Solution to the IEEE-CIS Second Technical Challenge with Machine Learning Modeling

Wenlong Wu
Electrical Engineering and Computer Science department
University of Missouri, Columbia
Columbia, MO, USA
ww6p9@mail.missouri.edu

Abstract—**Energy prediction has become an important topic in the energy field with the coming of the big data era. In the second IEEE-CIS technical challenge, we are provided with 3248 households' electricity consumption with historical half-hourly energy readings in 2017 and the goal is to predict the monthly electricity consumption in the coming year (2018). The time-series regression problems usually have two directions: statistical modeling and machine learning modeling. With the rise of the machine learning, we adopted the latter option that is more flexible and can learn the patterns from multiple households. Our solution has five steps in the machine learning modeling pipeline: data preprocessing, feature engineering, algorithm modeling, post-processing, and ensemble fusion. Details about each step are discussed in this report. In our solution, we used the Light Gradient Boosting Machine (LightGBM) model that learns the overall trends over 3248 buildings in a year and can also predict day-level energy prediction in 2018. We used Fuzzy C-Means (FCM) clustering algorithm to extract 12 clusters out of the 3248 buildings and built a LightGBM model for each cluster. Our final ensemble prediction achieves the lowest root Absolute Error (rAE) score and ranks as number one on the platform leaderboard.**

Our code (in Python) is publicly available on: https://github.com/waylongo/cis-challenge-energy-prediction.

I. Introduction

Energy prediction is a time-series regression problem. Usually, there are two approaches to solve this problem: statistic modeling and machine learning modeling. Statistic modeling runs on one-dimensional data and makes a prediction by decomposing the data into trends, seasonalities, and noise components. At the early stage of the competition, we tested ARIMA [1] and Facebook Prophet [2] models on the competition data and failed to leverage these statistical models in this competition. The reason why they do not perform well is that the provided data has a different range of monthly availabilities. For the households that meter readings started in December, the statistical models cannot learn the monthly trends in a year, and hence failed to decompose the data. Therefore, we moved to a machine learning modeling that can learn overall trends and patterns among 3248 households.

We followed the machine learning modeling pipeline: data preprocessing, feature engineering, algorithm modeling, data post-processing, and ensemble fusion. After we tested the statistical algorithms, we built a Light Gradient Boosting Machine (LightGBM) [3] model on the raw day-level energy data and used it as the baseline model. The root Absolute Error (rAE) score of the baseline model is pretty large because there are plenty of missing data filled with zeros and null values in the raw data that can destroy performance. We wrote some scripts to remove the missing data automatically and improved the rAE score significantly. However, there are some remaining corner cases of strange sensor patterns (outliers) like sudden energy reading increasing or decreasing, we could have created some rules to remove them but there are not many of them so we went through the household meter reading plots and removed some periods of data of 65 households manually. The rAE score was improved slightly by the manual sensor reading removal.

At the exploratory data analysis stage, we ran the t-distributed Stochastic Neighbor Embedding (t-SNE) [4] algorithm on the normalized month-level training data and found there are multiple clusters in the reduced-dimensional feature space. We ran Fuzzy C-Means (FCM) [5] with cluster number c=12 on the normalized month-level training data and found 12 well-separated clusters. During the modeling stage, we built a LightGBM model for each household cluster because the households within a cluster are like each other and share similar usage patterns. Besides the cluster-level analysis, we also ran a single LightGBM algorithm on all the household data, and this trained LightGBM model learned the overall trend among 3248 households in a year. Both cluster-level models and whole-level model have decent rAE scores. Near the end of the competition, we discovered that the sensor battery failure may also happen in the test set, so we created several rules to detect battery failure by checking the last few days' sensor readings in 2017. Detecting the failed sensor and setting them zeros improved the rAE score because these failed sensors have large weights contributing to the overall rAE score. We fused three predictions (cluster-level prediction, whole-level prediction, and the prediction using each household's average sensor readings value of November and December in 2017) in the end, and achieves the lowest rAE score on the platform leaderboard.

The rest of the report is organized as follows. Section II briefly reviews FCM and LightGBM. Section III introduces our solution methodology and Section IV discusses our approach's progression and advantages.

II. Background

II.A. Fuzzy C-Means

The Fuzzy C-Means (FCM) [5] is defined as the minimization of the following objective function:

$$J_{FCM}(U,V;X) = \sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik}^{m}||v_i - x_k||^2 \quad (1)$$

with the constraints

$$\sum_{i=1}^{c} u_{ik} = 1 \quad (2)$$

for all $k = 1, \dots, n$. Here, the fuzzifier parameter $m \in (1, \infty)$. Optimization of the FCM model can be performed by randomly initializing V and then alternatively updating U and V (alternating optimization) using the necessary conditions for extrema of J_{FCM},

$$u_{ik} = \left(\sum_{j=1}^{c}\left(\frac{||v_i - x_k||^2}{||\|v_j - x_k\|^2}\right)^{\frac{1}{m-1}}\right)^{-1} \tag{3}$$

$$v_i = \frac{\sum_{k=1}^{n} u_{ik}^m x_k}{\sum_{k=1}^{n} u_{ik}^m} \tag{4}$$

until a suitable termination criterion holds, for example when successive estimates of V change less than a threshold ε.

$$\max_{i=1,\dots,c} ||v_i - v_i'|| < \varepsilon \tag{5}$$

II.B. Light Gradient Boosting Machine

Light Gradient Boosting Machine (LightGBM) [3] is a gradient boosting framework. It makes use of tree-based learning algorithms that are considered to be very powerful when it comes to computation. There are two primary methods in ensemble learning: bagging and boosting. Bagging involves the training of many independent models and combining their predictions through some form of aggregation. Boosting instead trains models sequentially, where each model learns from the errors of the previous. Starting with a weak baseline, models are trained iteratively, each adding to the prediction of the previous to produce a strong overall prediction. In the case of gradient boosted decision tree, successive models are found by applying gradient descent in the direction of the average gradient, calculated with respect to the error residuals of the loss function of the leaf nodes of previous models. While other such tree-based approaches grow horizontally, the LightGBM algorithm grows vertically meaning it grows leaf-wise instead of level-wise. LightGBM chooses the leaf with large loss to grow. It can reduce more loss than a level wise algorithm when growing the same leaf. LightGBM was developed by the Microsoft team in 2016, and has a well-designed API [6], which is what we used in the competition.

III. Methodology

III.A. Solution Architecture

Figure 1 shows the overall architecture of our solution. The blue boxes represent data, the orange boxes represent models and the green boxes represent predictions.

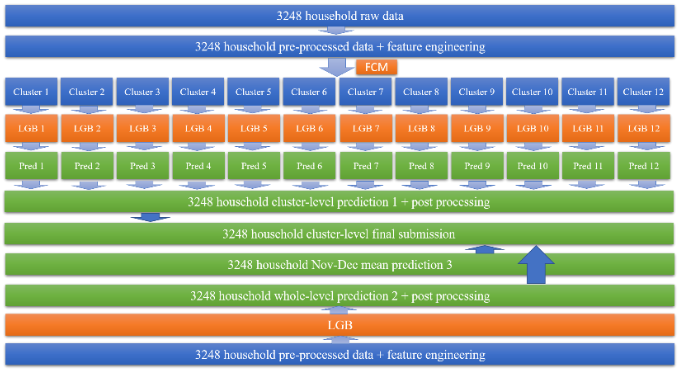
3248 household raw data
3248 household pre-processed data + feature engineering
FCM
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Cluster 10
Cluster 11
Cluster 12
LGB 1
LGB 2
LGB 3
LGB 4
LGB 5
LGB 6
LGB 7
LGB 8
LGB 9
LGB 10
LGB 11
LGB 12
Pred 1
Pred 2
Pred 3
Pred 4
Pred 5
Pred 6
Pred 7
Pred 8
Pred 9
Pred 10
Pred 11
Pred 12
3248 household cluster-level prediction 1 + post processing
3248 household cluster-level final submission
3248 household Nov-Dec mean prediction 3
3248 household whole-level prediction 2 + post processing
LGB
3248 household pre-processed data + feature engineering

Fig. 1. Solution architecture

III.B. Data Preprocessing

In this competition, we are provided with historical half-hourly energy readings of 3248 smart meters in 2017 but need to make predictions on the month-level, so the hour information is not that crucial for our understanding. We sum up each day's meter reading to do the day-level prediction. One example of the day-level smart meter readings plot is shown in Figure 2.

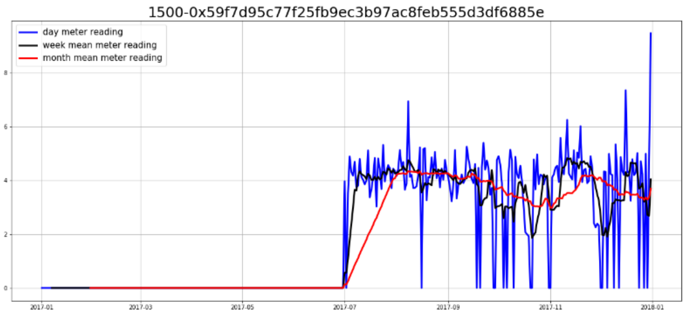

Fig. 2. One example of day-level smart meter readings plot

After we computed the day-level smart meter readings, we produced week mean meter readings (black line) and month mean meter readings (red line), as plotted in Figure 2. We can see from Figure 2 that not all households have data for all months in a year. Some households might start having records in June while others start having records in October, or even December. There are many missing data in the original training set, so we discarded most of the zero-reading and null-reading by removing them in the training if their 3-day window of mean meter readings are zero. That is, we removed a day's meter reading if it and its two neighboring days had zero or null meter readings. This is because the meter was not deployed or broken for some reason. In the example in Figure 2, all zero-readings before July were removed by this missing data removal method. This mechanism helps us remove most of the missing data (filled with zero and null in the training data). Besides, we went through each meter readings plot and removed 65 chunks of meter readings manually because there are some corner cases (outliers) in some households that sudden meter readings increase or decrease. We may have created some rules to remove those unusual sensor readings (outliers) but there are not many of them so we decided to manually remove them, which is more efficient and flexible compared with setting up rules if the data is small. Note that most meaningless data was removed by the automatic missing data removal mechanism.

III.C. Feature Engineering

Besides the historical meter readings in 2017, we are also provided with the weather data in 2017 (training data). However, the weather data in 2018 (testing data) is not provided, so it is impossible to leverage this information using machine learning modeling. We are also provided with additional information about households. However, more than 90% of the data in the provided table is missing, and so we did not use that information either. In this competition, we ended up using six features including categorical household ID, mean meter reading of the household, and time-related features like "day of week", "day of month", "month". We used cyclical feature encoding by taking the sine and cosine value of the month value so that January encoding and December encoding are similar.

Figure 3 provides a similarity visualization on the month feature cyclical encoding.

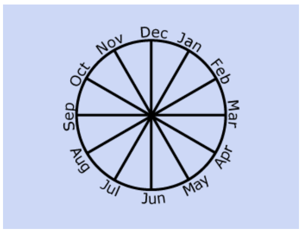

Fig. 3. Cyclical encoding on "month" feature

III.D. Algorithm Building

Before we ran algorithms on the training data, we built the t-SNE visualization on the min-max normalized month-level training data, as shown in Figure 4 (a.). The min-max normalized month-level training data is in 12 dimensions where each dimension represents a month's consumption data, filled with zero if missing.

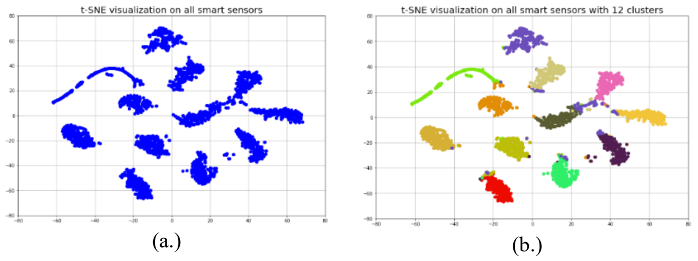

Fig. 4. (a) t-SNE visualization on the min-max normalized month-level training data (b) t-SNE visualization on the min-max normalized month-level training data colored by FCM clusters

As we can see in Figure 4 (a.), there are multiple clusters in the reduced-dimensional feature space. We ran fuzzy c-mean (FCM) algorithm with cluster number $c = 12$ on the min-max normalized month-level training data. The clustering result is shown in Figure 4 (b.). We carefully examined each cluster and found that they are relevant to the available months. So, during the modeling stage, besides building one single LightGBM model on all household meters, we built 12 separate LightGBM models for each cluster because each cluster has a different range of monthly availability and shares the same sensor pattern.

For local validation, we used shuffled stratified k-fold validation [7] with k=3 in this competition. That is, we used 2/3 of each household data for training and the remaining 1/3 of each household data for validation in each fold. The final prediction is the average of each fold's prediction.

The implementation of the LightGBM algorithm we used in this competition is from Microsoft research team [6]. The hyper-parameters we used for whole-level modeling and cluster-level modeling are included in our public code repository.

III.E. Post Processing

Post-processing has become an important component in data science competitions because there are usually some failed predictions by machine learning models that are not due to the underfitting of the models but due to the flaws in the data. One challenging problem we have in real-world sensor prediction is the sensor battery failure problem. We might also have this problem in both training data and testing data. Figure 5 shows a sensor battery failure example.

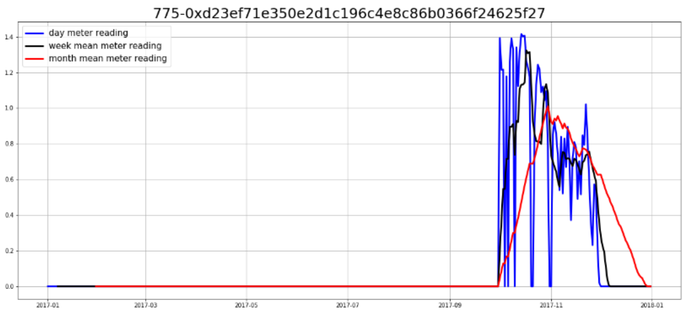

Fig. 5. One example of sensor battery failure.

As we can see in Figure 5, the sensor started in October 2017 and ended in November 2017. There are all zero readings in December 2017, which is a very unusual phenomenon. It was very likely that this sensor battery ran out or failed to work for some reason in December 2017. This sensor would continue to report zero until someone replaced the battery or fixed the sensor.

We created two rules to detect this type of sensor failure problem:

- Rule 1: If a household's last 31 days of sensor meter readings in 2017 are smaller than a threshold (a small number), we consider this household's sensor failed to work and make zero prediction for this household meter reading in 2018.

- Rule 2: If a household's last 31 days of sensor meter readings in 2017 are 20% lower than the average sensor meter readings in 2017, we consider this household's sensor problematic, and make predictions based on the latest month (November and December)'s sensor meter readings.

However, some sensors might resume working if people replaced the battery or fixed the broken sensor. The battery replacement by a human is an unpredictable behavior in the real world, so machine learning models cannot know this unpredictable behavior. With the above two rules, we detected around 32 suspicious sensors that might have the sensor failure problem.

In the next step of post-processing, we performed prediction smoothing because neighboring months should have similar energy consumption.

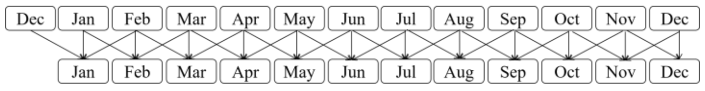

Fig. 6. Prediction smoothing scheme

Figure 6 shows the prediction smoothing scheme we used in this competition. For example, the smoothed "Apr" prediction is the average energy consumption value of "Mar", "Apr" and "May".

In the final step of post-processing, we did some up-scaling to the winter season (Nov., Dec., Jan. Feb. and Mar.) by multiplying an up-scaling scalar value that is larger than 1 (usually smaller than 1.15) because energy consumption increases on the cold days. Likewise, we did some down-scaling to the summer season (May., Jun. Jul., Aug., Sep.) by multiplying a down-scaling scalar value that is smaller than 1 (usually larger than 0.85) because energy consumption decreases in the warm days. The scaler value is a hyper-parameter that can be tuned.

III.F. Ensemble Aggregation

The final ensemble submission is the weighted average of three submissions. The first submission is generated by a single LightGBM model on the 3248 households. The second submission is generated by 12 separate LightGBM models on the 12 clusters of households. The last submission is generated using each household's average sensor readings value of November and December in 2017.

IV. Discussion

IV.A. Progression

The original training energy consumption data has lots of zero-reading and null-reading values so it is impossible for machine learning models to learn the patterns from the original raw data. We obtained a large root Absolute Error (rAE) score when running the LightGBM model on the raw energy data (rAE is around 2.20). After we ran the automatic data pre-processing that removes the data if their 3-day window of mean meter readings are zero, we got a significant improvement on the rAE score (rAE is around 0.92) because most of the meaningless energy readings were removed. We also manually removed 65 periods of household energy consumptions because of strange patterns, and the rAE score improves a little (rAE is around 0.87). We detected 32 suspicious meters using the two rules discussed in section III.E to detect battery depletion and sensor failure. We lowered the rAE to around 0.76 after setting them to zero, and got around 0.72 rAE after adjusting some of them to be the mean value or December value manually. The rAE value dropped to around 0.69 after applying prediction smoothing and scaling. The final ensemble of three submissions (cluster-level prediction, whole-level prediction and November-December mean constant prediction) produced the lowest rAE score (around 0.65) on the platform leaderboard because of model diversity.

IV.B. Our Approach Advantages

Our LightGBM model runs on the day level so it can not only provide monthly prediction but also daily prediction. Figure 7 shows a visualization of one household's monthly prediction and daily prediction.

As we can see in Figure 7, we only have one month (December) of training data for this household. The trained LightGBM model learns the overall trends of the 3248 households over one year so it can make a whole year's prediction even though only one month of data was given for this household. The bottom graph in Figure 7 also provides more accurate daily predictions for this household. Our final prediction achieves the lowest root Absolute Error (rAE) score and ranks as number one on the platform leaderboard.

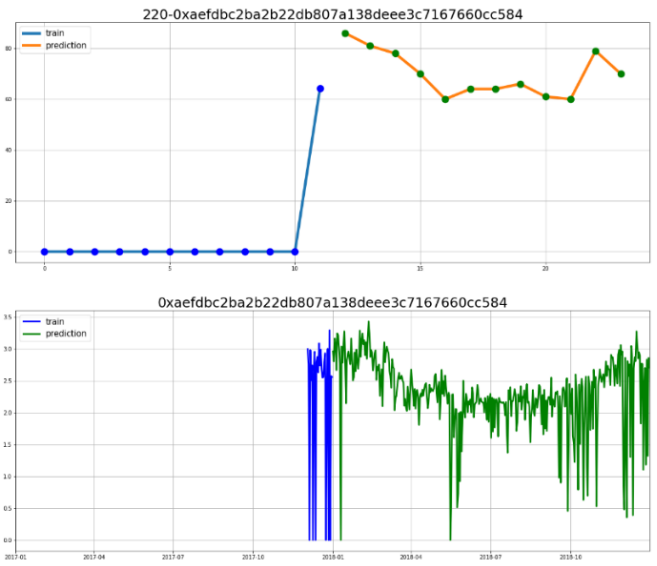

Fig. 7. One household's prediction visualization on monthly prediction and daily prediction

IV.C. Use of Computational Intelligence Techniques

The Computational Intelligence technique that we used in this competition is the fuzzy c-means (FCM) algorithm [5] that was proposed by Dr. James Bezdek. The FCM algorithm helps us find 12 clusters out of 3248 households at the exploratory data analysis stage. After studying each cluster carefully, we discovered each cluster is relevant to the different month availability so that we built a LightGBM model for each cluster and improved the rAE score in the final ensemble.

Acknowledgment

The author would like to thank Dr. James Keller and Dr. Mihail Popescu for their valuable comments and suggestions on this technical challenge. The author would also like to thank IEEE CIS, E.ON SE, and Dr. Isaac Triguero for hosting the technical challenge.

V. References

[1] S. Hillmer and G. Tiao, "An ARIMA-Model-Based Approach to Seasonal Adjustment", Journal of the American Statistical Association, vol. 77, no. 377, pp. 63-70, 1982. Available: 10.1080/01621459.1982.10477767.

[2] https://github.com/facebook/prophet

[3] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu. "Lightgbm: A highly efficient gradient boosting decision tree." In Advances in neural information processing systems, pp. 3146-3154. 2017.

[4] l. Maaten, and G. Hinton. "Visualizing data using t-SNE." Journal of machine learning research 9, no. Nov (2008): 2579-2605.

[5] J. Bezdek, R. Ehrlich and W. Full, "FCM: The fuzzy c-means clustering algorithm", Computers & Geosciences, vol. 10, no. 2-3, pp. 191-203, 1984. Available: 10.1016/0098-3004(84)90020-7.

[6] https://github.com/microsoft/LightGBM

[7] https://www.geeksforgeeks.org/stratified-k-fold-cross-validation