TO PREDICT HOW MUCH ENERGY A BUILDING WILL CONSUME

Submitted as project for Diploma in AI and ML at UoH and AAIC

Abstract

In this project a model for predicting metered building energy usage in the following areas: chilled water, electric, hot water, and steam meters is developed. For development of model the data comes from over 1,000 buildings, with better estimates of these energy-saving investments, large scale investors and financial institutions will be more inclined to invest in this area to enable progress in building efficiencies.

Saurav Kr Sahay

Sahay.saurav@gmail.com

Contents

1. Literature survey & Data Acquisition

1.1. Problem definition:

1.1.1. What is the problem all about?

In this project it is required to develop accurate models of metered building energy usage in the following areas: chilled water, electric, hot water, and steam meters. The data comes from over 1,000 buildings, with better estimates of these energy-saving investments, large scale investors and financial institutions will be more inclined to invest in this area to enable progress in building efficiencies.

Significant investments are being made to improve building efficiencies to reduce costs and emissions. The question is, are the improvements working. Thus, the main aim of the project is to predict the energy consumption of a building without retrofitting.

1.1.2. Why is this an important problem to solve?

Under pay-for-performance financing, the building owner makes payments based on the difference between their real energy consumption and what they would have used without any retrofits. The latter values have to come from a model. As per the given details the existing methods of estimation are fragmented and do not scale well. Some assume a specific meter type or don't work with different building types. Thus, it is important to come up with a method or say model through which a correct estimate of energy consumption by building without retrofitting work can be estimated. As this estimate need to be agreed upon by both building owners and agency which is going to do retrofitting work, the outcome need to be both accurate and interpretable. The model should also be overcome difficulties or error with existing practices. Thus it should be able to predict the energy consumption of buildings of several types as well as diverse metering arrangements.

1.1.3. Business/Real-world impact of solving this problem.

In today's world almost everything runs on electricity. The power generation sector is also one of the major contributors of emission and greenhouse gases. Apart from industrial and transportation use a sizable part of the electricity generated through the power sector is consumed in commercial/educational building for day-to-day activities like building cooling, chilled water, hot water etc. Thus, In the energy sector deploying new ways to reduce both energy usage and the carbon footprint of buildings while optimizing occupants' comfort and productivity. It is always beneficial as it will not only have a cost savings for the consumers but also have a significant impact on climate.

If it can be proven that with installation of new and advance building cooling devices large amount of money can be saved. It will encourage both service provider as well as building owner to go for these upgrades. Larger the saving on account of these upgrades faster will be recovery cost of new installation as well as adoption of innovative technologies.

1.2. Dataset:

1.2.1. Source of the dataset.

The data set is provided by American Society of Heating, Refrigerating and Air- Conditioning Engineers (ASHRAE) is an American professional association seeking to advance heating,

ventilation, air conditioning and refrigeration (HVAC&R) systems design and construction. ASHRAE has more than 57,000 members in more than 132 countries worldwide. Its members are composed of building services engineers, architects, mechanical contractors, building owners, equipment manufacturers' employees, and others concerned with the design and construction of HVAC&R systems in buildings.

The data set is floated by ASHRAE for a Kaggle completion organized in Year 2019. The data comes from over 1,000 buildings over a three-year timeframe. ASHRAE Technical Committee 4.7 which identifies, evaluates, develops, and recommends procedures for calculating energy performance for the built environment was mainly responsible for data collection. In addition to ASHRAE Technical Committee 4.7: Energy Calculations, the three organizations Singapore Berkeley Building Efficiency and Sustainability in the Tropics, BUDS Lab and Texas A&M Engineering Experiment Station have generously contributed to data collection.

1.2.2. Explanation of each feature and datapoint available.

SL No	Feature	Explanation	Data type
1	building_id	Foreign key for the building_metadata.csv. Which link building_id with site_id as all weather data is provide for site_id	int64
2	meter	The meter id code. Read as {0: electricity, 1: chilledwater, 2: steam, 3: hotwater}. Not every building has all meter types.	int64
3	timestamp	When the measurement was taken	object 8784
4	meter_reading	he target variable. Energy consumption in kWh (or equivalent). This is real data with measurement error, which will impose a baseline level of modeling error. For the site 0 electric meter readings are in kBTU. Multiplication by 0.2931 is required to get to model inputs into kWh like the other sites	float64
5	site_id	Foreign key for the weather files. Weather information for these 16 sites are given	int64
6	primary_use	Indicator of the primary category of activities for the building like Education, Manufacturing, Retail, office etc.	object
7	square_feet	Gross floor area of the building	int64
8	year_built	Year building was opened	float64
9	floor_count	Number of floors of the building	float64
10	air_temperature	Temperature in Degrees Celsius	float64

11	cloud_coverage	Portion of the sky covered in clouds in okta. With 0 Sky completely clear and 8 completely cloudy. While 9 Indicates error.	float64
12	dew_temperature	Degrees Celsius. The dew point is the temperature to which air must be cooled to become saturated with water vapor.	float64
13	precip_depth_1_hr	Millimeters. Precipitation is any product of the condensation of atmospheric water vapor that falls under gravitational pull from clouds. The main forms of precipitation include drizzling, rain, sleet, snow, ice pellets, graupel and hail.	float64
14	sea_level_pressure	Millibar/hectopascals	float64
15	wind_direction	Compass direction (0-360)	float64
16	wind_speed	Meters per second	float64

1.2.3. Data Size and any challenges you foresee to process it?

Training data contains two files “train.csv” of size 678 MB and “weather_train.csv” of size 7.5 MB containing data for year 2016. While the test data contains two files “test.csv” of size 1.5 GB and “weather_test.csv” of size 15 MB containing data for year 2017-18.One Building meta data of size 46 kB is which connect both Training data as well as both Test Data. Although the size of data set is not too large. However, the training data is relatively less i.e., only for one-year 2016 while the test data set is almost double i.e., for two years 2017 and 2018. This relatively less number of data points for training may result in some problem. However, this is only preliminary observation and any conclusion can only be made after doing detail analysis.

1.2.4. Tools (Pandas, SQL, Spark etc) that you will use to process this data.

As data is set relatively large to access it in python, Pandas will be required. SQL can also be used to combine the data available in different files. Since training data, itself is in two different files with some information available in metadata file, SQL is expected to come handy in merging of dataset. For preprocessing of data scikit-learn will be used.

1.2.5. Data Acquisition

1.2.5.1. Open Source data (or) via API (or) via scrapping (or) Generate synthetic Data

Data is floated by ASHRAE for Kaggle completion ASHARE - Great Energy Predictor III. It can be directly downloaded from Kaggle website after signing up for this closed completion. Kaggle also provides an API through which it can be downloaded or directly uploaded in google colab notebook.

```
>_  kaggle competitions download -c ashrae-energy-prediction
```

It is to be noted that if we wish to use the API for data download it is first required to create API token first in Kaggle website.

1.2.5.2. Can you acquire more data in addition to your primary source? If so, how?

One of the organizations which helped ASHRAE generously for data collection was BUDS Labs (https://www.budslab.org/) which is a scientific research group that leverages data sources from the built and urban environments to improve the energy efficiency and conservation, comfort, safety and satisfaction of humans. In their website they have link for the Building Data Genome 2 (BDG2) Data-Set (https://github.com/buds-lab/building- data-genome-project-2#the-building-data-genome-2-bdg2-data-set) for which also BUDS Labs has contributed. BDG2 is an open data set made up of 3,053 energy meters from 1,636 buildings. The time range of the times-series data is the two full years (2016 and 2017) and the frequency is hourly measurements of electricity, heating and cooling water, steam, and irrigation meters. A subset of the data was used in the Great Energy Predictor III (GEPIII) competition hosted by the ASHRAE organization in late 2019. The GEPIII sub- set includes hourly data from 2,380 meters from 1,449 buildings.

However, during the project only original dataset provided for Kaggle competition by ASHRAE for Great Energy Predictor III will be used.

1.3. Key metric (KPI) to optimize

1.3.1. Business Metric definition

As this is a regression problem where final output i.e. the predicted energy reading needs to be compared with real energy consumption. Thus, the objective is to minimize the error between the prediction and actual readings. Minimization of Root Mean Square Logarithmic Error (RMLSE) can be used as business metric.

RMSLE can be calculated as

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

Where:

ϵ is the RMSLE value (score)
n is the total number of observations in the (public/private) data set,
p_i is your prediction of target, and
a_i is the actual target for i.
$\log(x)$ is the natural logarithm of x

There are many performances metric that can be utilize to understand the performance of model. of the performance measurement metrices are

1.3.2. Why is this metric used?

There are many performances metric that can be utilize to understand the performance of model.

However we have selected this metric because this is a slight modification of widely accepted RMSE metric but it has a plus point that it is robust to outliers

1.3.3. Alternative metrics that can be used? Why are they not preferred in this case?

There are many performances metric that can be utilize to understand the performance of model. Like

- R-Squared (R^2) error
- Root Mean Squared Error (RMSE),
- Mean Absolute Error (MAE)
- Relative Squared Error (RSE)
- Mean Squared Error (MSE)
- Mean Absolute Percentage Error (MAPE)

1.3.4. Pros and cons of the metric used.

Some of the Pro of the RMSLE are

- Robustness of RMSLE to the outliers,
- RMSLE is not scale-dependent and is useful across a range of scales.
- It considers only the relative error between the actual value and the predicted value

While some of the cons are

- It penalizes the underestimation of the actual value more severely than it does for the Overestimation.

1.3.5. Where does this metric fail? Where should it not be used?

It penalizes the underestimation of the actual value more severely than it does for the Overestimation. It is relatively biased to overestimate. Thus, it should not be used at places where any overestimation needs to be penalized heavily.

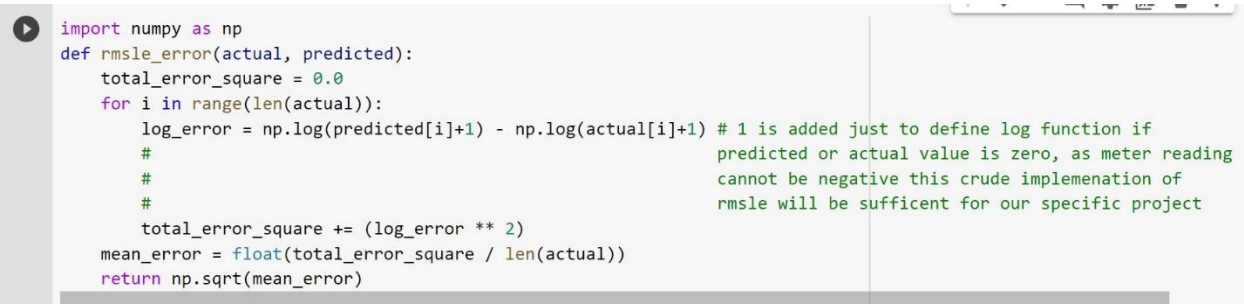
1.3.6. Where (on what type of problems is this metric used elsewhere in ML and Data Science?

This metric can be used other linear forecasting problem like,

- Forecasting the closing price of a stock each day.
- Forecasting product sales in units sold each day for a store.

- Forecasting unemployment for a state each quarter.
- Forecasting the average price of gasoline each day
- Forecasting next day weather
- Electricity consumption in a household
- Heart rate monitoring

1.3.7. Code: Implement this metric from scratch in Python using numPy/Pandas/Scipy only



```
import numpy as np
def rmsle_error(actual, predicted):
    total_error_square = 0.0
    for i in range(len(actual)):
        log_error = np.log(predicted[i]+1) - np.log(actual[i]+1) # 1 is added just to define log function if
        #                                                        predicted or actual value is zero, as meter reading
        #                                                        cannot be negative this crude implemenation of
        #                                                        rmsle will be sufficent for our specific project
        total_error_square += (log_error ** 2)
    mean_error = float(total_error_square / len(actual))
    return np.sqrt(mean_error)
```

1.4. Real world challenges and constraints

1.4.1. What real world constraints do you have while solving this problem?

At this stage it is very difficult to predict the real-world constraints. However, in general One of the main issues in Machine Learning problem solving is the absence of good data. As data quality is fundamental for the algorithms to work as proposed. Incomplete data, unclean data, and noisy data are the quintessential foes of ideal ML. Different reasons for low data quality are-

- Data can be noisy which will result in inaccurate predictions. This often leads to less accuracy and low-quality results. It is noted as one of the most common errors faced in terms of data.
- Incorrect or incomplete information can also lead to faulty programming through Machine Learning. Having less information will lead the program to analyze based on the minimal data present. Hence, decreasing the accuracy of the result

1.4.2. What are the requirements that your solution must meet?

The error both in training as well as test data set should be well within acceptable range.

1.5. How are similar problems solved in literature (technical articles & blogs, forums, research papers)

1.5.1. How is the problem mapped to an existing ML methodologies: Classification/ Regression/Time-series-forecasting/DeepLearning/RecSys etc?

Problem is basically a regression problem and falls under time series forecasting as future value of energy consumption needs to be estimated be estimated based on past Historic data. Deep Learning techniques can also be applied for this forecasting problem.

1.5.2. List all solution approaches that you think are relevant to your problem.

Some of the classical time series forecasting methods that can be used are

- Autoregression (AR)
- Moving Average (MA)
- Autoregressive Moving Average (ARMA)
- Autoregressive Integrated Moving Average (ARIMA)
- Seasonal Autoregressive Integrated Moving-Average (SARIMA)

Machine learning models that can be utilized for time series forecasting are

- Support Vector Regression (SVR)
- Random Forest Regression
- Xgboost Regression

Advance deep learning models that can be used

- ResNet
- Long Short Term Memory (LSTM)
- Bidirectional Encoder Representations from Transformers (BERT)

2. EDA and Feature Extraction

2.1. Creating single file for analysis

We have got three CSV files for training purposes. Each file contains some part of the information. First, we have combined all three files using pandas after converting them into dataframe.

For the site number “0” electric meter readings are in kBTU while for the rest of the sites data is in kWh. Thus, the meter reading of site “0” is multiplied by 0.2931 to convert it into kWh.

The plot for our target variable(“meter_reading”) w.r.t timestamp is shown below in Figure 1.

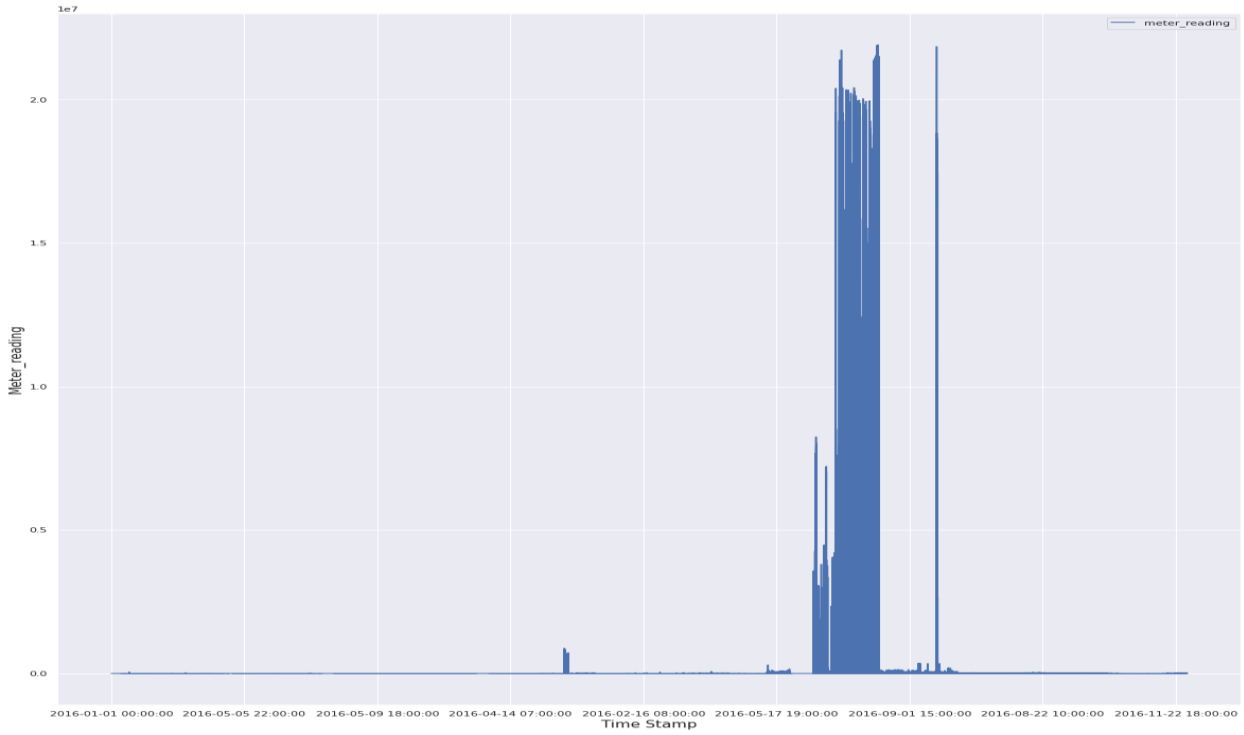


Figure 1 Meter Reading w.r.t Time

2.2. Finding Outlier

In the figure, we can see that the meter reading for some of the timestamps is erroneously high. We need to dig deep to find whether this high value is associated with some specific time duration or site_id or building_id.

```
df_train.meter_reading.describe()
```

```
count    2.021610e+07
mean     2.096420e+03
std      1.532356e+05
min      0.000000e+00
25%      1.794500e+01
50%      7.485000e+01
75%      2.503000e+02
max      2.190470e+07
```

we can see in the above details that there is a large difference between 75 percentile and 100 percentiles. So we dig deep for distribution of values between these two percentile, using quanltile function of pandas.

Table 1 Distribution of Meter Reading

Sl No	Percentile	Value	Multiplication factor w.r.t previous value
1	75	250.3	
2	85	489.334	1.95499
3	95	1455.08	2.973593
4	99	5178.24	3.558732
5	99.9	38671.9	7.468155
6	**99.99**	**6719767.27**	**173.7636**
7	99.999	19449967.80	2.894441
8	100	21904700.0	1.126208

In the above table also we can see that there is a very drastic increase (around 173 times) in value from 99.9 to 99.99 percentile, which further indicates presence of outlier or erronueous meter reading.

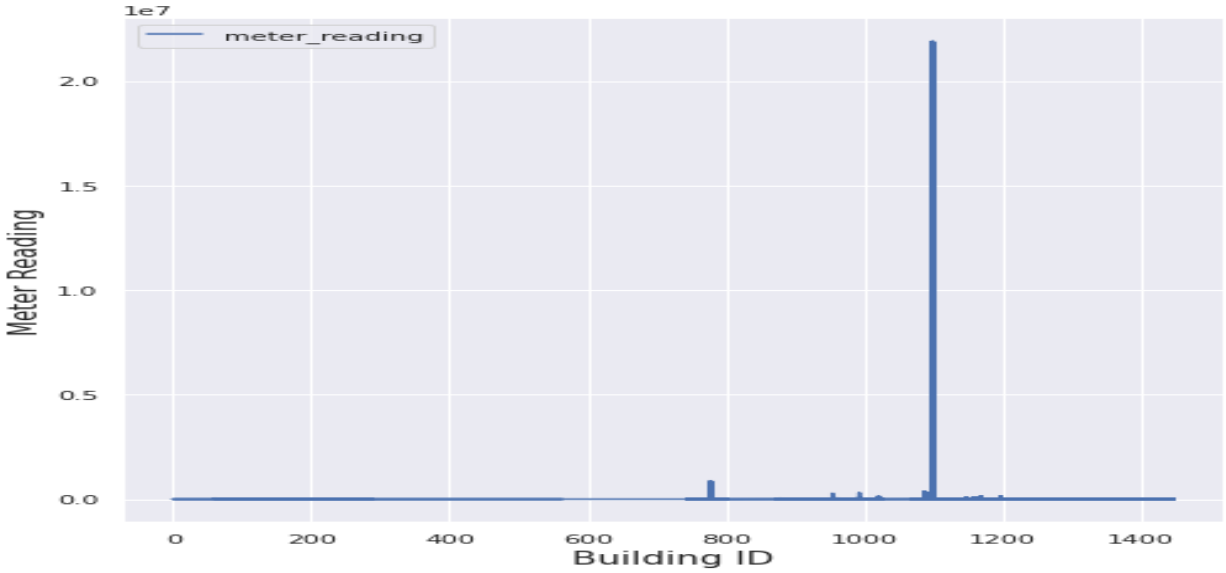


Figure 2 Building ID vs Meter Reading

It clearly shows that meter reading for some building betwwen 1000 and 1200 is extermaly high, let us find out the building id for higest meter reading

```
[11] df_train[df_train["meter_reading"]== df_train.meter_reading.max()]
```

	building_id	meter	timestamp	meter_reading	site_id	primary_use	square_feet
14405851	1099.0	2.0	2016-06-13 09:00:00	21904700.0	13	Education	332884.0

2.3. Removing outlier

We found that data for building id 1099 is very high. Thus we drop the rows containing values related to building id 1099 using following code.

```
df_train.drop(df_train[df_train["building_id"]== 1099].index, inplace= True)
```

And find out the next higest value.

```
df_train[df_train["meter_reading"]== df_train.meter_reading.max()]
```

	building_id	meter	timestamp	meter_reading	site_id	primary_use	square_feet
8513833	778.0	1.0	2016-09-09 17:00:00	880374.0	6	Entertainment/public assembly	108339.0

The same can also be seen as next peak near 800 level in the figure 2. Now we plot histogram plot of updated meter reading.

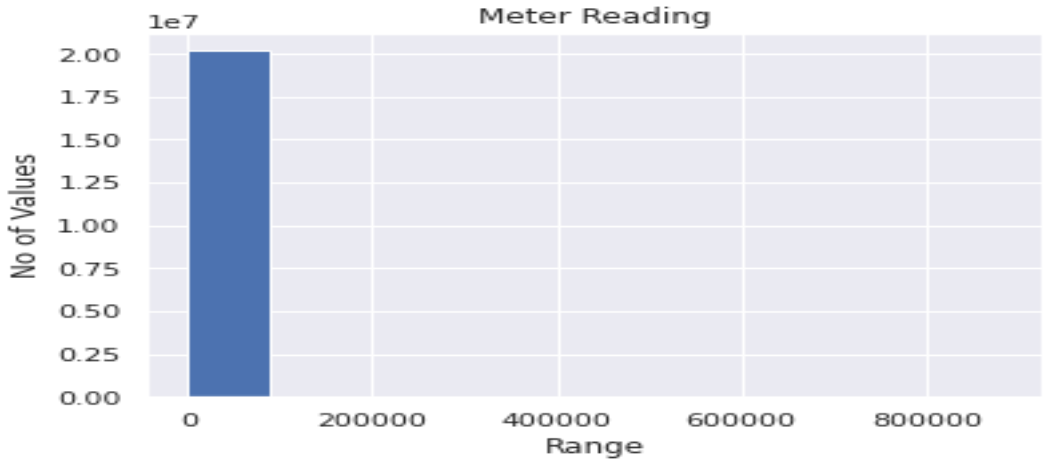


Figure 3 Histogram Meter Reading

2.4. Converting target variable for better visualization

Most of the values i.e 20195365 out of 20198534 which is 99.98 % lies between 0 to 88037. This is highly skewed distribution and look like lognormal of pareto distribution.
A new variable name log_meter_reading is created by taking log10 of meter reading value.

```
df_train["log_meter_readging"]= np.log10(df_train["meter_reading"]+1)
```

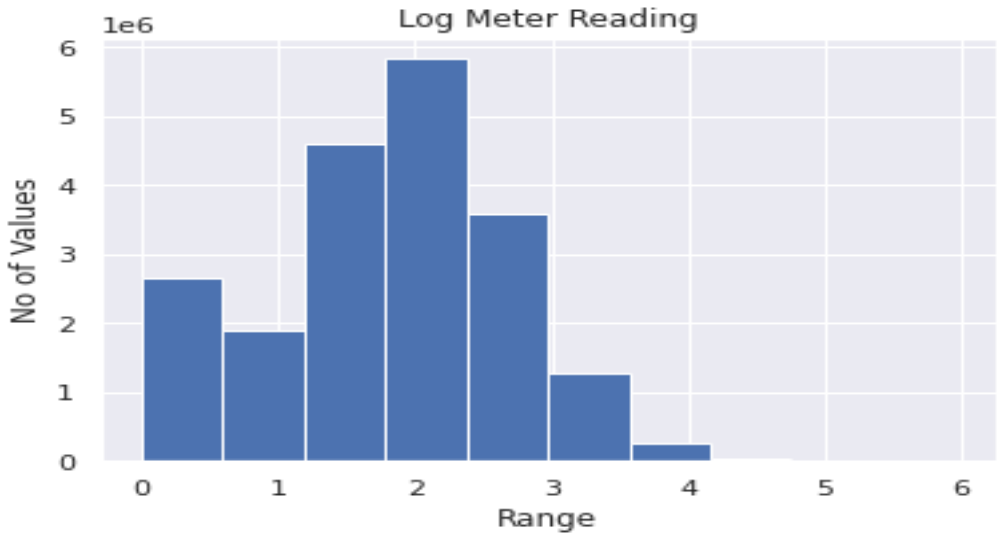


Figure 4 Histogram Log Meter Reading

2.5. Checking for missing values

Now we are checking for missing values.

```
df_train.isnull().sum() # checking for missing values
```

```
building_id              135
meter                    135
timestamp                  0
meter_reading            135
site_id                    0
primary_use              135
square_feet              135
year_built             12110214
floor_count            16691736
air_temperature           96654
cloud_coverage          8816843
dew_temperature          100136
precip_depth_1_hr       3749095
sea_level_pressure      1231427
wind_direction          1448617
wind_speed               143672
log_meter_readging          135
dtype: int64
```

We can see that many values are missing in year_built, floor_count, air_temperature, cloud_coverage, dew_temperature, precip_depth_1_hr, sea_level_pressure, wind_direction and wind_speed.

As we can see that many data are missing in a different category and we need to impute data or drop data for some of the categories which are not significant.

2.6. Creating new features for better understanding

A few more parameters like hour, day of the week, month are added for better visual analysis of the target variable.

```
df_train["timestamp"] = pd.to_datetime(df_train["timestamp"]) # Converting object type to datetime format for further analysis

df_train["hour"] = df_train["timestamp"].dt.hour # extrating hour from date time and creating new Varibale

df_train["DayofWeek"] = df_train["timestamp"].dt.dayofweek # extrating day of week from date time and creating new Varibale, Monday 0

df_train["Month"] = df_train["timestamp"].dt.month # extrating month from date time and creating new Varibale
```

Now mean of the log meter reading is plotted for the different hours of the day

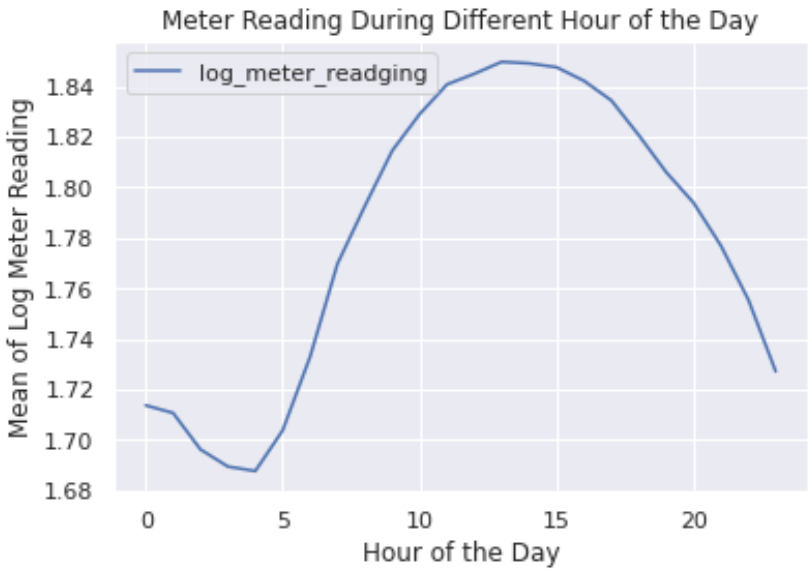


Figure 5 Meter Reading During Different Hour of the Day

We can see that the meter reading is minimum at around 4 hours while it starts ramping sharply from 5 Hrs and ramps quickly up to 9 Hrs, after that it ramps up relatively slowly up to 13 hours, post that it starts ramping down. It ramps down at a significantly faster rate from 17 Hours onwards. We can also see that mean meter reading normally remains higher from 9 Hrs upto 17 Hrs i.e during work hour.

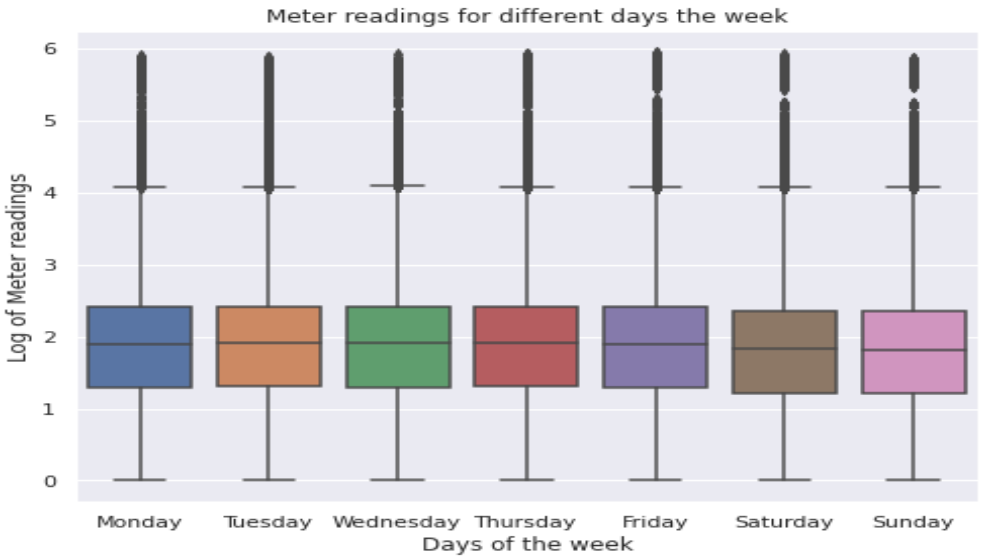


Figure 6 Meter Reading for different days of the Week

Variation of meter reading for different days are plotted, here we can see that for Weekend (i.e Saturday and Sunday) the 25 percentile, median and 75 percentile value is slightly lower than weekdays however it is not significant enough to make any conclusion.

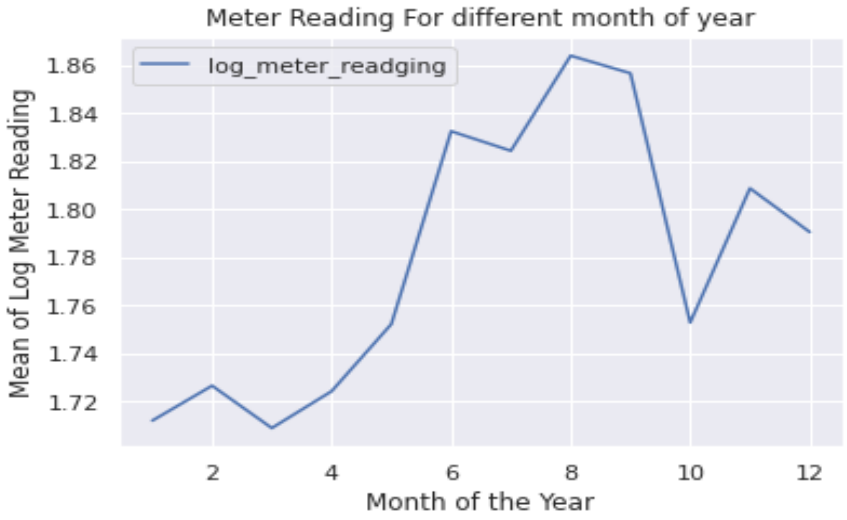


Figure 7 Meter Reading for different month of year

The average monthly consumption is plotted and it is observed that it is minimal from January to April and least in March. Post-April it increases rapidly and the maximum value is observed in August, after which it again starts dropping. in the initial months and then rises after April in the onset of spring. Although the plot ends in December however if we connect December and January a very shape drop between these two months is also noticed.

2.7. Analysis of other features

Now we are analyzing the different types of meters and found that most of the meters are measuring electricity followed by Chilled water and Steam and least number of meters are associated with hot water.

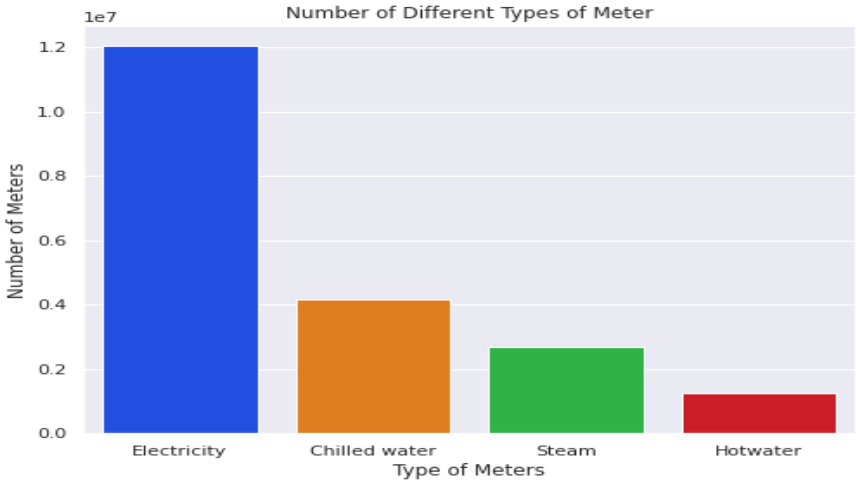
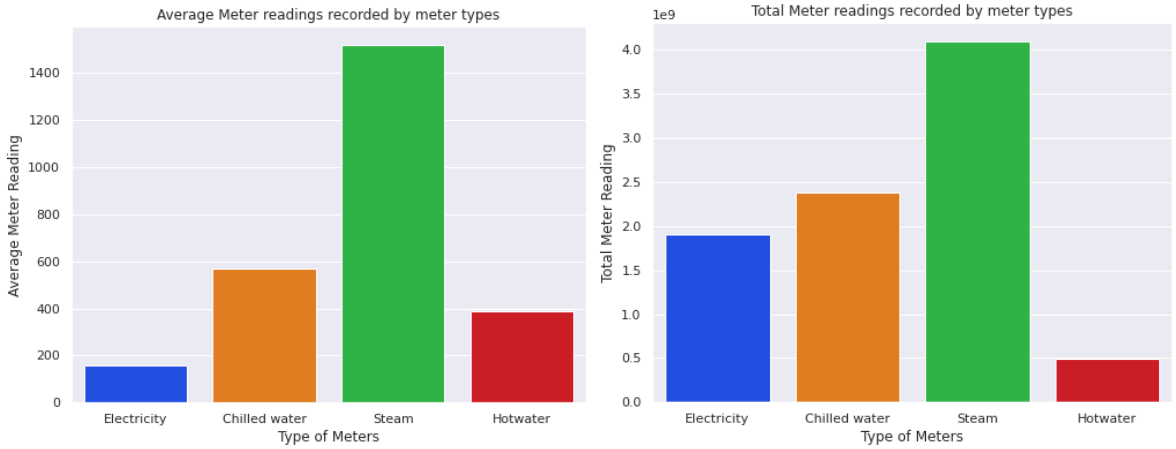


Figure 8 Number of Different types of Meter

Now plotting the average energy consumption and total energy consumption as recorded by a different type of meter.



Here we can see that both total energy consumption, as well as average energy consumption, is highest for steam meter while average energy consumption is minimum for Electricity meter and Total energy consumption is minimum for Hot water meter.

Now plotting the location of where these meters are installed. We found that most numbers of meters are at Educational building followed by offices, Entertainment/public assembly. While the Religious worship place and Utilities are represented by the least number of meters.

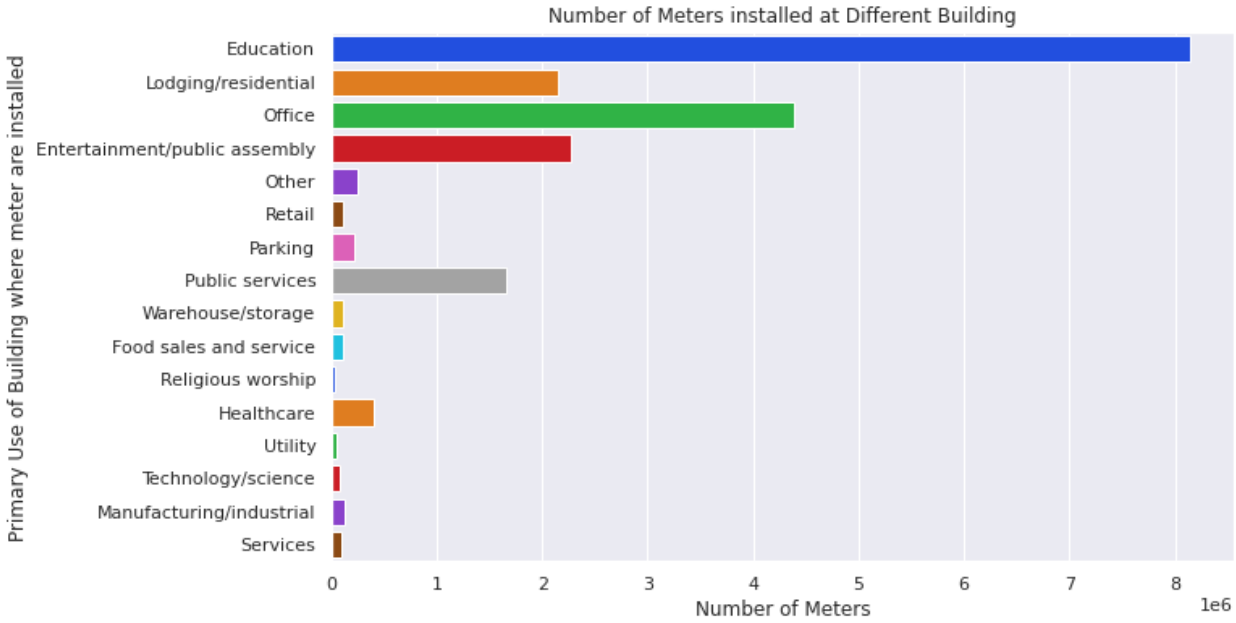
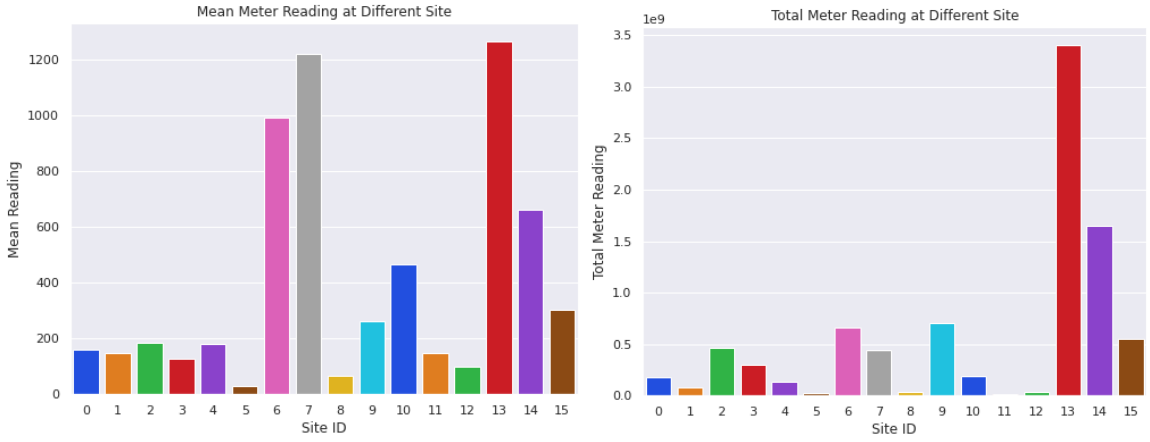


Figure 9 Number of Meters installed at Different Building

Now plotting the Mean meter reading and Total meter reaading at Different Site Id we found that while mean meter reading at Site 7 and Site 13 are highest and very close to each other. The Total meter reading is very high for location number 13.



we have plotted the log of average hourly usage for each of the primary usage category with different days of week. Following can be observed

- Almost for all building type barring Food sales and service and Lodging/residential the consumption during the weekday is significantly higher than that of weekend. This indicates that consumption patter during holiday and working day is different and this factor needs to be considered in modeling. However, in absence of information about exact location of the building like country, state etc, the holiday information cannot be utilized to its fullest.

- During the working day usage for educational institutes have the peak usage from morning 9 am until 4 pm and then the usage drops. This is expected as most educational institutes are operational during day time only.
- The consumption for enetertainmet/public assembly starts rising after 7 AM on weekday and after 10 AM on weekend continues to remain high upto 8 PM.
- Food Sales and Service remains almost unaffected by holiday the consumption starts ramping from 10 AM and peaking at round 3 PM its consumption remains higher side on during early night hour also.
- Health care witness a very Sharpe rise at around 9 AM and remain on higher side post that.
- Residential building consumption starts increasing from 6 AM and remains stable from 10 AM onwards
- Manufacturing/Industrial sector demand witness a very sharp peak from 5 AM and peaks at 9 AM after that it reduces slightly and remains stable upto 8 PM
- The office buildings, consume most of the energy in the daytime mainly from 10 AM to 5 PM. While consumption is significantly less during the weekend.
- For other primary use type building the consumption peaks at around 6 AM and remain high upto 8 PM.
- For parking Consumption is significantly lower during weekends. Further unlike other building type consumption is minimum during the day time.
- For public services consumption starts ramping from 5 AM and becomes maximum around 3 PM after that it starts reducing.
- For religious worship demand is higher during the day time with peaking in afternoon.
- For Relatil demand starts ramping sharply from 5 AM to 10 AM after that it ramin stable upto 6 PM
- For service demand peaks at around 8 AM and after that it reduces slowly
- For Technology/Science demand remain high from 9 AM to 6 PM.
- For utility demand peaks at around 6 AM and after that it reduces slowly
- For warehouse/storage demand remains on lower side during the day and it peak normally during early night hours

There are other variable also like square_feet, year_built, floor_count, air_temperature, cloud_coverage, dew_temperature, precip_depth_1_hr, sea_level_pressure, wind_direction and wind_speed. Variablity of these are plotted in the notebook

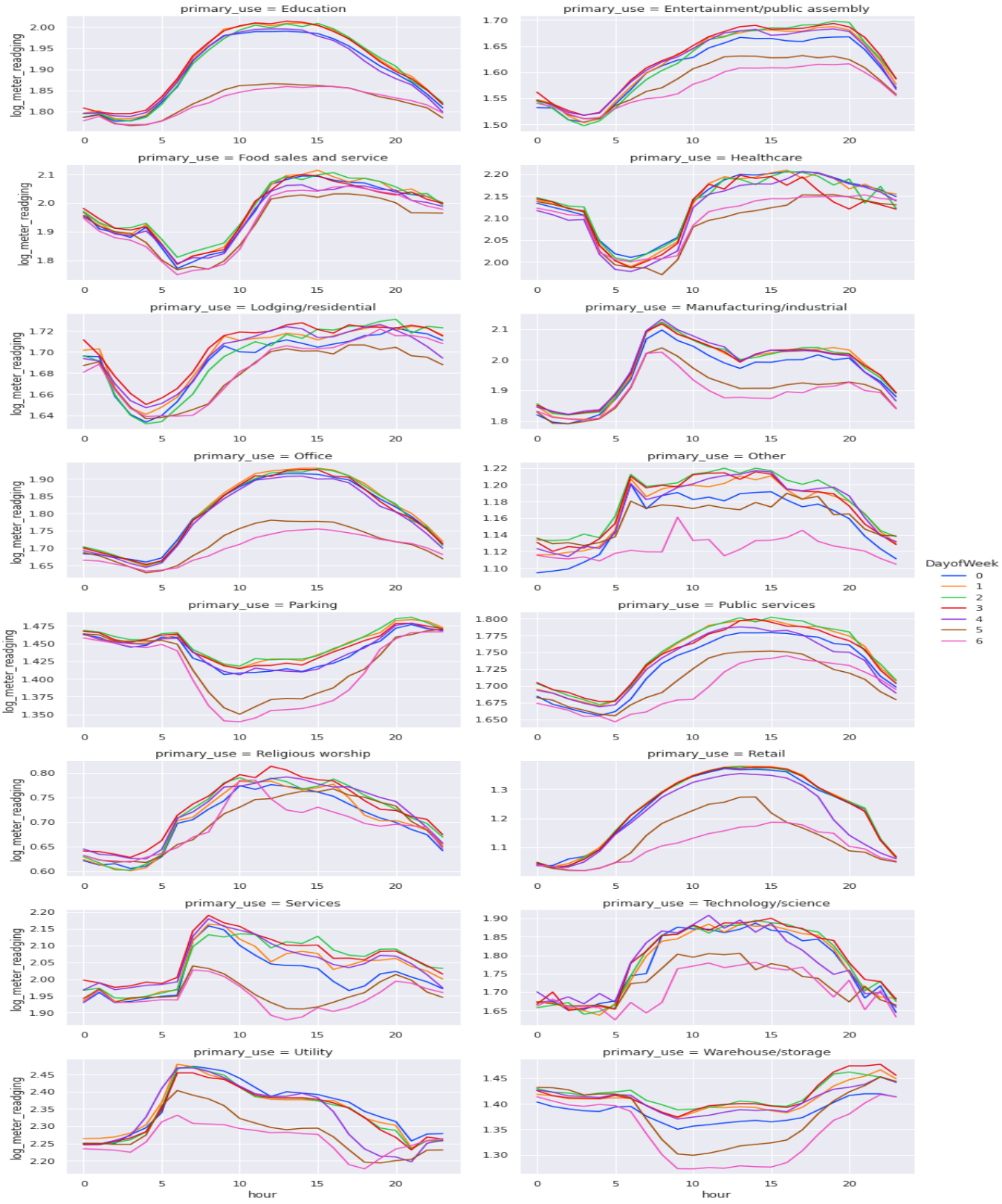


Figure 10 Hourly Energy Consumptions during different Days for different Primary use

2.8. Corelation Matrix for finding featue importance and extraction

We have plotted the correlation matrix of the features available. The square feet and floor count features have a strong positive correlation with the log of meter reading which is modified target variable. Further building ID and Site ID also shows a significant positive correlation with log of meter reading indicating that building at some specific location consumes relatively higher energy in comparison to other. A negative correlation with cloud coverage and air temperature is also observed.

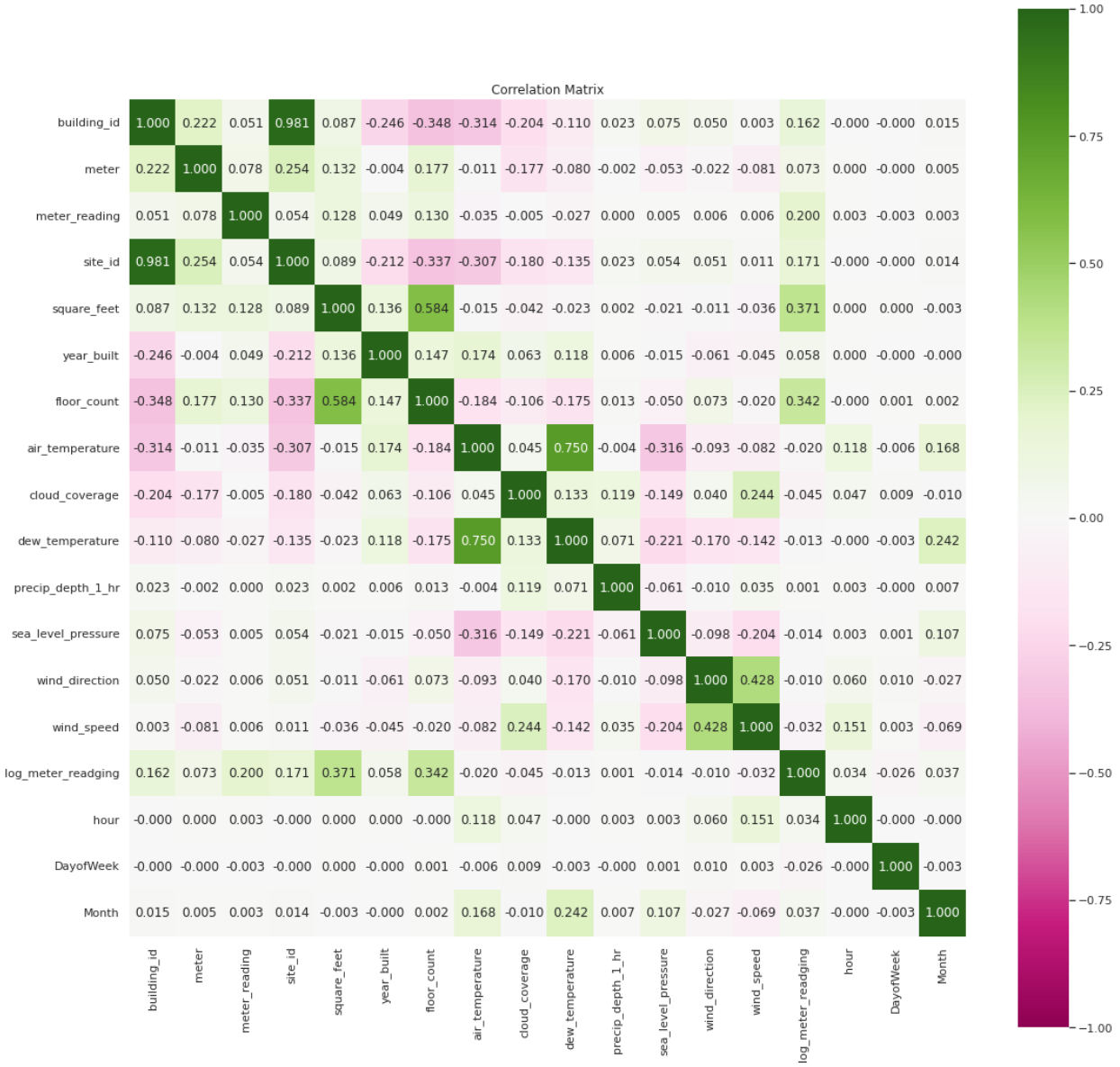


Figure 11 Correlation Matrix

3. Modeling and Error Analysis

3.1. Dropping zero meter values

As for real world only positive meter values are of use so we drop all the zero meter values there were very few zero meter reading only.

3.2. Finding Missing values

```
df_train.isnull().sum()*100/len(df_train) # Percentage missing value in train data
```

```
building_id            0.000737
meter                  0.000737
timestamp              0.000000
meter_reading          0.000737
site_id                0.000000
primary_use            0.000737
square_feet            0.000737
year_built            60.118224
floor_count           82.246186
air_temperature        0.488106
cloud_coverage        43.478054
dew_temperature        0.504996
precip_depth_1_hr     19.219744
sea_level_pressure     6.480747
wind_direction         7.234506
wind_speed             0.723466
log_meter_readging     0.000737
hour                   0.000000
DayofWeek              0.000000
Month                  0.000000
dtype: float64
```

We can see that top three missing values are floor_count(82.6%), year_bulit (56%) and cloud_coverage (43.6%). However, floor_count have second largest correlation cofficent with target variable which is log of energy meter reading. Highest being square_feet area, in the correlation matrix we can check that the correation coffiecent between square_feet and floor_count is 0.54. Even when many around 83 % of floor count data is missing so we can drop this floor_count variable. Because firstly many data are missing and second it is an redundent feature. Other two features which has high missing values are year_built and cloud_coverage. we are dropping the year_built as it dont have high correlation coffiecent with target variable also we dont have any speicific metric to impute this. We are not dropping cloud cover because with the help of site_id and hour of the day we can impute relatively easily.

Now the missing values are much lesser and can only be seen in we will impute this with median of feature

- cloud_coverage
- precip_depth_1_hr
- sea_level_pressure

we will impute this with median of feature

3.3. Adding new features

3.3.1. Holiday Information

No specific information about the location are provided so it becomes ***difficult to use Holiday package*** so we are restricting to weekend and weekday feature created by utilizing day of the week

3.3.2. Working hour feature

As energy consumption during the working hour is expected to be different from during the non-working hour so a new feature considering 6 AM to 8 PM as working hour is created. This time of 6 AM to 8 PM is considered based on visual analysis of energy consumption by different primary use

3.3.3. Relative Humity Feature

https://www.nature.com/articles/s41467-020-15393-8

The above article show that in many of the high energy consuming states, such as California and Texas, projections based on air temperature alone underestimates cooling demand by as much as 10–15% under both present and future climate scenarios. Our results establish that air temperature is a necessary but not sufficient variable for adequately characterizing the climate sensitivity of cooling load, and that **near-surface humidity** plays an equally important role.

Thus a new feature called relative humity is created

RH: =100*(EXP((17.625*TD)/(243.04+TD))/EXP((17.625*T)/(243.04+T)))

Temperature T (°) Dewpoint Td (°) Relative Humidity RH (%)

Temperature T (°) and Dewpoint Td (°) inputs/outputs to the equations are in

Source :- https://bmcnoldy.rsmas.miami.edu/Humidity.html

3.4. Base Line Model

For baseline model, we will use the site_id and primary use to find the mean readings for each combination

```
#log_rsmle score for baseline model
log_rmse_base_tr=math.sqrt(mean_squared_log_error(X_train['log_meter_readging'], X_train['y_pred_base']))
log_rmse_base_cv=math.sqrt(mean_squared_log_error(X_CV['log_meter_readging'], X_CV['y_pred_base']))
print("Log_RMSE for baseline model for train data:", log_rmse_base_tr)
print("Log_RMSE for baseline model for cv data:", log_rmse_base_cv)
```

```
Log_RMSE for baseline model for train data: 0.24545981573999626
Log_RMSE for baseline model for cv data: 0.25276768820709045
```

3.5. Test Train Split

We are going to do test train split based on time since for cross validation as for in future we are going to have data for upcoming period only, **thus K-Fold cross validation is not used ***

3.6. Reducing Memory use

As it was observed that Google Colab is crashing many a time with present data set. So Memory optimization technique is used code is taken from Kaggal repository and can be downloaded from https://www.kaggle.com/kernels/scriptcontent/3684066/download

It is inspired from

https://www.kaggle.com/arjanso/reducing-dataframe-memory-size-by-65

It uses the following approach

This notebook uses the following approach:

1. Iterate over every column
2. Determine if the column is numeric
3. Determine if the column can be represented by an integer
4. Find the min and the max value
5. Determine and apply the smallest datatype that can fit the range of values

```
X_train=reduce_mem_usage(X_train)
X_CV=reduce_mem_usage(X_CV)
```

```
Memory usage of dataframe is 1915.33 MB
Memory usage after optimization is: 950.68 MB
Decreased by 50.4%
Memory usage of dataframe is 478.83 MB
Memory usage after optimization is: 237.67 MB
Decreased by 50.4%
```

3.7. Label Binarization

As we have one feature which is primary use having multiple unique string values like

['Retail', 'Office', 'Education', 'Entertainment/public assembly', 'Warehouse/storage', 'Services', 'Lodging/residential', 'Parking', 'Public services', 'Healthcare', 'Manufacturing/industrial', 'Other', 'Technology/science', 'Food sales and service', 'Utility', 'Religious worship']

And this will become difficult to train with so we go ahead with creating dummy features with Binary 0 and 1.

```
X_train = pd.concat([X_train, pd.get_dummies(X_train.primary_use)], axis=1).drop(['primary_use'], axis=1)
X_CV = pd.concat([X_CV, pd.get_dummies(X_CV.primary_use)], axis=1).drop(['primary_use'], axis=1)
```

3.8. Dropping redundant features

There are few features which are not helpful during training as these are having less impact on output as seen during EDA analysis so these features are dropped.

'site_id', 'dew_temperature','timestamp'

Time stamp also need to be dropped as with this time it was not possible to train this feature.

3.9. Testing different Model

Now we are trying some Initial Machine learning model for training our model

3.9.1. Decision Tree Regressors

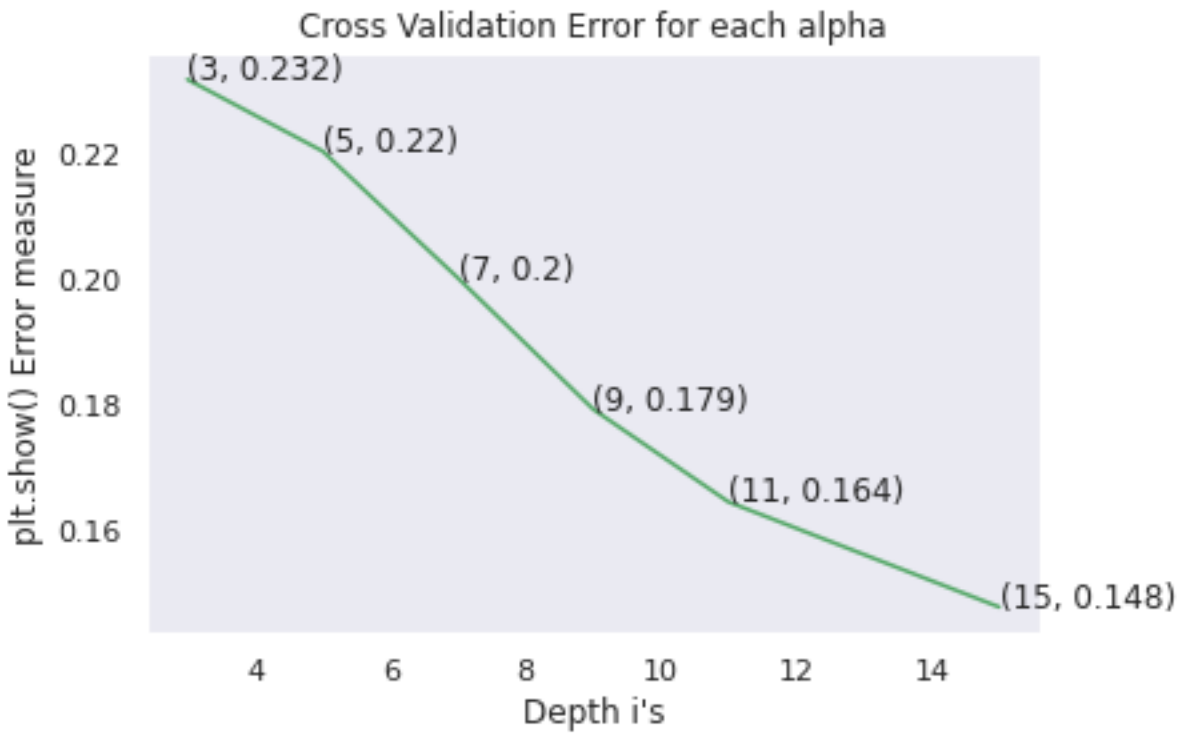
First we have tried Decision tree Model available in Sklearn. The **Decision Trees (DTs)** are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

We have tried decision trees for different depths [3,5,7,9,11,15]

```
cv_error_array=[]
depth=[3,5,7,9,11,15]
for i in depth:
  d_reg=DecisionTreeRegressor(max_depth=i,random_state=0)
  d_reg.fit(X_train,y_readings_tr)
  pred_d_reg=d_reg.predict(X_CV)
  cv_error_array.append(math.sqrt(mean_squared_log_error(y_readings_cv, pred_d_reg)))
  print("Log_rmse for max_depth:", i,'is',math.sqrt(mean_squared_log_error(y_readings_cv, pred_d_reg)))
```

```
Log_rmse for max_depth: 3 is 0.23200324504517203
Log_rmse for max_depth: 5 is 0.2205051779086939
Log_rmse for max_depth: 7 is 0.20019381969153877
Log_rmse for max_depth: 9 is 0.17936327374777192
Log_rmse for max_depth: 11 is 0.16442204924981432
Log_rmse for max_depth: 15 is 0.147135722114984
```

We also tried to find out the best alpha for our model which comes out to be 15



Train and CV loss at best alpha comes out to be

```
Train log_rmse for max_depth: 15 is 0.10673305977379793
CV log_rmse for max_depth: 15 is 0.14693363006866716
```

We save the parameters of final tunned model so that it can be used on test data

```
#Save the models to a pickle file for making predictions on test data in future
filename_reg='decision_tree_reg.sav'
joblib.dump(d_reg,filename_reg)
```

```
['decision_tree_reg.sav']
```

3.9.2. LGBM Regressor

LightGBM regressor with boosting_type='gbdt' is used. Which is Gradient Boosting for regression in Sklearn.

GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

Gradient Tree Boosting or Gradient Boosted Decision Trees (GBDT) is a generalization of boosting to arbitrary differentiable loss functions. GBDT is an accurate and effective off-the-shelf procedure

that can be used for both regression and classification problems in a variety of areas including Web search ranking and ecology

We have first used Mean Squared log error, it gave a good result and loss was reducing subsequently as number of estimators are increasing

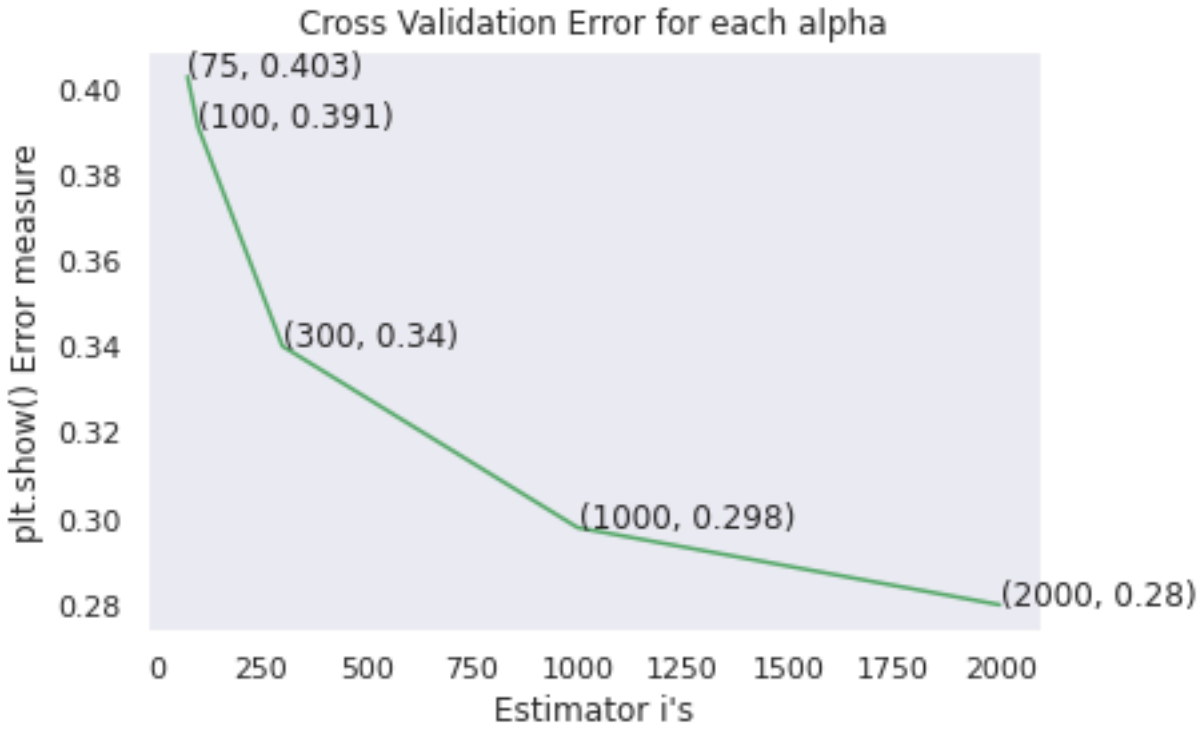
```
rmse for estimators: 75 is 0.16777824409221428
rmse for estimators: 100 is 0.16343208277587348
rmse for estimators: 300 is 0.14288685618420235
```

however, at 1000 number of estimators it crashed giving following error

```
ValueError: Mean Squared Logarithmic Error cannot be used when targets contain negative values.
```

So, we have switched to normal root mean square error and got these values

```
rmse for estimators: 75 is 0.4029137191068984
rmse for estimators: 100 is 0.3911731908146633
rmse for estimators: 300 is 0.34016068079636214
rmse for estimators: 1000 is 0.2979067178652192
rmse for estimators: 2000 is 0.2799544545587758
```



Train and CV loss at best alpha comes out to be

```
Train rmse for estimators: 2000 is 0.19996517559658614
CV rmse for estimators: 2000 is 0.2799548569612862
```

We save the parameters of final tunned model so that it can be used on test data

```
filename_reg='lgb_reg.sav'
joblib.dump(lgb_reg,filename_reg)
```

```
['lgb_reg.sav']
```

3.9.3. LGBM Random Forest Regressor

LightGBM regressor with boosting_type='rf' is used. Which uses Random Forest regression available in Sklearn.

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.

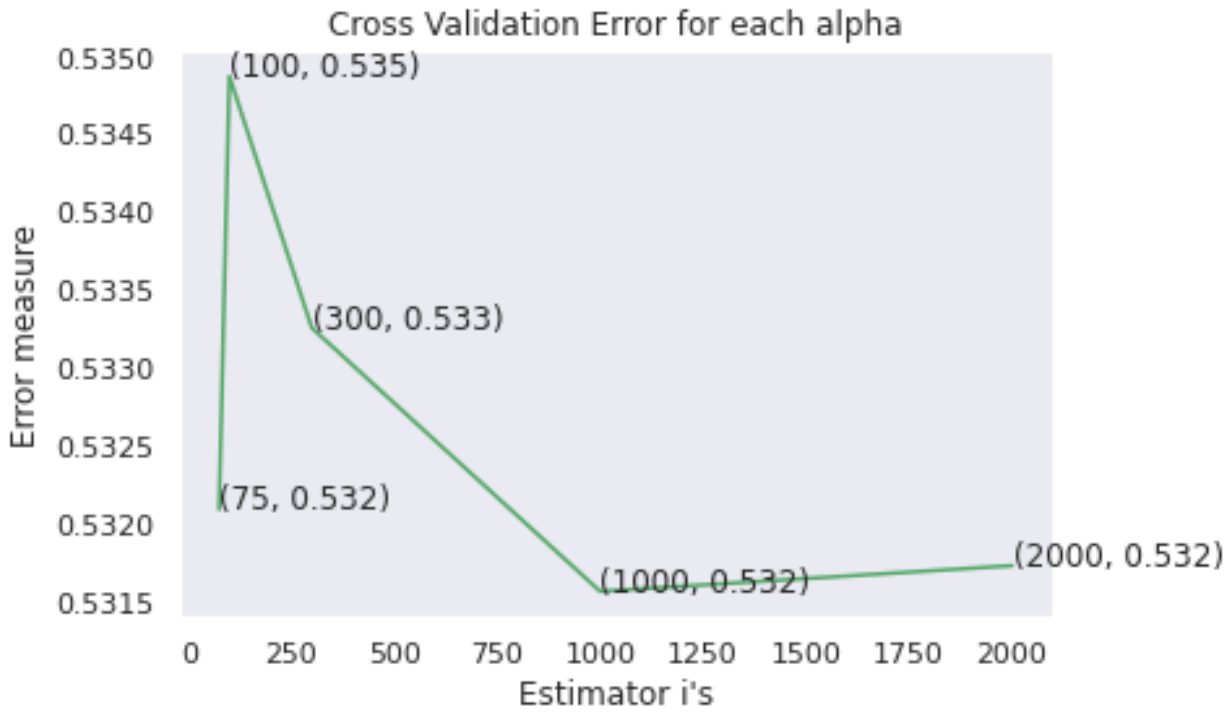
Furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size max_features.

The purpose of these two sources of randomness is to decrease the variance of the forest estimator. Indeed, individual decision trees typically exhibit high variance and tend to overfit. The injected randomness in forests yield decision trees with somewhat decoupled prediction errors. By taking an average of those predictions, some errors can cancel out. Random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. In practice the variance reduction is often significant hence yielding an overall better model.

RMSE model for different estimator values are tried and we got the below outcome

```
rmse for estimators: 75 is 0.532091039786465
rmse for estimators: 100 is 0.5348742792375081
rmse for estimators: 300 is 0.5332586787221064
rmse for estimators: 1000 is 0.5315654727349403
rmse for estimators: 2000 is 0.5317303882392718
```

Same is also plotted below



In the above picture we can see that loss are minimum when number of estimators are 1000.

3.10. Conclusion for simple regression-based model.

From the output it seems that among these three models, LGBM Random Forest Regressor is least useful while LGBM Regressor perform the best. We need to rerun Decision Tree Regressors model with only RMSE error to take the final call.

Further deep learning based models will also be analyzed under advance modelling section.

4. Advanced Modeling and Feature Engineering

We have tired few more models before selecting our models for final

4.1. Modeling continued

4.1.1. Catboost GBDT

Catboost GBDT model is not readily available in Google Colab and first same was installed

```
[ ] !pip install catboost
```

```
Collecting catboost
  Downloading catboost-1.0.4-cp37-none-manylinux1_x86_64.whl (76.1 MB)
     |████████████████████████████████| 76.1 MB 1.1 MB/s
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from catboost) (3.2.2)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.7/dist-packages (from catboost) (1.3.5)
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (from catboost) (0.10.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.7/dist-packages (from catboost) (1.19.5)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from catboost) (1.15.0)
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (from catboost) (5.5.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from catboost) (1.4.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.0->catboost) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.0->catboost) (2018.9)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->catboost)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->catboost) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packages (from plotly->catboost) (8.0.1)
Installing collected packages: catboost
Successfully installed catboost-1.0.4
```

Model with two different number of estimators is trained and their performance is as follows

```
Rmse for estimators: 1000 is 0.2994
Rmse for estimators: 1500 is 0.2997
```

As performance with n_estimator= 1000 is slightly better so same is used for final model.

4.1.2. Deep Learning based model

We have tried one deep learning-based model which is densely connected multilayer perceptron. Five sequential layers with different number of neurons are tired. Code for same is given below

```
[ ] inp_dim=X_train.shape[1]
    beta=[128,256,512,1024,2048]
    earlystop= EarlyStopping(monitor='val_loss', mode='min', patience=3)
    for i in tqdm(beta):

      model=Sequential()
      model.add(Dense(i, activation='relu', input_shape=(inp_dim,)))
      model.add(Dense(i*0.75, activation='relu'))
      model.add(Dense(i*0.5, activation='relu'))
      model.add(Dense(32, activation='relu'))
      model.add(Dense(16, activation='linear'))
      opt=Adam(0.0001)
      model.compile(optimizer=opt, loss= root_mean_squared_error)
      model.fit(X_train, y_readings_tr,epochs=15,batch_size=2048,validation_data=(X_CV,y_readings_cv),callbacks=earlystop)
```

Root mean squared error is not directly available in keras so we have created it separately

```
import keras.backend as K
```

```
#https://stackoverflow.com/questions/43855162/rmse-rmsle-loss-function-in-keras
# As root mean square error is not directl available in Keras
def root_mean_squared_error(y_true, y_pred):
        return K.sqrt(K.mean(K.square(y_pred - y_true)))
```

Finally based on minimum RMSE following structure is selected

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 2048)              65536

 dense_1 (Dense)             (None, 1536)              3147264

 dense_2 (Dense)             (None, 1024)              1573888

 dense_3 (Dense)             (None, 32)                32800

 dense_4 (Dense)             (None, 16)                528

=================================================================
Total params: 4,820,016
Trainable params: 4,820,016
Non-trainable params: 0
_________________________________________________________________
None
```

However, the error that we got is not appreciating and it is shown below

```
Rmse for Epoch: 6/15 is 0.7673
```

Post 6 epoch the stop decreasing, and we stop future iteration using early stop feature.

4.2. Comparing different models

Model	Best Hyperparameter	Best RMSE
Baseline	Mean	0.655
Decision Trees	max_depth = 15	0.3677
LightGBM GBDT	n_estimators = 2000	0.2767
Random Forest	n_estimators = 1000	0.5312
Catboost	n_estimators = 1000	0.2997
MLP	5 Dense Layer, Relu Activation	0.7673

The best performance we got is from LightGBM GBDT model while the worst is of MLP surprisingly it is even worst than Baseline model. We could not try RNN based Deep learning models like LSTM, GRUs due to computation resource crunch.

4.3. Feature importance in different model

We have taken top five features for our top performing 4 models and shown the same below

Model	Feature-1	Feature-2	Feature-3	Feature-4	Feature-5
Decision Trees	square_feet	building_id	meter	air_temperature	Month
LightGBM GBDT	building_id	square_feet	meter	Month	air_temperature
Random Forest	building_id	square_feet	meter	air_temperature	sea_level_pressure
Catboost	square_feet	building_id	meter	air_temperature	Education

Here we can see that building_id or square_feet is top two feature while meter is third most important feature for all the four model.

4.4. Predicting on Kaggel data

4.4.1. Processing data for prediction

In line with data cleaning and data imputation on train data same process is followed for processing test data also

```
df_data_test = pd.read_csv("test.csv")
df_test_weather = pd.read_csv("weather_test.csv")
df_building_metadata = pd.read_csv("building_metadata.csv")
#
#
df_test = pd.merge(df_data_test,df_building_metadata,how='outer', on ="building_id")
df_test = pd.merge(df_test,df_test_weather,how="outer",on= ["site_id","timestamp"]) # creating one final file with all training data together
```

```
# Covering and adding
df_test["timestamp"] = pd.to_datetime(df_test["timestamp"]) # Converting object type to datetime format for further analysis
df_test["hour"] = df_test["timestamp"].dt.hour # extrating hour from date time and creating new Varibale
df_test["DayofWeek"] = df_test["timestamp"].dt.dayofweek # extrating day of week from date time and creating new Varibale, Monday 0
df_test["Month"] = df_test["timestamp"].dt.month # extrating month from date time and creating new Varibale
```

```
df_test.drop(['year_built', 'floor_count'], axis=1,inplace=True) # Droping year built and floor_count
```

```
# imputing misssing data
#cloud coverage
df_cloud_coverage=df_test.groupby(['site_id','hour'])['cloud_coverage'].transform('median')
df_test['cloud_coverage'].fillna(df_cloud_coverage,inplace=True)

# precip_depth_1_hr
df_precip_depth_1_hr =df_test.groupby(['site_id','hour'])['precip_depth_1_hr'].transform('median')
df_test['precip_depth_1_hr'].fillna(df_precip_depth_1_hr, inplace=True)

# sea_level_pressure
df_sea_level_pressure =df_test.groupby(['site_id','hour'])['sea_level_pressure'].transform('median')
df_test['sea_level_pressure'].fillna(df_sea_level_pressure, inplace=True)

# wind_direction
df_wind_direction =df_test.groupby(['site_id','hour'])['wind_direction'].transform('median')
df_test['wind_direction'].fillna(df_wind_direction, inplace=True)

# wind_speed
df_wind_speed =df_test.groupby(['site_id','hour'])['wind_speed'].transform('median')
df_test['wind_speed'].fillna(df_wind_speed, inplace=True)

# dew_temperature
df_dew_temperature =df_test.groupby(['site_id','hour'])['dew_temperature'].transform('median')
df_test['dew_temperature'].fillna(df_dew_temperature, inplace=True)

# air_temperature
df_air_temperature=df_test.groupby(['site_id', 'hour'])['air_temperature'].transform('median')
df_test['air_temperature'].fillna(df_air_temperature, inplace=True)
```

```
df_test['cloud_coverage'].fillna(df_test['cloud_coverage'].median(), inplace=True)
df_test['sea_level_pressure'].fillna(df_test['sea_level_pressure'].median(), inplace=True)
df_test['precip_depth_1_hr'].fillna(df_test['precip_depth_1_hr'].median(), inplace=True)
```

```
# Creating new features
df_test["Weekend"] = df_test["DayofWeek"] > 4
df_test['WorkingHour']= df_test['timestamp'].apply(lambda x: 1 if x.hour >=6 and x.hour <=20 else 0)
df_test['relative_humidity']= 100*((np.exp((17.625*df_test['dew_temperature'])/
                                           (243.04+df_test['dew_temperature'])))/(np.exp((17.625*df_test['air_temperature'])/
                                                                                         (243.04+df_test['air_temperature']))))
```

```
# Binning
df_test = pd.concat([df_test, pd.get_dummies(df_test.primary_use)], axis=1).drop(['primary_use'], axis=1)
```

```
#dropping redundent features
df_test.drop(['site_id', 'dew_temperature','timestamp'],axis=1,inplace=True)
```

4.4.2. Final Score on Kaggel for different models

We have tried to make prediction on public using our top 4 model. We have achieved a public score of 1.392 with our best model which is LightGBM GBDT model. The score of different model is shown below There is certainly a scope of improvement which we will work in the future.

5 submissions for Saurav Sahay

Sort by Select...

All Successful Selected

Submission and Description	Private Score	Public Score	Use for Final Score
Random_Forest_predictions.csv 7 minutes ago by Saurav Sahay Random_Forest_Predictions	2.062	1.848	☐
Catboost_predictions.csv 17 minutes ago by Saurav Sahay Catboost predictions	1.713	1.420	☐
LightGBM_GBDT_predictions.csv 33 minutes ago by Saurav Sahay LightGBM GBDT	1.608	1.329	☐
Decision_tree_predictions (1).csv an hour ago by Saurav Sahay Log conversion corrected	1.800	1.451	☐

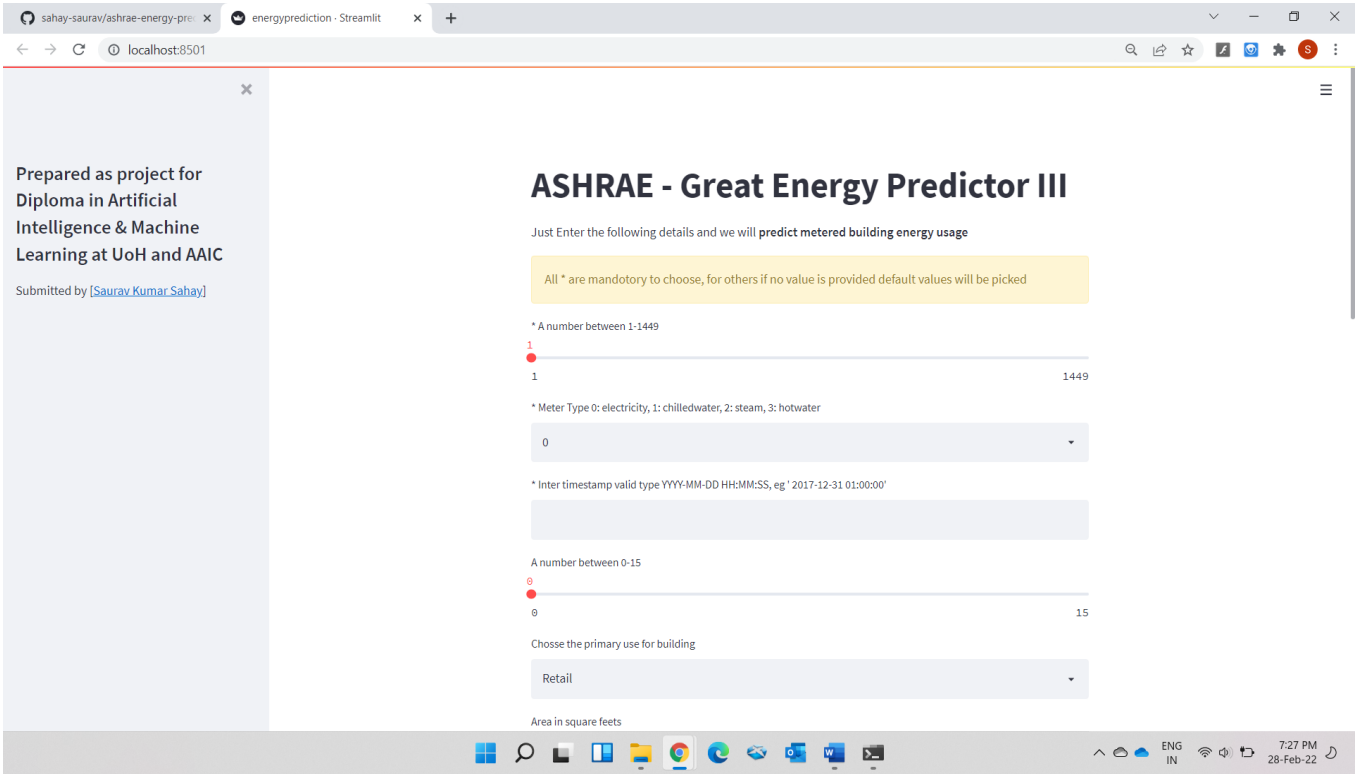
5. Deployment and code repository

5.1. Streamlit application

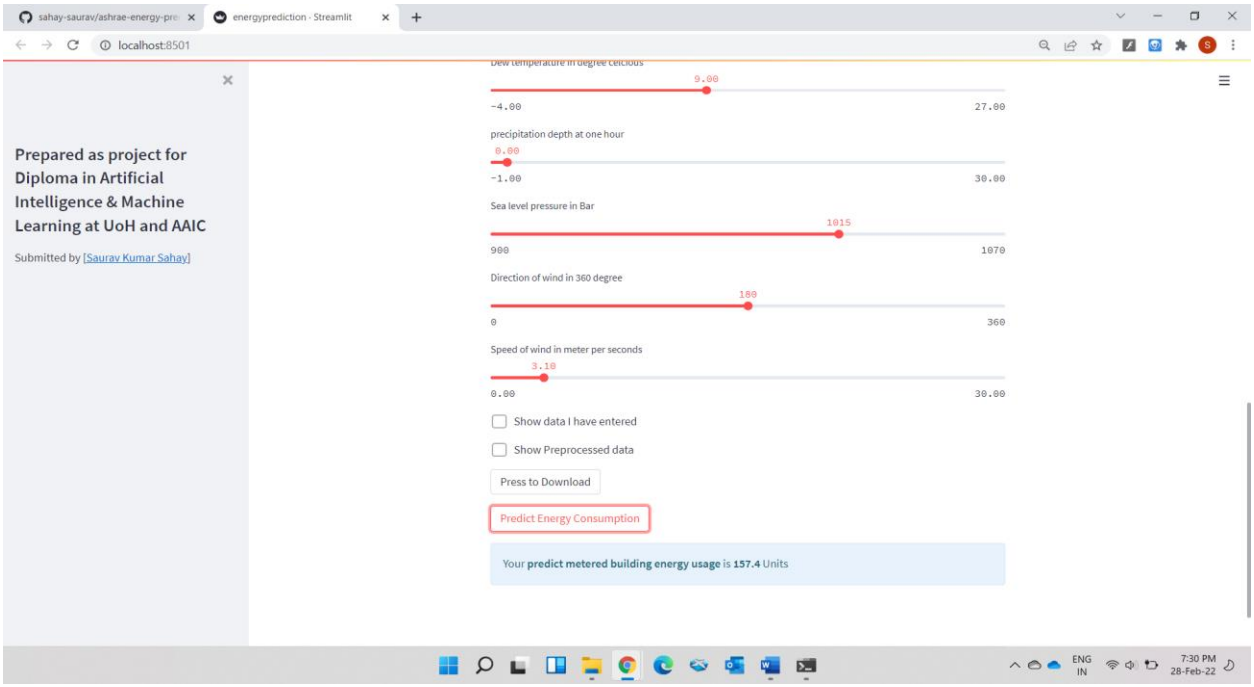
Streamlit is a tool that allows engineers to quickly build highly interactive web applications around their data, machine learning models, and pretty much anything.

An application with our final model is prepared using streamlit application some of the screenshot for the application are as follows

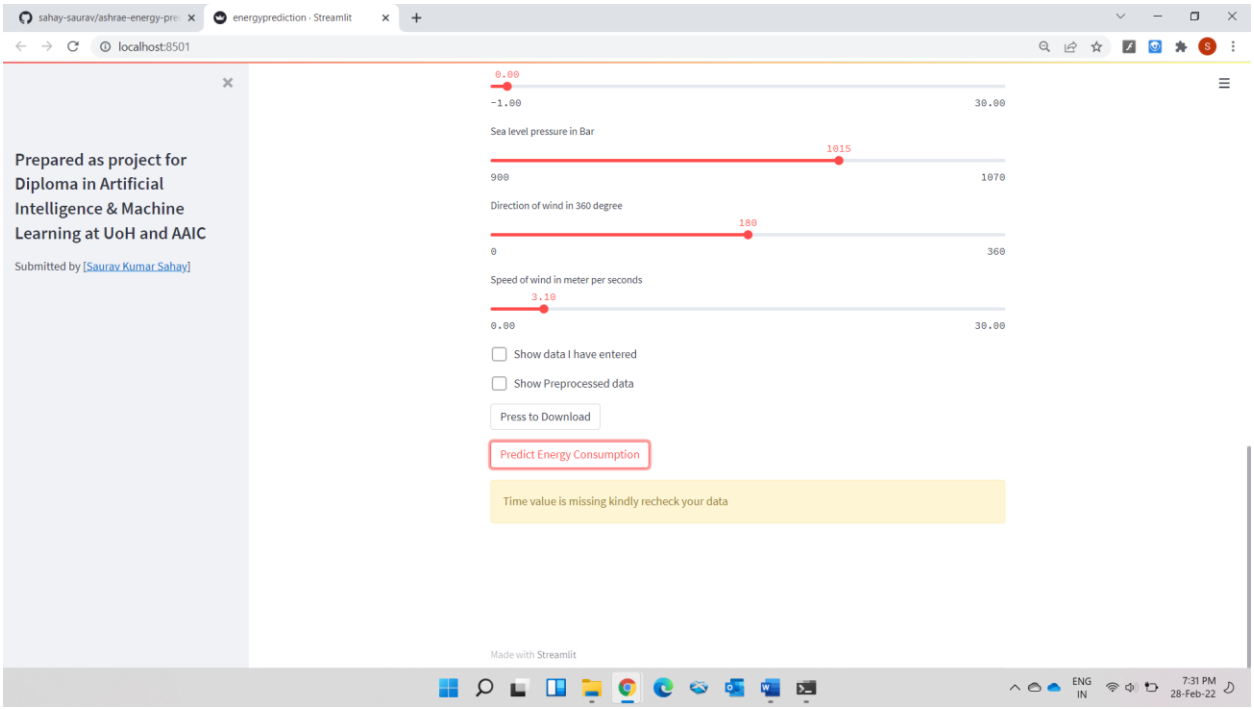
```
PS C:\Users\saurav\Downloads\ashrae-energy-prediction\Streamlit_final> streamlit run energyprediction.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.7:8501
```



Picture showing a fillable format



Picture showing predicted values once data are entered



Picture showing missing time value

5.2. Github repository

All codes used for the project along with writeup are uploaded in github and available at following repository

https://github.com/sahay-saurav/ashrae-energy-prediction

6. References.

ASHRAE https://www.ashrae.org/

ASHRAE https://en.wikipedia.org/wiki/ASHRAE

ASHRAE- Great Energy Predictor III- A Machine Learning Case Study https://medium.com/analytics-vidhya/ashrae-great-energy-predictor-iii-a-machine-learning-case-study-a01a67eb048d

ASHRAE GREAT ENERGY PREDICTION(III) https://vishalg148256.medium.com/ashrae-great-energy-prediction-iii-d4b9330d64bb BUDS

Lab https://www.budslab.org/

Buds-lab / building-data-genome-project-2 https://github.com/buds-lab/building-data-genome-project-2

Building Data Genome Project - Collecting Open Data Sets for Building Performance Research https://arxiv.org/abs/2007.06933

https://www.researchgate.net/project/Building-Data-Genome-Project-Collecting-Open-Data-Sets-for-Building-Performance-Research

What's the Difference Between RMSE and RMSLE https://medium.com/analytics-vidhya/root-mean-square-log-error-rmse-vs-rmlse-935c6cc1802a

End-to-End Introduction to Evaluating Regression Models https://www.analyticsvidhya.com/blog/2021/10/evaluation-metric-for-regression-models/

Ensembles for Time Series Forecasting http://proceedings.mlr.press/v39/oliveira14.pdf

Ensemble Learning for Time Series Prediction http://inds08.uni-klu.ac.at/INDS2008/INDS08_Ensemble_Learning_for_Time_Series_Prediction.pdf

How to Choose the Right Forecasting Technique by John C. Chambers, Satinder K. Mullick, and Donald D. Smith https://hbr.org/1971/07/how-to-choose-the-right-forecasting-technique

The Complete Guide to Time Series Analysis and Forecasting https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775