inside different categorical features. First one is “numeric to categorical”, another one is “categorical to categorical”. For the former one I used :

- Mean, max, min, median, sum, skewness, kurtosis
- IQR : q75 - q25
- IQR_ratio : q75 / q25
- MAD : Median Absolute Deviation : median(|x - median(x)|)
- Beyond1std : Calculating the ratio beyond 1 std
- Mean varience : mean / std
- Range : max - min
- Range_ratio : max / min
- HL ratio : The ratio of the samples higher and lower than the mean
- Shapiro-Wilk Statistic & Jarque-Bera Statistics
- diff and ratio : “x - mean(x)” or “x / mean(x)”
- Z-score : (x-mean(x)) / std(x)

Latter one I used :

- n_distinct : number of unique
- Shannon entropy
- freq1name : the name of most frequently appeared ones inside the group
- freq1ratio : the number of most frequently appeared ones / group size

Additionally, I also use the lat&lon information of samples and public land price data, then created ~200 new features :

- K-means with Haversine distance on lat and lon, number of cluster = 3,5,7,11,15,21,30,50,70, and treat them as categorical features
- Distance to cluster
- Spatial features : calculating the number of sample, mean of square meter, mean of number of floor of buliding, mean of How old is the building within a 100m, 200m, 500m, 1km radius. Imho it can somehow capture and extract more infomation of the location of sample.

- land price : Since the land price data only have thousands of sample, in order to join them into trainset and testset, I use a approach that is similar to way I create spatial features. For any sample in train or test, just calculating the mean of land price within a 50m, 100m, 200m, 500m radius.

Miscellaneous :

- text based counts : the counts of punctuation, digits, latin alphabet, Cyrillic alphabet, Kanji, Hiragana, Katakana on {Location + Access}
- Bag of character (2-5 grams), then apply tfidf, ~40000 sparse features in total
- Count encoding
- n-way interaction

Modeling and Ensemble

At the beginning, I feed all the features into Lightgbm (~4000 dense + ~40000 sparse). And obviously my model severely overfitting with such a big amount of features, so I just generating the order of importance from Lightgbm with full dataset, then choose top 150 features (or gain > 0.001) to train, it gave me ~1500 improvement on local CV.

By droping some group of features or change the target into "price per square meter" and repeat the steps I mentioned above, I am able to generate some diversity for ensembling. For the final blending, I just applied the optim() function in R with Nelder-Mead solver on level-0 meta-features(7 models) to find the best weight to blend, with following formula : x1 * model_1^x2 + x3 * model_2^x4 + …… + x13 * model_7^x14 + x15". It works very well both on local CV and public/private leaderboard.

Some comments

1. In this competition we are not given any time-related information. So checking the mean price of some groups may helps you to identify whether there some big difference between train and testset. Also be careful with using public land price data, same reason, we don't know the date, naively using h31 price is kinda dangerous (but I still did it :P).

2. RMSE is definitely not the ideal metric for this competition, My local CV is 14105, if I exclude the samples with high prices (>500k), the RMSE score is around ~10300, and, the RMSE on those high pricing samples is around ~170000. And there are only 123 high pricing sample in the trainset. IMHO, MAE or MAPE probably works better.

3. As I mentioned before, There's a sample with wrong labeled target. Actually I have strange habit that I'll always check the row of the most biased samples after training a model. So I soonly discovered that the sample with id=5776, the price is labeled as 1203500, but my prediction said it's around ~120000. After looking deeper into data, it seems a little bit hilarious that you need pay 1.2 million yen per month just for a small room(1K, 20.53m2). So I highly suspect that this sample have wrong target. And this is also the reason why I quit this competition in the middle stage, no one can ensure that this will not happened in the test set, especially the case that the low price is marked as high price.