## MODEL SUMMARY

### Background

Competition Name: ASHRAE - Great Energy Predictor III

Team Name: 不用leakage上分太难了

Private Leaderboard Score: 1.235

Private Leaderboard Place: 4

[For each team member]

Name: Jun Yang

Location: Chengdu, Sichuan, China

Email: [1136337803@qq.com](mailto:1136337803@qq.com)

My team has only one member, which is myself. I'm pursuing a master's degree in electronics and communication engineering at the University of Electronic Science and Technology of China. I have done some research on power forecasting in the past. In order to get medals to improve my resume and exercise my machine learning skills, I participated in this competition. I spent two full months on the competition, from the start to the end.

### Summary

My solution is weighted blend of three single models:

1. A 2 folds XGBoost model;
2. A 5 folds XGBoost model trained by meter type;
3. A 3 folds LightGBM model shared by a public kernel.

The most important features are building\_id, air\_temperature, square\_feet and so on.

For model (1), training with GPU takes about 8 minutes;

For model (2), training with GPU takes about 24 minutes;

For model (3), training with CPU takes about 26 minutes;

So it takes about 58 minutes to train my models.

### Features Selection / Engineering

**Original Features**

For the three models, most of the original features of the data itself are very useful, including

“building\_id”, “meter”, “site\_id”, “primary\_use”, “square\_feet”, “air\_temperature”.

**Feature Transformations**

I first converted the “timestamp” to get some time features, including “DT\_hour(hour)”, “DT\_day\_week(weekend)”, “DT\_day\_month” and so on.

“DT\_hour(hour)” and “DT\_day\_week(weekend)” features are useful for every model.

I also perform log1p conversion on the “meter\_reading”.

**Aggregate features**

Then I created some aggregate features by grouping “building\_id” or “site\_id”, including “building\_id\_uid\_enc”, “site\_id\_uid\_enc”, “building\_id-m\_nunique”, “g\_meter\_site\_id\_uid\_enc”, which represent the distribution of meter types in a certain building or site.

I also created aggregated features by grouping “site\_id” and “hour”, including “DT\_w\_dew\_temp”, which represents the temperature distribution of a site in the morning, middle, and evening of the week.

**Target Encoding**

I do some aggregation on “meter\_reading\_log1p” by grouping building\_id, including “mean”, “median”, “min”, “max”, “std”, “skew”.

**Weather Lag Features**

I use weather features and sliding windows to generate lag features. The span of the sliding window is mainly 3, 18, 72. The main operations are “mean”, “median”, “min”, “max”, “std”, “skew”.

**Exception Label**

I generate the feature “exception” by labeling the outliers, but this feature is not used for training.

**Feature Selection**

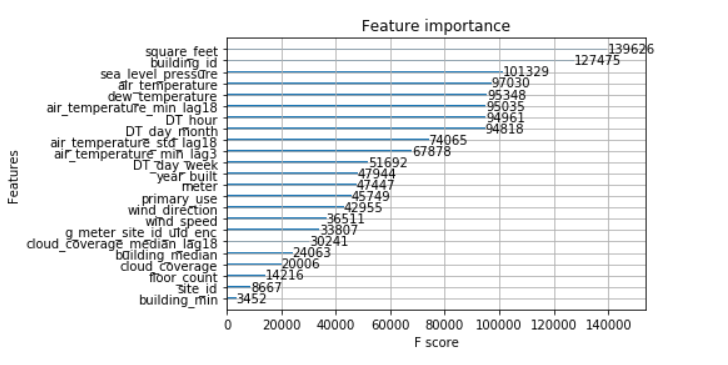
I randomly sample one-tenth of the full training set to generate a sub-training set, and then randomly sample two-tenths of the full training set to generate a simulation test set. The simulation test set is generated without using the sub-training part of the full training set. After that, I used the sub-training set and the simulation test set to generate the features mentioned in the sections above. I often generate a set of features first and then add them one by one to the training process. I only use this feature in the full training set when the oof score of the sub-training set and the score of the simulation test set increase at the same time.

I think it may be that different K values of K-fold cross-validation cause different models to have different preferences on features.

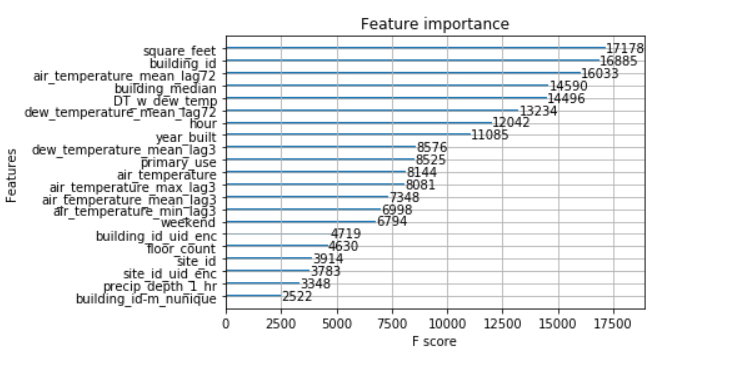
I have not used external data during the training phase.

**Feature Importance Plot**

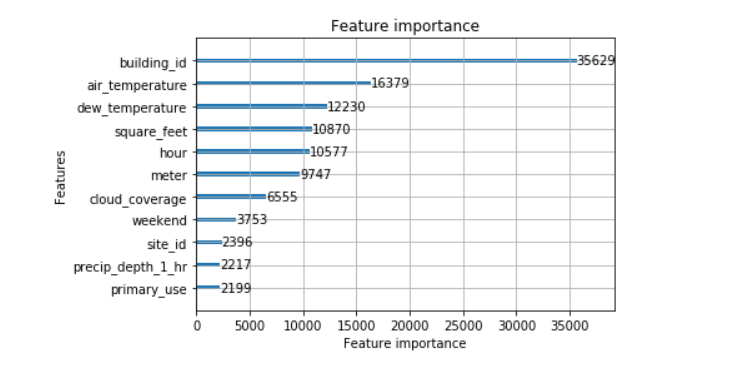
'wind\_speed' and 'wind\_direction' show interaction in the 2 folds XGBoost model, but not in other models. For a 2 folds XGBoost model, the feature importance plot is as follows:



For a 5 folds XGBoost model trained by meter type, the feature importance plot is as follows:



For a 3 folds LightGBM model shared by a public kernel, the feature importance plot is as follows:



### Training Method(s)

I trained two XGBoost models using K-fold cross validation without shuffling:

1. One is a 2 folds XGBoost model;
2. another is a 5 folds XGBoost model trained by meter type.

The difference is that during the data processing stage, I did not delete the outliers, but excluded the outliers in the current fold during the training process of each fold.

My final submission is a weighted blend of my own two models with

1. a 3 folds LightGBM model shared by a public kernel.

I used the public kernel **ASHRAE: Leak Validation and more** and oof validation score to determine the weight, and the final weight was determined as:



### Interesting findings

As we all know, the most important part of this competition is the **detection of outliers**. In the early days of the competition, I spent a week carefully observing the meter\_reading of each meter in each building and manually marking the data that I thought were abnormal, then training a simple model and making predictions, and then using the feedback from Public Leaderboard adjusted the anomaly labels, and finally got a set of anomaly labels that accompanied me throughout the competition.

In addition, since the timestamp of the weather dataset does not match the timestamp of the training dataset, **the timestamp of the weather dataset should be aligned**.

Another important thing for the competition is **post-processing**. Because the abnormal meter\_reading is mostly 0, removing the abnormal data will make the average value of the overall model prediction higher, so it is very important to use post-processing to correct this deviation. I used the public kernel **ASHRAE: Leak Validation and more** and oof validation score to determine the post-processing coefficient, and the final coefficient was determined as:



### Simple Features and Methods

**Best Single Model**

My best single model is model (2): a 5 folds XGBoost model trained by meter type, which uses 21 features, with a public score of 1.065 and a private score of 1.253. I post-processed the submission of model (2) using a post-processing coefficient of 0.9 and got a public score of 1.057 and a private score of 1.248.

**simplified model**

Since model (2) is trained by meter type, model (2) has 4 sub-models. For each sub-model, use the top ten features of the sub-model's feature importance to train a new sub-model, and finally get a new simplified model. Using the post-processing coefficient of 0.9 to post-process the simplified model prediction, we can get a public score of 1.074 and a private score of 1.250.

Each sub-model of the simplified model uses only 10 features, and the entire simplified model uses a total of 13 features.

### Model Execution Time

For model (1), training with GPU takes about 8 minutes, generating predictions with GPU takes about 5 minutes;

For model (2), training with GPU takes about 24 minutes, generating predictions with GPU takes about 5 minutes;

For model (3), training with CPU takes about 26 minutes, generating predictions with CPU takes about 28 minutes.

So it takes about 58 minutes to train my models, and about 38 minutes to generate predictions.

For simplified model, training with GPU takes about 13 minutes, generating predictions with GPU takes about 3 minutes.