1. data explore:
   1. classified variables into 4 by pattern richness and
2. month effect model: split 2016 10/11/12 month data randomly into 2 sets, train 2016 & 2017 rest of the data together with half of 10/11/12 data, predict on
   1. All data used in training.
   2. half of 10/11/12 data used in training;
   3. other half of 10/11/12 data as validation

And measure median error on each of the prediction target.

Number of samples for 10/11/12: 8542.

1. Train without sale\_month as feature:

IS predict miss median: -0.0000324

IS predict miss median on month: 0.0033405

OS predict miss median: 0.0058031

1. Train with sale\_month as feature:

factor is fairly used in prediction, avg feature importance rank 10.

IS predict miss median: -0.0000361

IS predict miss median on month: 0.0036768

OS predict miss median: 0.0053516

Road map (by time) of experiments:

1. Data exploration, classify variables into 4 categories.
2. Try heuristic feature selection. Class 1&2 / 1 / 1&2&4 / 1&2&3&4. Use all shows better performance, meaning LightGBM can well squeeze any information if exists, and well filter out not affected by useless factors. (Surprisingly, LGB can directly make use of high-cardinality features.) Full\_raw features LGB performs better IS but worse OS, use this as benchmark for all following research.
3. Naïve feature engineering: combination of dollar\_tax, tax\_dollar\_land, tax\_dollar\_structure, tax\_dollar\_total, performs better on cv but worse on testing. (params get refreshed here, but not on later fe experiments)
4. FE framework 1 & 2, brute force iterate all combinations of numerical variables grouby categorical variables + operation set = {diff from mean, abs\_diff from mean} (NOT normed here). Each time for one categorical variable, takes all numerical variables, select by feature importance:
   1. Framework 1: keep the first fixed number (100) of most important features in each round.
   2. Framework 2: keep the top percentage (80%) of most important features. A newly engineered feature will only be selected if both (1) ranks top p% within all features, (2) ranks top p% within all new features.

Without re-search of params, does not even see improvements in in-sample CV.

1. Linear-reg + LGB stacking. Found rough linear relationship between year\_built(which is a very important feature as reflected in LGB feature-importance). Since Tree is not efficient at capturing linear relationship, first extract this linear relationship between y and year\_built, then modeling the rest error with LGB. Shows no improvement from full\_raw features.

Design the concept of generic-model (a function that takes in training data and make prediction on testing data). When CV for some ideas, an existing cv function is not available, so need to write a CV function myself. More importantly, within CV all operations should be applied on training data only, leaving validation data intact to get comparable CV results (especially important for outlier handling operations). Since it is much more generic, all following research use generic model cv.

1. Outlier handling, 4 strategies:
   1. Cap / floor y outliers.
   2. Remove y outliers.
   3. Remove x outliers.
   4. Set NA to x outliers.

Without re-search of params, all shows no-better in CV performance compared to full\_raw features.

1. Combination with SVR (Linear SVR is used here as sample size too large for generic SVR to be efficiently experimented). Following combinations has been tried:
   1. SVR alone. (significantly worse than LGB)
   2. SVR + LGB stacking, SVR as first level. (same scale of performance as LGB alone, but worse.)
   3. SVR + LGB blending, using linear regression to determine weights. (significant worse performance.)
   4. SVR + LGB blending, equal weighted. (well lies in the middle of SVR and LGB alone respectively.)
2. 2-layer model: first classify samples into 2 groups by characteristic of y (soft classification, i.e. gets prob of falling into each group); then for each group predict samples as-if it belongs to the group; last linearly combine the results weighted by probs from first step. Construction of layers tried:
   1. Classify by abs error, i.e. y > quantile(0.75) and y < quantile(0.25) as one group.
   2. Classify by sign of error, i.e. y > 0 as one group.
   3. Classify by median of error, i.e. y > quantile(0.5) as one group.

Observations:

1. Abs\_error CV result is too bad to be considered.
2. Sign\_error and mid\_error shows similar performance, but both worse than full\_raw features LGB.
3. Tried equal weighted blending of sign\_error and raw\_lgb, CV gets improved but testing result gets worse. So it does not capture any significantly new structure.
4. A new design of feature engineering framework (FE3). Test one new feature at a time, for each feature combine it with raw model, we can have this new model’s CV and the new feature’s rank in feature importance. Then we sort by CV for all new features, out of those lowest error ones (should be smaller than benchmark model), pick those with high rank (can have handful of features with rank higher than 5) as new features. While we have noticeable improvements in CV, result on LB is bad, meaning it could even overfit to local CV seed.
5. Finally, we have 2017 data, with more data, local CV and LB scores gets easily (finally) improved.
6. Due to lack of time, run a reduced version of FE3, only run single-factor rank, but no CV. Only pick few highest ranked ones as new features. Does not improve LB score.
7. Second thought on class 3 (high cardinality) variables, the categories might be too high for lgb to capture pattern (need to understand details of LGB’s algo to handle categoricals though). So, let’s remove them and find a ‘best’ substitute for it. The way is to find a group of numerical variables, for a given categorical, for each numerical, create a group\_mean new feature. For this group of group\_mean new feature, add them to raw model and rank feature importance. Pick the best one as the delegate to original categorical. Non-disappointedly, all best-candidates perform significantly than original categorical. Use these new delegates of class3 vars, LB gets improved. (Maybe there is a reason Class1&2&4 performs better than raw-full in stage2).
8. Take a look at 2-layer (Sign clf + lgb) on 2-year data, no improvement on local CV, does not worth to test on LB.
9. Accident discovery!! Noticed that Q4 IS predictions are biased (first found a non-zero mean, which is not a correct angle of view though), so use a magic number to shift it will improve prediction, provided that private LB follows the same distribution as in training. Indeed the biggest jump of LB score ever, put me into medal-zone. But calm down, need to find a more systemic way to handle this effect.
10. Actually, using mean to handle it is not correct while evaluation uses MAE, the correct metric is median, meaning if we have a non-zero median, we can shift all data by reducing it to zero to improve MAE.
    1. Define y\_train – lgb\_pred = error.
    2. Noticed that with lgb\_pred:
       1. Global error median is effectively reduced to 0.
       2. Month-group error median is well away from 0.
    3. Acceptable as month is not used as a factor for training.
    4. Although, even include month as a factor, its feature importance rank goes up to 10, still does not shift month-group error median.
    5. Since direct training can shift global median to 0, then we re-train on month-group error to make it zero.
11. 2-step lgb.
    1. First train x against y, get step1\_error data set and model\_step1.
    2. Then for given month\_group, train x against step\_error, getting model\_step2.
    3. Predict for OS month\_group as model\_step1(x) + model\_step2(x).
    4. For step2, due to much smaller size of sample, use sub-sample of all features, manually selected based on data-exploration implications.
    5. Effectiveness of this method relies on same distribution of error in training month-group data and testing month-group data. So predicting public LB is fine, but whether 2016 Q4 characteristic can be projected to 2017 is a problem.
    6. When testing predicting from 2016 to 2017 on other month, only looking at median. 2step lgb does not always reduce median to 0, sometimes even make it worse. So, let’s cross fingers distribution does not change for Q4 and as mentioned in 20171016 diary, 2step lgb captures more than median.
12. A side effect is, month-group 2016-2017 bias is reduced if train with outliers removed (outliers are removed symmetrically, so should not impact on median much, unless outlier distribution differs quite much between month groups). Since from stage 6, outlier does not harm much on scores, might as well use it as a standard. Moreover, intuitively, removing outliers can better help capture patterns.
13. So, prediction from 2016 to 2017 is bad, can we make it better by make a one more step to adjust 2016 -> 2017 prediction error. Proved to be not working, as explained in diary 20171015.
14. Retrain the model! Gets very different num-boosting rounds from before, so if condition allowed, do re-train when possible when with new data / features / framework.
15. Blending with catboost. Details see diary 20171017.

20171015:

1. Class3 feature engineering, use ‘best’ group mean to substitute original high cardinal class variables, successfully improved both IS cv and OS score.
2. Step3 prediction research, does not improve final error median when using 2016 month data to train model and predict 2017 month. It is reasonable, joined year 2016 -> 2017 training does not differentiate months, so this type of joined training only learns the ‘mean’ error pattern across months. Yet according to actual 2016 & 2017 error median, it is quite different (sign) across months. So cannot expect to improve each month’s result through joint training.
3. ‘best’ number of boosting rounds can be learned by length of cv\_hist of LightGBM.cv. Re-param-search for all models again. Lower learning rates leads to more stable model (for month-train, it means smaller best\_n\_rounds std.)
4. Outlier-removed results shows better results in predicting from 2016 to 2017 after step 2

20171016

1. Param-re-search with outlier removed shows very different (larger) suggested n-rounds for lgbs, meaning previous conclusion about outlier-remove effect is not correct, should not use same parameter set for outlier-removed version. Intuitively, with outlier removed, patterns are more obvious and easier to be captured with more training.
2. Moreover, training month model on error shows further lower of cv error, which was not observed with with-outlier & rough param version, showing effectiveness of further month-train.
3. Maybe the whole research process (feature engineering, stacking, 2-layer etc.) should be started all over again with outlier removed.
4. 2 versions of final 2step lgb, one treats 10/11/12 combined as one group, the other treats as separate. The separate one performed significantly worse on LB (even with re-tuned parameter), meaning small sample size is really hard for algo to capture stuff.
5. Final 2step lgb gives 0.064144, which seems to be the limit it can push. Take a further look at its effectiveness, it even does not well adjusted median to 0 for in-sample data (2016 month error). So it does better for more than median adjustment (which it actually did very bad), maybe captured some local x-y dependence structure for each month.
6. Treasure founded, only\_catboost (an existing kernel on discussion board) itself gains a score similar to 2step-lgb, meaning it implicitly incorporated bias adjustment for each month. Directly blending it with 2step-lgb push pushes the score up to 0.064073.

20171017

1. Two version of LGB:
   1. V1: step1 train with cv iter rounds with 20% boosting considering sample size increase (ensembles of 5 param sets). Step2 train with 4 ensembles (2 iter rounds for 2 param sets).
   2. V2: step1 boosting rounds reduced to 10%. Step2 use 4 ensembles (4 iter rounds for 1 param set).
2. First, several problems with catboosting original script:
   1. Did not use 2017 props to predict 2017 submission.
   2. Unnecessarily removed 2017 tax properties for 2017 training sample processing.
   3. Used original census\_and\_block column as numerical.
   4. Break up date feature to ‘days’, we don’t have this info for OS prediction, so it has to specify a day for 2017 LB prediction and it assigned as random day of 01.
   5. Used different label for same month / same quarter of different year.
3. 5 versions of catboosting.
   1. V1: resolve problem a) and b), as a benchmark model.
   2. V2: resolve problem c), use census and block as categorical variables.
   3. V3: based on V2, resolve problem d) and e), remove ‘day’ feature use same month and quarter for 2016 and 2017, but keep feature year\_month, year\_quarter hoping it may help purify some local structure. Make separate prediction for 10, 11 and 12.
   4. V4: based on V3, combine 10, 11, 12 as one month value.
   5. V5: based on V4, remove year\_month and year\_quarter feature.
4. Takebacks:
   1. Catboost is really good at handling bias and categorical variable, its by-month IS error is directly un-biased, worth reading its tech doc on how this is done.
   2. Unlike lgb, using census and block as categoricals directly improves prediction power.
   3. Separate month predicts makes it worse, less training sample is indeed a problem. Same observation as in 2step lgb.
   4. Removing year\_month and year\_quarter further improves LB score. Comforting to see reality confirms intuition.
   5. LGB V1 performs better than V2 when combined with catboost. Hard to tell which part (step1 or step2) actually helped.
5. Concerns:
   1. Due to lack of time, no local CV is performed for catboost. Inspired by discussion board, there is another good way to validate prediction for 2017. Hold out one or two 2017 months to mimic private LB data set. This hold-out group could be switched to get a more thorough evaluation.
   2. Not enough ensembles for catboot V3,4,5 due to lack of time, again.
   3. Both 2step lgb and catboost relies month-group distribution stationarity, maybe fragile to predict from 2016 to 2017. If with one more submission option and more time, will setup second candidate as raw 1step lgb + catboost without date feature.
   4. No time to further look into the catboost kernal’s handling of outliers and data cleaning. Could be something more subtle there.