Prediction of Energy Consumption of Buildings based on Building Characteristics and Weather Data

Prabesh Dhakal[1]

[1]*Registration Number: 3032362, Email: prabesh.dhakal@stud.leuphana.de*

Abstract

In order to predict the energy consumption of buildings that use different types of energy devices and that are located at different sites, a gradient boosting approach named CatBoost is applied. The data is processed using interpolation and smoothing techniques to reduce noise. Feature engineering is performed to identify important features. Finally, models are fitted to sites and energy meters separately, which are then averaged to create a more robust model. The results indicate that CatBoost produces satisfactory outcome with lower effort required in handling categorical features. However, which model is chosen at the end depends on the error metric used to evaluate the models. This and further limitations are discussed.

Keywords: energy consumption prediction, regression, gradient boosting, CatBoost

1 Introduction

In late 2019, American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE)[1] organized a competition on Kaggle, a data science platform, in which the objective was to build a model that can predict energy consumption of buildings using facts about the building and time indexed weather information.[2] The last time the association organized a similar competition was over 25 years ago.

Electrical energy production and usage is one of the most important activities as it powers many aspects of modern economies. Of all the electricity produced in the year 2018, 74 percent of the production involved non-renewable sources. In addition, industry and buildings account for 90 percent of the energy consumed in the year 2018 (International Energy Agency 2019). ASHRAE claims that with better estimates of energy-saving investments, large scale investors and financial institutions will be more inclined to invest in this area to enable progress in building efficiencies (ASHRAEa 2019). This competition plays an important role in advancing the research landscape in the area of energy consumption (ASHRAEa 2019) by allowing tests of "state of the art" machine learning techniques in the domain of building energy use (ASHRAEb 2019).

A lot of work can be done in the area of research with regards to the improvement of efficiency in the production and consumption of electricity (Schaeffer *et al.* 2012; Catenhusen 2004). Machine learning models have gained popularity in the energy industry where they are applied for tasks like demand prediction, cost prediction, wind speed estimation, storage panning, etc. (Mosavi *et al.* 2019).

This paper has been structured in the following manner: Section 2 contains information about the data, Section 3 details the approach used to deal with the data, the training method, validation process, as well as the error metrics used, Section 4 contains the results of the project, and discussions are presented in Section 5. Appendices are presented at the end and contain relevant information pertaining to the paper.

Machine Learning Lab (WiSe2019/20) (2020)

Submission Date: 2020 Mar. 02

1. ASHRAE is an American professional association that operates in ore than 132 countries as of early 2020 and works on advancing the design and construction of heating, ventilation, air conditioning and refrigeration systems.
2. Link to the competition page: https://www.kaggle.com/c/ashrae-energy-prediction

2 About the Data

The data set consists of three different files that contain *building metadata*, *time indexed weather data*, and *time indexed energy usage data*. The data contains information about 1449 buildings that are used for various purposes - such as education, residence, office, etc. - and are spread throughout 16 different sites (referenced with `site_id`). Following are the columns that each of the files contained; see Appendix I for description on each of them:

1. building_metadata.csv:
   `site_id, building_id, primary_use, square_feet, year_built, floor_count`
2. weather_train.csv:
   `timestamp, site_id, air_temperature, cloud_coverage, dew_temperature, precip_depth_1_hr, sea_level_pressure, wind_direction, wind_speed`
3. train.csv:
   `timestamp, building_id, meter, meter_reading`

The resulting data frame contains 20.2 million rows, the same number of rows as `train.csv`. To obtain one single table on which the algorithm could be trained on, a table merge was performed taking `building_id` and `timestamp` as the *primary keys*.

The target variable is `meter_reading` which is the amount of energy (in kWh) consumed by a given building in a given hour interval. In addition, there are four different types of meters for different use cases, each that could apply to a given building: chilled water, electric, hot water, and steam meters.

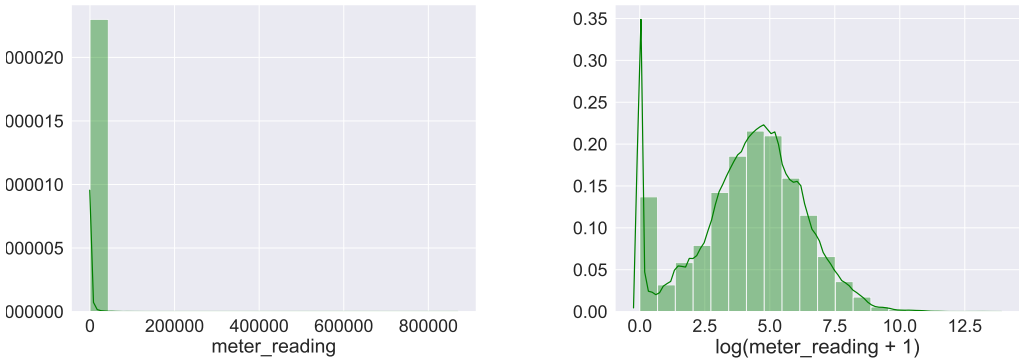

Figure 1. Distribution of the raw target variable (left) and log transformed target variable (right).

The `meter_reading` variable from the training data has strong positive skew with most of the observations falling at zero energy because meter readings are by their nature non-negative. Even after performing a $log(1 + x)$ transformation, the meter reading variable has a slight positive skew with a considerable proportion of the observations lying at 0 as can be seen in the Figure 1.

3 Method

This section contains information on how the project was carried out.[3] The modeling process has additionally been summarised in Figure 2. Details on data processing and feature engineering, selection of the training method, training and tuning the model, and final evaluation have been discussed in individual subsections.

3. Link to the code: https://github.com/prabeshdhakal/mll_predicting_energy_consumption

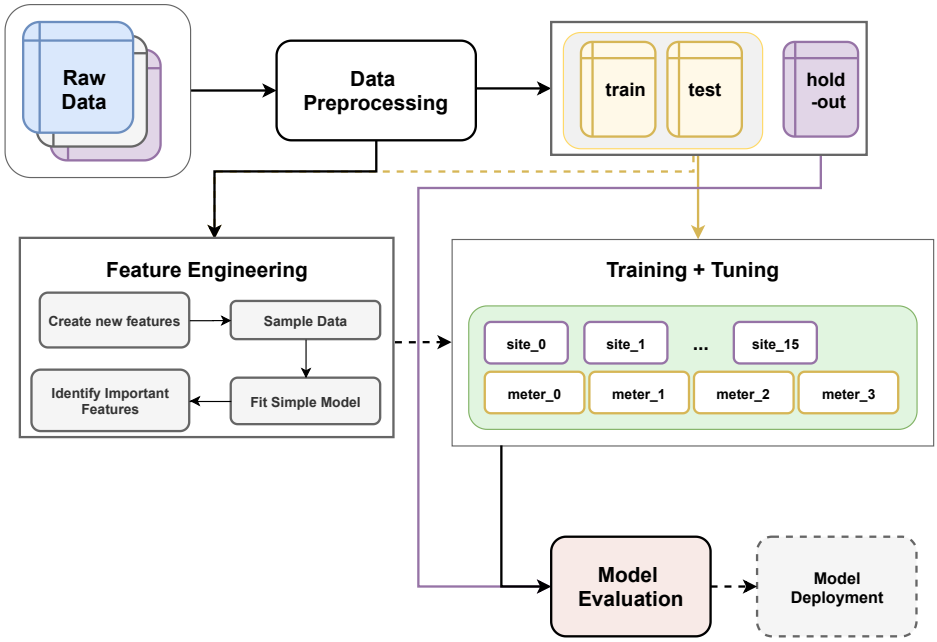

Figure 2. Overvew of the modeling process.

3.1 Data Preprocessing and Features Engineering

The weather data file was preprocessed first. Since the data is timestamped, interpolation was identified as an appropriate method for dealing with missing values. Furthermore, the temperature and pressure related data was smoothed using the Savitzky-Golay Filter. Savitzky-Golay Filter is a method of smoothing noisy signal which preserves moments of higher orders better than other models, preserving the width and amplitudes of the peaks (Savitzky and Golay 1964). Its use in this step was motivated by its application in similar domains in many tasks related to time indexed signals (Barak 1995; Schettino, Duque, and Silveira 2016; Oliveira *et al.* 2018).

After dealing with the weather data, the remaining files (building_metadata.csv and train.csv) were loaded and combined to form a single data table. This stage of data preprocessing involved the identification of anomalies such as long streaks of missing values and outliers. Site 0 in particular did not have any meter reading for the first 140 days. Hence, the corresponding rows were deleted. Also, one of the buildings had exceptionally high values for meter reading. These rows were removed as well. Additionally, some new features were created; most notably: flags for features with missing values in the weather data and time series related features such as day of the month, day of the week, etc. Then, the table was split into 3 parts, one for training, one for test (used for validation of the trained models), and one as a holdout for final model evaluation. Each of these files were then further split into 16 files - one for each site- and 4 files - one for each type of meter. The train and test files were used for feature selection, training, and tuning of the model.

In order to remove unimportant features for each site and for each meter, the training files were sampled, and a CatBoost regression model (discussed in the next subsection) was fitted to each of them. Lastly, feature importance was recorded from these models.

3.2 Training Method Selected

Given the dataset provided in the competition, the task of predicting energy consumption is a regression problem that requires a model that can handle a combination of numeric and categorical features. In addition to the diversity of the data types of the columns, the dataset itself was very

large, which means that the efficiency of the learning algorithm in terms of training time is also important.

Ensemble methods reduce the variance of estimators and subsequently may improve the quality of prediction, despite the individual classifiers of the ensemble performing poorly (Hastie, Tibshirani, and Friedman 2009; Dvornik, Schmid, and Mairal 2019). For this reason, the ensemble method was identified as the correct approach to training an algorithm instead of an individual learners like decision tree or support vector machine. In particular, "CatBoost", one of the implementations of an ensemble method called gradient boosting, was chosen.

Boosting is one of the most widely used ensemble methods in machine learning (Russell and Norvig 2016). Boosting refers to a family of algorithms that are able to convert weak learners, which are only slightly better than random guess, to strong learners, which are very close to perfect performance (Zhou 2012). Specifically, *gradient boosting*, which optimizes the empirical loss via steepest gradient descent in function space, is a powerful machine-learning technique, and has remained the primary method for learning problems with heterogeneous features, noisy data, and complex dependencies (Mayr *et al.* 2014; Prokhorenkova *et al.* 2017). In context of regression, the loss function is typically mean squared error and mean absolute error (Hastie, Tibshirani, and Friedman 2009).

CatBoost, which stands for *Categorical Boosting*, is a relatively new gradient boosting method that has become popular in recent times because of its ability to train robust models while being significantly more efficient in some applications compared to its rivals, XGBoost and LightGBM.[4] In addition to its training efficiency, CatBoost also supports categorical variables out of the box, providing options for using permutation techniques, target-based statistics, etc.(Prokhorenkova *et al.* 2017)

CatBoost has two primary novel approaches in creating gradient boosted decision trees: (i) it handles categorical features and deals with them during training instead of using tables with preprocessed categorical variables, and (ii) it uses a new schema for calculating leaf values when selecting the tree structure in order to reduce overfitting. CatBoost handles categorical features by performing random permutation of the dataset and computing average label value for each example with the same category value placed before the given one in the permutation. Similarly, CatBoost builds trees in two steps: determining the tree structure, and setting values in leaves after the tree structure is set. While CatBoost performs the second step using traditional gradient boosted decision trees approaches, it uses a novel method to determine the tree structure.[5] (Prokhorenkova *et al.* 2017). The general building stages for a sing tree in CatBoost is as follows:

Algorithm 1 Steps involved in building a single CatBoost tree

1: Preliminary calculation of splits: possible values of objects are divided into disjoint ranges, *buckets*, delimited by threshold values to determine the possible ways to split data into buckets; information from this step is used in choosing the tree structure
2: (Optional) Transforming categorical features into numerical features
3: (Optional) Transforming text features into numerical features
4: Choosing the tree structure
5: Calculating values in leaves

In addition, CatBoost also provides an overfitting detector which requires evaluation data in order to function. One of the overfitting detector which was implemented in this project works as follows: if there are no improvements in the validation set loss value for n past iterations, where n is an arbitrary integer value signifying the number of iterations, the model is considered overfitted

4. See (Prokhorenkova *et al.* 2017) for exact performance metrics in different test cases.
5. See Appendix 5 for how features are handled and how tree structure is determined by CatBoost.

and the training process is stopped.

Models were trained for the sites and the meters separately. The training file was separated into 16 parts, one for each site, then models were trained for each site, and the model was then combined. This model is named "site model" for convenience. Similarly, the training file was again separated into 4 parts, one for each meter, then a model was trained for each file, and the model was combined. This model is named "meter model" for convenience. Then, predictions were made using the "site model" and the "meter model" separately, and finally, the two sets of predictions were averaged to get the final predictions. This averaged model is named "combined model".

3.3 Baseline Model

The baseline model, which serves as a benchmark against which a trained regression model can be compared, was chosen to be the mean of the target variable of the training dataset. In other words, this model naively predicts the mean figure of the training dataset when a prediction is made.

3.4 Hyperparameter Optimisation

The hyperparameters were optimised using a 3-fold cross validation across a grid of possible values for the four key hyperparameters of the CatBoost regression model: *iterations*, *depth of the trees*, *learning rate*, and the *amount of L2 regularization* applied to each tree. The `GridSearchCV` method provided by the `scikit-learn` library in Python 3 was used for this step.

3.5 Methods of Evaluation

Three different types of model evaluation metrics were selected: root mean squared error, root mean squared logarithmic error, and mean absolute error.

Root Mean Squared Error (RMSE)

Arguably the most common metric used in regression problems, RMSE, calculated by taking a squared root of the mean squared error (MSE), is preferred to MSE because it is on the same scale as the data (Hyndman and Koehler 2006). The following formula is used to calculate RMSE:

Let y_i be the true value, and $\hat{y}_i$ the predicted value for a given row of inputs X_i,

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{1}^{N}(y_i - \hat{y}_i)^2}$$

The underlying assumption of RMSE is that the errors are unbiased and follow a normal distribution (Chai and Draxler 2014), which cannot be ascertained for the dataset in this project. Additionally, since the difference between the true value and the predicted value is squared for each observation, RMSE weighs large errors heavily, which means that if certain sites or meters produce a large deviation, the RMSE even though the errors may be smaller in other meters or sites (Giuliani, Henze, and Florita 2016).

Mean Absolute Error (MAE)

MAE was chosen as the second metric of the trained model's performance evaluation. Tracking MAE is useful as it is also on the same scale as the data and is not as prone to outliers as RMSE is (Hyndman and Koehler 2006). The formula used to calculate MAE is as follows:

Let y_i be the true value, and $\hat{y}_i$ the predicted value for a given row of inputs X_i,

$$MAE = \frac{1}{n}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Root Mean Squared Logarithmic Error (RMSLE)

While RMSE and MAE have been widely used in model evaluation for many years, there is no

consensus on the most appropriate metric for model errors (Chai and Draxler 2014). Since, the original competition also selected RMSLE as the primary evaluation criteria, it was chosen as the third criteria for model evaluation. As opposed to RMSE, which punishes over-estimations, RMSLE punishes under-estimation and is calculated using the following formula:

Let y_i be the true value, and $\hat{y}_i$ the predicted value for a given row of inputs X_i,

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(y_i + 1) - log(\hat{y}_i + 1))^2}$$

The models were trained on a desktop computer with Intel Core i7 6700 3.2GHz processor with 8 processing threads and 64GB memory.

4 Results

Once the models were trained, predictions were made with the holdout set as the final evaluation of the model. Three different models were evaluated: the models trained on each site separately ("site models"), models trained on each meter separately ("meter models"), and a combined model which is the average of the predictions made by "site models" and "meter models", named "combined model". Figure 3 depicts the predictions made by each model on the first 100 rows of the holdout dataset.

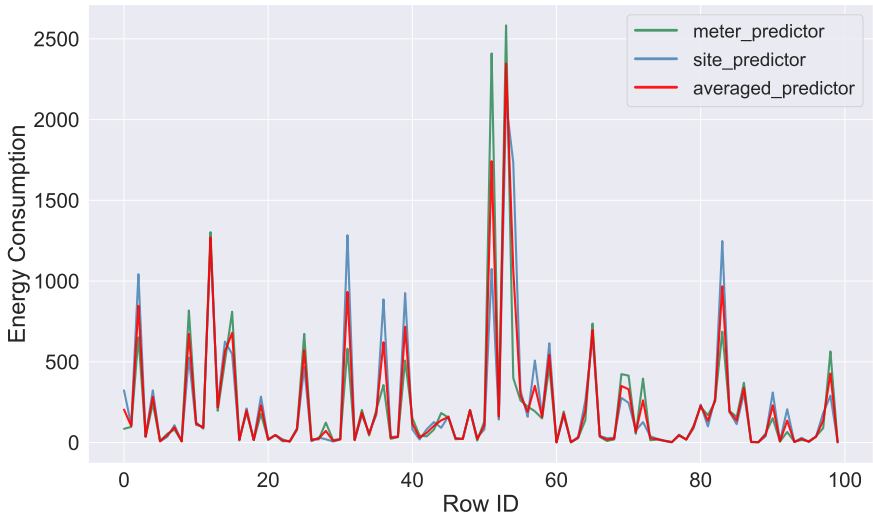

Figure 3. Predictions made by each of the three models on the first 100 rows of the holdout set.

Loss figures of each model for the three chosen metrics have been summarised in Table 1. One can see in the table that if the root mean squared error (RMSE) was the sole evaluation metric, the models (trained on sites, trained meters, and the average of two models) does not perform much better than the baseline model. However, if one considers mean absolute error (MAE) figures, one can see that the trained models are significantly better than the baseline.

Further, if one only considers the MAE figures, the model trained for different sites seems to have outperformed the rest of the trained models and the model trained for the different meter types performs poorly. If only RMSE and MAE are considered, the combined model created by averaging predictions made by "site models" and "meter models" seems to perform worse than the "site models". However, based on the root mean squared logarithmic error (RMSLE) metric, the

combined model performs slightly better than "site models".

Table 1. Error values for prediction on the holdout dataset for different models.

Error Metric	model:baseline	model:site	model:meter	model:combined
RMSE	4145.9845	4049.4	4060.5841	4053.6748
RMSLE	2.8482	1.1344	1.1875	1.1313
MAE	615.5186	303.5662	319.4877	305.8833

Table 2 summarises the error figure for the training, validation, and test set for the final "combined model". The error value for the RMSE and MAE metrics was lowest for the test set, and the error value for RMSLE metric was lowest for validation set. For two of the three metrics, RMSLE and MAE, the error values were highest for the training dataset. The validation set had the highest value for RMSE.

Table 2. Training, validation, and test (holdout) set error values for the combined model.

Error Metric	train	valid	test
RMSE	4127.8104	4485.113	4053.6748
RMSLE	1.1378	1.13	1.1313
MAE	316.7129	311.352	305.8833

Figure 4 depicts the prediction made by the final "combined model" in comparison to the true value and the baseline predictor. It is clear from the figure that, for the first 100 rows of the holdout dataset, the final model is relatively better at predicting values under 1000, but fails at predicting spikes in the energy consumption.

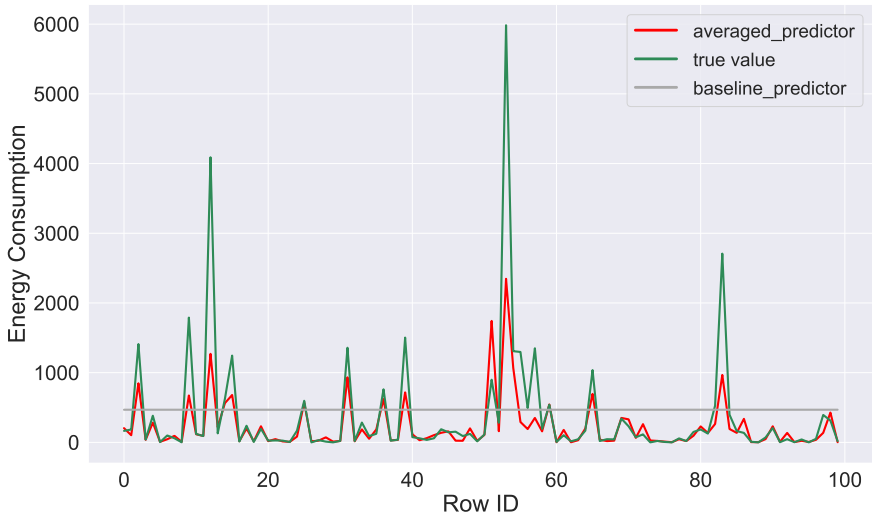

Figure 4. Predictions made by the combined model in comparison to the true values for the first 100 rows of the holdout set.

5 Discussion

The primary purpose of this paper was to predict energy consumption of buildings using the buildings' characteristics and weather data with the application of machine learning. As such, the data was cleaned in order to reduce noise, feature engineering was performed in order to extract more information from existing features, the features were then selected, and models were fitted to the resulting training dataset.

It is clear from the result that a gradient boosted regression model based on CatBoost is able to identify some pattern in the buildings' characteristics and weather information successfully and predict the energy consumption with moderate success. The model fails to predict cases with sudden spike in energy consumption, as is evident in Figure 4.

Similarly, the model also shows some inconsistencies in the error rates for training, validation, and test datasets: the error rates for the test dataset was the lowest, whereas the training set had higher error rates (see Table 2). One of the reasons for the high error values for the training dataset could be that the training set is three times as large as both the validation and test sets, and as such could have more of the large spikes in energy usage which contributed to a bad performance of the final "combined model". Furthermore, the RMSE model for the baseline model and the trained models were also very close (see Table 1). The reason for this could be because RMSE weighs large errors heavily, as mentioned in the methods of evaluation section above. These inconsistencies in the error metrics, show that selecting multiple metrics for model evaluation is of paramount importance.

Limitations of the study: This study was as much about features engineering and feature selection as it was about identifying an appropriate regression method to model energy consumption of the buildings. This means that a domain expert in the field of electrical energy consumption could have additional information to be able to identify a better method of data preprocessing and features engineering; most notably, dealing with missing values, creation of time series related features, and identification of outliers. Thus, conducting this study in collaboration with a domain expert could have resulted to a better model overall. Furthermore, alternative boosting methods, notably LightGBM by Ke *et al.* (2017), which has a different method of creating trees, could have been tested. Finally, training additional models based on approaches different from boosting and combining the different types of models could also yield better results.

References

ASHRAEa. 2019. *ASHRAE - Great Energy Predictor III.* https://www.kaggle.com/c/ashrae-energy-prediction.

ASHRAEb. 2019. *The Research and Technical Activities Report.* https://www.ashrae.org/File%5C%20Library/Technical%5C%20Resources/Research/RT_REPORT_A19.r1.pdf.

Barak, P. 1995. "Smoothing and Differentiation by an Adaptive-Degree Polynomial Filter." *Analytical Chemistry* 67 (17): 2758–2762. doi:10.1021/ac00113a006. https://doi.org/10.1021/ac00113a006.

Catenhusen, W.-M. 2004. "The significance of research and Education for Renewable Energies."

Chai, T., and R. R. Draxler. 2014. "Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature." *Geoscientific Model Development* 7 (3): 1247–1250. doi:10.5194/gmd-7-1247-2014. https://www.geosci-model-dev.net/7/1247/2014/.

Dvornik, N., C. Schmid, and J. Mairal. 2019. *Diversity with Cooperation: Ensemble Methods for Few-Shot Classification.* http://arxiv.org/pdf/1903.11341v2.

Giuliani, M., G. P. Henze, and A. R. Florita. 2016. "Modelling and calibration of a high-mass historic building for reducing the prebound effect in energy assessment." *Energy and Buildings* 116:434–448. ISSN: 0378-7788. doi:https://doi.org/10.1016/j.enbuild.2016.01.034.

Hastie, T., R. Tibshirani, and J. H. Friedman. 2009. *The elements of statistical learning: Data mining, inference, and prediction / Trevor Hastie, Robert Tibshirani, Jerome Friedman.* 2nd ed. Springer series in statistics. New York: Springer. ISBN: 0387848576.

Hyndman, R. J., and A. B. Koehler. 2006. "Another look at measures of forecast accuracy." *International Journal of Forecasting* 22 (4): 679–688. https://ideas.repec.org/a/eee/intfor/v22y2006i4p679-688.html.

International Energy Agency. 2019. *World Energy Outlook 2019.* 810. doi:https://doi.org/https://doi.org/10.1787/caf32f3b-en. https://www.oecd-ilibrary.org/content/publication/caf32f3b-en.

Ke, G., Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. 2017. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." In *Advances in Neural Information Processing Systems 30,* edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 3146–3154. Curran Associates, Inc. http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf.

Mayr, A., H. Binder, O. Gefeller, and M. Schmid. 2014. "The evolution of boosting algorithms. From machine learning to statistical modelling." *Methods of information in medicine* 53 (6): 419–427. doi:10.3414/ME13-01-0122.

Mosavi, A., M. Salimi, S. Faizollahzadeh Ardabili, T. Rabczuk, S. Shamshirband, and A. Varkonyi-Koczy. 2019. "State of the Art of Machine Learning Models in Energy Systems, a Systematic Review." *Energies* 12 (7): 1301. ISSN: 1996-1073. doi:10.3390/en12071301. http://dx.doi.org/10.3390/en12071301.

Oliveira, M. de, N. Araujo, R. da Silva, T. da Silva, and J. Epaarachchi. 2018. "Use of Savitzky–Golay Filter for Performances Improvement of SHM Systems Based on Neural Networks and Distributed PZT Sensors." *Sensors* 18, no. 2 (January): 152. ISSN: 1424-8220. doi:10.3390/s18010152.

Prokhorenkova, L., G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. 2017. "CatBoost: unbiased boosting with categorical features." http://arxiv.org/pdf/1706.09516v5.

Russell, S., and P. Norvig. 2016. *Artificial Intelligence: A Modern Approach.* Pearson Education Limited. ISBN: 9781292153964.

Savitzky, A., and M. J. E. Golay. 1964. "Smoothing and Differentiation of Data by Simplified Least Squares Procedures." *Analytical Chemistry* 36 (8): 1627–1639. doi:10.1021/ac60214a047. eprint: https://doi.org/10.1021/ac60214a047. https://doi.org/10.1021/ac60214a047.

Schaeffer, R., A. S. Szklo, A. F. P. de Lucena, B. S. M. C. Borba, L. P. P. Nogueira, F. P. Fleming, A. Troccoli, M. Harrison, and M. S. Boulahya. 2012. "Energy sector vulnerability to climate change: A review." *Energy* 38 (1): 1–12. ISSN: 0360-5442. doi:https://doi.org/10.1016/j.energy.2011.11.056. http://www.sciencedirect.com/science/article/pii/S0360544211007870.

Schettino, B. M., C. A. Duque, and P. M. Silveira. 2016. "Current-Transformer Saturation Detection Using Savitzky-Golay Filter." *IEEE Transactions on Power Delivery* 31 (3): 1400–1401. doi:10.1109/tpwrd.2016.2521327.

Zhou, Z.-H. 2012. *Ensemble methods: Foundations and algorithms / Zhi-Hua Zhou.* Chapman & Hall/CRC machine learning & pattern recognition series. Boca Raton: CRC Press. ISBN: 978-1-4398-3005-5.

Appendix I: List of Column Names

Table 3. Description of columns in the data set.

File Name	Column Name	Description
train.csv	timestamp	450
	building_id	unique identifiers for buildings
	meter	one of the four types of meters that record the amount of energy consumed
	meter_reading	the target variable; measured in KWh (or equivalent)
building_metadata.csv	site_id	unique identifier for sites
	building_id	unique identifier for buildings
	primary_use	main use of a given building
	square_feet	square footage of a floor of a given building
	year_built	year when the buildings were built
	floor_count	the no. of floors in a given building
weather_train.csv	timestamp	350
	site_id	unique identifiers for sites
	air_temperature	(recorded hourly)
	cloud_coverage	(recorded hourly)
	dew_temperature	(recorded hourly)
	precip_depth_1_hr	amount of rain (recorded hourly)
	sea_level_pressure	(recorded hourly)
	wind_direction	(recorded hourly)
	wind_speed	(recorded hourly)

Appendix II: CatBoost Details

Here are notes on how CatBoost handles (categorical and numeric) features, and the algorithm that it uses for estimation of gradient.

CatBoost's method of handling features

Assuming a dataset of observations $D = (X_i, Y_i)_{i=1...n}$, where $X_i = (x_{i,1}, ..., x_{i,m})$ is a vector of m (categorical and numerical) features, and Y_i is a *label value*, supposing $\sigma = (\sigma_1, ..., \sigma_n)$ is the permutation, the following substitation is made:

$$x_{p,k} = \frac{\sum_{j=1}^{p-1} ([x_{\sigma_j,k} = x_{\sigma_p,k}] Y_{\sigma_j} + a * P)}{\sum_{j=1}^{p-1} ([x_{\sigma_j,k} = x_{\sigma_p,k}] + a)} \tag{1}$$

Here P is a prior value and $a > 0$ is the weight of the prior. For regression problems, the prior P is the average label value in the dataset.(Prokhorenkova *et al.* 2017)

Algorithm for Gradient Estimation

Let F^i be the model constructed after building i trees and $g^i(X_k, Y_k)$ be the gradient value on k-th training sample after building i tree. To make the gradient $g^i(X_k, Y_k)$ unbiased w.r.t. the model F^i, F^i needs to be trained without the observations X_k. For each example X_k, a separate model M_k is trained, which is never updated using a gradient estimate for this example. Then, the gradient on X_k is estimated with M_k and this estimate is used to score the resulting tree.(Prokhorenkova *et al.* 2017)

Here is the pseudo-code from Prokhorenkova *et al.* 2017 that explaines how this is performed:

Algorithm 2 Model update and calculation of model values for gradient estimation

input : $\{(X_k, Y_k)\}_{k=1}^n$ ordered according to σ, the number of trees I
$M_i \leftarrow \theta$ for $i = 1 \dots n$
for $iter = 1, 2, \dots, I$ **do**
    for $i = 1, 2, \dots, n$ **do**
        for $j = 1, 2, \dots, i-1$ **do**
            $g_j = \frac{d}{da} Loss(y_j, a)$ where $a = M_i(X_j))$
        end for
        $M \leftarrow LearnOneTree((X_j, g_j) for j = 1 \dots i-1)$
        $M_i \leftarrow M_i + M)$
    end for
end for
return : $M_1, \dots, M_n, M_1(X_1) \dots M_n(X_n)$

All M_i share the same tree structures.

Declaration of Authorship

I hereby declare that I am the author of the project report that I am submitting. The report is entirely my own original work except where otherwise indicated. Any use fo the works of any other author, in any form, is properly acknowledged at their point of use.

_____

Prabesh Dhakal, March 02, 2020