Great Energy Predictor III Competition Using Light GBM

Rex Zhang
Michtom School of Computer Science
Brandeis University
Waltham, MA
rexzjy@brandeis.edu

Xiao Zhong
Michtom School of Computer Science
Brandeis University
Waltham, MA
xzhong@brandeis.edu

Nate Zhou
Michtom School of Computer Science
Brandeis University
Waltham, MA
mzhou@brandeis.edu

ABSTRACT

In this competition, we aim to develop accurate predictions of metered building energy usage in the following areas: chilled water, electric, natural gas, hot water, and steam meters. The data comes from over 1,000 buildings over a three-year time frame. we trained model using a Light GBM and we are able to achieve a public Leaderboard RMSLE score of 1.01 without using leakage, and 0.964 using leakage, which currently ranked at 162/3428. It is worth noticing that the public leaderboard RMSLE score is calcualted using only 22% of test dataset, and due to the problem of leak data, the private Leaderboard score and final ranking of our result may fluctuate after the competition ends.

1. INTRODUCTION

Significant investments are being made to improve building efficiencies to reduce costs and emissions. Under pay-for-performance financing, the building owner makes payments based on the difference between their real energy consumption and what they would have used without any retrofits. The latter values have to come from a model. Current methods of estimation are fragmented and do not scale well. Some assume a specific meter type or donâĂŹt work with different building types. The competition seeks to retroactively measure probably future hourly energy consumption for 2 years based on past year consumption. The training data for three years hourly energy consumption is collected from thousands of buildings across 16 sites measuring electricity, chilled water, steam and hot water consumption. To aid in the analysis, building attributes such as square feet, floor count etc. are collected as well as various ambient weather conditions at the 16 sites in which the buildings under study are located.

2. DATA EXPLORATION

2.1 Initial Data Exploration

For this kaggle contest, we are provided with three main categories of datasets including the Building_meta which contains information of each individual building (site_id, primary_use, square, floor, year_build); training and testing body which contains information of the meter_id, targeted meter_reading and the corresponding timestamp; and the weather_training and testing data that provides the weather information of each site at each corresponding timestamp. The data organization of this competition is clear and up to reorganization for specific training methods. In the train dataset, the data is organized in sorted building_id sequence for chronological timestamps. And in the test and target dataset, the first 104 building are assigned additional data_id for time ahead of the provided dataset that is used for testing and submission.

The setting of the feature data is per site_id base whereas every building designated is located at certain location whereas weather report for that specific location is available. Overall, there are 15 sites which means that the building is sparsely located at 15 distinct part of United States.

However, the rather concentrating location of designated building causes trouble. Since certain building meta info is publicly accessible online, the clustering of building leads to data leakage, a problem that we will address in some later part.

2.2 Interpretation of Feature Influence as Perturbation

The data set we have containing over 1400 buildings in training set and around 100 specific buildings as target to be predicted. The energy usage in each building are intrinsically different. That is to say that all of the external feature, such as weather information, could only contribute perturbations to the intrinsic energy usage pattern of buildings. This intuitive observation is based on our first trial to directly applied lightGBM on this dataset. We will give a detailed discussion of the way we extract general pattern in the section of smooth moving average.

2.3 Data Processing

Due to the interpretation of the zero entry of the training data, it is unclear of the effect of including them into latter training process. More likely, they are not going to offer usable information for the training because following data are approximately all non-zero. Thus, to maintain the uniformity of the training data into a concrete model, we drop meter_id 0 for building 0 to 104 from timestamp earlier than 2016-5-20.

In order to incorporate our intuition into the training process, we need some data processing procedures. Basically, we use the moving average method to normalize the original data point-wise to get the perturbation information. After getting those normalized data, we could centralize them by subtracting 1 point-wise. In another word, the resulting local changing rate is the perturbation we want to look at. Then what left to do is to train model to find those perturbation for targets basing on indicators and apply the inverse

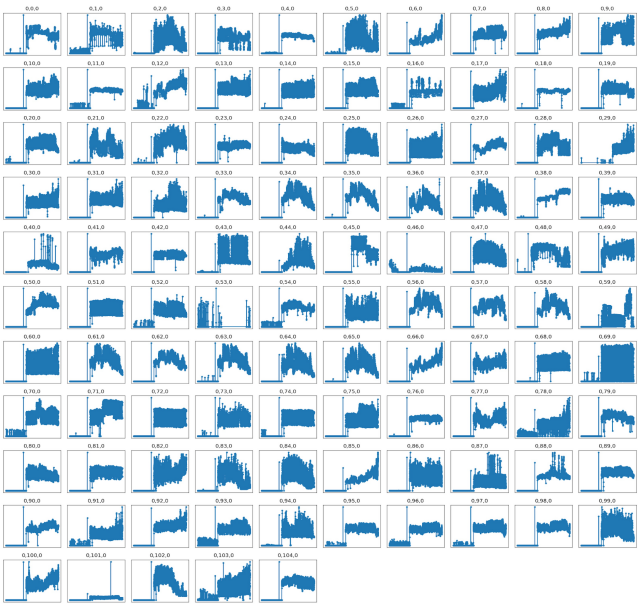

Figure 1: Training Data Visualization

Figure 2: Training Data Visualization

normalization process to re-scale the result.

2.3.1 Training Data

Training data contains over 20 million single meter_reading entry for buildings. However, a simple visualization of data has shown that for meter_0 of building ranging from 0 to 104, the meter_reading are all zero within timestamp 2016-1-1 to 2016-5-20. The summary of the training data is shown in table 1 and the 0 entry is shown in figure 1 and figure 2.

2.3.2 Testing Data

The testing data contains over 40 million single meter_reading entries corresponding to each row_id of the final submission file.

2.4 Visualization of Data

2.4.1 Training Data

We then on plot more of the training data by timestamp. The locally station tendency of the plot have reaffirmed our initial assumption of perturbation. (Figure 3, 4, 5)

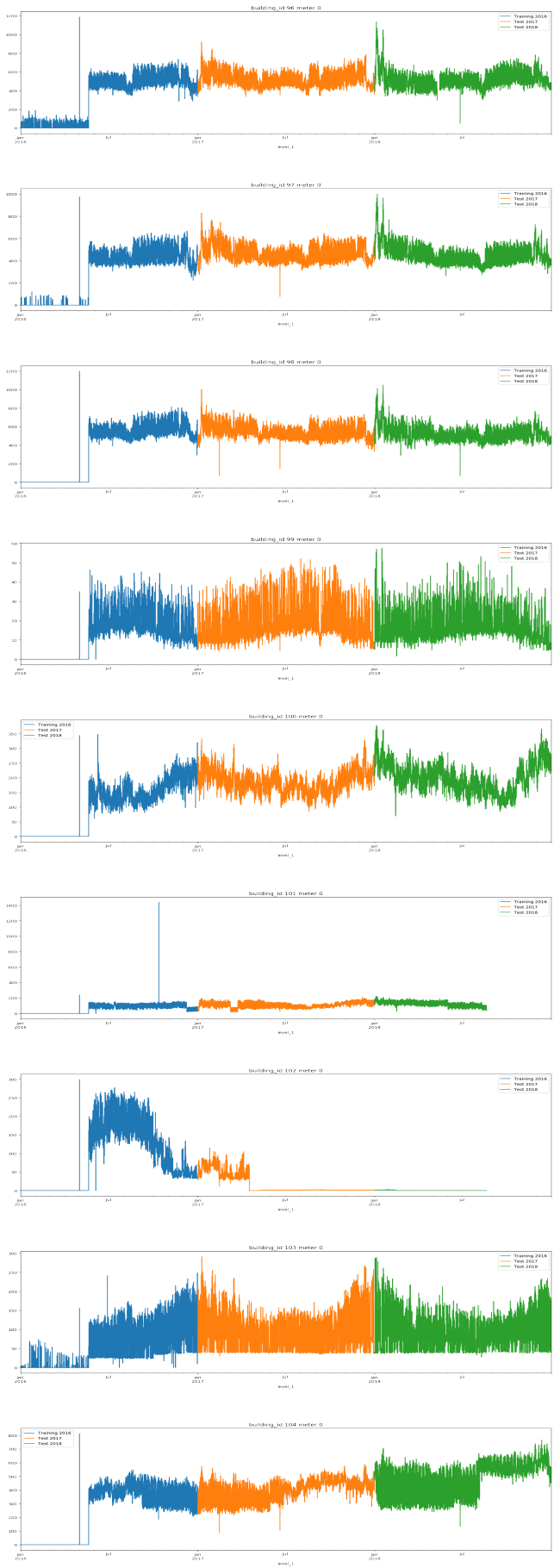

Figure 3: Training data visualization by timestamp

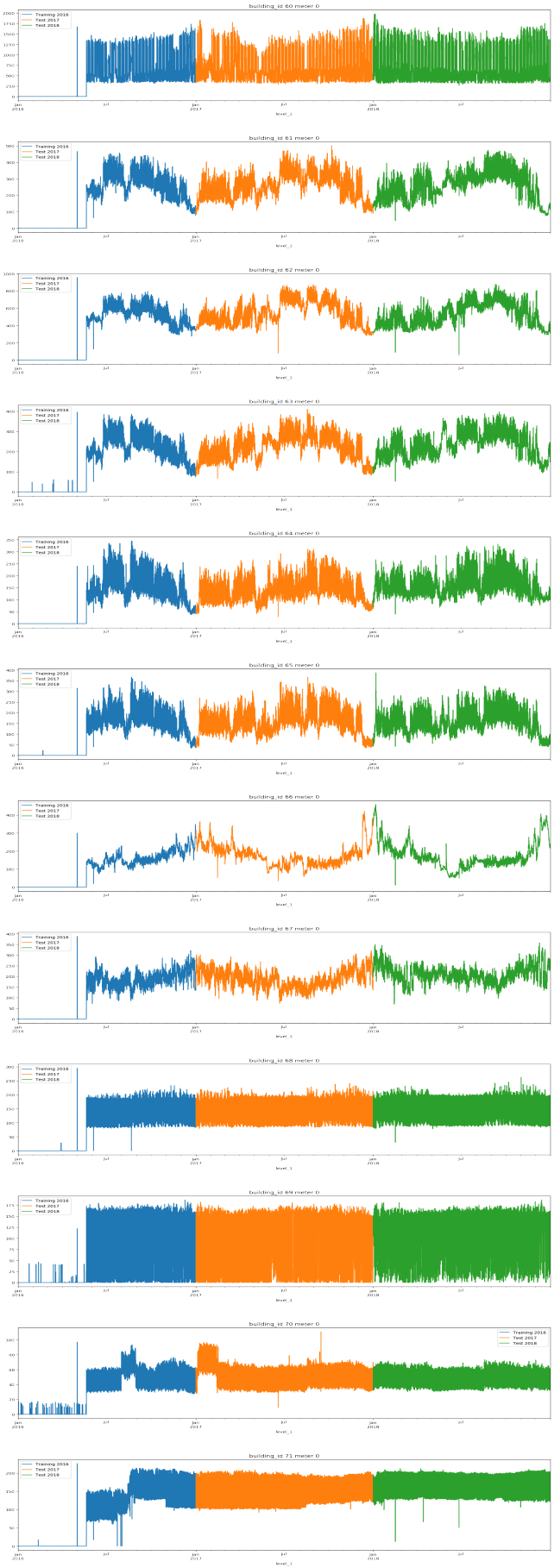

Figure 4: Training data visualization by timestamp

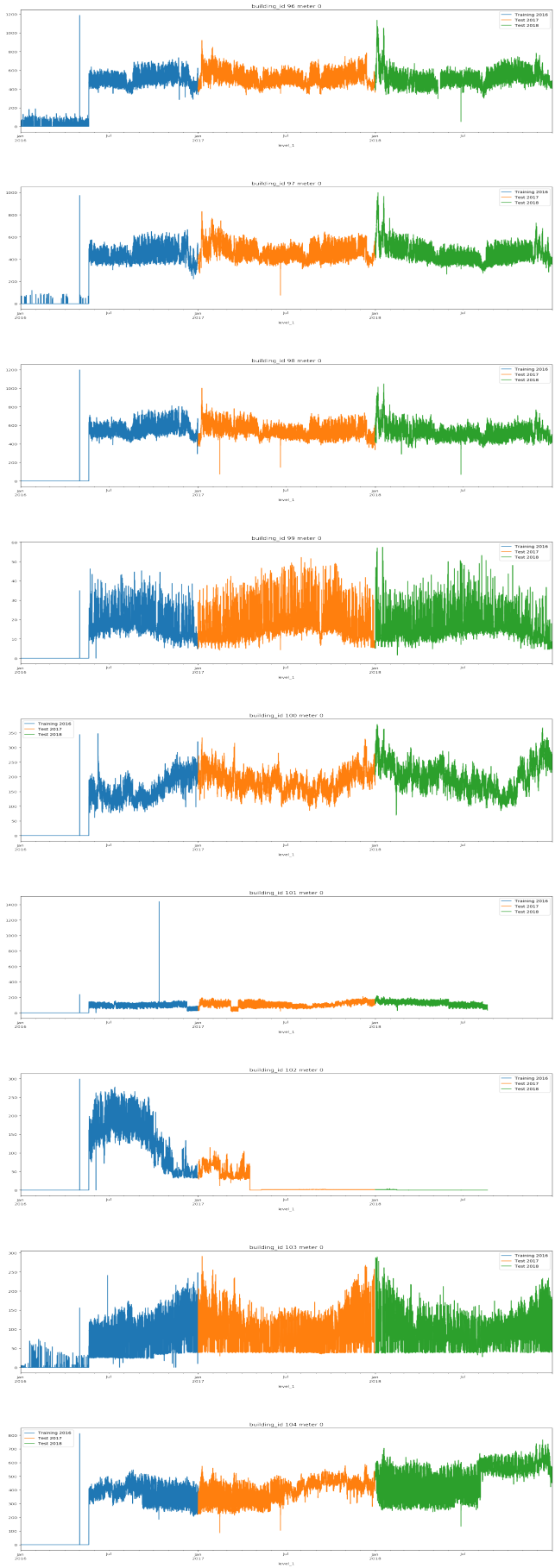

Figure 5: Training data visualization by timestamp

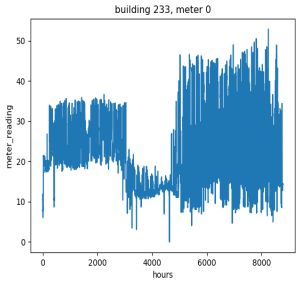

Figure 6: Original Data of building 233 meter 0

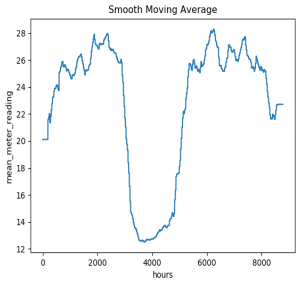

Figure 7: Smoothing Moving Average of building 233 meter 0

3. SMOOTH MOVING AVERAGE AND PERTURBATION EXTRACTION

To better capture the relations between given features and the meter reading we want to predict, we need to further process our data. For example, if we look at the data of building 233 and meter 0 and plot the meter reading column, we will see that the data behaves like a high frequency wave and it oscillates around a core curve. If we compared this with the picture we drew previously, we could tell that in general different buildings and meters will have different core curve. Thus, this core curve is a kind of intrinsic feature of those meters and buildings. We could extract this core curve from original data by computing the smooth moving average. The figure shows the example of smooth moving average of building 233, meter 0. After getting the smooth moving average, we could proceed forward to get the perturbation of the data basing on the core curve. This could be done by normalizing meter reading columns by its corresponding moving average and then subtract 1 to re-centralize. For the building 233, meter 0, we have the perturbation shown in figure 8.

Our goal, then, is to train a model to make the correct prediction of the perturbation instead of the whole meter reading. More precisely, the perturbation instead of the original meter reading serves as the feature to be learned. This is a reasonable way to train the model, since, to make use of all of those data from more than 1400 building, we need to train a model to learn general pattern related to weather and other features. While, the core curve is specific to each building and meter which is not expected to be learned by the model, so we want to extract the perturbation out of original data by normalizing basing on smooth moving average. Therefore, after training, to get the final result for those target meter reading, we add the perturbation predicted by the model back to the smoothing moving averages computed for previous year. This could be done by going through a reverse process of normalization. Notice that this is basing on the assumption that smooth moving averages or those core curves will stay stable i.e won't change too much from 2016 to 2017.

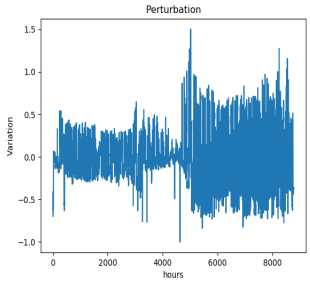

Figure 8: Perturbation of building 233 meter 0

4. MODEL STRUCTURE

4.1 LightGBM

LightGBM is a graident boosting framework that uses tree-based learning algorithm. It is one of the most successful machine learning algorithm, and is proven to be useful in kaggle competition. Since our training dataset contains more than 20,000,000 rows of data, using Light GBM would provide optimal speed, low memory usage, and better accuracy when training.

For the implementation, we first use K-fold cross-validation. Specifically, we use a 5-fold cross-validation for the training dataset and the test dataset. And then we train the model for each fold and output the predictions.

For the choice of parameters of Light GBM, due to the Bayesian nature of Light GBM, we implemented an Bayesian hyperparameter optimization function to probabilistically select the optimum values for each parameter after every round of evaluation. This approach is better than Gird search or Random search because it incorporate past results in objective functions. The implementation involves using the python module optuna. In this function, since we are dealing with regression in this task and the metrics used in the competition is 'Root Mean Squared Logarithmic Error', we define MSE as the evaluation metrics, and we optimize parameters of LightGBM through running the model on our data. In each set of of hyperparameters, we use 5-fold cross validation on the training data. The GBM model will be trained with early stopping, where estimators are added to the ensemble until the validation score has not decrease for 100 iterations (estimators added). Cross validation and early stopping will be implemented using the LightGBM cv function. For the objective function, it will return the loss of the iteration, and the algorithm will minimize this value by choosing the next hyperparameters based on the past results. Next we define the domain of paraemters. For parameters such as learning rate, it will be a uniform distribution, and we use hp.loguniform. For parameters such as num_leaves that has discrete distribution, we use hp.quniform. For hyperparameters that depend on other hyperparameters, we use nested conditional state-

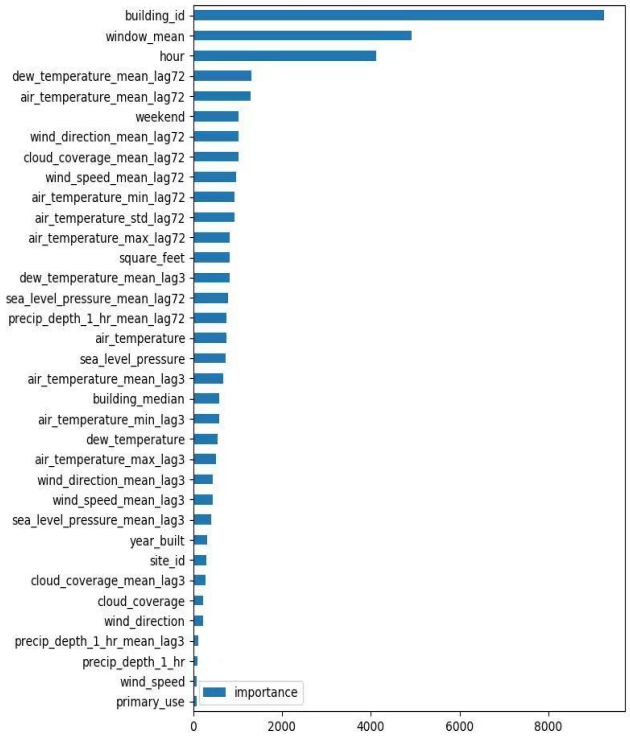

Figure 9: Feature Importance by site id

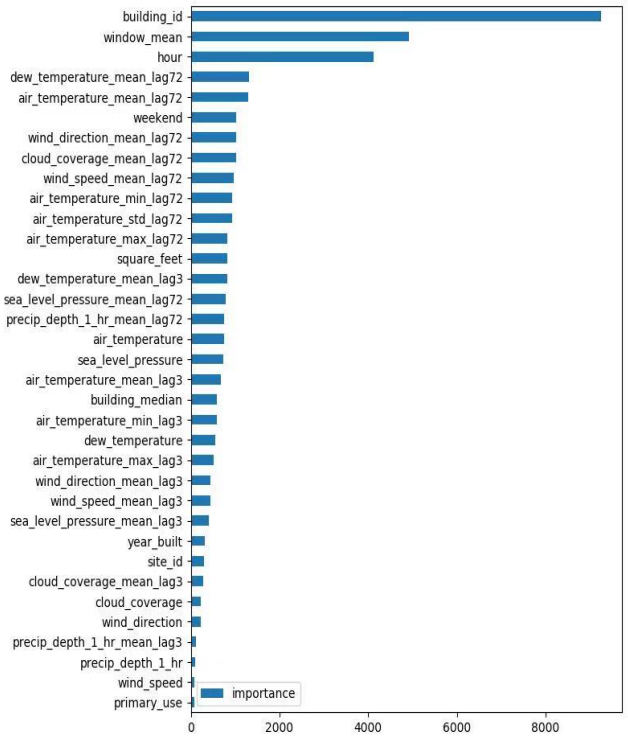

Figure 10: Feature Importance by site id

ments. Now we can define the complete space with 13 Light GBM parameters to be optimize. We use trail object to store past results, and use TPE for optimization algorithm. Now we use the fmin function to minimize our loss score and choose the best hyperparameters.

4.2 Features Importance

The feature importance graphs is presented in figure 9 to figure 17. Figure 9 to figure 13 are feature importance for separate models trained on 15 site ids, and we average models for every three site ids and obtain the results of 5 feature importance graphs. Figure 14 to figure 17 are feature importance for separate models trained on 4 different meter types. We observe that there are a lot of features that are not effective in our model. For future improvement, we will delete these features.

5. BLENDING

5.1 leakage

AS is previously mentioned in initial data exploration section, the concentration of the building designated causes public accessible leakage. In this Kaggle competition, through some participants effort, the real data for site_0, site_1, site_2, site_3, site_4, site_15 has been revealed belonging to some university campuses and publicly posted onto the forum. Undoubtedly, the newly exposed data has been revolutionary for the existing model since it extends the available data and allow for further validation of already trained model. Also, the leakage data is of help for the training of our model, since the stationary assumption that we granted for perturbation, we need real value to base our perturbation. Thus,

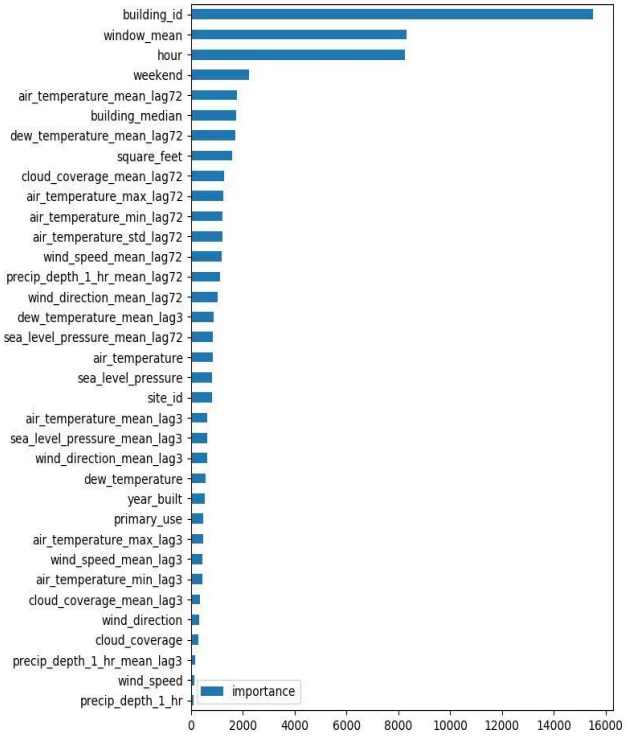

Figure 11: Feature Importance by site id

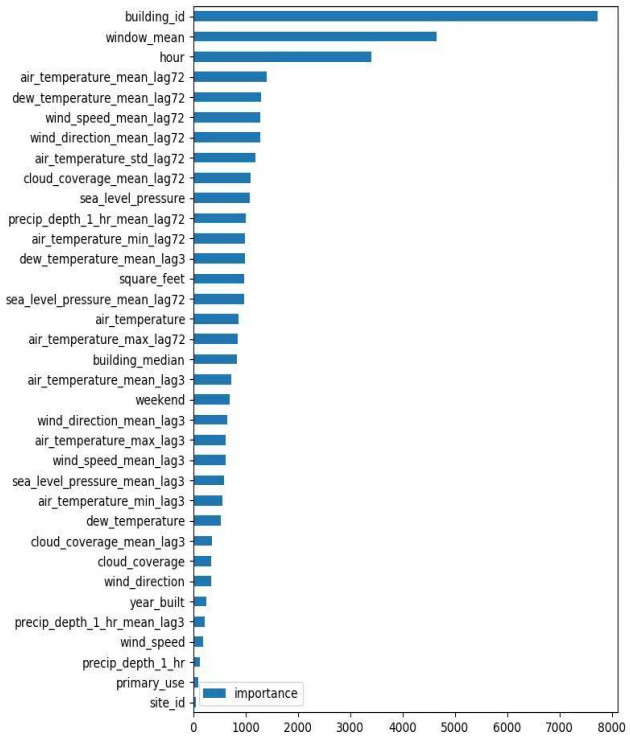

Figure 12: Feature Importance by site id

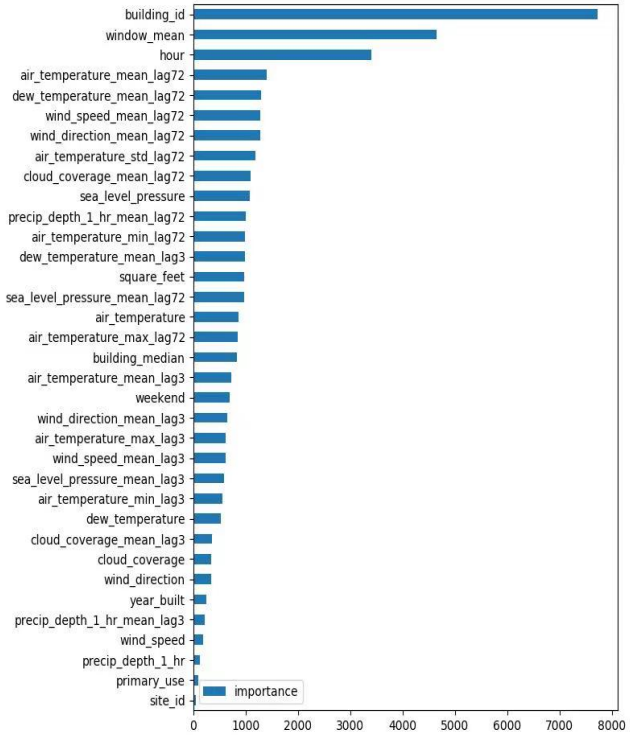

Figure 13: Feature Importance by site id

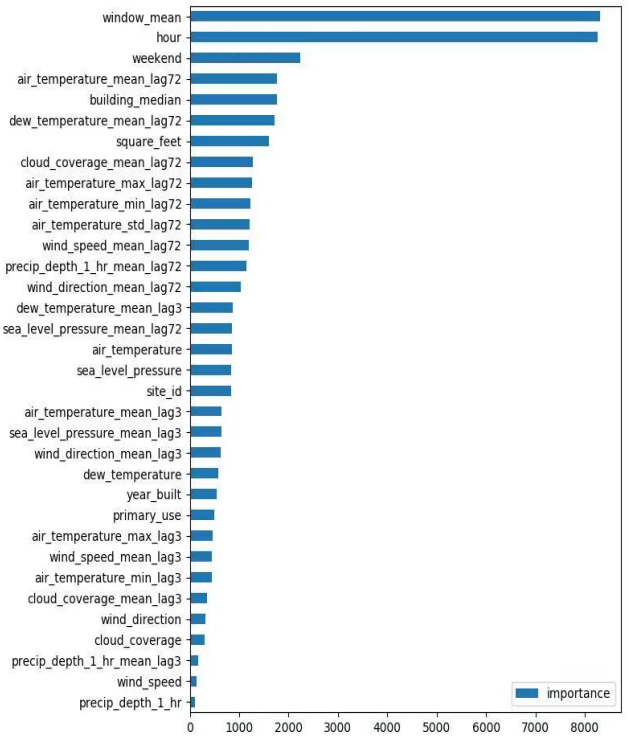

Figure 14: Feature Importance by meter type

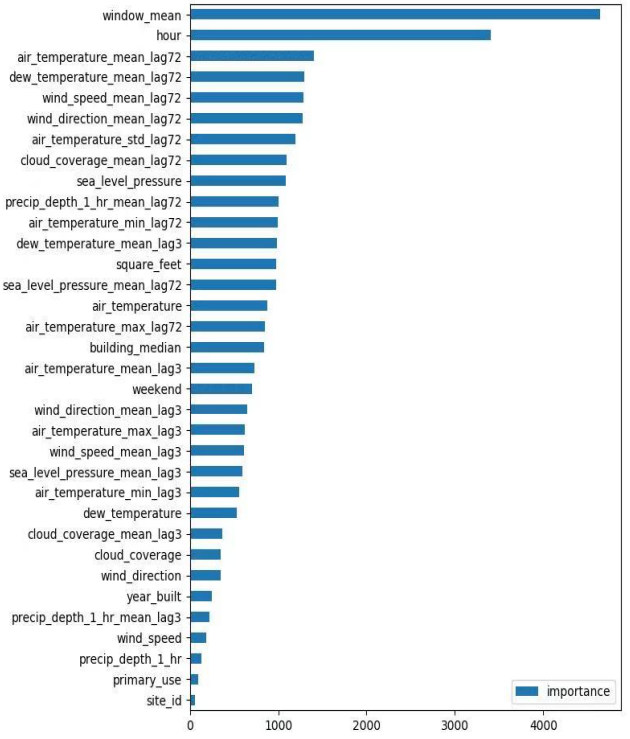

Figure 15: Feature Importance by meter type

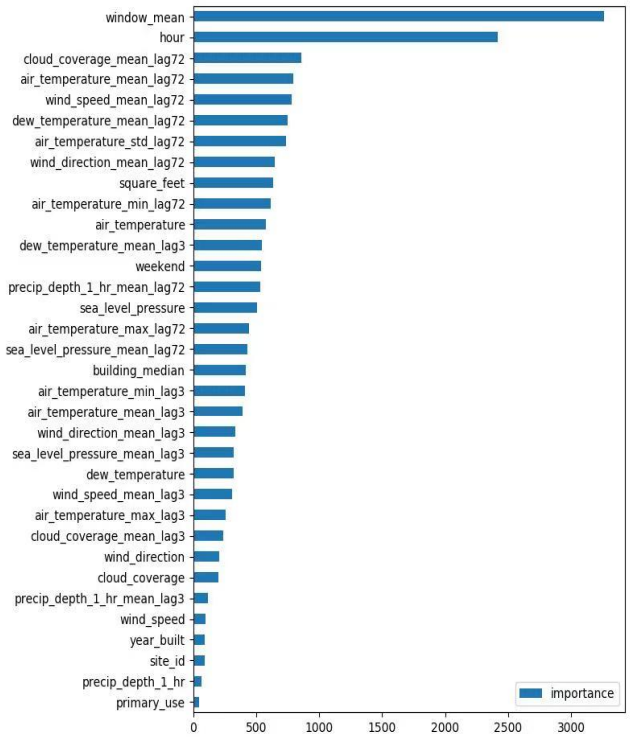

Figure 16: Feature Importance by meter type

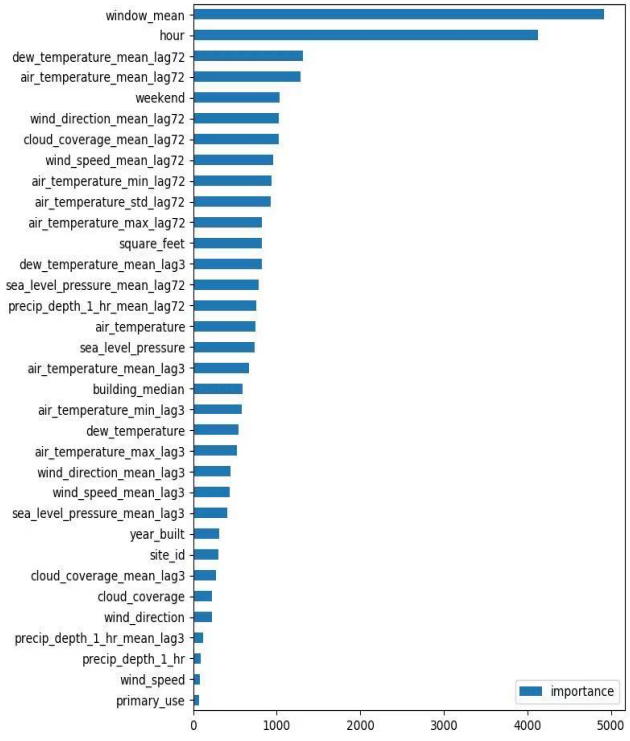

Figure 17: Feature Importance by meter type

the 0 value of meter_0 of building 0 to 104 for a considerable amount of time can be well compensated by the corresponding timestamp of data dating a year later for related meter and buildings.
Furthermore, we propose a validation method of any model based on heuristic search utilizing the leakage data by blending our trained model to search for the best weight possible.

5.2 heuristic search

For further boosting the result, we adopt a blending method basing on searching the best result within certain range.
The boosting method is kind of combing the result from different models, under the assumption that they are all giving a relatively decent result. In this case, we could expect that the results from models are just the ground-truth distribution plus noises. Our goal for adopting boosting method is to reduce the effect of these noise on the prediction task.
We blending our results by combining them linearly with coefficients in range 0.1 to 0.7 associated to each result. Then we partition this range into a grid with 50 data points and search through this 50 points to find the one giving the best outcome. In another word, we make our blending task into a optimization problem with in a linear space of coefficients and the criterion for scoring the result is log of mean square error. Notice that the error term is computed by measuring our blending results with parts of the leakage data. The leakage data made available by the sponsor only containing a small portion of the target we want to predict, but it could serve as a helpful guide for us blending. In our case, we search thorough those 50 data points of coefficients to find the one fitting best to the leakage data.

6. CONCLUSIONS

Our final result using our model achieved a public Leaderboard RMSLE score of 1.01 without using leakage, and 0.964 with the use of leakage, which currently ranked at 162/3428. It is worth noticing that the public leaderboard RMSLE score is calcualted using only 22% of test dataset. Hence the more accurate performance of our model should be evaluated with the rest of the test datasets after the competition ends. In terms of model training, we are able to achieve best results by training separate models for each site ids, and average our results. To further improve our results, we believe that training separate models for each building may be a better approach. However if we train models for each building, the limitation of data size may be a problem. Another concern that may impact the test result is the leak data. We believe that current competition participators who achieve low RMSLE score is achieved by using publicly available meter data downloaded from website that contain meter readings of buildings in the test dataset, which constitutes about 3% of the total data. The competition organizer mentioned that the private test dataset will not include meter readings that can be publicly downloaded. Nevertheless, the problem of leakage is still a controversial issue of this competition, as most participators are using internet searching and reverse engineering to achieve low RMSE score, which is not the intent of this Kaggle competition.

7. ACKNOWLEDGMENTS

Thanks to Professor Pengyu Hong, Long Sha, Ziyu Liu, and Zhaonan Li for your support this semester!