Modelling Energy Usage

Dongtao Jiang | Springboard Data Science Career Track | 9/29/2020

Table of Contents

Introduction

Energy efficiency in building systems is a big concern regarding cost saving and environmental impact. The state-of-art buildings systems design mandate improvement in energy savings in the areas including electric, chilled-water, hot-water and steam. If certain technology has been implemented this year aiming to reduce energy usage, one would expect to see a drop in energy usage next year. Suppose we do see a drop, however, is the drop an effect of weather change or new technology? Since weather plays an important role in energy usage. it is important to develop a model that is able to separate the two kinds of effect and allow stakeholders to have confidence in applying new technology. 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.

The data comes from over 1,000 buildings at several different sites around the world and the local weather data over a three-year timeframe.

Dataset

The dataset was obtained from the following link,

https://www.kaggle.com/c/ashrae-energy-prediction/data

train.csv

- building_id - Foreign key for the building metadata.
- meter - The meter id code. Read as {0: electricity, 1: chilled water, 2: steam, 3: hot water}. Not every building has all meter types.
- timestamp - When the measurement was taken
- meter_reading - The target variable. Energy consumption in kWh (or equivalent). Note that this is real data with measurement error, which we expect will impose a baseline level of modeling error.

building_meta.csv

- site_id - Foreign key for the weather files.
- building_id - Foreign key for training.csv
- primary_use - Indicator of the primary category of activities for the building based on EnergyStar property type definitions
- square_feet - Gross floor area of the building
- year_built - Year building was opened
- floor_count - Number of floors of the building

weather_[train/test].csv

Weather data from a meteorological station as close as possible to the site.

- site_id
- air_temperature - Degrees Celsius
- cloud_coverage - Portion of the sky covered in clouds, in oktas
- dew_temperature - Degrees Celsius
- precip_depth_1_hr - Millimeters
- sea_level_pressure - Millibar/hectopascals
- wind_direction - Compass direction (0-360)
- wind_speed - Meters per second

test.csv

The submission files use row numbers for ID codes in order to save space on the file uploads. test.csv has no feature data; it exists so you can get your predictions into the correct order.

- row_id - Row id for your submission file
- building_id - Building id code
- meter - The meter id code
- timestamp - Timestamps for the test data period

Software Packages

- Python 3.6
- Pandas 1.0.3
- Numpy 1.15.4
- Sklearn 0.23.1
- Matplotlib 3.2.1
- Seaborn 0.10.0
- Scipy 1.10
- Catboost 0.24.1
- LightGBM 2.3.1

Data Wrangling

The following procedures have been carried out,

- Conform timestamp to datetime format
- Merge datasets based on building_id, site_id and timestamp

- train set, test set
- buidings_meta_df
- weather data

– Change data types to reduce memory usage

– Correct unit

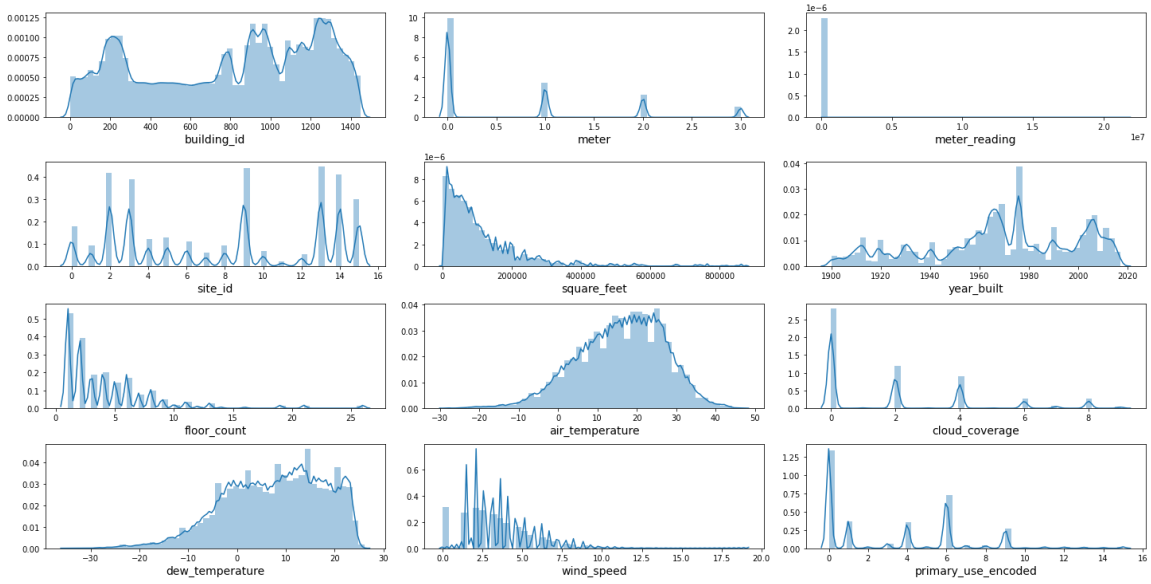
- Site 0 were not properly converted to units of kWh and are in kBTU
- Multiply by 0.2931 to get to model inputs into kWh like the other sites, and 3.4118 to get back to kBTU for scoring

Exploratory Data Analysis

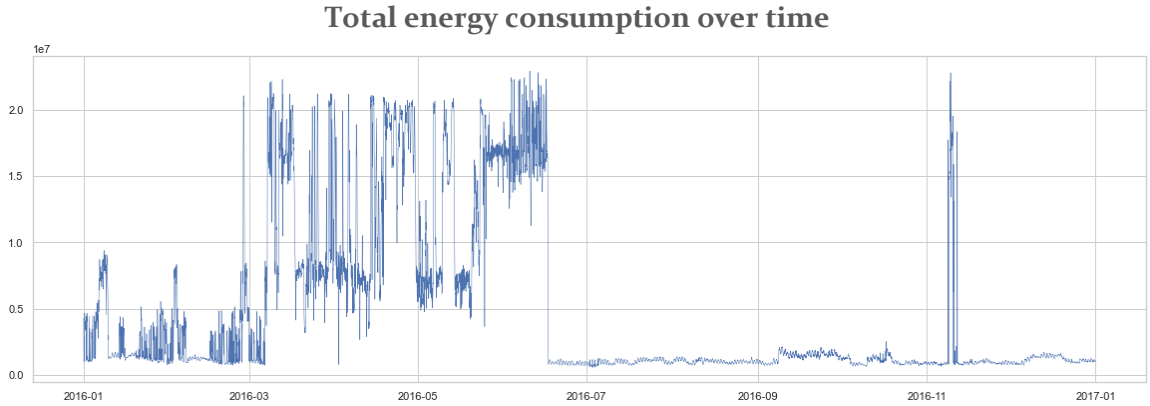
Distribution of variables

Meter reading is severely skewed to larger numbers. Majority of readings are small values. Square feet, floor count, wind speed, cloud coverage is also skewed to larger values.

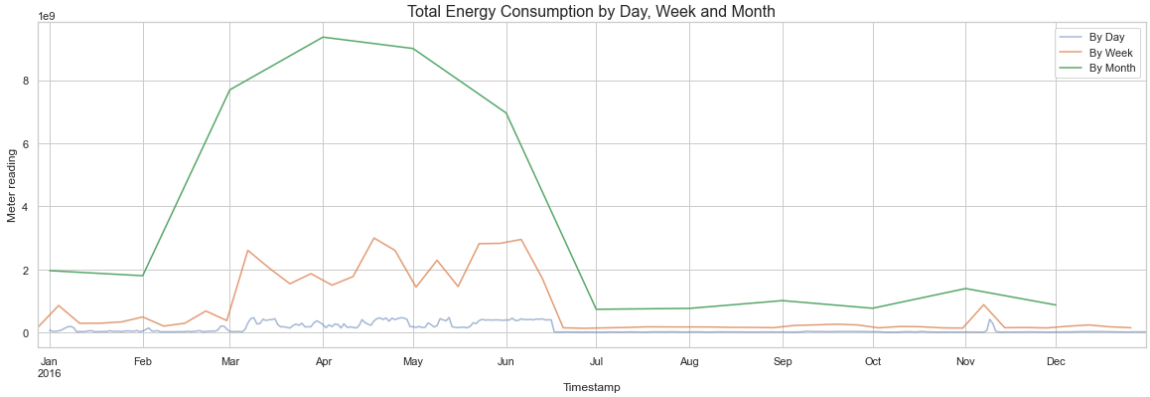
There are more readings for electricity type of meters. it is followed by {1: 'chilled water', 2: 'steam', 3: 'hot water'}



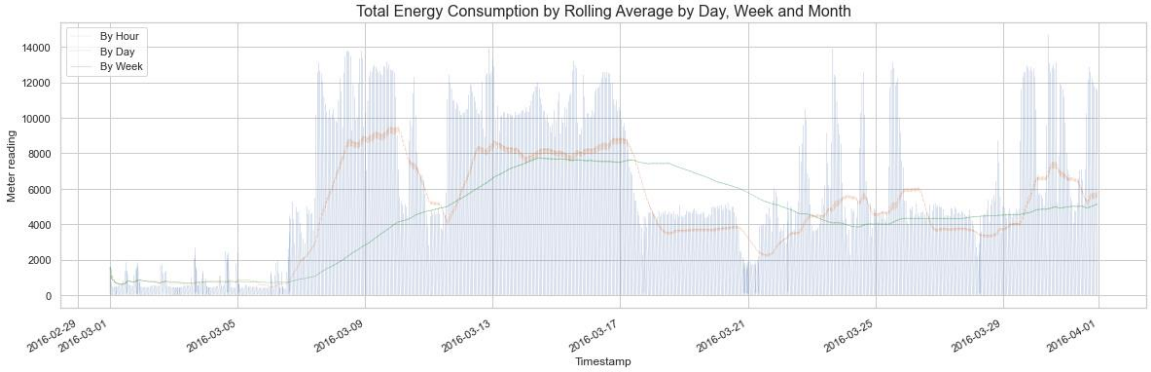
There is unusually high energy usage from March to June and in a few days in November, as shown below.



Energy usages are shown below with different time interval: day, week and month. Monthly usage smooths out all fluctuations from smaller time intervals, showing a jump in energy usage in spring season.



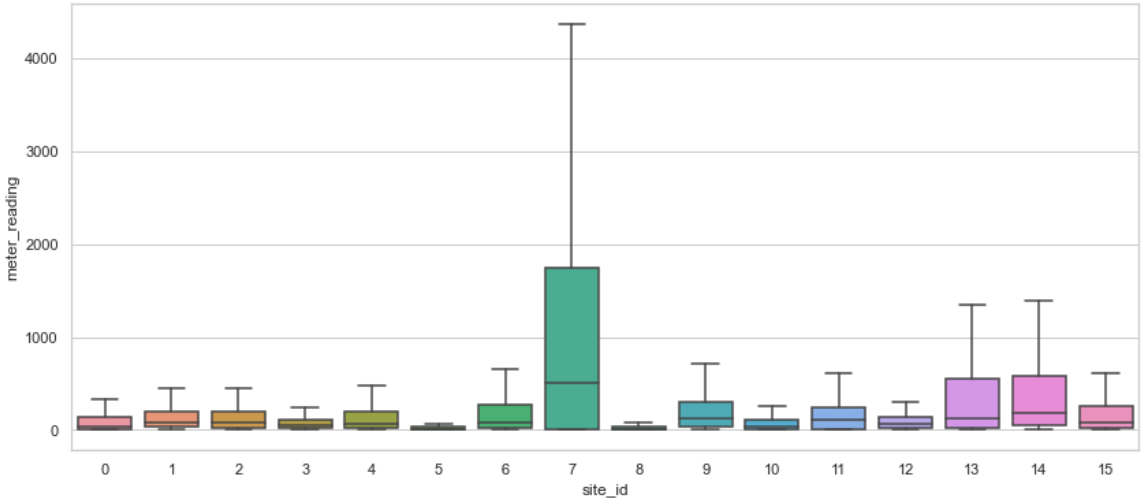
The plots with the rolling average by hour, day and week are show below. The larger the window, the smoother the curves, and the more it lags behind. The rolling average will be used for modelling in the machine learning.



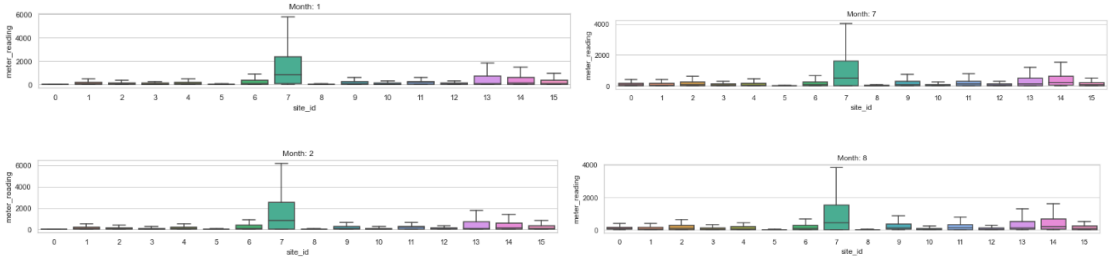
Site specific yearly and monthly energy consumption

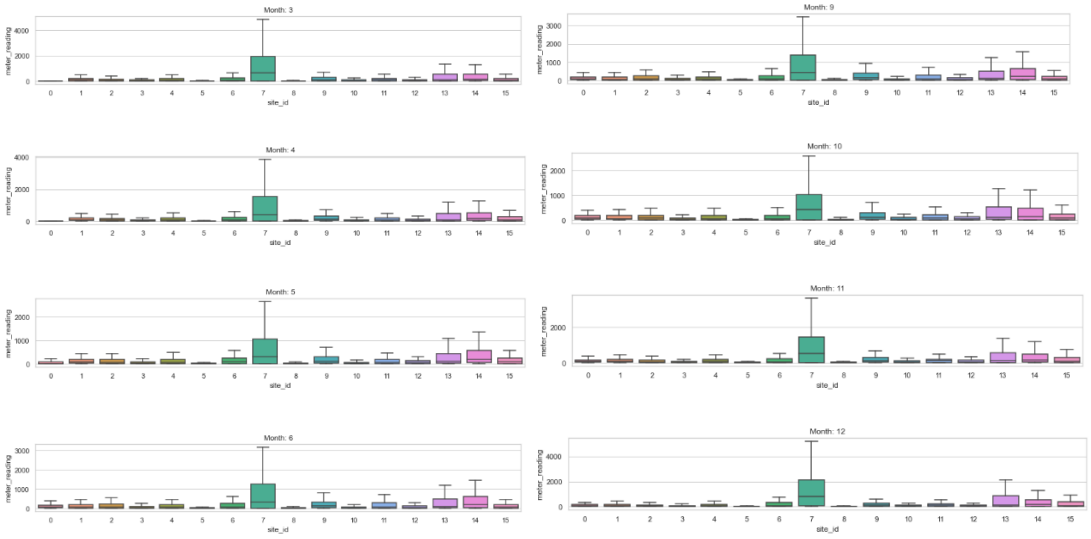
It is interesting to note that site 7 consumes many times higher energy than other sites both yearly and monthly. It also has a very large variation.

Yearly Consumption at Different Site

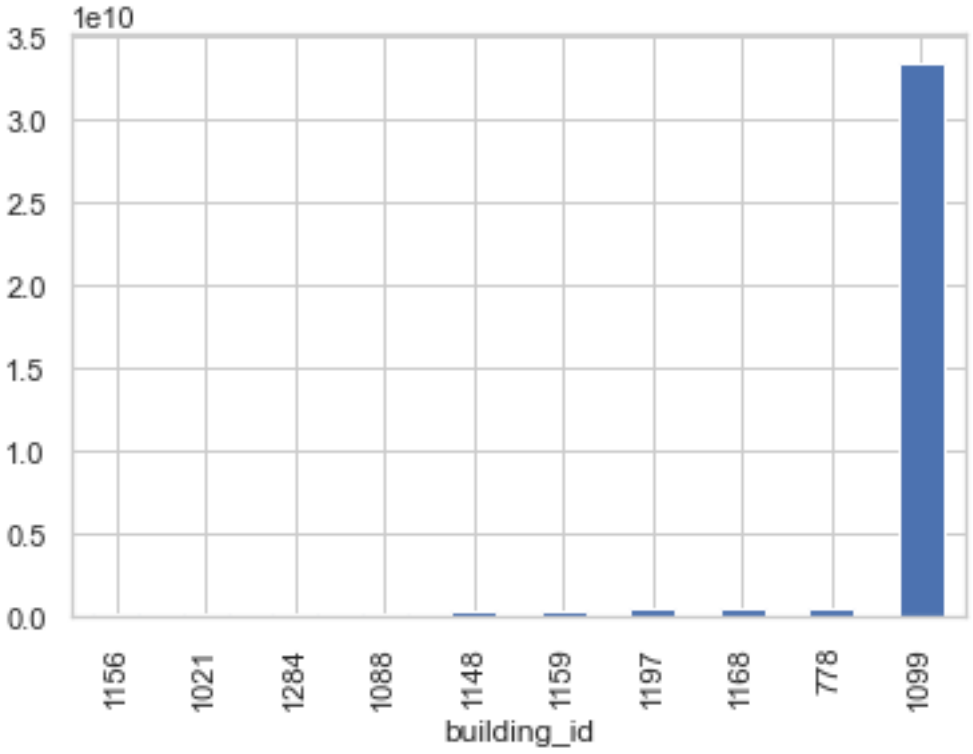


Monthly Consumption at Different Site



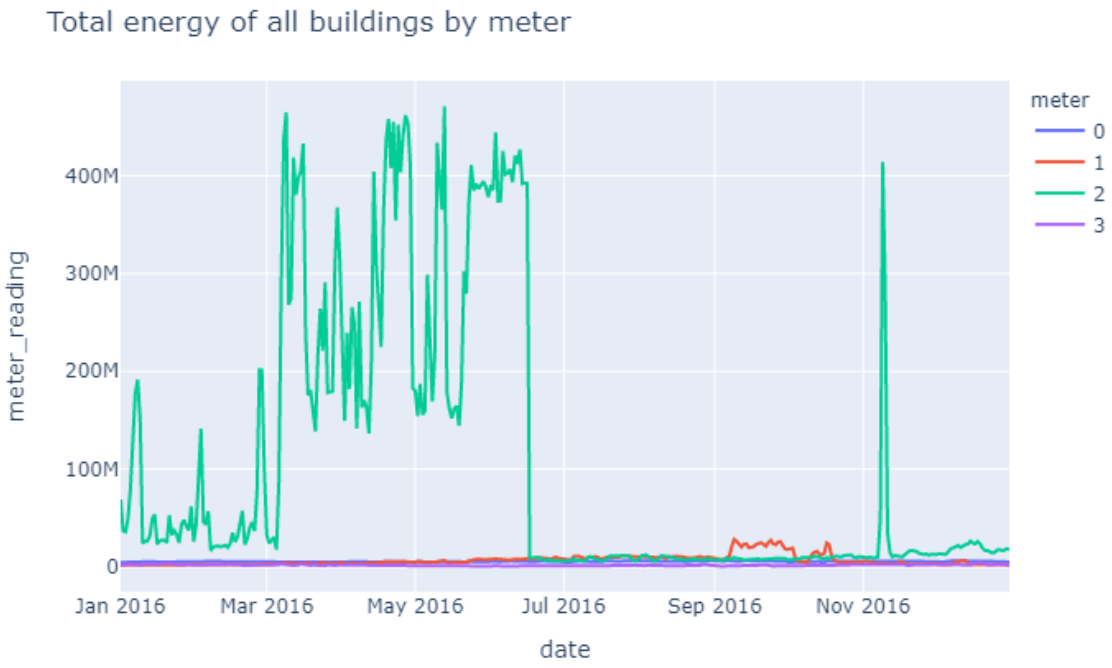


It is found that building_id 1099 eats up much more energy than all others. It is at least about 100 times higher that of other buildings. The building 1099 belongs to site 13 and is an Educational institution.



Energy types

Steam is the major usage compared to chilled water, electricity and hot water.

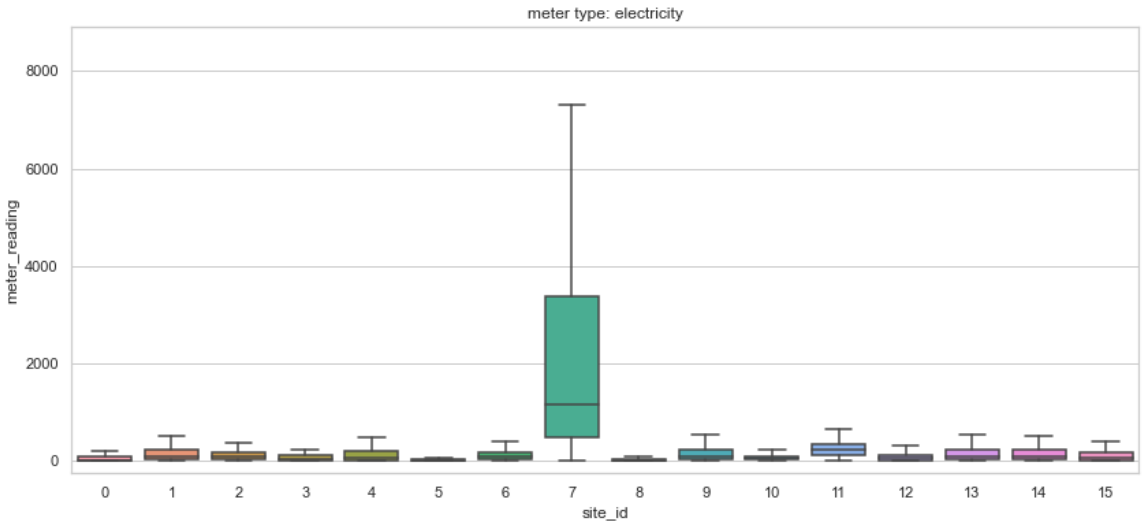


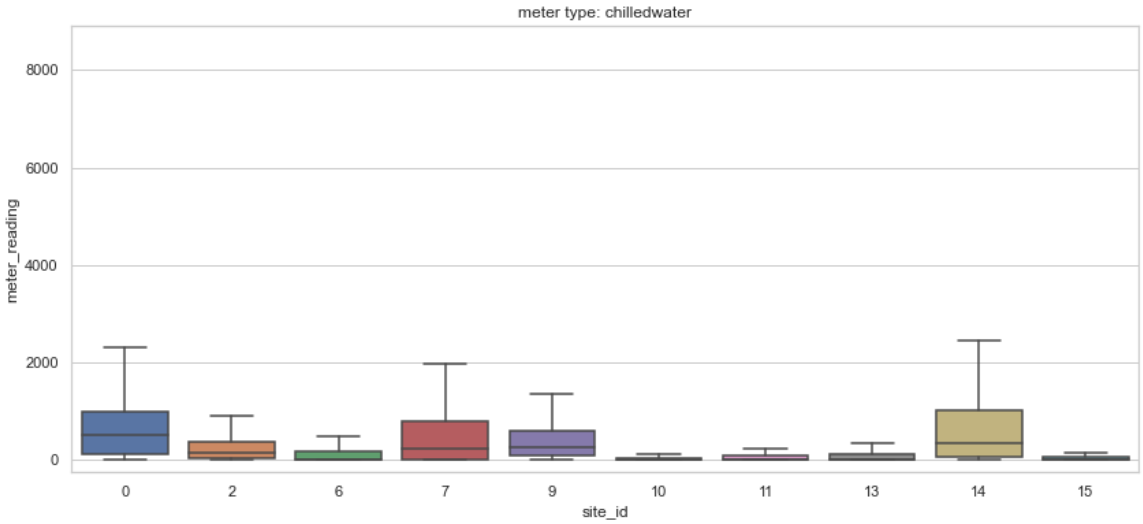

```
{0: 'electricity', 1: 'chilled water', 2: 'steam', 3: 'hot water'}
```

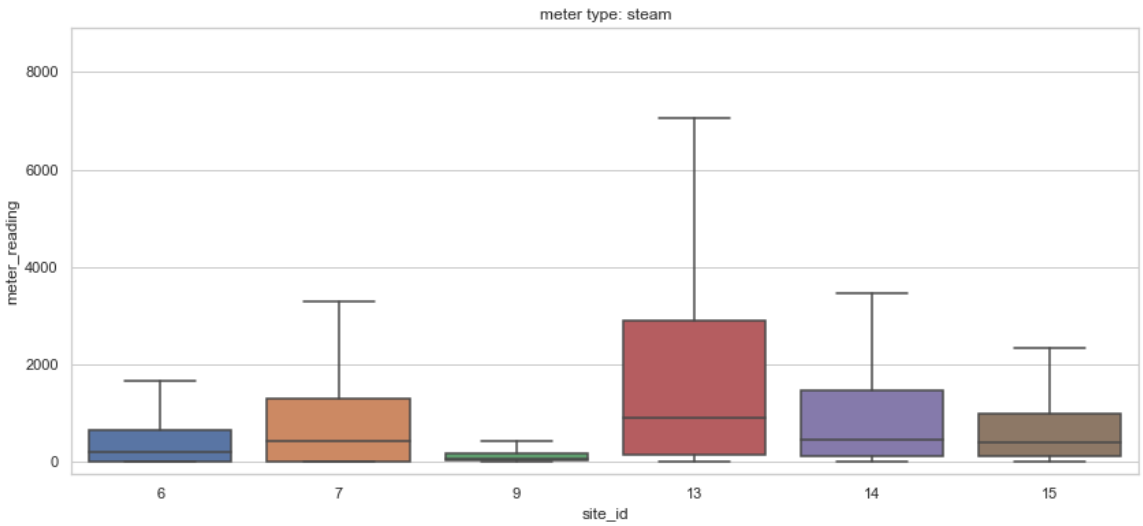
While all sites have electricity reading, not every site has other types of meter readings.

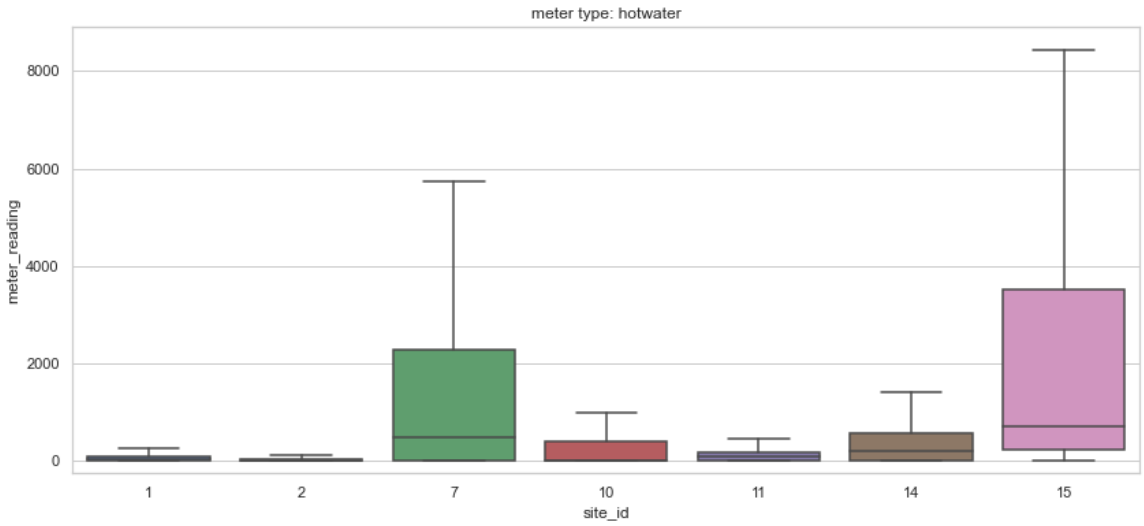
Steam and hot water in a couple of sites use a lot of energy than other meter types.

Site 7 consumes higher energy in each meter type than most other sites.


meter type: electricity
meter_reading
8000
6000
4000
2000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
site_id

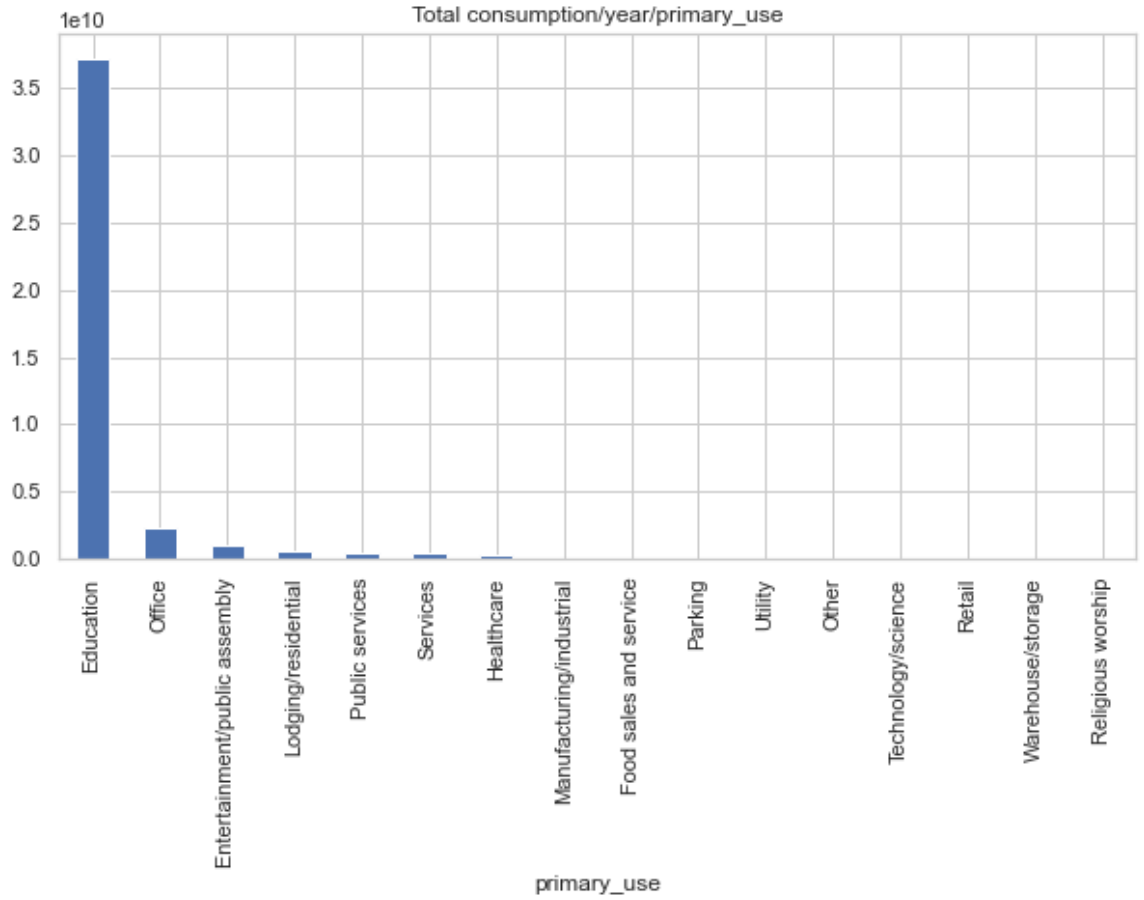

meter type: chilledwater
meter_reading
8000
6000
4000
2000
0
0 2 6 7 9 10 11 13 14 15
site_id


meter type: steam
meter_reading
8000
6000
4000
2000
0
6 7 9 13 14 15
site_id

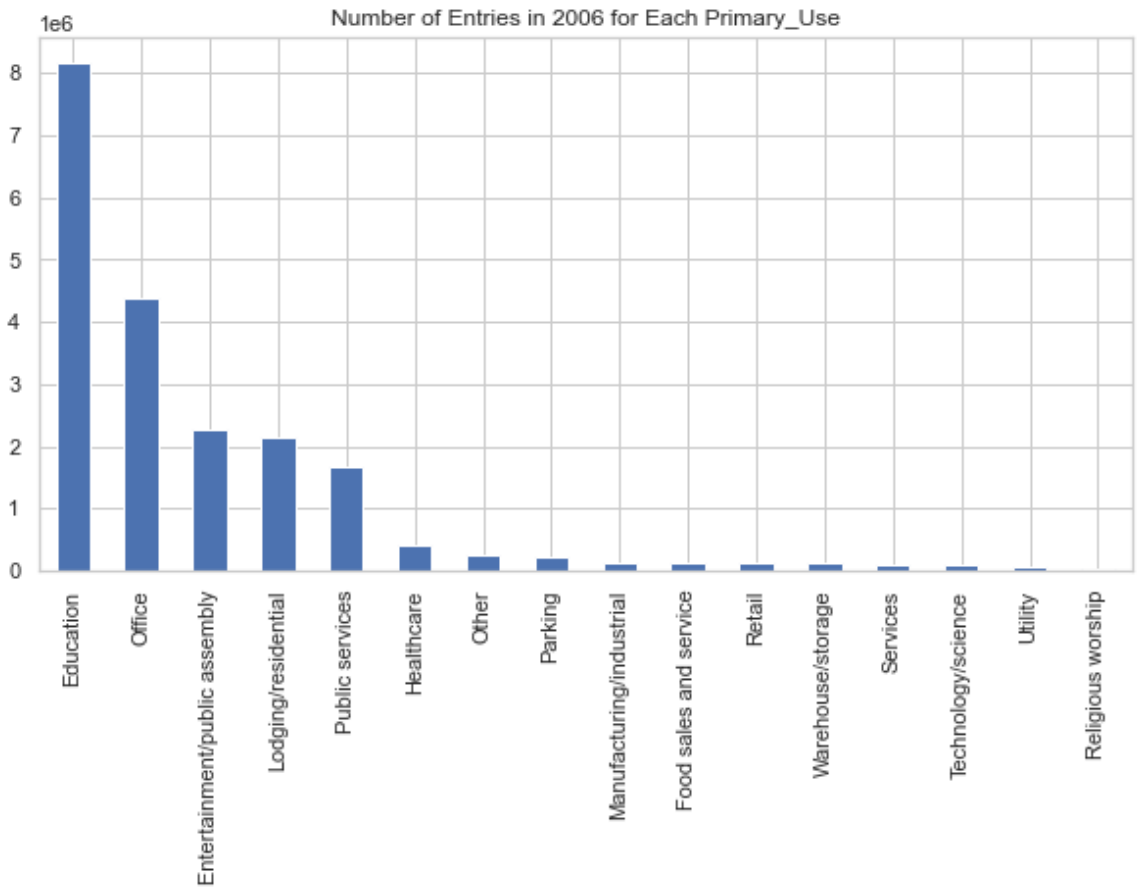


primary_use

Education is predominant in energy consumption. It is followed by office, entertainment/public assembly, lodging/residential, etc.



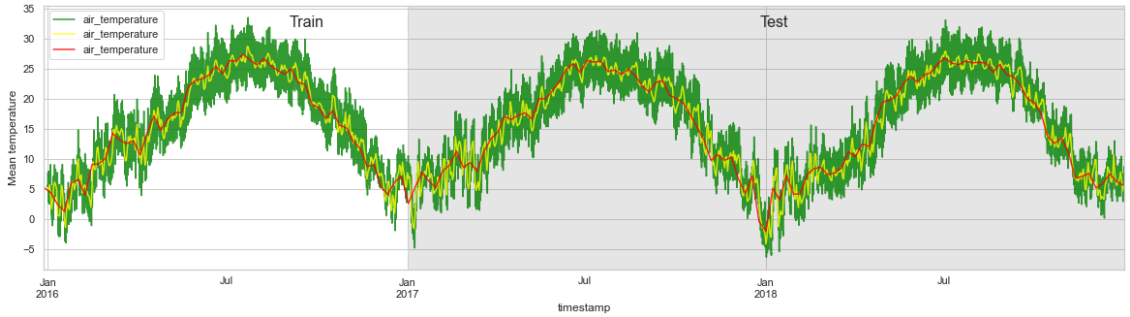
The number of total entries for Education is also far greater than other primary_use, explaining its high usage.



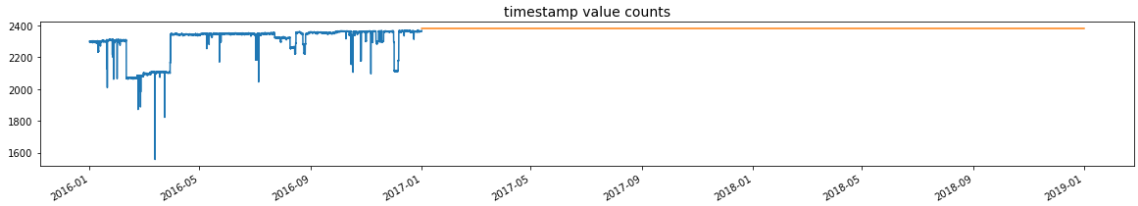
Air temperature

Air temperature in the test set has very similar pattern as in the train set or following two years.

The peak consumption of energy during the spring season is not corresponding well with the air temperature pattern.

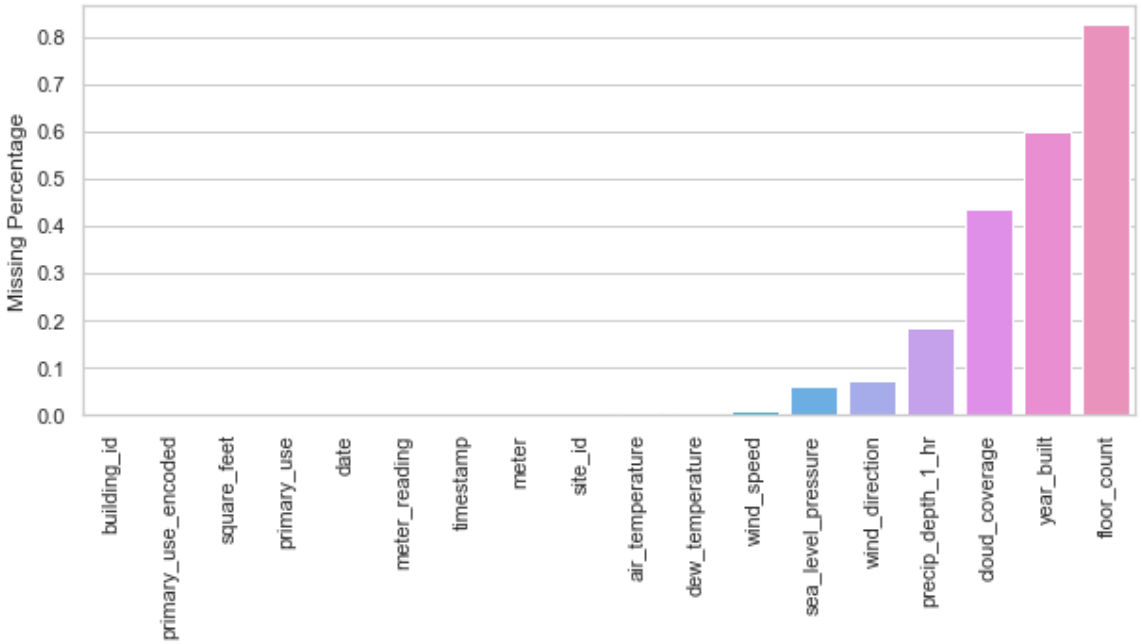


Missing timestamps compared with test dataset



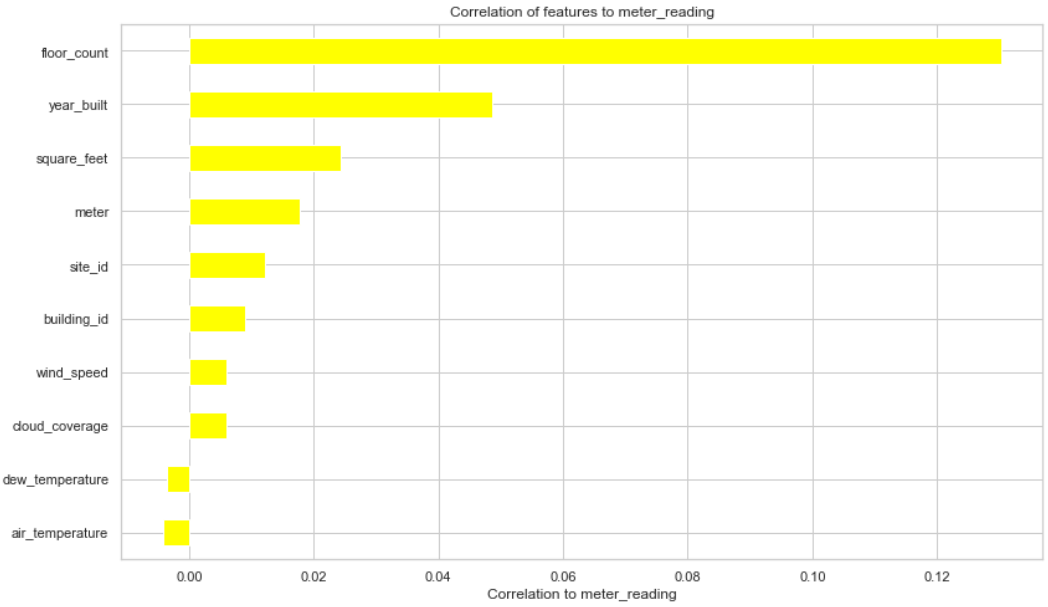
The blue is for year 2006 as the train set. The orange is for year 2017 and 2018. Compared with test set, the training set has missing records in many hours and days.

Top ranking in missing values: floor_count, year_built, cloud_coverage with more than 40% missing.



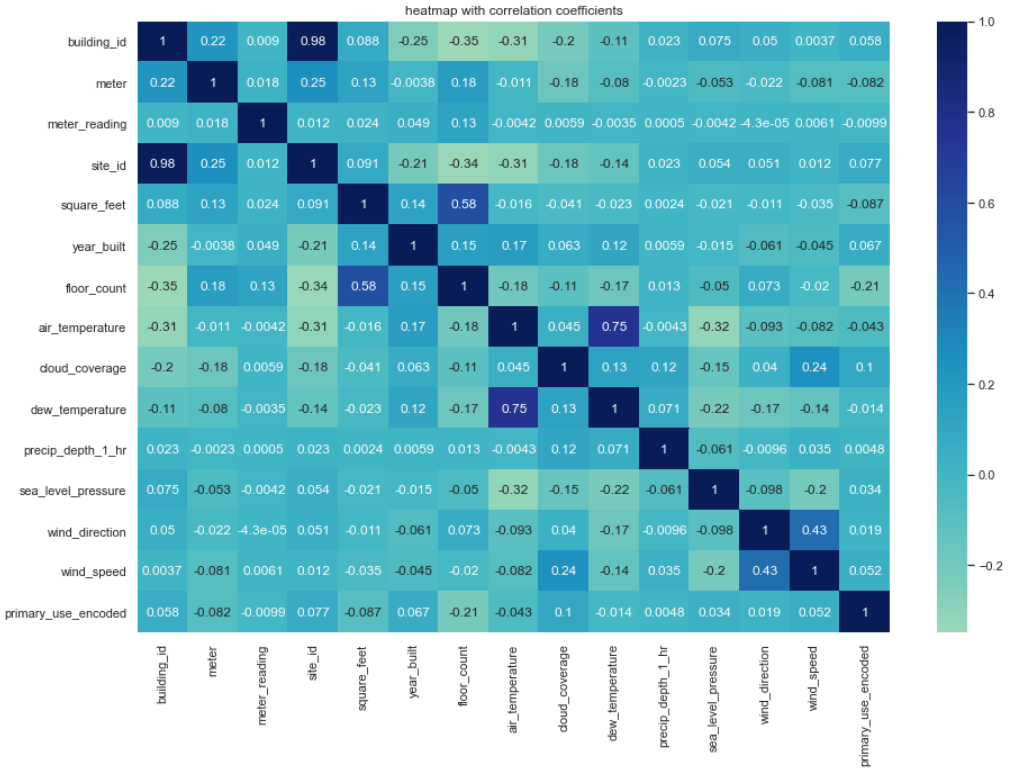
Correlation

The correlation to meter reading is plotted in bar graph. Not everything in there are useful. For example, the values of site id, building id, meter have nothing to do with meter readings. Floor count and square footage have strong correlation to meter reading makes sense because the more energy is demanded for larger service area. The lower the air temperature and dew temperature, the higher energy usage is reasonably revealed.



From the heatmap, it can be seen that site id is well correlated to building id. This certainly reflects the reality. Floor count and square feet are also related to each other.

The correlation coefficient between dew temperature and air temperature is 0.75, which agrees well with meteorology study.



Machine Learning

Independent Variables / Predictors

- Building ID
- Meter Type {0: electricity, 1: chilled water, 2: steam, 3: hot water}.
- Timestamp - When the measurement was taken
- Site ID - Foreign key for the weather files.
- Primary Purpose / Use
- Square Feet
- Year Built
- Floor Count
- Air Temperature
- Cloud Coverage
- Dew Temperature
- Precipitation Depth in One Hour
- Sea Level Pressure
- Wind Direction
- Wind Speed
- Hour
- Day of Year
- Week of Year
- Month

Target Variable

- Meter Reading

Imputation

Missing air temperature is best interpolated linearly instead of using mean value.

The missing values in 'year_built', 'floor_count', 'cloud_coverage', 'dew_temperature', 'precip_depth_1_hr', 'windspeed' were imputed using SimpleImputer with means.

Feature engineering

Add **hour, day of year, week of year, month** as new features.

Wind speed is converted to Beaufort scale.

One-hot encoding

LightGBM has a built-in mechanism to convert categorical variables to one-hot encoding. It also allows faster computing.

Metrics

The root mean squared error, denoted RMSE, and percent mean absolute error (PMAE), will be used to evaluate the model.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

$$PMAE = \frac{\sum_{j=1}^{n}(|y_i - \hat{y}_i|)}{\sum_{j=1}^{n}|y_i|}$$

RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.

The second metric gives a better idea to the stakeholders about how far the prediction is away from actual value percentage wise.

Cross Validation

KFold splits and Time Series Splits are used respectively for validation purpose to prevent overfitting.

Models

Gradient boosting techniques based on decision tree have become a strong army in machine learning forces. Because of its non-linear nature of problem solving, it is suited for all kinds of machine learning needs such as recommender, ranking, classification, linear regression, etc. Gradient boosting uses an additive strategy starting from a rough prediction decision (mean or median) and becoming more and more powerful in prediction by adding up all the predictions step by step. Each prediction is made based upon the previous prediction. To decide split point or the node in the decision tree, the loss function is needed to find the minimum.

The usual gradient boosting pre-sort all samples in the dataset and use all variables to find all possible split points and evaluate the best split based on loss function. The computation time can be extremely long with a large dataset.

To overcome the hurdle, LightGBM uses one-side sampling method by sorting the gradients, which keeps the samples with larger gradients and eliminate a large number of samples with lower gradients. This downsizing not only greatly save computation time but also proves to be without sacrificing prediction accuracy.

Catboost, however, takes a unique strategy called ordered boosting that solve the leakage problem that is inherit in encoding categorical data with target statistics. The idea is to take the advantage of permutation and each dataset is treated like time series. The previous model learned in the past is used to predict the samples in the current time and the iteration continues on until all samples are scanned.

Learning results

Catboost

The following table compares the initial test scores with time features being set as numeric or categorical. It is found that using additional time features as integer got slightly better score.

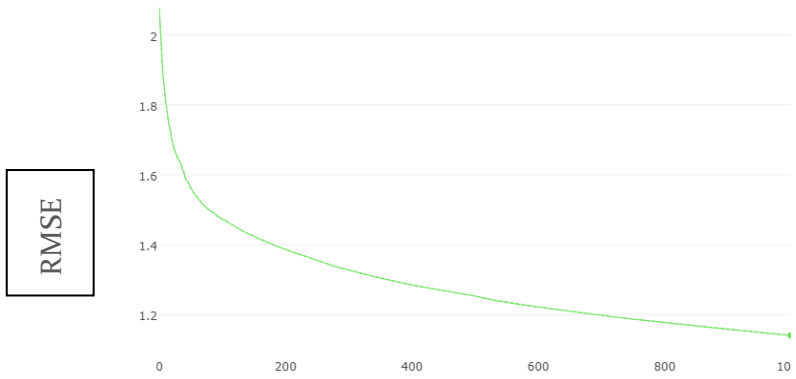
1 categorical variable: ['primary use'] numeric: ['hour', 'dayofyear', ' weekofyear', 'month']	5 categorical variables ['primary_use', 'hour', 'dayofye ar', 'weekofyear', 'month']
bestScore = 1.563167039	bestScore = 1.5948642

By setting the early_stopping_rounds, CatBoost is able to continue iteration until detection of overfitting. It takes Catboost 1 hour 38 min to get the best score 1.140.

A snap shot of the timed progression of scores during iterations.

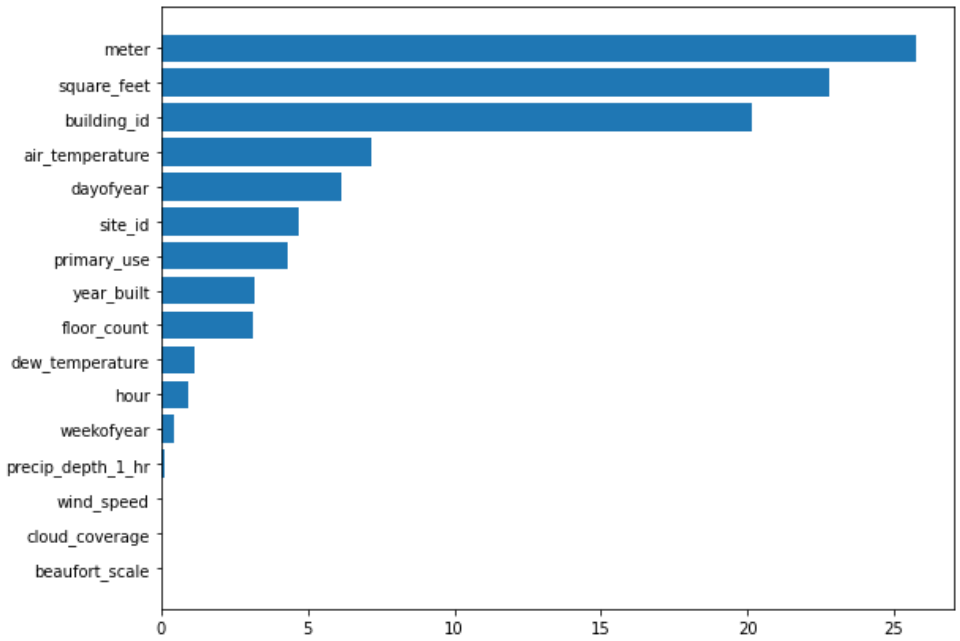
0:	learn: 2.0773102	test: 2.0772248	best: 2.0772248 (0)	total: 7.39s	remaining: 2h 3m 1s
1:	learn: 2.0347985	test: 2.0347067	best: 2.0347067 (1)	total: 12.6s	remaining: 1h 45m 6s
2:	learn: 1.9952108	test: 1.9951207	best: 1.9951207 (2)	total: 17.4s	remaining: 1h 36m 38s
3:	learn: 1.9627101	test: 1.9626161	best: 1.9626161 (3)	total: 21.6s	remaining: 1h 29m 31s
4:	learn: 1.9346027	test: 1.9344940	best: 1.9344940 (4)	total: 25.4s	remaining: 1h 24m 18s
5:	learn: 1.9088951	test: 1.9088068	best: 1.9088068 (5)	total: 28.7s	remaining: 1h 19m 7s
6:	learn: 1.8813429	test: 1.8813015	best: 1.8813015 (6)	total: 32.4s	remaining: 1h 16m 35s
7:	learn: 1.8597902	test: 1.8597429	best: 1.8597429 (7)	total: 37.2s	remaining: 1h 16m 49s
8:	learn: 1.8412133	test: 1.8411592	best: 1.8411592 (8)	total: 40.6s	remaining: 1h 14m 25s
9:	learn: 1.8275006	test: 1.8274505	best: 1.8274505 (9)	total: 44.5s	remaining: 1h 13m 29s
10:	learn: 1.8125809	test: 1.8125151	best: 1.8125151 (10)	total: 48.9s	remaining: 1h 13m 13s
100:	learn: 1.4723753	test: 1.4721771	best: 1.4721771 (100)	total: 6m 8s	remaining: 54m 41s
200:	learn: 1.3865690	test: 1.3862559	best: 1.3862559 (200)	total: 11m 55s	remaining: 47m 22s
300:	learn: 1.3274386	test: 1.3271378	best: 1.3271378 (300)	total: 18m 5s	remaining: 42m 1s
400:	learn: 1.2847891	test: 1.2842049	best: 1.2842049 (400)	total: 24m 20s	remaining: 36m 20s
500:	learn: 1.2535299	test: 1.2528898	best: 1.2528898 (500)	total: 30m 35s	remaining: 30m 28s
600:	learn: 1.2219053	test: 1.2212602	best: 1.2212602 (600)	total: 36m 51s	remaining: 24m 27s
700:	learn: 1.1981293	test: 1.1975124	best: 1.1975124 (700)	total: 42m 55s	remaining: 18m 18s
800:	learn: 1.1778082	test: 1.1772590	best: 1.1772590 (800)	total: 48m 51s	remaining: 12m 8s
900:	learn: 1.1587090	test: 1.1581088	best: 1.1581088 (900)	total: 54m 47s	remaining: 6m 1s
999:	learn: 1.1410593	test: 1.1404469	best: 1.1404469 (999)	total: 1h 38s	remaining: 0us

Progress of Catboost Regression

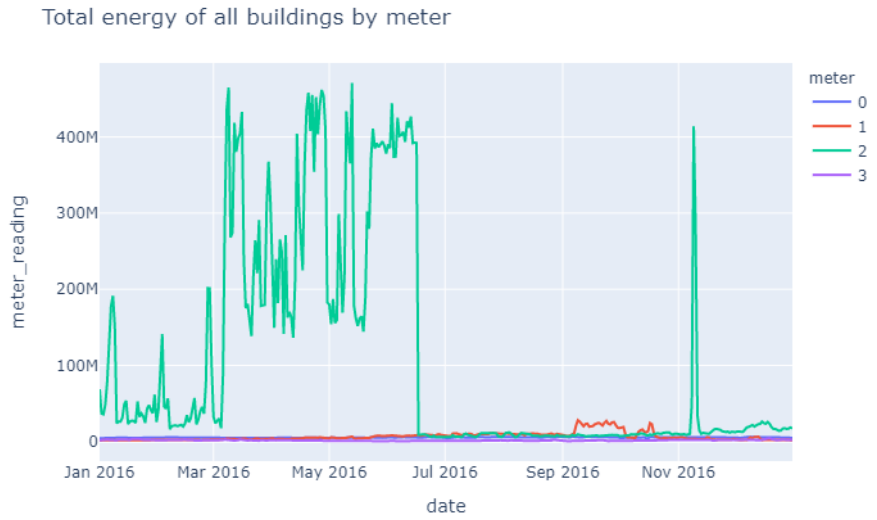


Iteration

The root mean squared error drops very fast in the first 50 or so iteration from 2.07 to 1.56. After that point, the progression becomes slower and slower.



Meter type has the highest feature importance, which is in line with our EDA previously that shows significantly differently usage of the four distinctive types of energy (see below again).




```
{0: 'electricity', 1: 'chilled water', 2: 'steam', 3: 'hot water'}
```

The feature importance of square_feet, building_id is slightly smaller than that of meter. The 4th highest feature importance is air temperature, which drops more than 50% than the higher-ranking features.

The feature importance of dayofyear, hour, weekofyear are in descending order, indicating daily sum of energy usage is more suited for information gain. hourly usage is probably too noisy and weekly usage averages out some information.

Among the weather features, air temperature is the most important. Precipitation has very minimal effect on energy. dew_temperature is in between.

The wind speed/beaufort scale and cloud coverage show no effect on energy usage.

LightGBM

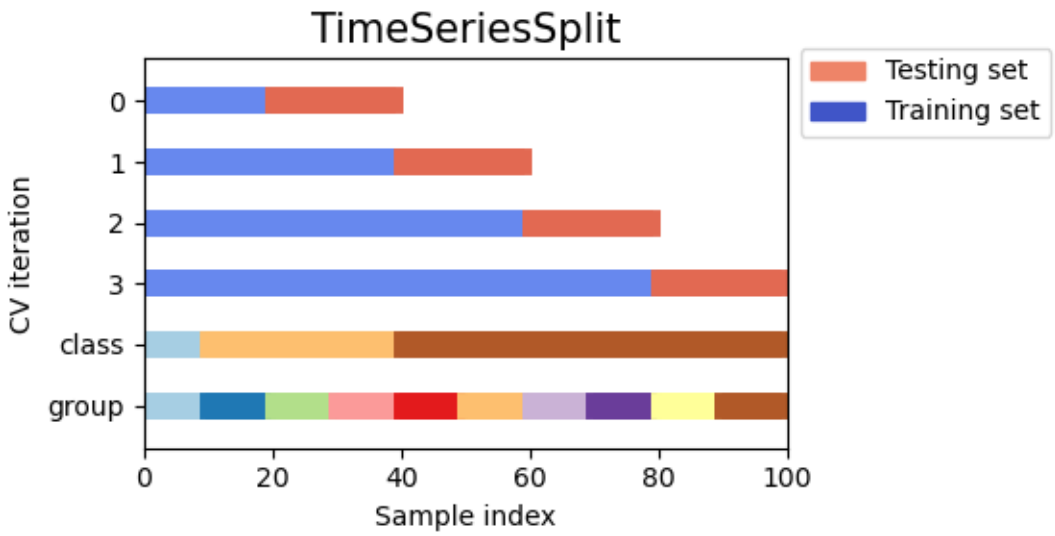
Initial test scores (RMSE) from using LightGBM were about 1.2, which is a lot better than using Catboost (~1.6). Computation time is about 12 min, which is about 3 times shorter than using Catboost to get the similar score. For this particular task, LightGBM performs much better that Catboost. It seems that prediction shift caused by target leak is not a severe problem, which LightGBM ignores while Catboost prevents.

KFold cross validation results are compared in the table below. Time features like '['hour', 'dayofyear', 'weekofyear', 'month'] are treated either as integer or category. It is found that those time features as integer significantly improve the test score from 1.283 to 1.020. This result is similar to what is found in CatBoost. By removing those additional time features, the test score also slightly improves but not as much as keeping them as integer.

LightGBM RMSE test scores, KFold = 5

int: ['hour', 'dayofyear', 'weekofyear', 'month'] cat: ['primary_use'] 'feature_fraction': 0.85	cat: ['hour', 'dayofyear', 'weekofyear', 'month'] cat: ['primary_use'] 'feature_fraction': 0.85	remove ['hour', 'dayofyear', 'weekofyear', 'month'] cat: ['primary_use'] 'feature_fraction': 0.85
0.952	1.424	1.235
0.963	1.209	1.195
0.941	1.277	1.069
1.050	1.225	1.204
1.191	1.278	1.336
average: 1.020	average: 1.283	average: 1.208

Time series split



TimeSeriesSplit with 5 folds was used for cross validation purpose. The scores are shown below. It can be seen that the scores vary much with each time split. However, the final model delivers the best score, 0.95, which is better than that last one of 1.19 obtained by using the KFold. This is actually caused by the difference in the splits hence the train/test sets in respect to the difference in time series splits and KFold splits. Looking further into the splits, it is the more training samples from the time series splits that make the prediction more accurate.

Normal KFold Splitting before Smoothing

int: ['hour', 'dayofyear', 'weekofyear', 'month'] cat: ['primary_use'] 'feature_fraction': 0.85
0.961
1.381
1.675
1.138
0.951
average: 1.222

Time series rolling average

In all decision tree-based learning algorithms, finding the best splitting point of tree node is critical in terms of predication accuracy and efficiency. If the locally neighboring data has a big variation that is not important to large scale prediction, it is beneficial to smooth it out. This not only helps increase accuracy but also save computing time. LightGBM uses

one-side sampling that takes advantage of higher gradients, or information gain in the partial samples. Sample smoothing can potentially suppress the high gradients in the local data, delivering additional benefits to the gradient boosting method. Therefore, by smoothing the data, we can reduce some the noisy signal to help with tree growing.

Pandas rolling method is used to do the smoothing. Rolling window is best set as 'day'. The two tables below show the 5 folds cross validation test scores using the smoothed data. It can be seen that the average score decreases from 1.222 to 0.973, a significant improvement in performance. Time series splits are also applied to the smoothed dataset. There is a further increase in predication accuracy by the score average. Percent mean absolute error is used to evaluate the algorithm. This metric allows one to see the prediction is about 10% off the actual in overall.

Normal KFold Splitting after Smoothing

int: ['hour', 'dayofyear', 'weekofyear', 'month'] cat: ['primary_use'] 'feature_fraction': 0.85	
root mean squared error	percent mean absolute error
0.878	0.084
1.943	0.211
1.301	0.129
0.249	0.033
0.491	0.047
average: 0.973	average: 0.102

Time Series Splitting after Smoothing

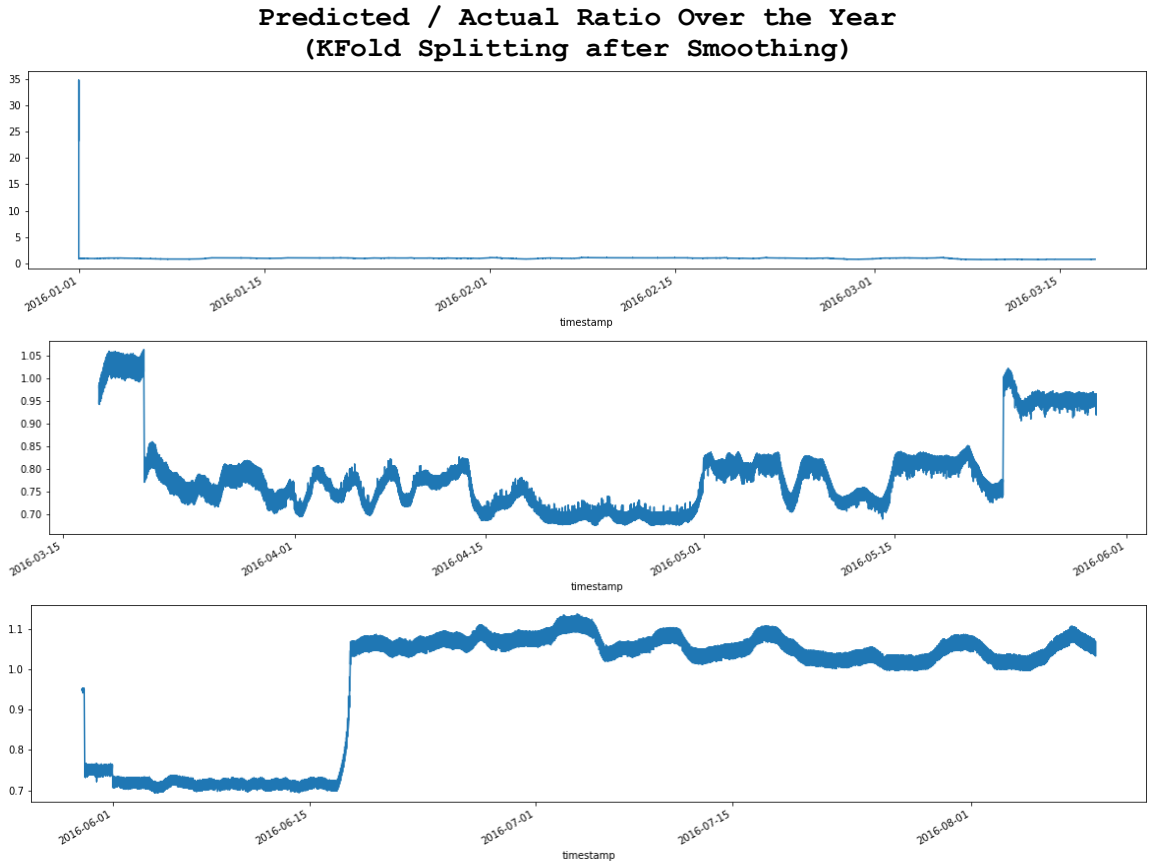
int: ['hour', 'dayofyear', 'weekofyear', 'month'] cat: ['primary_use'] 'feature_fraction': 0.85	
root mean squared error	percent mean absolute error
1.687	0.191
1.270	0.140
0.114	0.015
0.271	0.039
0.522	0.044
average: 0.773	average: 0.086

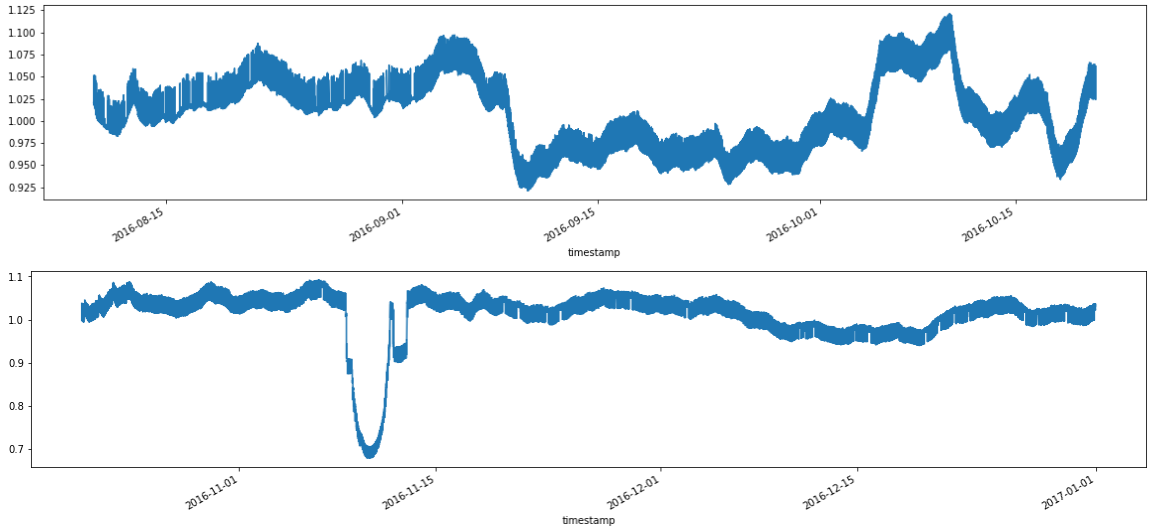
The ratio of predicted value against the actual is plotted in order to investigate how well the trained model works from sample to sample. Overall, the predicated value fluctuates around 1.0 in the maximum range from 0.7 to 1.3. The exceptionally high ratios close to 35 in the beginning of January in the KFold splitting is mostly likely caused by data entry

error. However, it was found removing those data doesn't help the prediction. This is probably because there are only about 1000 samples, a very small percentage of all 20 million samples, which doesn't affect the overall prediction performance.

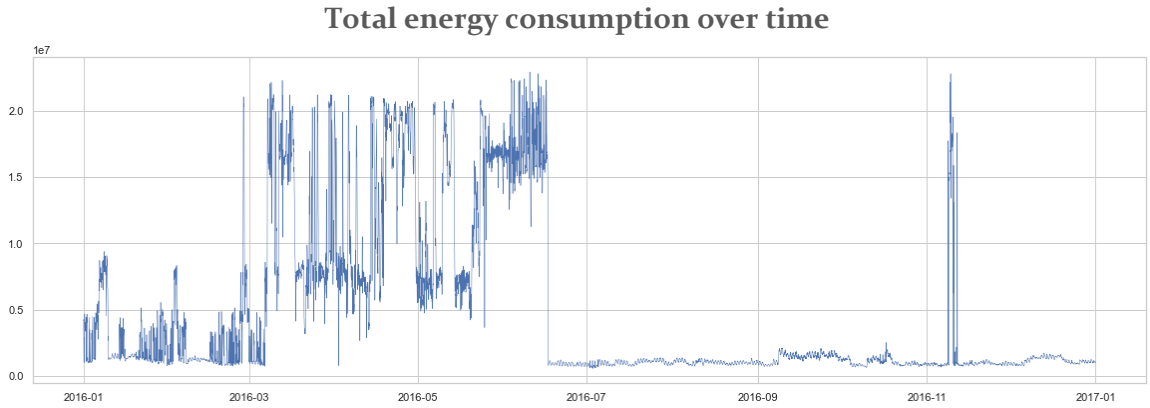
In both cases of splitting, March through July see much less amount of predicted energy usage that the actual, whereas, November through December see the predicted is very close to the actual. Building 1099, which consumes a lot of energy in November, is found to be responsible for it.

The abrupt deep drop in November corresponds with the unusually high usage observed in EDA section earlier. On March 7th and June 17th, there is a sharp transition in prediction that is very well corresponding with the abrupt transition from low energy usage season to high energy usage season or vice versa. Therefore, the models fail to catch the transition and result in significant prediction error. It would be a good idea to separate the dataset from the transition and train the models separately.





Time, Year-Month-Day



Influence of number of splits

Instead of using 5 splits, 6 KFold splits of the dataset is experimented. The purpose is to separate the sample roughly according to the month. However, there is no effect on model performance in doing so.

Feature importance

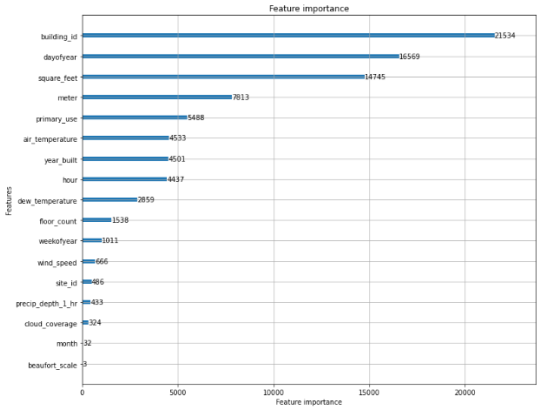
The ranking of feature importance is shown in the figure below, which is very different from that of Catboost. For example, the most important feature is building id for LightGBM whereas the top one is meter for Catboost. The importance of Dayofyear is also elevated for LightGBM to 2nd position from 5th for Catboost. The ranking of square feet

remains similar, being either 2nd or 3rd in ranking. The ranking of air temperature also remains similar, being either 4th or 5th in position. The ranking of year_built is either 6th or 7th. The importance of meter drops from 1st to 4th. The importance of site id drastically reduces from to 13th for LightGBM from 6th for Catboost.

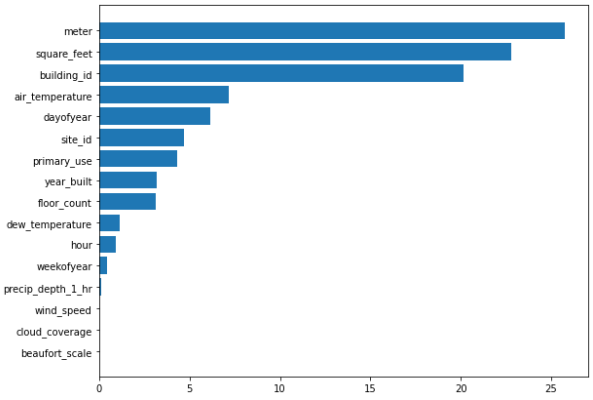
However, these top features like building id, day of year, square feet, meter, air temperature, regardless their relative ranking, are captured by both algorithms. These are indeed the predominant factors influencing the usage from domain knowledge point of view. It is worth noting the model successfully pick out air temperature as an important feature. Therefore, the model can be appropriately used to predict energy usage for the following years based on weather change. This would allow the effect of newer technology investment or change on energy saving to be separated from that of the weather change.

The relative ranking of dayofyear, hour and weekofyear remains the same order. Weather data like wind speed / beaufort scale, precipitation and cloud coverage remains as the least important features.

LightGBM



CatBoost



Another note is the air temperature data is treated as a whole when the models try to find the best splitting point using this variable/feature. However, it is the local air temperature that would affect the energy usage of a specific building. So, one of the future works can be modelling energy usage building wise.

It is worth noting that other factors, for example, budget/funding/project situation, decision making by the administrators when or what time to turn on AC also have great impact on energy usage and its timing. Therefore, the dataset for modeling work is always limited. Therefore, a better working model has to keep these human factors in mind so as to accordingly adjust relevant parameters in the model.

Conclusions

A rich energy usage dataset coming from over 1,000 buildings around the world combined with local weather data provides a wonderful machine learning opportunity for predicting energy usage. Exploratory data analysis finds the distribution of the data is highly skewed with respect to meter readings, energy type, square footage, primary use, etc. The energy usage is exceptionally high from March to June and a few days in November. There is also significantly higher energy usage on site #7 compared with other sites. Building #1099 on an educational site was found to consume at least 70 times more energy than any other buildings. Among the energy types, steam is the predominant over almost all year. Education is predominant in energy consumption. It is followed by office, entertainment / public assembly, lodging / residential, etc. The air temperature variation over the year doesn't corresponds well with the overall energy usage. The correlation heat map reveals that the lower the air temperature and dew temperature, the higher energy usage. The correlation coefficient between dew temperature and air temperature is 0.75, which agrees well with meteorology study.

Necessary data wrangling is conducted. Missing air temperature is interpolated linearly. Other missing values in 'year_built', 'floor_count', 'cloud_coverage', 'dew_temperature', 'precip_depth_1_hr', 'wind_speed' were imputed using SimpleImputer with means. New features like hour, day of year, week of year, month are added.

Since this dataset is well-structured and tabulated, the decision tree-based algorithms are best suited for machine learning. Catboost and LightGBM learning algorithms based on gradient boost are applied to the energy dataset. It is found Catboost is about 3 times slower than LightGBM to achieve similar score. For LightGBM, treating time features as integer significantly improve the test score (RMSE) from 1.283 to 1.020. The additional time features seem to have minimal effect on test score. Data smoothing by rolling average (set day as window size) significantly improves the score from 1.222 to 0.973. Time series splitting is found to improve the prediction, as compared with normal KFold splitting.

The ratio of predicted value against the actual is plotted in order to investigate how well the trained model works at the local level. It helps us find possible data entry errors in the first 1000 samples in the dataset, or the very beginning of the year. March through July see much less amount of predicted energy usage that the actual, whereas, November through December see the predicted is very close to the actual. The models fail to catch the transition and result in significant prediction error. It would be a good idea to separate the dataset from the transition and train the models separately.

The feature importance obtained by LightGBM is very different from that of Catboost. For example, the most important feature is building id for LightGBM, whereas the top feature is meter or energy type for Catboost. However, these top features like building id, day of year, square feet, meter, air temperature, regardless their relative ranking, are captured by

both algorithms. These are indeed the predominant factors influencing the usage from domain knowledge point of view. Weather data like wind speed / beaufort scale, precipitation and cloud coverage remains as the least important features.

It is worth noting the model successfully pick out air temperature as an important feature. Therefore, the model can be appropriately used to predict energy usage for the following years based on weather data. This would allow the effect of newer technology investment or change on energy saving to be separated from that of the weather change.

Lastly, other factors, for example, budget/funding/project situation, decision making by the administrators when to turn on AC, etc., also have great impact on energy usage and its timing. Therefore, it is necessary to have the human factors in mind when building a realistic model.

Future Work

- Check with domain expert's possible data entry errors in the January meter readings because this the only occurrence with off-the-chart prediction.
- Separate the dataset from the transition where the high energy usage season change to lower or vice versa and train the models separately.
- Separate samples according to meter or energy type for machine learning individually
- Check with domain expert on how budget/funding/project situation, decision making by the administrators when or what time to turn on AC, etc., impact on energy usage and its timing.
- Predict energy usage building wise. This way the local weather data can play a more important role.
- Continue feature engineering by adding new variables such as holidays because it affects energy usage for obvious reasons