BACKGROUND

How much does it cost to cool a skyscraper in the summer? A lot! And not just in dollars, but in environmental impact. Thankfully significant improvements are being made to improve building efficiencies. But are these improvements working? Existing methods to estimate the impact of improvements are fragmented, and do not scale well. Some assume a specific meter type (e.g. chilled water, electric, hot water, or steam), or perform poorly on specific building types.

Building load forecasting could be an avenue to develop a cost-effective, scalable, approach to improvement savings estimation. Figure 1 depicts how this approach would be implemented. Basically, historical building energy consumption is used to predict future energy consumption. After improvements have been implemented, reported energy consumption is compared to the prediction.

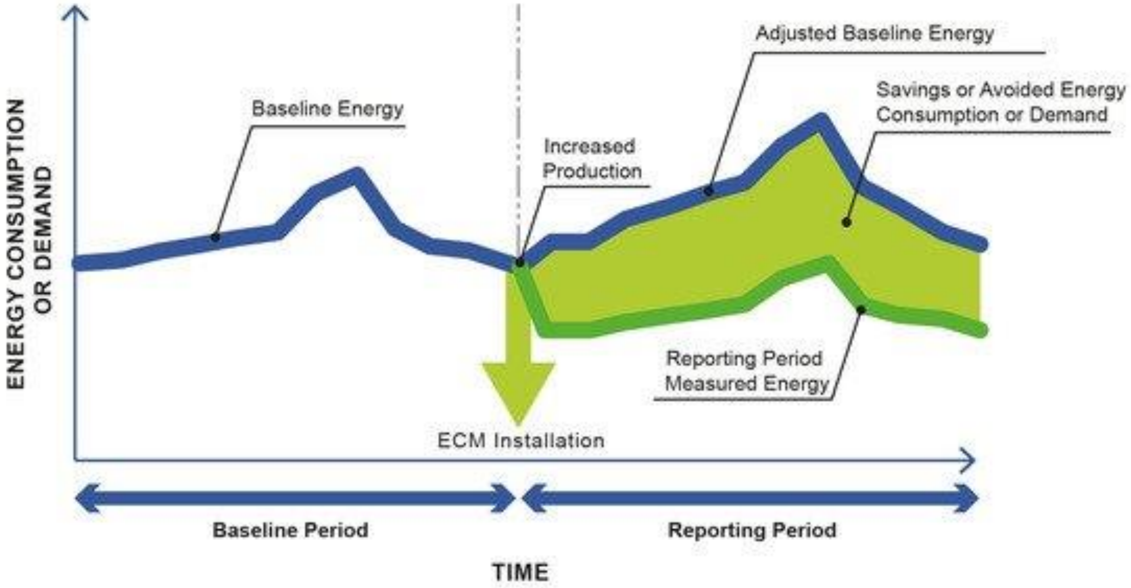


Figure 1: Building Load Forecasting approach to Improvement Savings Estimation

To evaluate the potential of this approach, a Kaggle competition was created. The goal of the project is to predict the energy usage for thousands of buildings that contain at least one of the following meters: chilled water, electric, hot water, steam.

PROJECT DATA DESCRIPTION

Hourly readings for available meter types in each building is provided for the year 2016. Also, hourly building-location weather data is provided for 2016 and 2017. Using this data, I predict the metered building energy usage for 2017.

DATA EXPLORATION

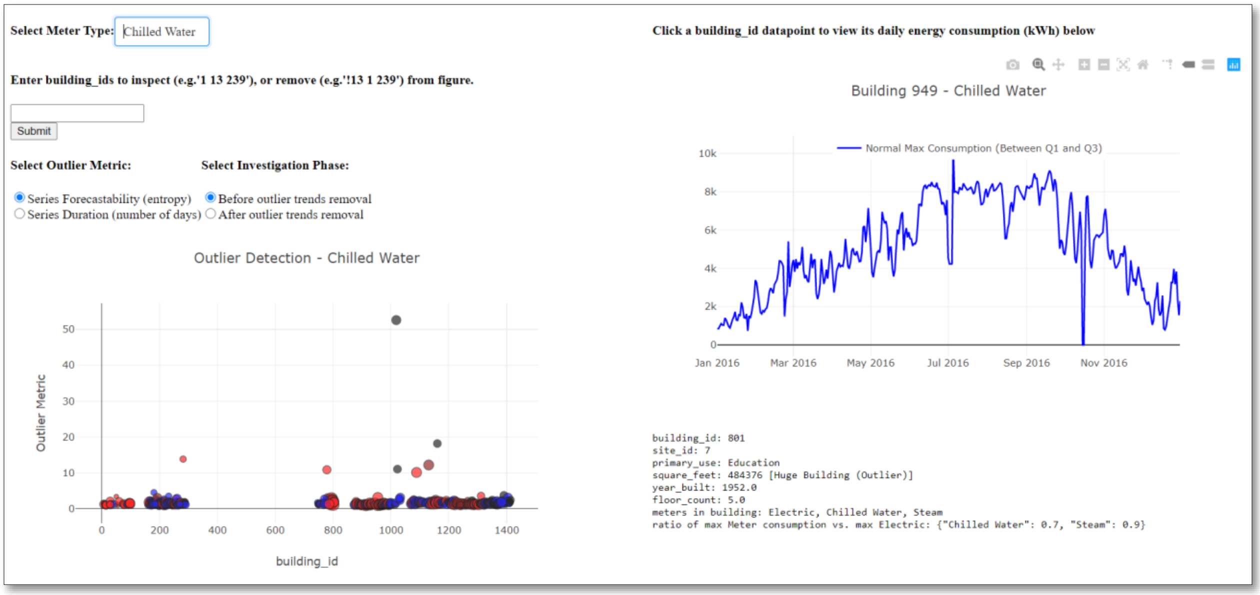
For this project, we are provided the following information:

- Building ID: Identifiers for 1449 buildings.
- Building Characteristics:
  - Primary Use: Explicit categories include Education, Office, Parking, Entertainment, Retail, e.t.c. Any building use that doesn't fall into one of 15 explicit classes is grouped into a 16th category, "Other".
  - Square Feet: Total floor area for each building.
  - Year Built: The oldest and newest were constructed in 1900 and 2017 respectively. This information was provided for only 46.6% of buildings in the dataset.
  - Floor Count: The max and min number of floors across all buildings is 1 and 26 respectively. This information was provided for only 24.5% of buildings in the dataset.
- Site ID: An identifier for the location of each building. There are 16 locations.
- Train Weather Data: Hourly air temperature, dew temperature, cloud coverage, hourly precipitation depth, sea level pressure, wind direction, and wind speed for the year 2016. A breakdown for the percentage of missing values per weather characteristic is provided in figure 5.
- Forecast Weather Data: Hourly weather data for the years 2017 and 2018.
- Meter ID: Identifiers for each meter type; Electric (0), Chilled Water (1), Steam (2), Hot Water (3)
- Meter Reading: Hourly meter reading for 2016 across all building meters. Because some buildings have multiple meters, we end up with 2380 time series across all 1449 buildings.

Metered Building Energy Usage

I explored the readings provided for all building meters using Plotly-Dash interactive visualizations. Before building dashboards, it was beneficial to understand the usage demands being met by the energy systems that these meter types track.

- Electric: Could support all sorts of demands, including heating, cooling, lighting, appliances, plug loads, e.t.c.
- Chilled Water: Would typically support cooling demands. Hence, readings would likely be higher during summer.
- Steam: Would typically support heating demands. Hence, readings would likely be higher during winter.
- Hot Water: Would typically support heating demands. Hence, readings would likely be higher during winter.



Select Meter Type: Chilled Water

Enter building_ids to inspect (e.g.'1 13 239'), or remove (e.g.'!13 1 239') from figure.

Select Outlier Metric: ● Series Forecastability (entropy) ○ Series Duration (number of days)

Select Investigation Phase: ● Before outlier trends removal ○ After outlier trends removal

Click a building_id datapoint to view its daily energy consumption (kWh) below

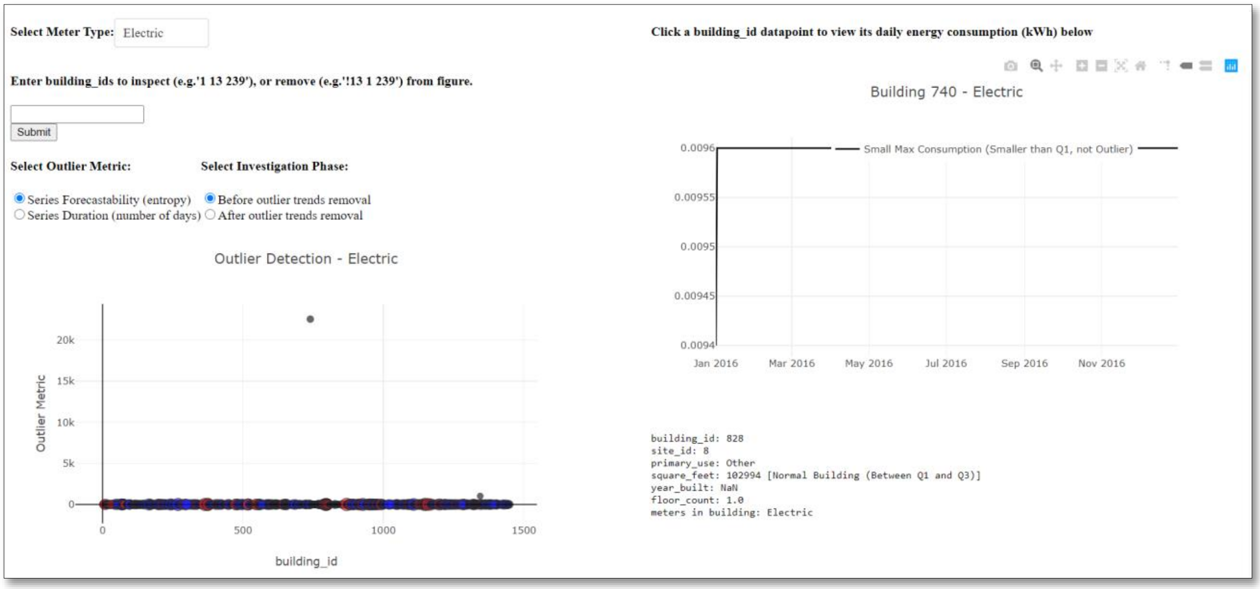
```
building_id: 801
site_id: 7
primary_use: Education
square_feet: 484376 [Huge Building (Outlier)]
year_built: 1952.0
floor_count: 5.0
meters in building: Electric, Chilled Water, Steam
ratio of max Meter consumption vs. max Electric: {"Chilled Water": 0.7, "Steam": 0.9}
```

Figure 2: Dashboard for meter reading exploration

Upon choosing a meter type, hovering and clicking datapoints within the Outlier Detection scatter plot, (and sometimes entering the IDs for buildings to "focus on", or "be removed" from the scatter plot) the dashboard for meter reading exploration (figure 2) reveals:

- Total daily energy consumption during 2016 on a line chart
- Characteristics for each building, including:
  - Location, primary use, year built, floor count, available meter types
  - Comparison of max energy consumption across all meter types in the selected building, so I can identify the primary energy systems for said building.
  - Size of this building compared to all others in the dataset (i.e. within the interquartile range, smaller than typical, larger than typical, outlier size).
  - Max daily energy consumption for this meter compared to all others in the dataset (i.e. within the interquartile range, smaller than typical, larger than typical, outlier consumption).
- Outlier meter readings.

For outlier detection, my primary strategy was to find faulty meters. To do this, first, I leveraged time series entropy to compute the complexity of each time series. Compared to other meters of the same type, a meter with steady daily consumption could be faulty. I plot the inverse of the time series entropy in order to set up these meters to standout on the scatter plot. For instance, the meter readings, shown in figure 3, for the "normal sized" Building 740 are likely faulty. A building of this size with only one energy system shouldn't have such low, unfluctuating, energy consumption (compared to the rest of the dataset).



Select Meter Type: Electric

Enter building_ids to inspect (e.g.'1 13 239'), or remove (e.g.'!13 1 239') from figure.

Submit

Select Outlier Metric:

● Series Forecastability (entropy)
○ Series Duration (number of days)

Select Investigation Phase:

● Before outlier trends removal
○ After outlier trends removal

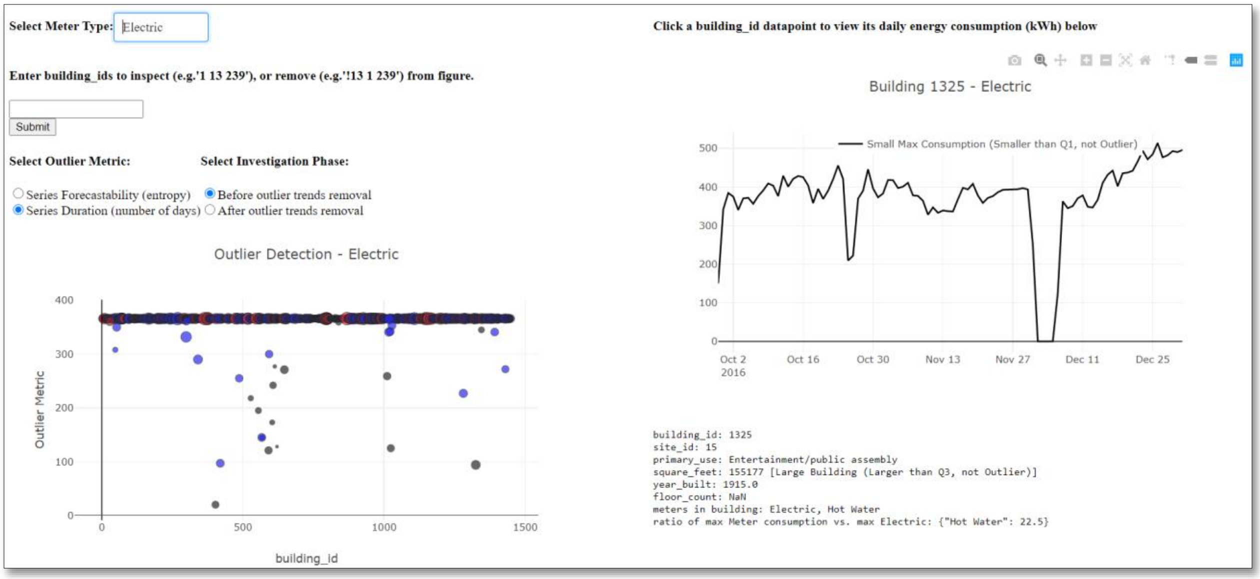
Click a building_id datapoint to view its daily energy consumption (kWh) below

```
building_id: 828
site_id: 8
primary_use: Other
square_feet: 102994 [Normal Building (Between Q1 and Q3)]
year_built: NaN
floor_count: 1.0
meters in building: Electric
```

Figure 3: Example of faulty meter (identified using time series entropy)

The inverse of the entropy for electric meter readings at Building 740 is quite high, at 22500. Upon inspecting the population of buildings for each meter type, an entropy-inverse cutoff was chosen. Meters with an entropy-inverse above these cutoffs were removed from the study. The corresponding cutoffs for Electric, Chilled Water, Steam, and Hot Water meters are 3.5, 3.0, 3.5, and 6.0 respectively.

Another strategy for identifying outliers involved identifying the readings duration. For predicting energy consumption 1 year into future, it isn't advisable to train with less than 1 year of data. Figure 4 reveals that we have 3 months of usage readings for the electric meter in Building 1325. Furthermore, the scatter plot on the dashboard reveals that several meters tracked consumption for less than a year.



Select Meter Type: Electric

Enter building_ids to inspect (e.g.'1 13 239'), or remove (e.g.'!13 1 239') from figure.

Submit

Select Outlier Metric:

○ Series Forecastability (entropy)
● Series Duration (number of days)

Select Investigation Phase:

● Before outlier trends removal
○ After outlier trends removal

Click a building_id datapoint to view its daily energy consumption (kWh) below

```
building_id: 1325
site_id: 15
primary_use: Entertainment/public assembly
square_feet: 155177 [Large Building (Larger than Q3, not Outlier)]
year_built: 1915.0
floor_count: NaN
meters in building: Electric, Hot Water
ratio of max Meter consumption vs. max Electric: {"Hot Water": 22.5}
```

Figure 4: Example of faulty meter (identified using time series duration)

Lastly, some behaviors that signal potentially faulty readings weren’t captured by the above outlier metrics. All electric meters in locations with Site ID “0” have electric readings of 0 for the first 4.5 months of the year. Likewise, for all meter types in locations with Site ID “15”, the reported electric readings is zero from February 11, 2016 till March 29, 2016.

Ultimately, 483 of the original 2380 meters were excluded from the study. This is about 20.3% of the provided dataset. Figure 5 reveals the breakdown of the exclusions, per meter type.

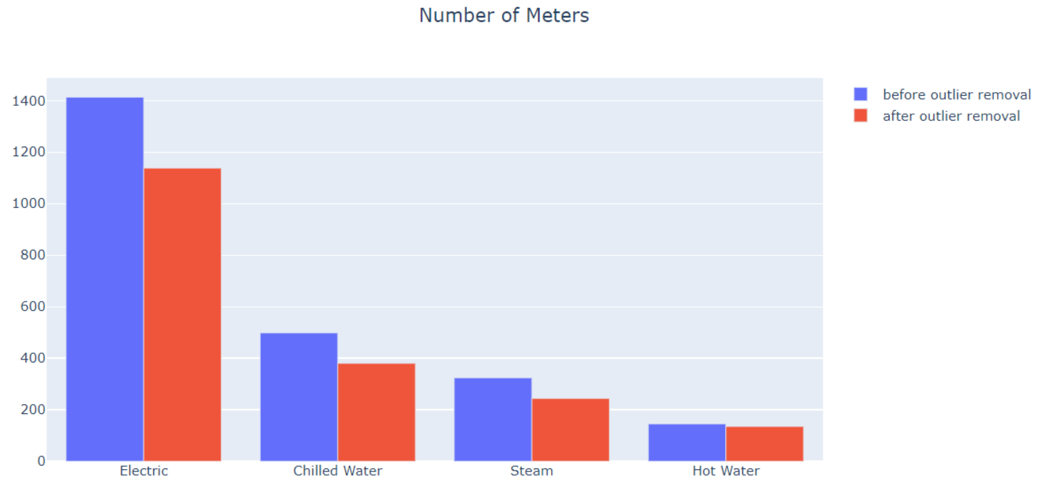


Figure 5: Number of meters before and after outlier removal

Weather Data

Figure 6 shows the percent of missing data for each provided weather characteristic that could influence energy demand. We are missing about half (49.5%) of the cloud coverage data. Figure 7 reveals the dashboard that I create to explore the weather dataset.

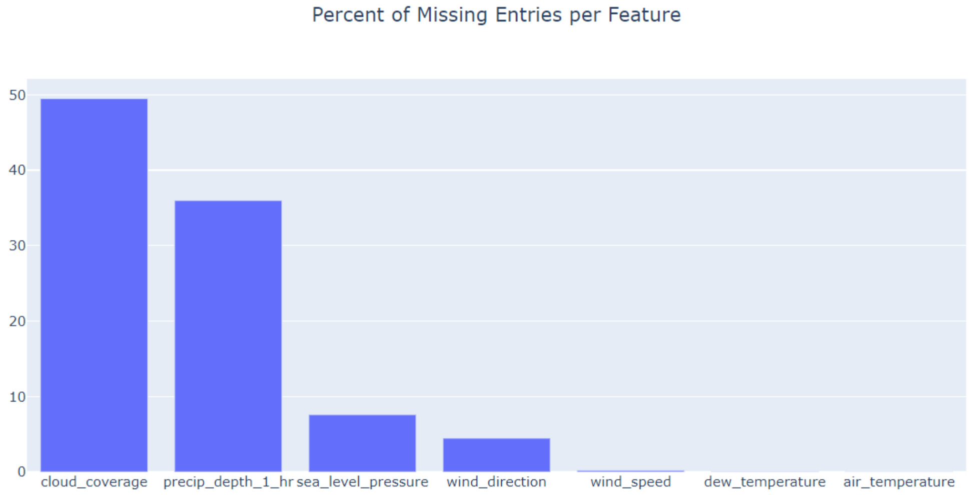
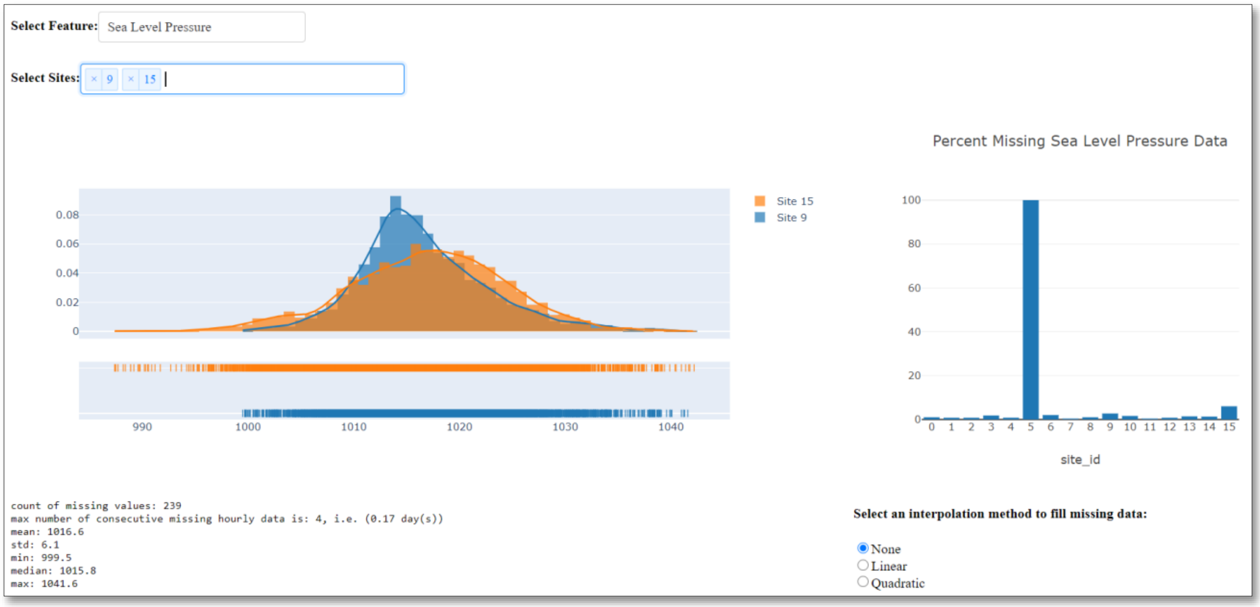


Figure 6: Percentage of missing weather train data



Select Feature: Sea Level Pressure

Select Sites: × 9 × 15

```
count of missing values: 239
max number of consecutive missing hourly data is: 4, i.e. (0.17 day(s))
mean: 1016.6
std: 6.1
min: 999.5
median: 1015.8
max: 1041.6
```

Select an interpolation method to fill missing data:

● None
○ Linear
○ Quadratic

Figure 7: Dashboard for weather data exploration

Upon choosing a weather characteristic, entering at least one Site ID, hovering over the rug plot (and sometimes selecting an interpolation method to fill missing data), the dashboard reveals:

- The percentage of missing weather data for each location
- For any time series, one can see the:
  - Count of missing data
  - Maximum number of consecutive missing hourly data. This would affect the strategy for making estimates that fill the missing entries.
  - The statistical description of the time series. Observing the change in statistical description of the data (and distplot curve) before and after filling missing values could reveal when interpolation strategies yield favorable or unfavorable results.

While exploring Precipitation Depth, I realized that the percentage of missing data in Site 7 and Site 11 are the same (91.5%). This random number seemed too exact. It prompted further investigation. I found out that Site 7 and Site 11 are the same location. Two other locations that are the same are Site 0 and Site 8. These findings were beneficial downstream, as I was able to avoid entering duplicate data into machine learning algorithms that predict missing weather data. This helped reduce algorithm train time.

FEATURE ENGINEERING

Before filling out missing weather data, these steps were taken:

- Wind direction was transformed into its x-axis and y-axis components. This will help downstream algorithms to understand that 359 degrees is much closer to 0 degrees than it is to 250 degrees.
- Precipitation Depth was removed from the study. Compared to other features, it doesn't contain a lot of information (as most values for this feature is zero).

- Relative humidity was created using the August-Roche-Magnus approximation. This would allow downstream algorithms harness the explicit relationship between air (dry bulb) temperature and dew (point) temperature.

$$relative\ humidity = 100 \times \frac{e^{\left(\frac{17.625 \times T_{dewpoint}}{243.04 + T_{dewpoint}}\right)}}{e^{\left(\frac{17.625 \times T_{drybulb}}{243.04 + T_{drybulb}}\right)}}$$

Figure 6 (above) reveals the percentage of missing weather data for each characteristic. These missing entries were estimated using:

- Interpolation: For features with consecutive missing hourly entries that span less than a day.
    - Linear Interpolation: The dashboard (see figure 7) revealed that quadratic interpolation for wind speed and wind direction yields unwanted impact on the distributions for these features. Hence linear interpolation was implemented.
    - Quadratic Interpolation: Typical daily air and dew temperature doesn’t exhibit linear behavior. Hence, quadratic interpolation was implemented.
- Machine Learning: For features with consecutive missing data that span more than one day.
    - Ridge Regression: This was carried out to fill missing entries for Sea Level Pressure. Inputs for this prediction include filled out wind speed, wind direction, air temperature, dew temperature, and relative humidity. The root mean square error (RMSE) on the predictions is ~7. Compared to typical sea level pressure values, ~1000, this error is quite small.
    - Random Forest Classifier: This was carried out to fill missing entries for Cloud Coverage. Inputs for this prediction include filled out wind speed, wind direction, air temperature, dew temperature, relative humidity, and sea level pressure. To generate the best possible estimates:
        - Stratified K Fold Cross Validation was used to set up the data for classifier parameter tuning. This ensures that underrepresented cloud coverage classes are available within each broken-out train and test folds.
        - Scikit-Learn’s GridSearchCV was set to select the parameter combination that yields the best macro-averaged F1-score. This ensures that the best parameter tuning combination weighs algorithm performance across all classes evenly.
        - Other classifiers that were evaluated include K Nearest Neighbors, Support Vector Machines, AdaBoost, Gradient Boosting, Light Gradient Boosting Machine, and Multi-layer Perceptron (Neural Network). Compared to the mentioned alternatives, Random Forest has the second fastest parameter tune time, and best F1 Macro.
        - For each classifier, the Accuracy and RMSE was determined for the best parameter settings because these metrics are easy to communicate. Random Forest has an accuracy of 65.53%, and a RMSE of ~2. The available classes that describe cloud coverage are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 oktas.

Lastly, a binary weekday-weekend feature was created to capture the distinction between occupant behavior across weekdays and weekends.

BUILDING LOAD FORECAST

Forecast Evaluation Metric

For this project, the forecast evaluation metric is the Root Mean Square Log Error (RMSLE). To explain RMSLE, I compare it to the more ubiquitous RMSE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(pred_i - actual_i)^2}$$

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}[\log(pred_i + 1) - \log(actual_i + 1)]^2}$$

For both metrics, a value of zero signals a perfect prediction. In the case of RMSLE, we take the log of predictions and actual values. Hence RMSLE is used when one doesn't want to penalize huge differences between predicted and actual values. The below bullets and table 1 reveal differences between both metrics.

- Case 1: both predicted and actual values are small: RMSE and RMSLE are almost the same
- Case 2: either the predicted or actual value is large: RMSE > RMSLE
- Case 3: both predicted and actual values are big: RMSE > RMSLE (compared to RMSE, RMSLE is almost negligible)

Table 1: RMSLE vs RMSE Examples

Example	Prediction	Actual	RMSE	RMSLE
Case 1	0.5	1	0.5	0.287
Case 2	0.5	1200	1199	6.69
Case 3	1000	1200	200	0.182

There are 2 ways that working with RMSLE would influence this project:

1. RMSLE cannot execute for negative numbers. Hence, it would flag an error when we have negative predictions for energy usage. This is beneficial because such a forecast is not plausible.
2. RMSLE could be more favorable towards certain meter types. Recall that certain energy systems meet "near zero" or low demand for almost half a year. For instance, chilled water system energy consumption is typically low during the winter. For these low meter readings, RMSLE functions like RMSE. On the other hand, RMSE could report more conservative numbers for meters with higher year-round energy consumption.

Leaked Energy Consumption for Forecast Evaluation

For this project, I have no access to the data that would help me evaluate my forecasts. Thankfully, some leaked future meter readings were compiled by a Kaggle User. The time range for these leaked readings span both the train and test durations (i.e. 2016 and 2017). To validate the accuracy of this leak, I check the RMSLE between the original train meter readings, and those within the leak. A value of 0.057 reveals that both train datasets are quite similar. To account for the differences between both sets that prevent the RMSLE from being 0, I train my time series algorithms on the leak, as opposed to the provided train data for the competition.

Unfortunately, only 392 of the 1897 meters that remain in my study are available within the leak. Furthermore, none of these 392 meters represent the steam meter type.

Forecast Approach and Results

Using the provided 2016 meter readings, and the weather data for both 2016 and 2017, I predict the energy usage for 2017. For my forecasts, I use Facebook Prophet (FP). Besides being faster (and easier to deploy) compared to ARIMA based algorithms, I leverage this algorithm for the following reasons:

1. FP can recognize and handle the multiple seasonality (daily, weekly, monthly, yearly) that could be present in sub-daily data.
2. Several meters are missing hourly data. FP can handle timestamps that are not evenly spaced.
3. FP allows users to set capacities. Capacities allow me to constrain the trend for predictions. Hence, I can set Facebook Prophet to make only predictions greater than zero for energy usage.

Note that FP capacities do not control seasonality. Hence, some generated forecasts still contain negative energy usage. When this occurs, FP seasonality components were manually tuned. Unfortunately, the duration for each time series is too short to support hyperparameter tuning. Hence, to manually tune, I applied several combinations of:

- Turning off one of the multiple seasonality (daily, weekly, monthly, yearly)
- Changing FP's default *seasonality_mode* from *additive* to *multiplicative*
- Updating the value for the *seasonality_prior_scale*

Of the 392 possible forecasts, 313 (80%) were plausible (i.e. contain only positive predictions). Further parameter tuning could yield plausible forecasts for the remaining 79. The plausible forecasts have a RMSLE of 0.94. The RMSLE for the best performing algorithm within the Kaggle competition is 1.231. It would appear that my algorithm is better than those within the competition. However, there are two differences between this work, and the objectives for the competition:

1. I predict 1 year into the future (usage for just 2017). Competitors predict 2 years into the future (2017 and 2018)
2. I forecast for 313 meters. Competitors forecast for all 2380 meters (this includes meters that are presumed to be faulty, and meters which tracked consumption for less than a year).

Figure 8 reveals the breakdown of RMSLE across meter types. Compared to hot water (1.65) and chilled water (2.15) meters, the RMSLE for predictions on electric meters (0.59) is low. A possible explanation for this is that both chilled water and hot water meters track usage that is typically seasonal. For almost half the year, seasonal usage is closer to zero. The RMSLE on predictions at these low values would typically be higher than those for higher usage.

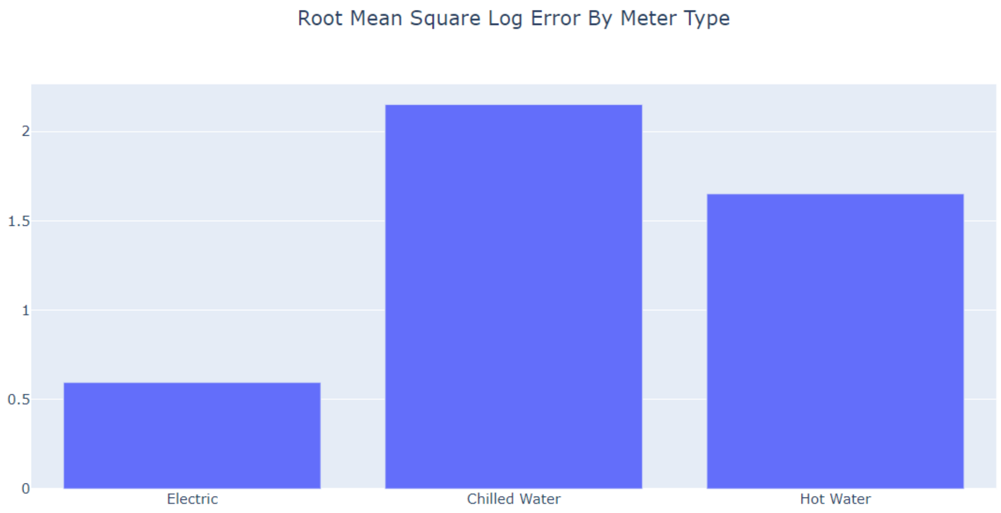


Figure 8: RMSLE by meter type

Generally, it was easier to generate forecasts with only positive energy usage for certain meter types. 251 (electric), 93 (chilled water), and 47 (hot water) meter types make up the 392 generated forecasts. After some manual parameter tuning, plausible predictions were generated for about 80% of these forecasts. Figure 9 reveals that the percent of plausible forecasts were 91%, 85%, and 45% for the corresponding electric, hot water, and chilled water meters. Additional parameter tuning could yield plausible forecasts for the remaining meters. However, these results could support a conclusion that generating plausible energy usage predictions is more computationally expensive for certain meter types.

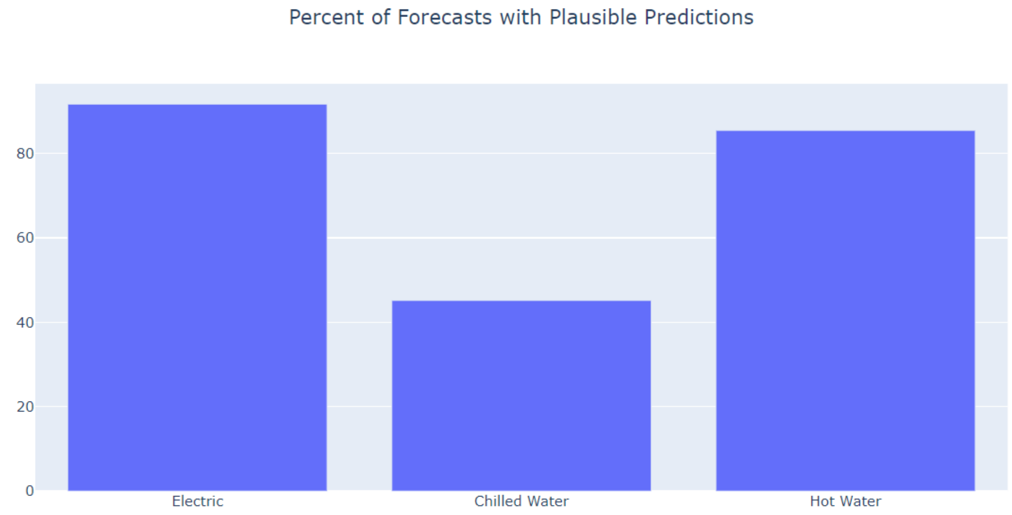


Figure 9: Percent of forecasts with only positive predictions by meter type