Ashrae-Great Energy Predictor III (2019)

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

Github Link: https://github.com/MichaelSetyawan/RegensteinKaggleAshrae

Authors:

Qinzi Cao, Yang Xiang, Vincent Lee, Michael Setyawan, Manaswi Mishra, Reshma Patil

Problem statement:

Develop accurate prediction models for meter readings of buildings based on the meter type: chilled water, electric, hot water, and steam meters.

Background:

Current methods to reduce costs and emissions of energy improve building efficiencies are not enough. 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. Methods in use of estimation are fragmented and do not scale well. Some assume a specific meter type or don't work with different building types.

Dataset:

The data comes from over 1,000 buildings over a three-year timeframe (train dataset: Jan-Dec 2016, test dataset: Jan 2017-Dec 2018). 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.

Files	Features	Data Size	Timeframe
Building_metadata	site_id, building_id, primary_use, square_feet, year_built, floor_count	1449	--
Train	building_id, meter, timestamp, meter_reading	20216100	Jan 2016-Dec 2016
Test	row_id, building_id, meter, timestamp	41697600	Jan 2017 – Dec 2018
Weather_train	site_id, timestampe, air_temperature, cloud_coverage, dew_temperature, precip_depth_1_hr, sea_level_pressure, wind_direction, wind_speed	139773	Jan 2016 – Dec 2016
Weather_test	site_id, timestampe, air_temperature, cloud_coverage, dew_temperature, precip_depth_1_hr, sea_level_pressure, wind_direction, wind_speed	277243	Jan 2017 – Dec 2018

Features in details

Building_metadata

- site_id - Foreign key for the weather files.
- building_id - Unique ID. Foreign key for train.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
- floorcount - Number of floors of the building

Train/Test

- building_id - Foreign key for the building metadata.
- Meter - The meter id code. Read as {0: electricity, 1: chilledwater, 2: steam, 3: hotwater}. *Not every building has all meter types.*
- timestamp - When the measurement was taken
- Meter_reading - The target variable. Energy consumption in kWh (or equivalent). Not present in Test file

Weather_train/Weather_test

- site_id - City/location 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

Evaluation Metric:

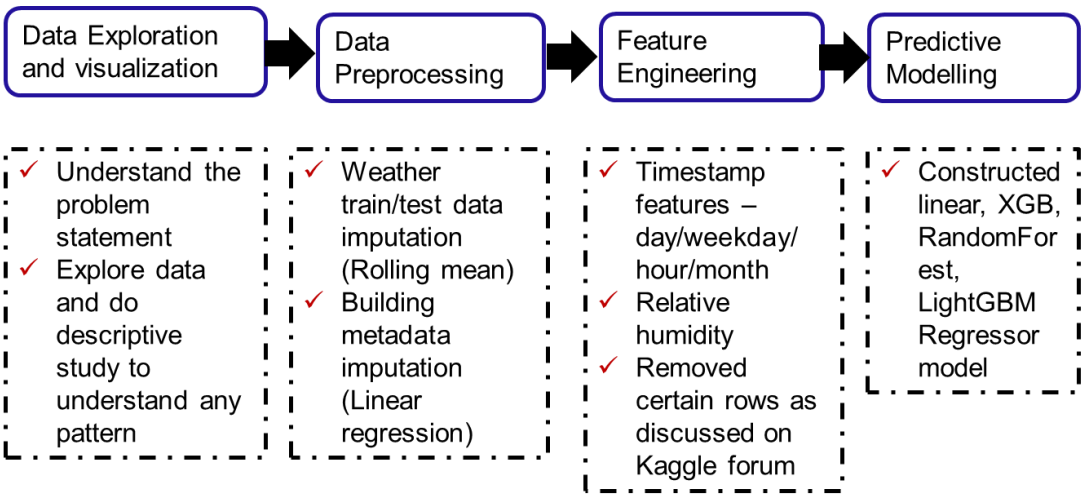
Evaluation metric for this data is Root Mean Squared Logarithmic Squared Error. The RMSLE is calculated as:

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

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

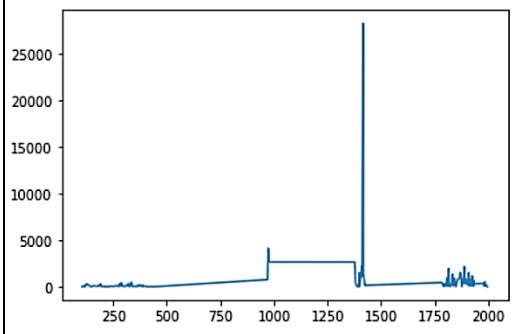
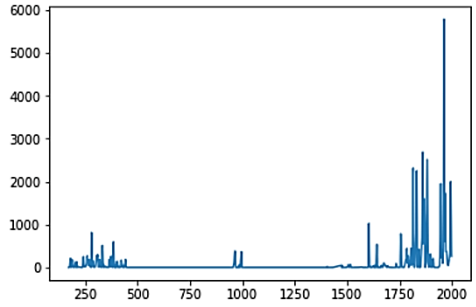
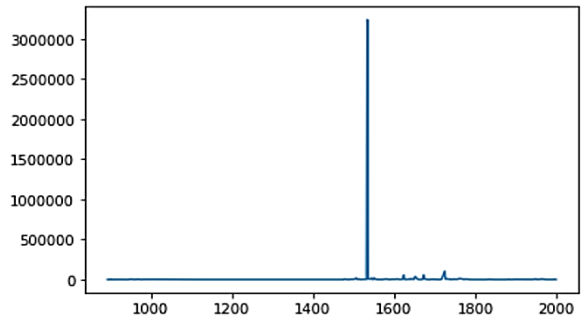
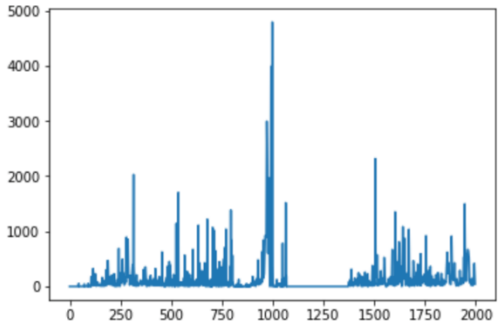
Explanation of RMSLE function: https://www.slideshare.net/KhorSoonHin/rmsle-cost-function

Process Flow:

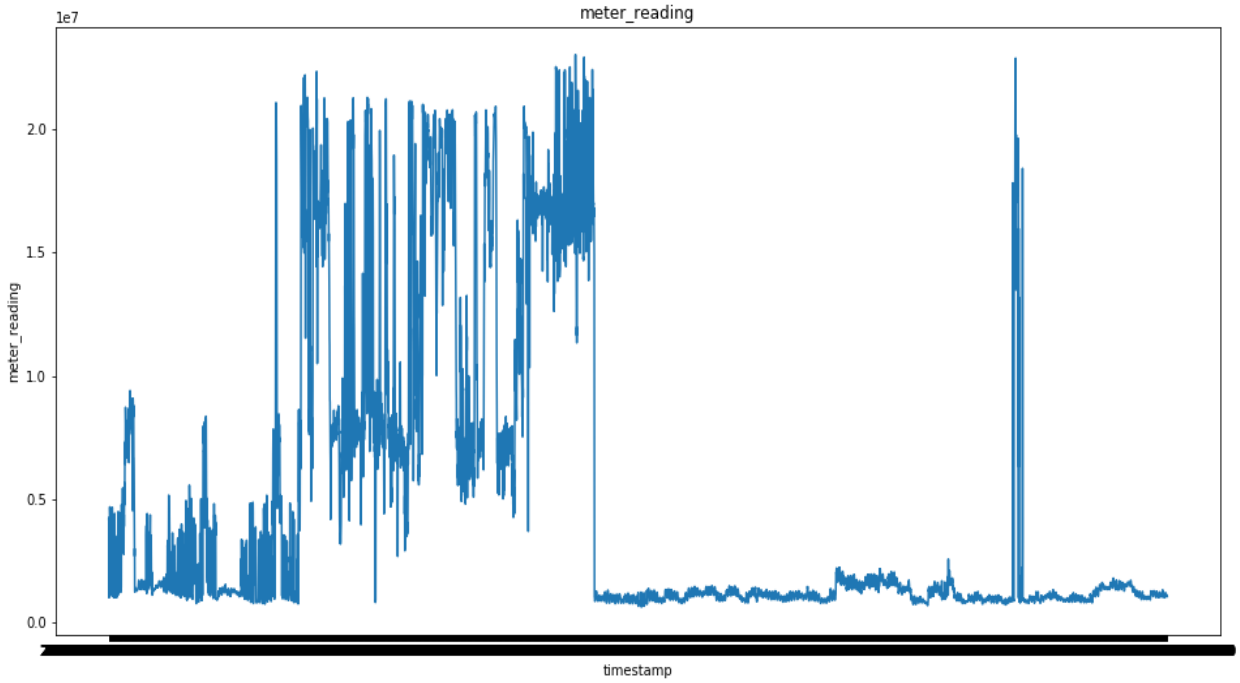


Data Exploration and Visualization:

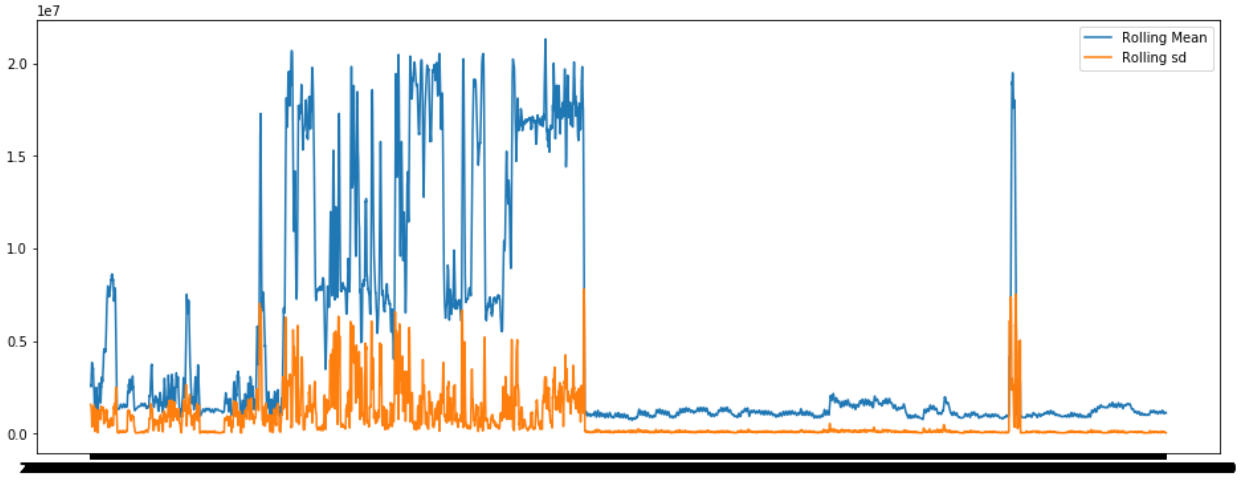
Meter type

Hot water	**Chilled Water**
	
Steam	**Electricity**
	

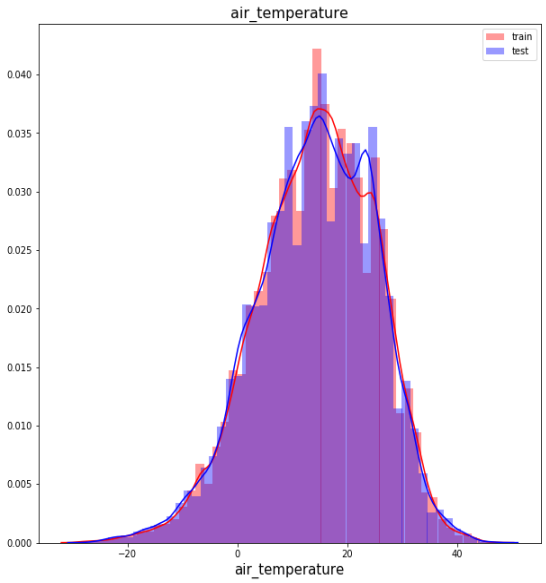
Overall meter reading by timestamp

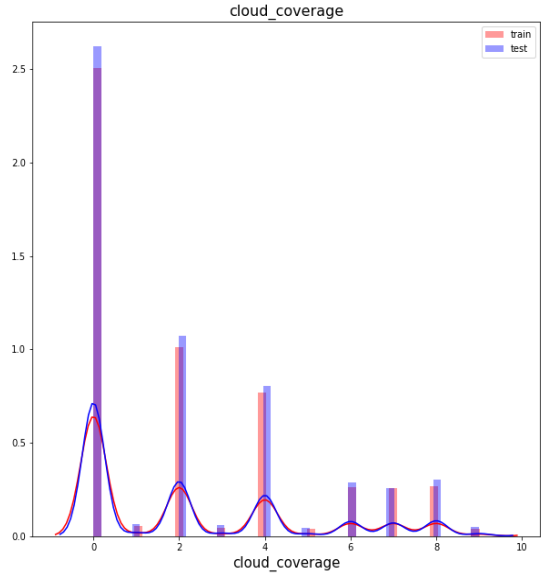


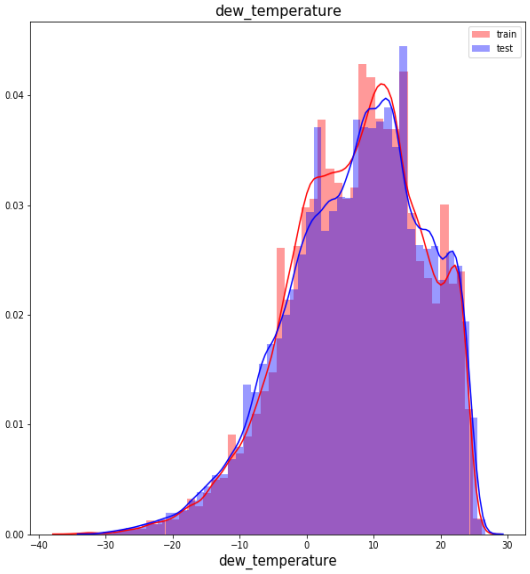
Visualization of rolling mean and sd of meter reading by timestamp

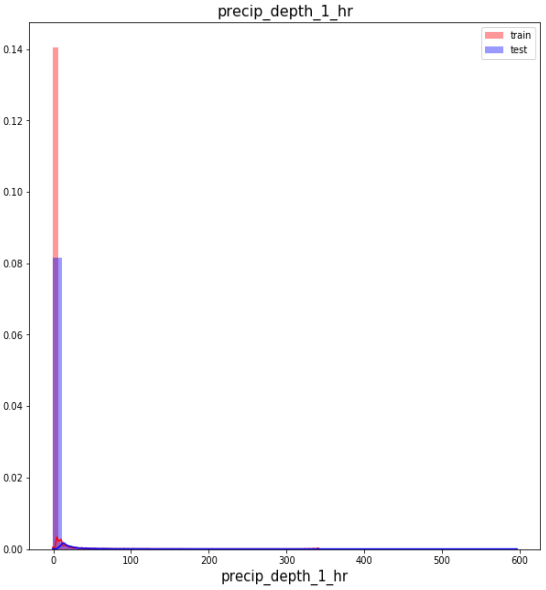


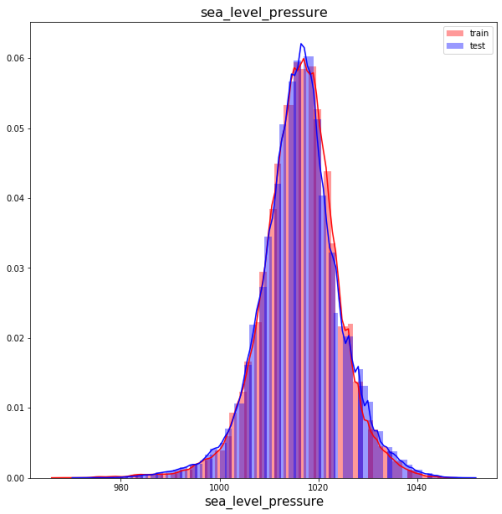
Distribution plots for variables in weather_train and weather_test

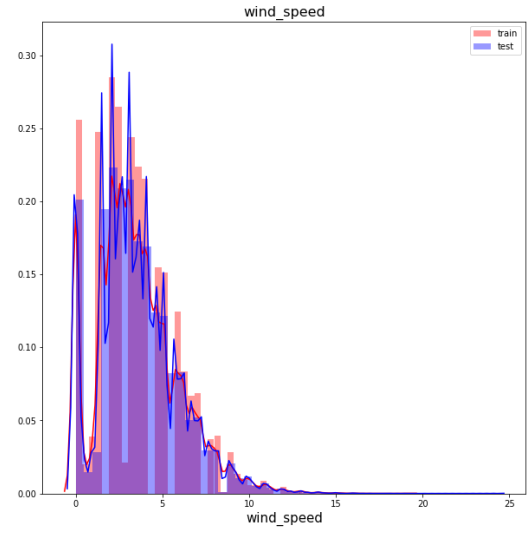




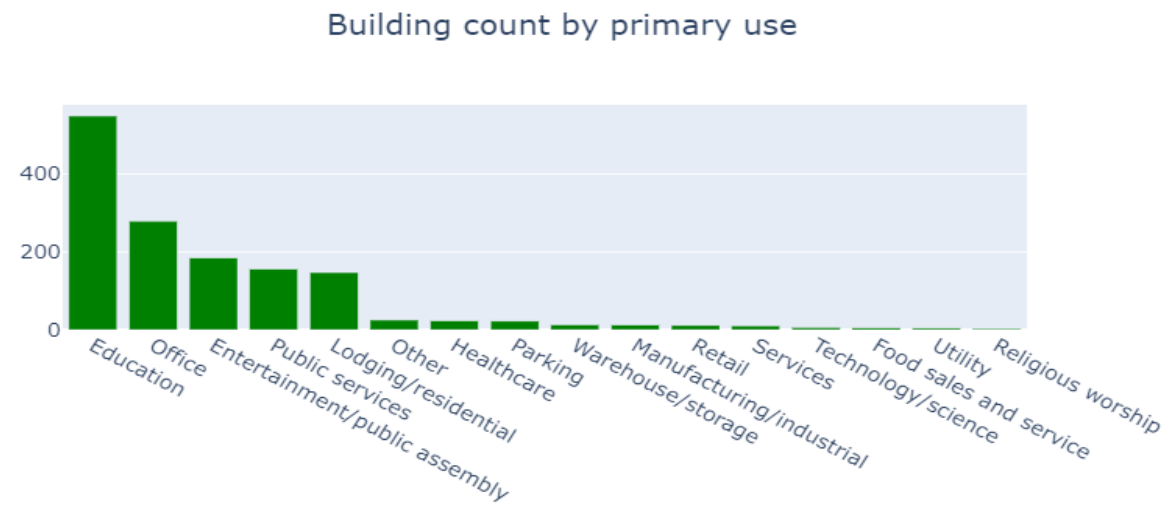


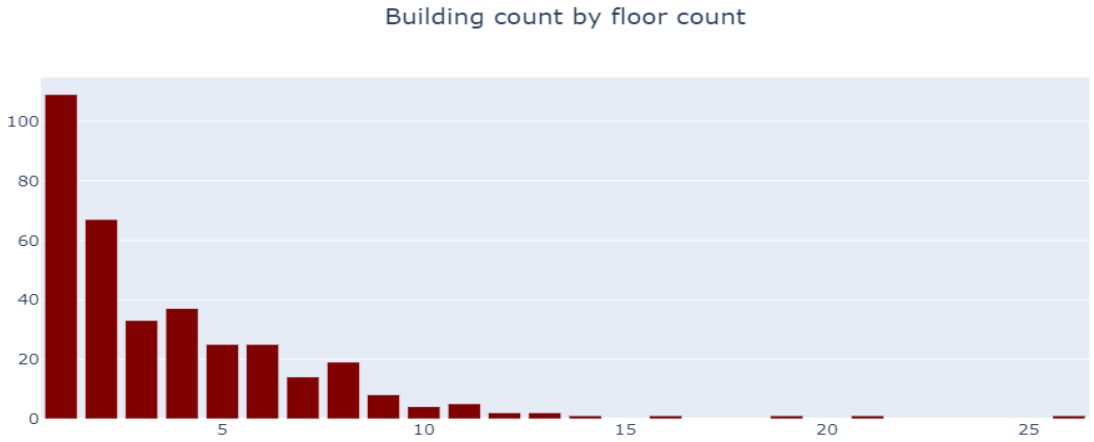


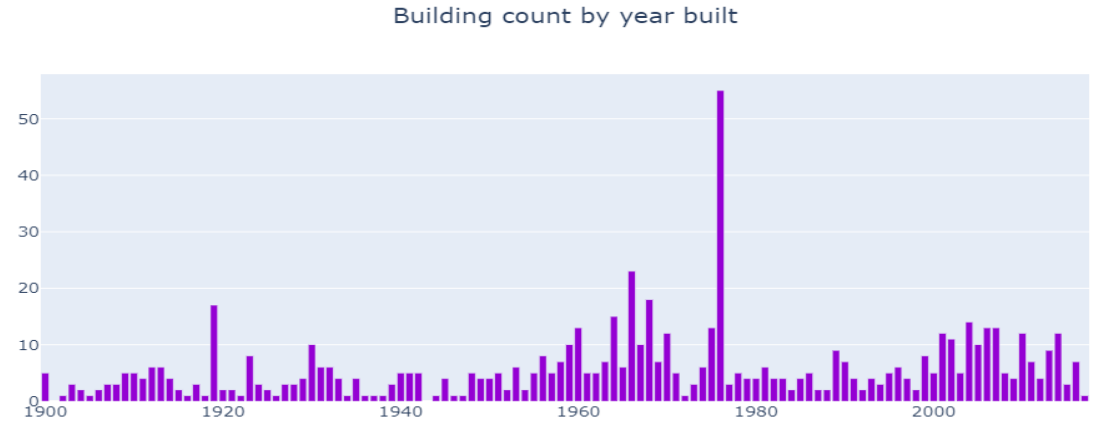




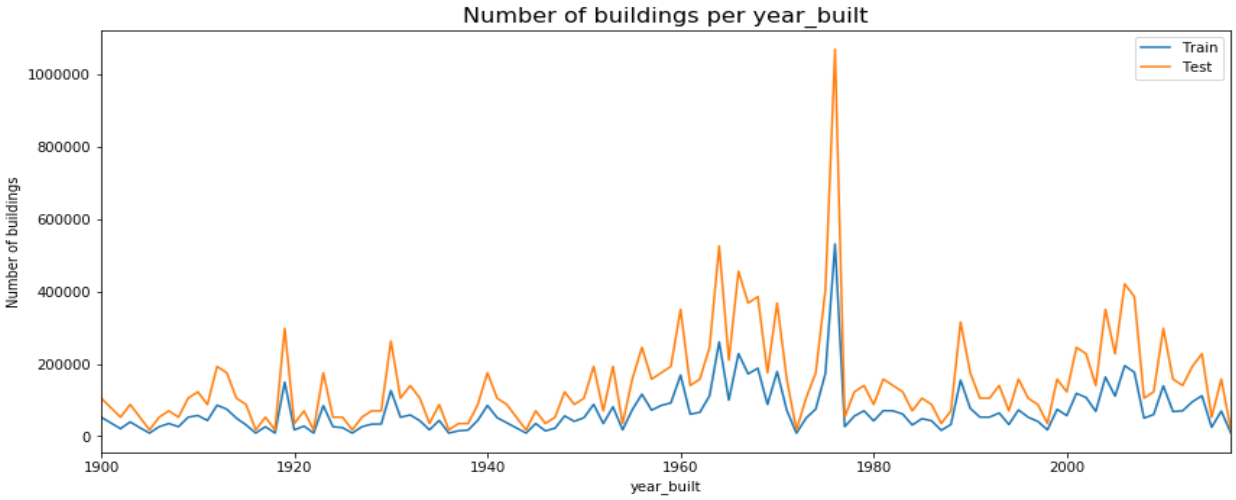
Building counts by primary use







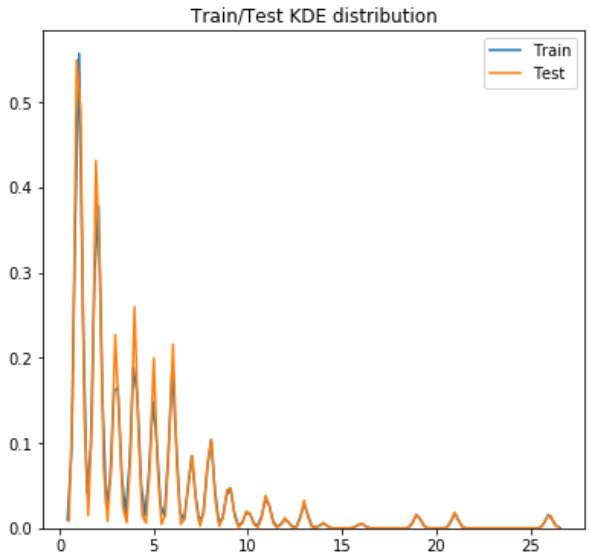
Visualizations for merged datasets (training with building and weather, test with building and weather)



Kernel distribution estimation (KDE) distribution plot of building floor count between train and test

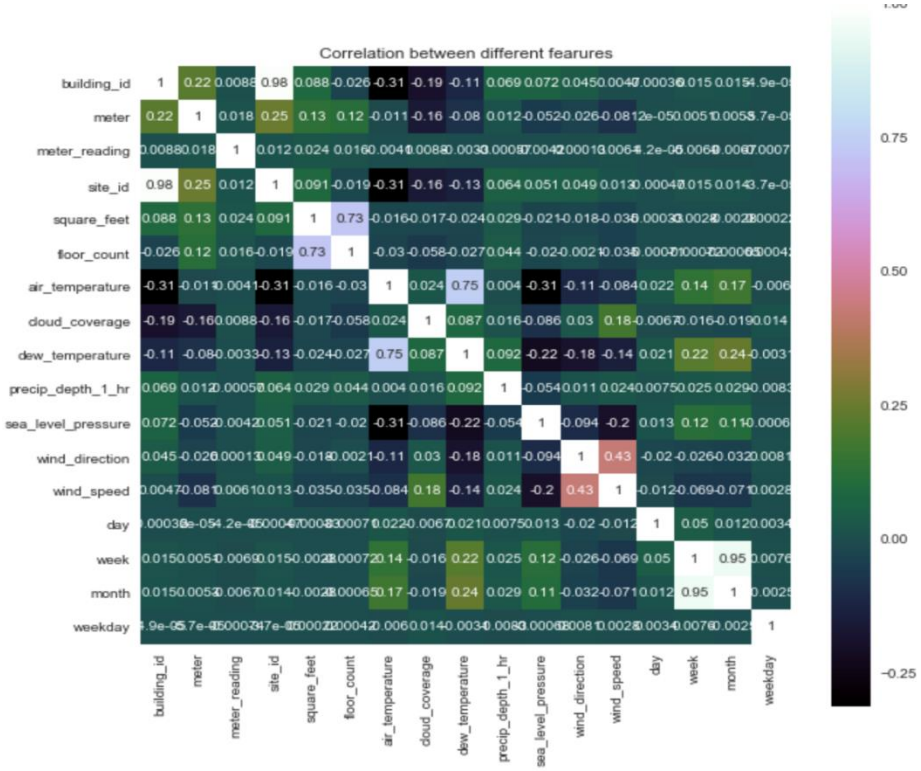
KDE is a algorithm that creates a smoother curve compared to a histogram given a set of data. The algorithm takes a parameter called bandwidth that affects the estimates on how “smooth” the curve results in. The concept of weighting the distances of observations from a particular point, x, can be expressed mathematically as follows. The variable K represents the kernel function. Using different kernel functions will produce different estimates.

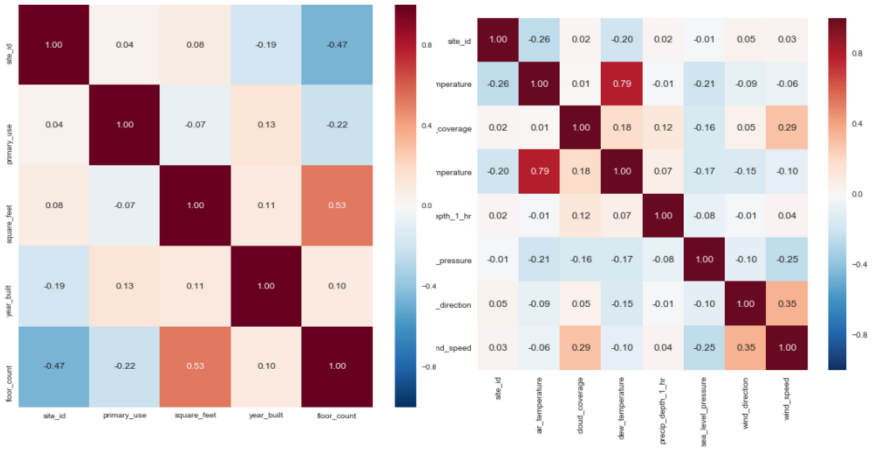
$$\hat{f}(x) = \sum_{observations} K(\frac{x - observation}{bandwidth})$$

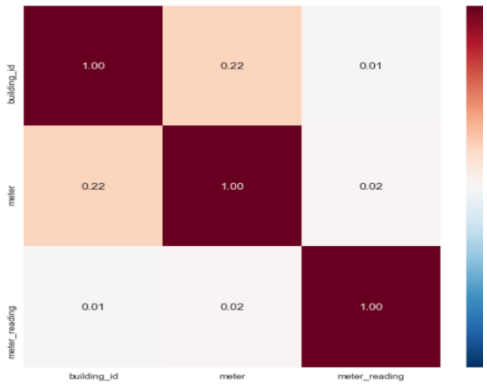


Train features:

Correlation matrix with Train features







Data Preprocessing:

Imputation of building metadata

Missing value count: year_built and floor count features have 53% and 75% missing values.

site_id	0.000000
building_id	0.000000
primary_use	0.000000
square_feet	0.000000
year_built	0.534161
floor_count	0.755003

Year_built imputation:

- Year_built value is imputed by ‘most frequent’ value from ‘Simple Imputer’ package.

Floor_count imputation:

- We build a linear regression model to impute floor_count from features such as 'site_id', 'primary_use', 'square_feet'. Accuracy of prediction on validation dataset is 65%:
    - Use LabelBinarizer to convert the categorical features to binary form including ‘site_id’, ‘primary_use’, ‘square_feet’ and use them to predict floor_count;
    - Use the existing ‘floor_count’ data as the train dataset, so the missing values of ‘floor_count’ are the test data we need to predict;
    - Identify the relationship between existing train dataset and the data we have on other three features;
    - Build the linear regression model to predict the test data.

Imputation of weather – train/test data

Weather_train

site_id	0.000000
timestamp	0.000000
air_temperature	0.000375
cloud_coverage	0.506588
dew_temperature	0.001179
precip_depth_1_hr	0.344781
sea_level_pressure	0.076702
wind_direction	0.044618
wind_speed	0.001659

Weather_test

site_id	0.000000
timestamp	0.000000
air_temperature	0.000393
cloud_coverage	0.494895
dew_temperature	0.000808
precip_depth_1_hr	0.359791
sea_level_pressure	0.075966
wind_direction	0.044844
wind_speed	0.002175

Weather data imputed by sequential manners by 3 conditions

1. Forward rolling window for 72 hours grouped on Site_id:
    - Group by the site_id because the same location tends to share similar weather features following a period of time;
    - Adopt rolling function from Python to do the calculation;

- Use observation of future 72 hours (3 days);
- Calculate the mean of observations and use the mean to fill in the NAs;

2. Backward rolling window for 72 hours grouped on Site_id
    - Group by the site_id because the same location tends to share similar weather features following a period of time;
    - Adopt rolling function from Python to do the calculation;
    - Use observation of past 72 hours;
    - Calculate the mean of observations and use the mean to fill in the NAs;
3. Average of feature value on particular timestamp when whole site_id is missing:
    - Some weather data are missing for the whole site_id;
    - Use the timestamp as the condition and set the timestamp to be equal;
    - In the same location, time can be an important feature to determine the weather data because they tend to share the same weather at the same time.

Feature Engineering:

1. **Holiday break:**
   Created holiday_break feature as most of the buildings are educational. Hoilday break intervals include summer, Thanksgiving, Christmas breaks mainly.
2. **Relative humidity:**
   Relative humidity is computed from air_temperature and dew_temperature. Relative humidity gives the ratio of how much moisture the air is holding to how much moisture it could hold at a given temperature.

   This can be expressed in terms of vapor pressure and saturation vapor pressure:

   RH = 100% x (E/Es)

   where, according to an approximation of the Clausius-Clapeyron equation:

   E = E0 x exp[(L/Rv) x {(1/T0) - (1/Td)}]
   Es = E0 x exp[(L/Rv) x {(1/T0) - (1/T)}]
   E = 6.11*10.0(7.5*air_temperature/(237.7+ air_temperature))**
   Es = 6.11*10.0(7.5*dew_temperature/(237.7+dew_temperature))**
   RH = (E/Es)*100

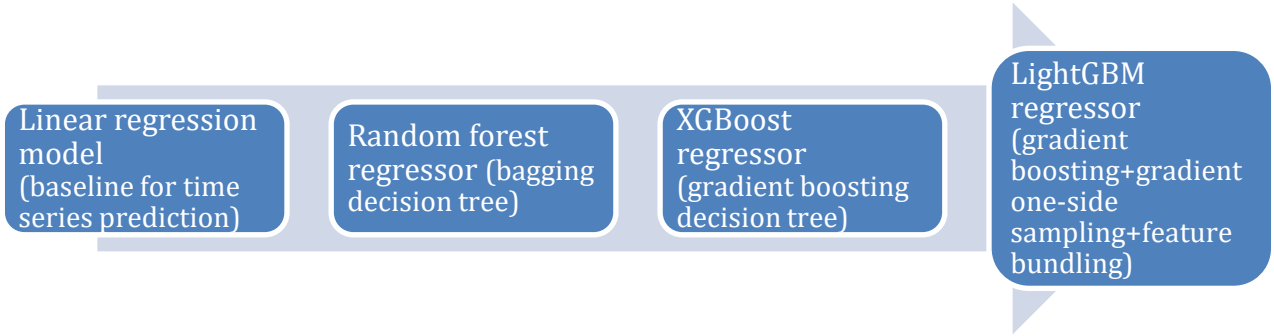
3. **Daytime**
   Computed weekday, day, month from timestamp

4. **Seasonality**
   Converted months into 4 seasons as winter, spring, summer and fall

5. **#Site_ID=0 meter reading for electricity is incorrect, so need to convert it into kWh**
6. **Removed rows where meter reading is zero**

Predictive Modeling

Model selection:



1. Constructed Linear Regression Model:
(File: model2_log.py)

Model	Categorical features	Description	Public scores
Linear Regression Model	'site_id', 'primary_use', 'meter', 'weekday', 'month', 'is_holiday_break', 'day_hour', 'season'	- Removed all the categorical features; - Clean up the data and separate them into train dataset and test dataset; - Removed the meter reading that equals to 0, took log of the train data and antilog the predicted results	1.87

2. XGBoost:
(File: XGBoost.py)

Model	Categorical features	Description	Public scores
XGBoost Model	'site_id', 'primary_use', 'meter', 'weekday', 'month', 'is_holiday_break', 'day_hour', 'season'	- Removed all the categorical features; - Used the basic XGB Regressor Model; - Removed the bad rows between the timestamp from '2006-01-01' to '2006-05-20' as well as the meter reading that is equal to 0 Limitation: Heavy computational power doesn't allow us to take into account all features	1.87 (submissionQF_XGboost.zip)

3. Kfold_RandomForest & XGBoost:
(File: kfold_randomforest.py & kfold_XGBoost.py)

Model	Categorical features	Description	Public scores
Random Forest +KFold(5)	'site_id', 'primary_use', 'meter', 'weekday', 'month', 'is_holiday_break', 'day_hour', 'season'	- Removed all the categorical features with 'day' and 'timestamp' column; - Use StratifiedKfold function to split the train and validation based on the 'building_id'; - For Random Forest model, the training set is spilt to 5 smaller data sets and the	1.57 (submissionsQF+kfoldrf.csv)

		model is trained using 4 of the folds as the training data	
XGBoost +KFold(2)	'primary_use', 'meter', 'weekday', 'month', 'is_holiday_break', 'day_hour', 'season', 'building_id'	- For XGBoost model, the training set splits to 2 smaller data sets and the model is trained using 2 of the folds as the training data	1.47 (submissionsKfoldXGB.zip)

4. LightGBM Regressor Model

Summary of models:

Model	Categorical features	Description	Public scores
LightGBM	'site_id', 'primary_use', 'meter', 'weekday', 'month', 'is_holiday_break', 'day_hour', 'season'	Removed 'building_id'	1.51 (submissionsQF_lgb.7z)
LightGBM	'primary_use', 'meter', 'weekday', 'month', 'is_holiday_break', 'day_hour', 'season', 'building_id'	Removed 'site_id'	1.33 (submissions_allV2.7z)
LightGBM +KFold(5)	'primary_use', 'meter', 'weekday', 'month', 'is_holiday_break', 'day_hour', 'season', 'site_id', 'building_id'	Selected best model predictions out of 5 by looking at RMSE score on validation (hold out) dataset on KFold	1.278 (submissions_allV5_model1.7z)
LightGBM +KFold(5)	'primary_use', 'meter', 'weekday', 'month', 'is_holiday_break', 'day_hour', 'season', 'site_id', 'building_id'	Averaged prediction output by best iterated model all 5 KFold model.	1.276 ('submissions_allV5_mean.7z')-best model
LightGBM +KFold(3)	'primary_use', 'meter', 'weekday', 'is_holiday_break', 'day_hour', 'season', 'site_id', 'building_id'	Divided train datasets into 2 halves as Jan -June and Jul-Dec. Final outcome is average predictions of two models on test datasets. (KFold – average prediction)	1.35 (submissions_2fold_half.7z)
LightGBM +KFold(3)	'primary_use', 'meter', 'weekday', 'is_holiday_break', 'day_hour', 'season', 'site_id', 'building_id'	Built 16 different models (4 *4=meter*season) by partitioning data by meter type and season for train and test. (KFold – average prediction)	1.302 (submission_infi_meter_on_season_build.gz)
LightGBM+ KFold(3)	'primary_use', 'meter', 'weekday', 'is_holiday_break',	Built 1449 models for each building by dividing train and test data into each building type. (KFold – average prediction)	1.345 (submission_infi_buildingwithmeter.gz)

Model	Categorical features	Description	Public scores
	'day_hour', 'season', 'site_id'		
LightGBM+ KFold(3)	'primary_use', 'weekday', 'is_holiday_break', 'day_hour', 'season', 'site_id', 'building_id'	Built 4 different models by dividing train and test datasets into 4 metertypes (KFold – average prediction)	1.297 (submission_infi _meter.gz)

How did the models run on large training and test datasets?

As the models require high computational power, the models are run using the compute clusters hosted at the Research Computing Center (RCC) at the University of Chicago. Following are the learning points from using RCC resources:

- Either use terminal (for mac) or use SSH for windows to connect with midway2 server
- Windows users can directly use SSH to upload or download files because they are visible
- For Mac users, terminal is not a visualized tool so either use commands to upload and download files, or use FileZilla
- Running jobs that have large memory, we can adopt the bigmem2 to do interactive computing by typing 'sinteractive --partition=bigmem2 --ntask=1 --cpus-per-task=8 --mem=128G'
- If running jobs by using interactive computing, there will be a queue where everyone is using it based on high or low priority

Lesson learnt:

- Winners' solution: Ensemble (ridge regression) of several models (LightGBM, CatBoost, Neural Network) improves accuracy
- Usage of too many features lead to overfitting

Report Date: 18/01/2020