Machine Learning Engineer Nanodegree

Capstone Project

Kally Wenying Wu
November 8th, 2019

I. Definition

In this project, I'll develop machine learning models of metered building energy usage in the following areas: chilled water, electric, hot water and steam meters. The dataset used for training comes from over 1000 buildings over a three-year timeframe.

Project Overview

Buildings consume about 40% of the total energy use in the United States. In recent years, significant investments have been made to improve building energy consumption to lower operational cost and reduce environmental footprint. Predicting energy used by heating, ventilating, and air-conditioning systems is important for HVAC diagnostics, system control, system identification, as well as energy management and optimization.

Under the current pay-for-performance financing plan, building owners make payments based on the difference between their real energy consumption and what they would have used without any retrofits. The latter values come from an estimation model. However, building energy estimation models are challenging to build and current methods of estimation are fragmented and many of them do not scale well.

Problem Statement

The goal of this project is to use machine learning algorithms to building energy estimation models based on historic usage rates and historic weather data. Machine learning algorithms produce accurate energy consumption forecasts, and they can be used to implement energy saving policies. I will work with different regression models such as linear regression, support vector machine, boosting and a small neural network to predict energy consumption using site-specific data such as weather information and historic energy usage. My motivation of working on the project comes from my previous experience in architecture / building industry and my goal is to produce more accurate prediction models to improve the efficacy of energy conservation and lower the cost of pay-for-performance financing.

Metrics

The evaluation metric for this competition is Root Mean Squared Logarithmic Error (RMSLE). I use RMSLE here instead of RMSE (Root Mean Squared Error) because both predicted and true values in this project are huge numbers and RMSLE penalizes the underestimation of the actual values more severely than it does for overestimation.

The RMSLE is calculated using the formula below:

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

Where:
n is the total number of observations in the data set,
p_i is your prediction of target,
a_i is the actual target of i, and
log(x) is the natural logarithm of x.

II. Analysis

Data Exploration

For this project, I'm using datasets from a Kaggle competition: Ashrae Energy Prediction III. The Kaggle competition datasets contain the following files:

train.csv
Historic meter reading data by timestamp for the building

- building_id - Foreign key for the building metadata.
- meter - The meter id code. Read as {0: electricity, 1: chilled water, 2: steam, 3: hot water}. Not every building has all meter types.
- timestamp - When the measurement was taken
- meter_reading - The target variable. Energy consumption in kWh (or equivalent). Note that this is real data with measurement error, which I expect will impose a baseline level of modeling error.
- building_meta.csv
- site_id - Foreign key for the weather files.
- building_id - Foreign key for training.csv
- primary_use - Indicator of the primary category of activities for the building based on EnergyStar property type definitions
- square_feet - Gross floor area of the building
- year_built - Year building was opened
- floor_count - Number of floors of the building

The train.csv file has 20,216,100 entries.

	building_id	meter	timestamp	meter_reading
0	0	0	2016-01-01 00:00:00	0.0
1	1	0	2016-01-01 00:00:00	0.0
2	2	0	2016-01-01 00:00:00	0.0
3	3	0	2016-01-01 00:00:00	0.0
4	4	0	2016-01-01 00:00:00	0.0

building_meta.csv
Building metadata including the building use, square ft area and year build information. Building_meta has 1,449 entries.

	site_id	building_id	primary_use	square_feet	year_built	floor_count
0	0	0	Education	7432	2008.0	NaN
1	0	1	Education	2720	2004.0	NaN
2	0	2	Education	5376	1991.0	NaN
3	0	3	Education	23685	2002.0	NaN
4	0	4	Education	116607	1975.0	NaN

weather_[train/test].csv
Weather data from a meteorological station as close as possible to the site. Dataset includes precipitation, cloud coverage, air temperature and etc.

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

The weather_train file has 139,773 entries and weather_test file has 277,243 entries.

	site_id	timestamp	air_temperature	cloud_coverage	dew_temperature	precip_depth_1_hr	sea_level_pressure	wind_direction	wind_speed
0	0	2016-01-01 00:00:00	25.0	6.0	20.0	NaN	1019.7	0.0	0.0
1	0	2016-01-01 01:00:00	24.4	NaN	21.1	-1.0	1020.2	70.0	1.5
2	0	2016-01-01 02:00:00	22.8	2.0	21.1	0.0	1020.2	0.0	0.0
3	0	2016-01-01 03:00:00	21.1	2.0	20.6	0.0	1020.1	0.0	0.0
4	0	2016-01-01 04:00:00	20.0	2.0	20.0	-1.0	1020.0	250.0	2.6

test.csv contains the meter, building id and timestamp that I will be predicting on.

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

The test.csv file has 41,697,600 entries.

	row_id	building_id	meter	timestamp
0	0	0	0	2017-01-01 00:00:00
1	1	1	0	2017-01-01 00:00:00
2	2	2	0	2017-01-01 00:00:00
3	3	3	0	2017-01-01 00:00:00
4	4	4	0	2017-01-01 00:00:00

sample_submission.csv contains all the future data that I need to predict on. sample_submission has 24,936,697 entries.

	row_id	meter_reading
0	0	0.0
1	1	0.0
2	2	0.0
3	3	0.0
4	4	0.0

Running a simple statistical analysis on the dataset shows the mean and standard deviation of meter reading based on different building types.

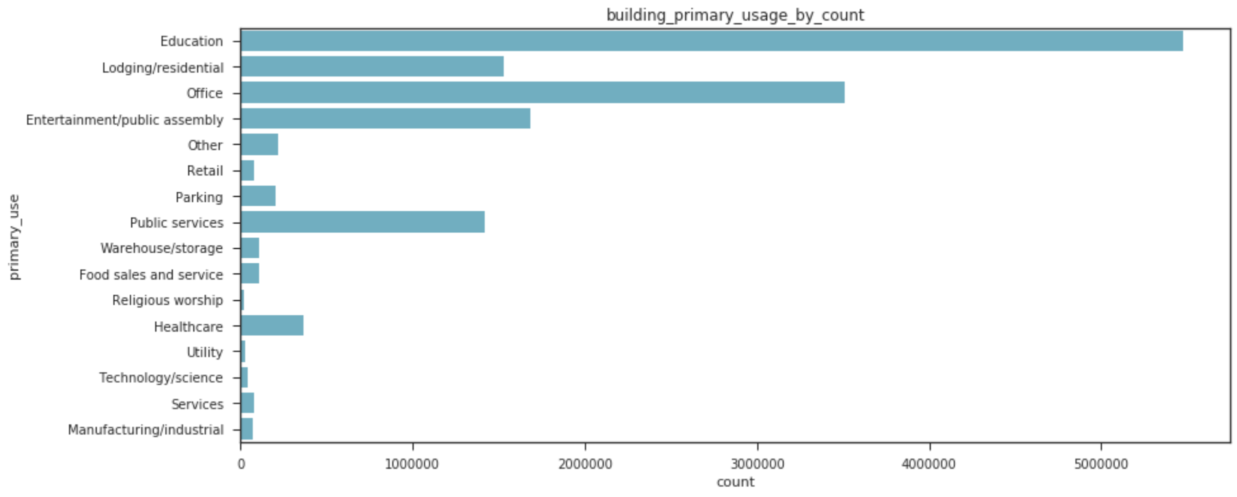
	primary_use	meter_reading_mean	meter_reading_std
0	Education	6410.346163	288503.022712
1	Lodging/residential	327.948961	1133.527547
2	Office	606.529602	3688.813310
3	Entertainment/public assembly	540.230224	11645.436617
4	Other	143.073003	458.542699
5	Retail	170.388693	367.559422
6	Parking	172.075026	693.211104
7	Public services	305.621619	1284.002356
8	Warehouse/storage	54.393144	66.802817
9	Food sales and service	305.341700	570.281145
10	Religious worship	3.969560	3.101675
11	Healthcare	764.286871	1386.096818
12	Utility	634.306124	803.768303
13	Technology/science	134.744648	248.018141
14	Services	4715.030703	15621.874757
15	Manufacturing/industrial	322.756190	533.493546

After joining train.csv with weather_train.csv files based on site_id and timestamp, I created a training data frame with 20,216,100 entries. The dataset has many missing values. Running df_train.isnull().sum(), I calculate the percentage of missing values on the training dataset based on the variable names:

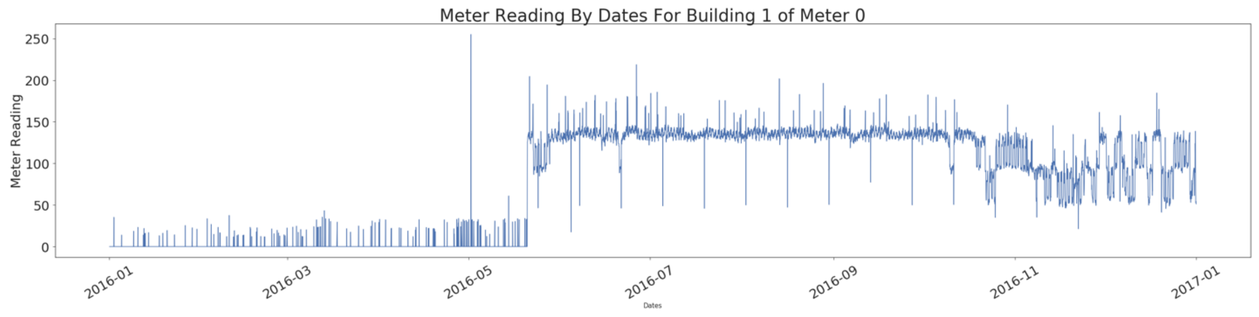
year_built	59.99%
floot_count	82.65%
air_temperature	0.48%
cloud_coverage	43.66%
dew_temperature	49.53%
precip_depth_1_hr	18.54%
sea_level_pressure	6.09%
wind_direction	7.17%
wind_speed	71.07%

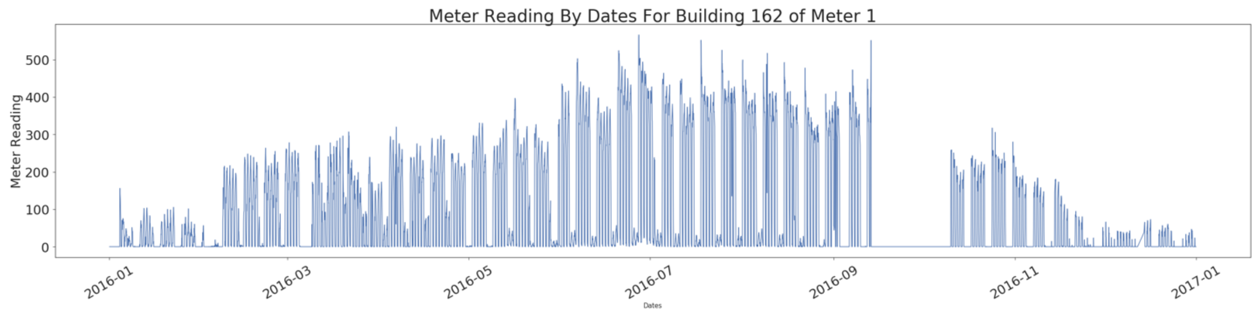
Exploratory Visualization

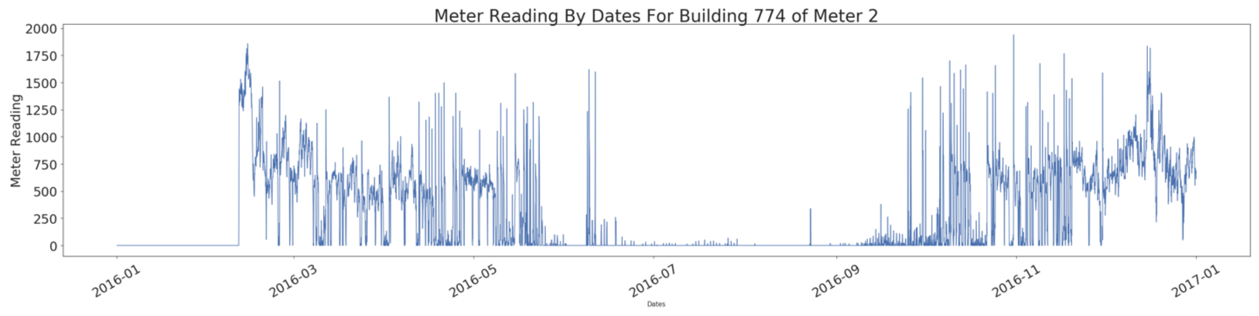
I conducted exploratory data analysis on the dataset and visualized the primary usages of the buildings. As the plot indicates, the most common building usage is educational.

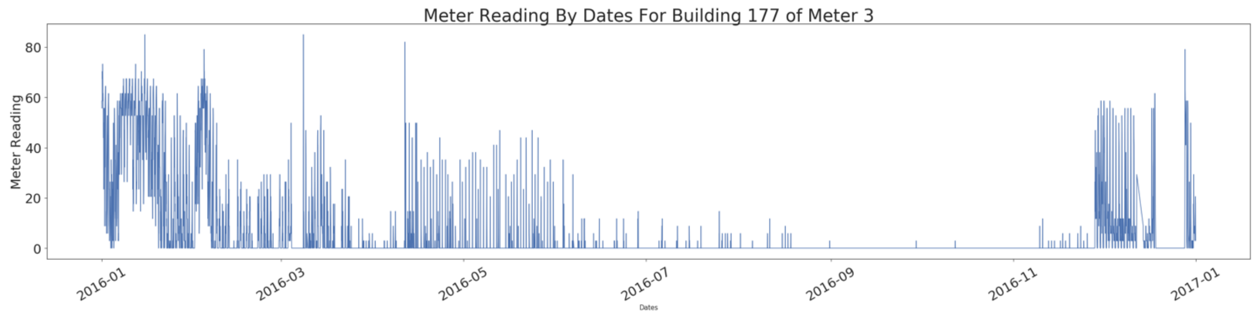


I created several plots of building energy consumption by building id, meter type and site id. As the plots indicate, building energy usage varies with seasons.

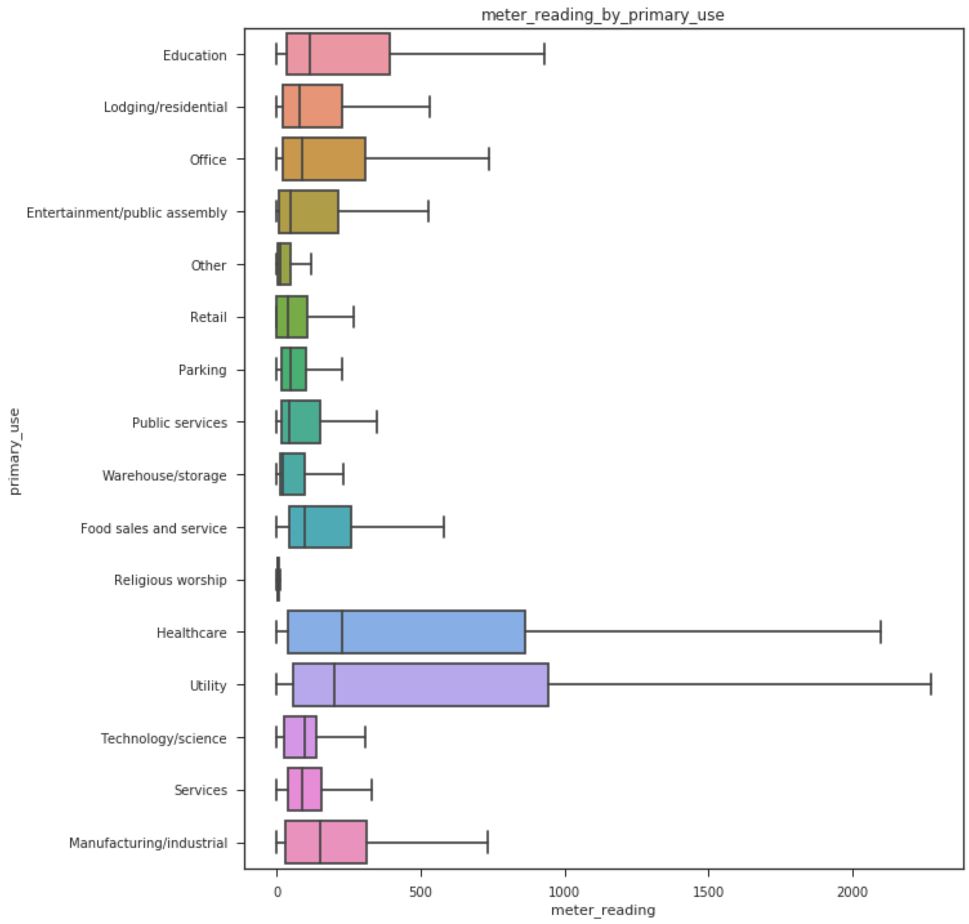


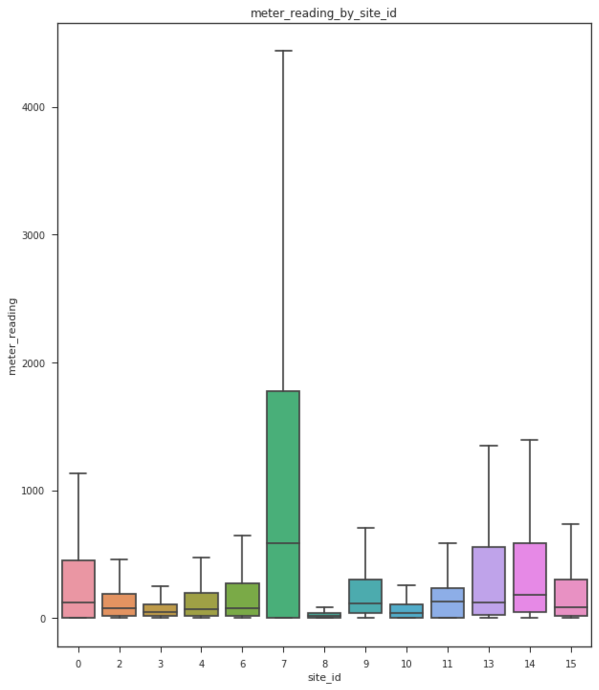


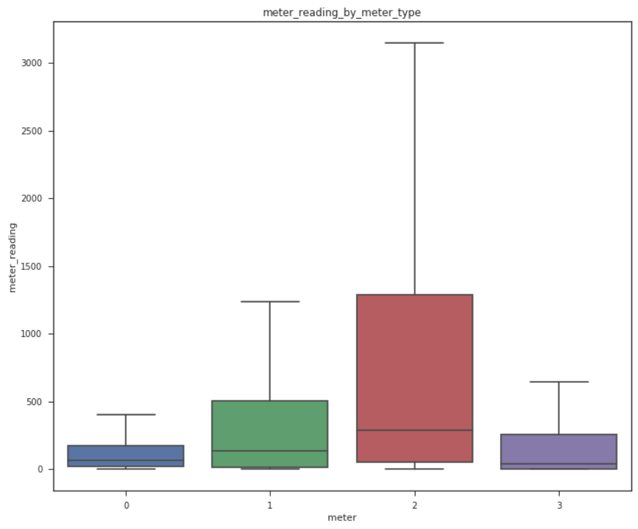


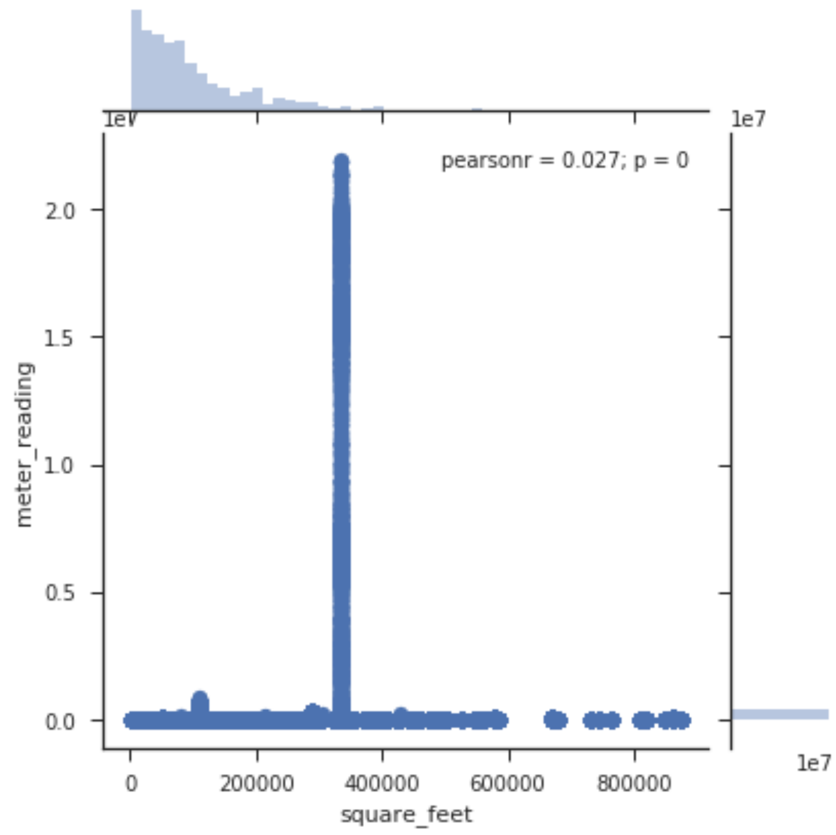


I created a few boxplots trying to understand the relationship between meter reading, meter type, primary usage and square feet. Building energy consumption varies with primary use, meter_type, site id and square feet.


meter_reading_by_primary_use
primary_use
Education
Lodging/residential
Office
Entertainment/public assembly
Other
Retail
Parking
Public services
Warehouse/storage
Food sales and service
Religious worship
Healthcare
Utility
Technology/science
Services
Manufacturing/industrial
0
500
1000
1500
2000
meter_reading

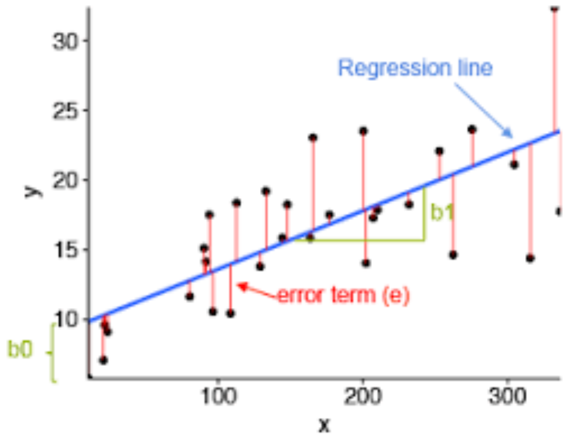

meter_reading_by_site_id
meter_reading
4000
3000
2000
1000
0
0
2
3
4
6
7
8
9
10
11
13
14
15
site_id


meter_reading_by_meter_type
meter_reading
3000
2500
2000
1500
1000
500
0
0
1
2
3
meter



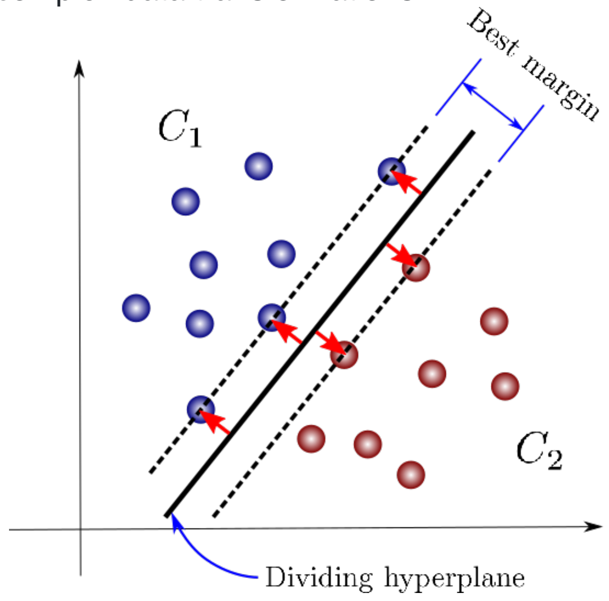
Algorithms and Techniques

To solve the problem, I will first train a linear regression model as my baseline using following features from EDA: 'building_id', 'meter', 'site_id', 'primary_use', 'square_feet' and 'month'. Linear regression is a simple method that is easy and intuitive to use. However, the model doesn’t do a good job modeling non-linear relationship and it’s also very sensitive to anomalies in the dataset.



[Linear regression model]

Next, I will experiment with more computationally expensive models such as support vector machine (SVM) regressor and LightGBM boosting algorithm. SVM is a supervised machine learning algorithm that works well with classification and regression problems by applying complex data transformations.



[Support vector machine hyperplane]

LightGBM is a gradient boosting framework that uses a tree-based learning algorithm. LightGBM has fast training speed and is very memory efficient.

Lastly, I will work with a small long short-term memory neural network. LSTM networks are well-suited for making predictions based on time-series data since they were developed to deal with exploding and vanishing gradient problems when training traditional RNNs.

I will use RMSLE as my evaluation metrics to compare the performances of these solutions against the baseline model.

Benchmark

For the benchmark, I fit a linear regression model on the training dataset after applying ridge regression cross validation for hypermeter tuning. My baseline model returned a RMSLE score of 4.3460. Similarly, a lasso linear regression model returned a RMSLE score of 4.3464.

III. Methodology

Data Preprocessing

As part of data preprocessing step, I split train.csv into training, validation and test sets using the 80/20 rule. The training set contains 9,576,523 entries of historic building energy usage data from 4 different meter types: chilled water, electric, hot water and steam meters, the validation set has 2,394,131 entries and the test set has 2,992,664 entries.

Based on the percentage of missing data, I dropped columns that have more than 40% missing values: ‘year_built’, ‘floor_count’, ‘cloud_coverage’ and ‘dew_temperature’. I also applied filtering to remove entries that have missing values.

As part of the data preprocessing steps, I converted the timestamp column into datetime objects, and factored the each datetime object to year, month, day and hour variables.

Implementation

First, I worked with a support vector machine model. Given the size of the training dataset, I picked a stochastic gradient descent regressor, which is known for successfully solving large-scale problems that has more than 100,000 training examples. My SVM model output a RMSLE score of 4.2734, which is a slight improvement from the linear regression baseline.

Second, I worked with a LightGBM model. To prevent the model from overfitting, I applied early stopping using a validation set. Using the default hyperparameter settings, my LightGBM model achieved a RMSLE score of 2.2994, which is a significant increase in performance from the baseline.

Lastly, I worked with a small long short-term memory (LSTM) neural network. A sequential model which is a linear stack of layers is used. The first layer is a LSTM layer with 30 memory units and it returns sequences. This is done to ensure that the next LSTM layer receives sequences and not just randomly scattered data. The second layer is an LSTM layer with 16 memory units and it returns sequences too. A dropout layer is applied after the second layer to avoid overfitting of the model. Finally, I have the last layer as a fully connected layer and neuron size set to 1, because I need to output one predicted energy consumption result. By applying early stopping and training the model for 50 epochs, my LSTM model achieved a RMSLE score of 2.1849, making it the best-performing model among the three.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm (LSTM)                  (None, 1, 30)             4440      
_________________________________________________________________
lstm_1 (LSTM)                (None, 1, 32)             8064      
_________________________________________________________________
dropout (Dropout)            (None, 1, 32)             0         
_________________________________________________________________
lstm_2 (LSTM)                (None, 1, 16)             3136      
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 16)             0         
_________________________________________________________________
dense (Dense)                (None, 1, 1)              17        
=================================================================
Total params: 15,657
Trainable params: 15,657
Non-trainable params: 0
_________________________________________________________________
```

Refinement

The lightGBM.cv function evaluates the performance of the model-building procedure using a k-fold cross validation in helping people getting a sense of getting the error + stat uncertainty. Lightbgm.cv() allows one to evaluate performance on a k-fold split with fixed model parameters.

For better accuracy results, I lowered the learning_rate to 0.001 and increased number of iterations to 1000. Cross validation result shows that the best num_boost_round is 1000. By calling the lightgbm.train method, I get a RMSLE score of 3.6835 on the test set.

Similarly, I refined the existing LightGBM model again by increasing the number of leaves to 100 for better accuracy results. This model returned a RMSLE score of 3.3692 on the test set.

By including weather parameters, I retrained a LSTM model using 9 features instead of 5. The first layer is a LSTM layer with 30 memory units with returned sequences. This is done to ensure that the next LSTM layer receives sequences and not just randomly scattered data. The second layer is an LSTM layer with 32 memory units [with returned sequences]. The third layer is an LSTM layer with 16 memory units [with returned sequences]. A dropout layer is applied after the second and the third layers to avoid overfitting. Finally, a last layer is added as a fully connected layer with its neuron size set to 1. By applying early stopping and training the model for 50 epochs, my LSTM model achieved a RMSLE score of 2.1840, which is a slight 0.0009 improvement from the vanilla LSTM model.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_4 (LSTM)                (None, 1, 30)             4800      
_________________________________________________________________
lstm_5 (LSTM)                (None, 1, 32)             8064      
_________________________________________________________________
dropout_2 (Dropout)          (None, 1, 32)             0         
_________________________________________________________________
lstm_6 (LSTM)                (None, 1, 16)             3136      
_________________________________________________________________
dropout_3 (Dropout)          (None, 1, 16)             0         
_________________________________________________________________
dense_2 (Dense)              (None, 1, 1)              17        
=================================================================
Total params: 16,017
Trainable params: 16,017
Non-trainable params: 0
_________________________________________________________________
```

IV. Results

Model Evaluation and Validation

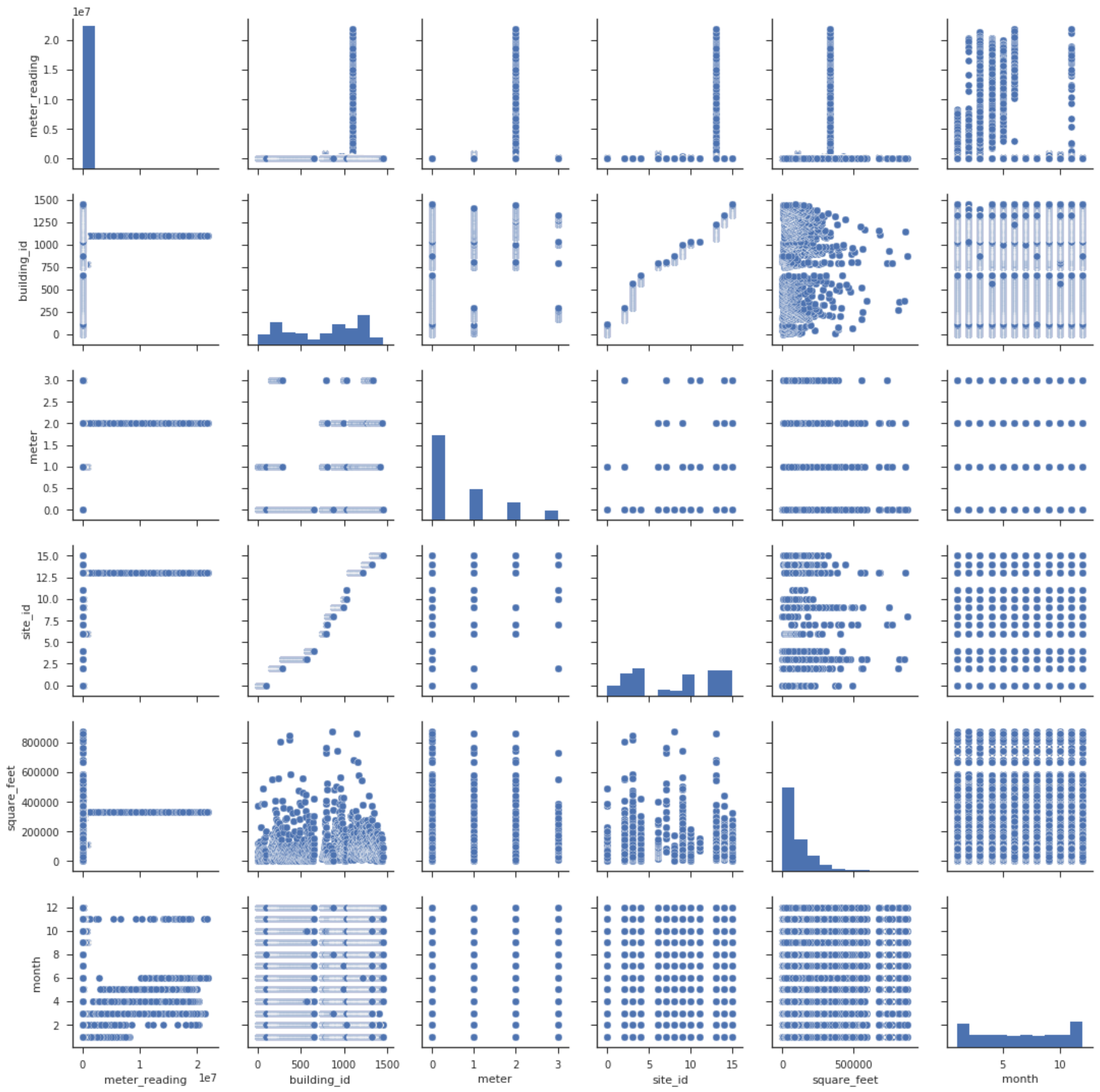
Model	RMSLE score
Ridge Linear Regression (baseline)	4.3460
Lasso Linear Regression	4.3464
SVM Regressor	4.2734
LightGBM (vanilla)	2.2994
LightGBM (learning_rate)	3.6836
LightGBM (num_of_leaves)	3.3692
LSTM (vanilla)	2.1849
LSTM (with weather info)	2.1840

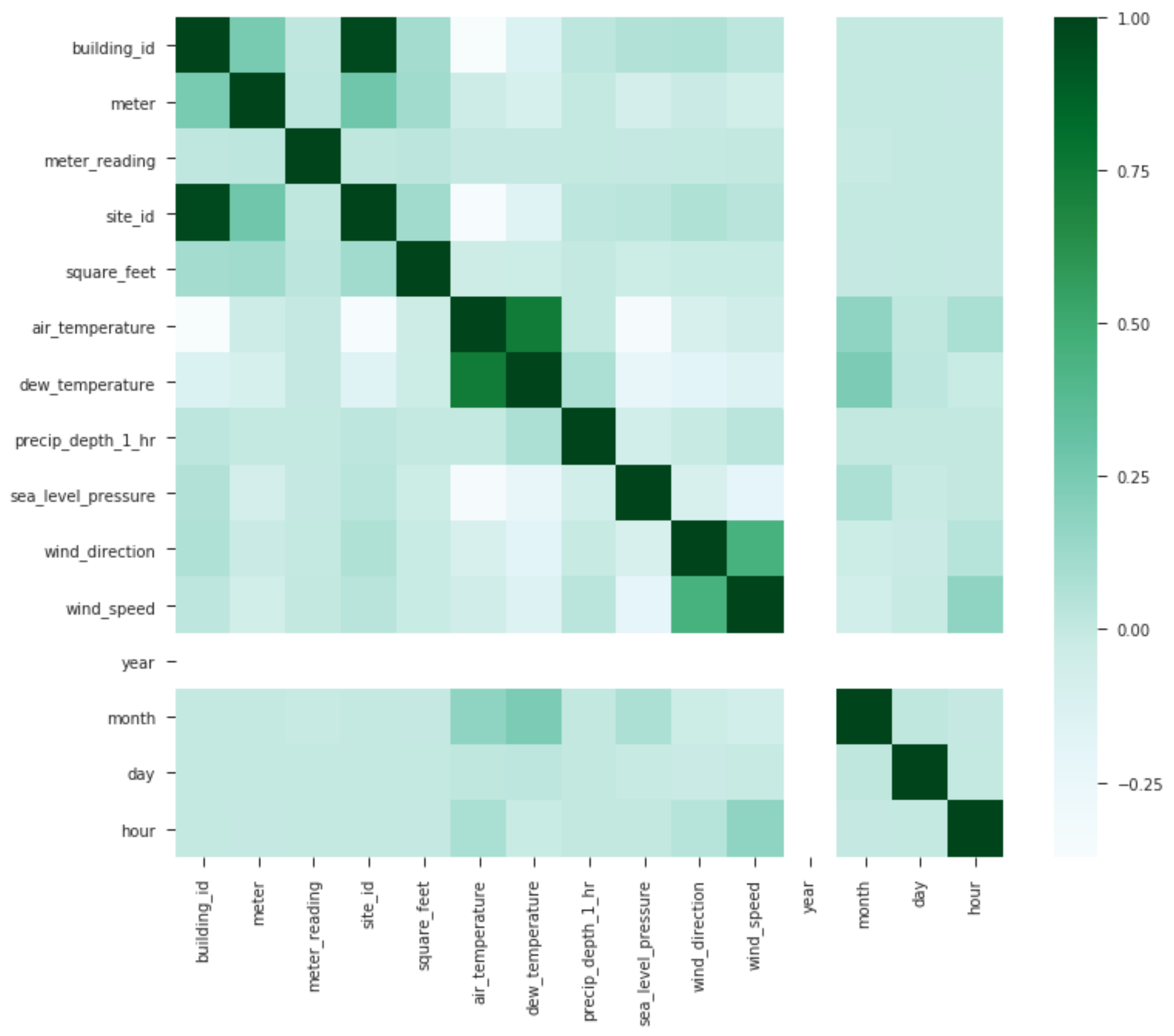
Using RMSLE as an evaluation metrics, the LSTM model with weather information turns out to be the best performing model that achieved a RMSLE score of 2.1840, which is a significant 49.75% decrease from the RMSLE score of the linear regression baseline model.

V. Conclusion

Free-Form Visualization

Using the pairwise plot and the correlation plot, I further investigated the non-linear relationship between ‘meter reading’ and other variables such as ‘building_id’, ‘meter’, ‘site_id’, ‘square_feet’ and ‘month’.





Reflection

The process used for this project could be summarized with the following steps:

1. The initial problem was defined, and the relevant dataset was preprocessed.
2. The dataset was analyzed using exploratory data visualization.
3. A baseline model was trained and evaluated using relevant features from data visualization.
4. Several further models were trained and refined using hyperparameter tuning.
5. The models were evaluated using RMSLE scores.

Improvement

If I were to continue working on this project, below are a number of additional areas that could be explored:

- *Train the model with more features such as historical weather data by consulting with domain experts*
- *Work with ensemble methods using existing models*
- *Experiment with another neural network architecture*

References:

- Kreider, J.F., and Haberl, J.S. Sat. "Predicting hourly building energy use: The great energy predictor shootout -- Overview and discussion of results". United States.
- ASHRAE – Great Energy Predictor III https://www.kaggle.com/c/ashrae-energy-prediction
- Building energy consumption prediction, a comparison of five machine learning algorithms, http://cs109-energy.github.io/
- LightGBM Parameters Tuning, https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html