ASHRAE - Great Energy Predictor III

Machine Learning Lab - Project
Group Name : Desi Boyz
Ash, Samarth and Shishir

Agenda

- Problem overview
- Feature engineering
- Model fitting
- Final model selection
- Summary

Motivation

Ask and Acquire

Problem Statement

Current method used to charge for energy usage : “pay-for-performance”

We aim to predict the mean hourly energy consumption of buildings with various energy meter types.

Dataset: Hourly energy meter readings, weather data and building characteristics

- Aggregated data at a daily level

Features	
Building Characteristics	site_id
	building_id
	primary_use
	square_feet
	meter
Weather Data	air_temperature
	dew_temperature
	wind_speed
	wind_direction
	cloud_coverage
	precip_depth_1_hr
DateTime data	month
	day

Target
mean hourly energy consumption per day

792,863 observations
13 potential predictors

Models

1) Linear Regression : To set a baseline

2) Random Forest : predictions derived from a collection of trees trained parallely

3) XGBoost : a parallelized implementation of Gradient Boosting, where predictions are derived from a collection of trees trained sequentially.

Process

Feature Engineering and Pipeline creation

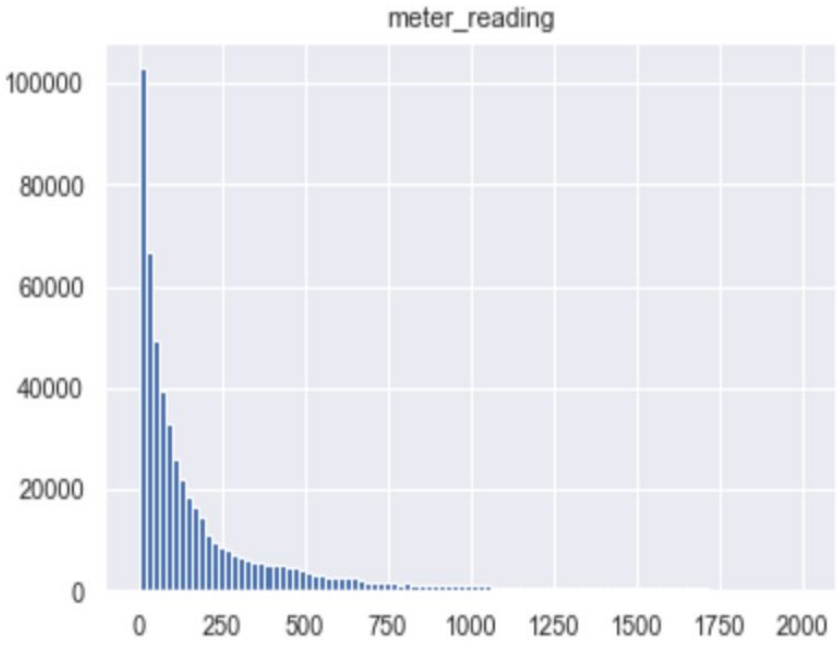
- Convert the datetime to day of week and month columns
- For numeric columns, imputed missing values with the median value of the column
- For categorical columns, used label encoding

Hyperparameter tuning

- Using the pipeline, performed random search to find optimal hyperparameters
    - To increase generality we performed a 5-fold CV

Evaluation Metric

Our north-star metric is *median absolute error*. We chose it due to the skewed distribution of y variable.

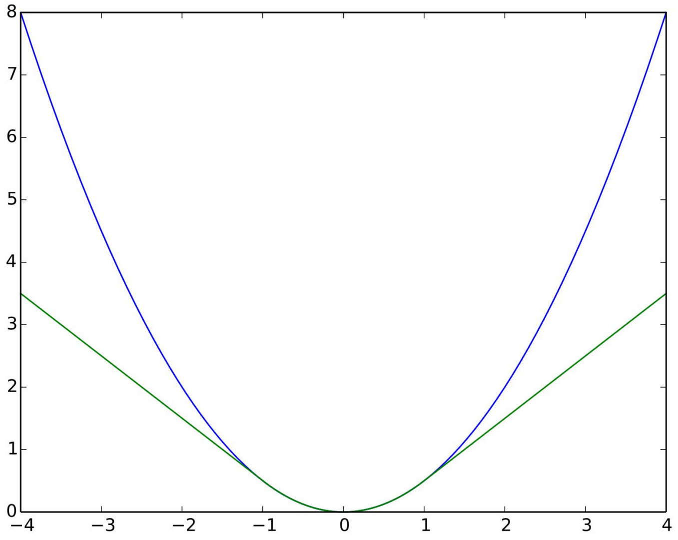


Loss Function

Our friend from first assignment - the Huber Loss function

Reasons for choosing Huber-Loss:

- Less sensitive to outliers in data
- Continuous
- Differentiable



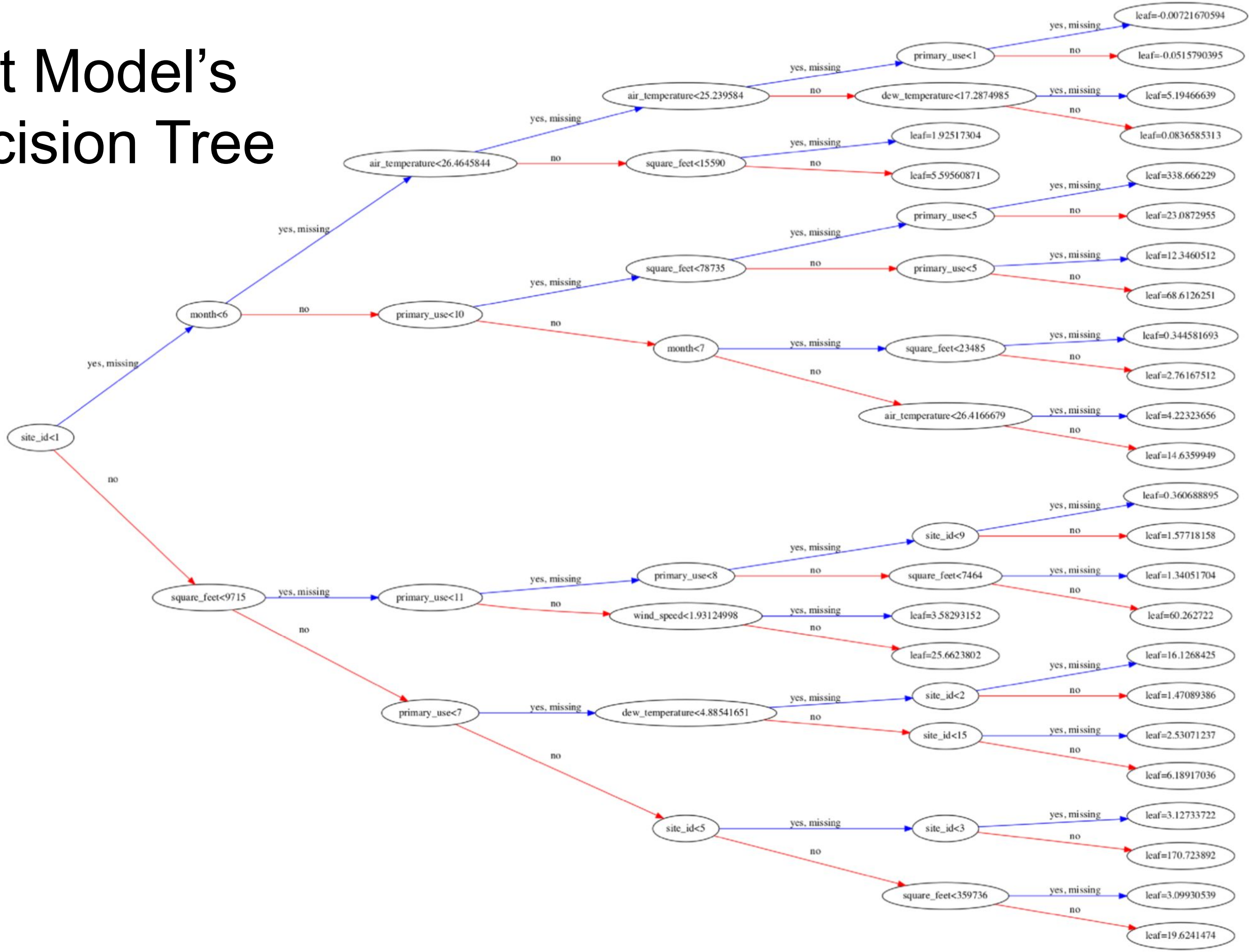
$$L_\delta(a) = \delta^2\left(\sqrt{1 + (a/\delta)^2} - 1\right)$$

$$\delta = 1$$

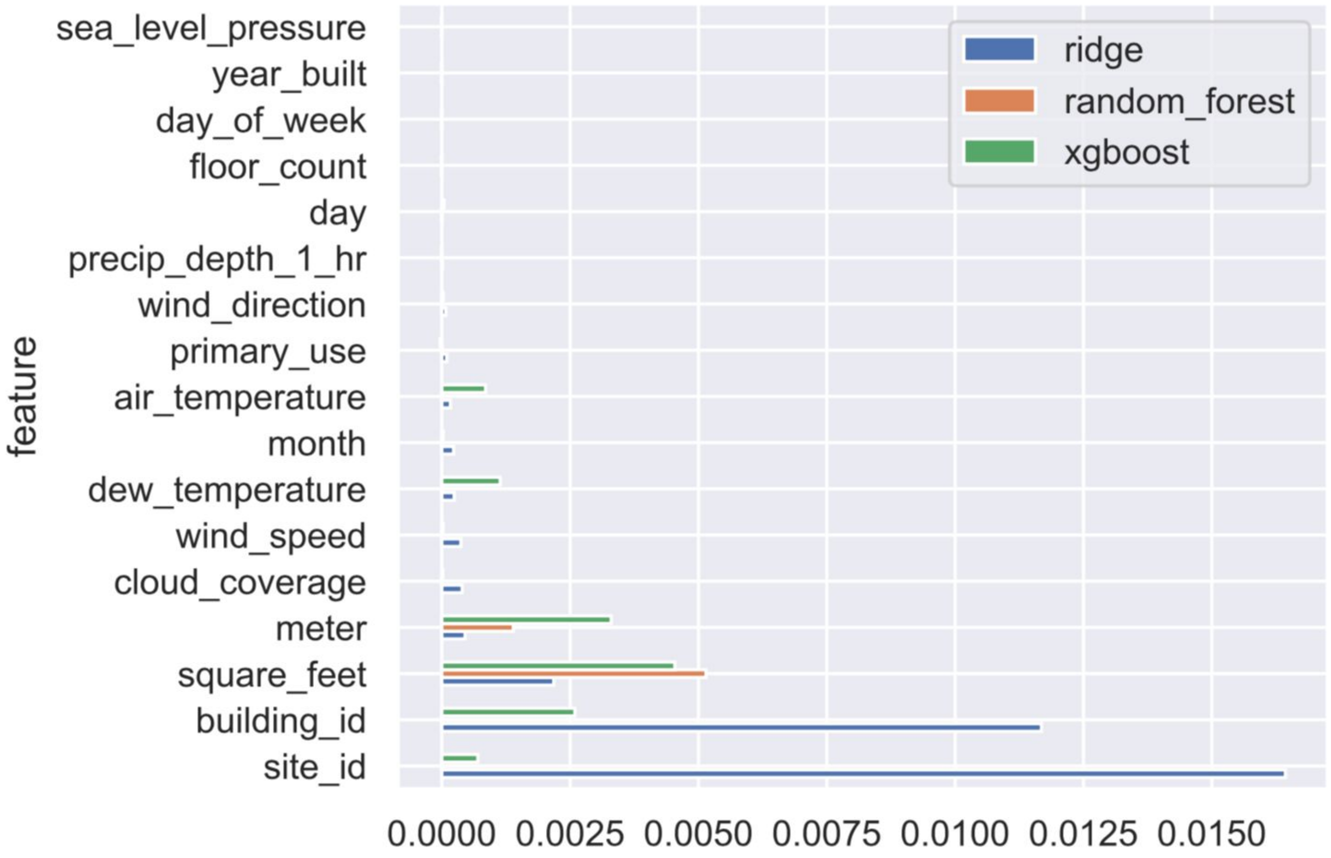
```python
def huber_approx_obj(train, preds):
    """
    Function returns gradient and hessein of the Pseudo-Huber function.
    """
    d = preds - train
    h = 1  ## constant
    scale = 1 + (d / h) ** 2
    scale_sqrt = np.sqrt(scale)
    grad = d / scale_sqrt
    hess = 1 / scale / scale_sqrt
    return grad, hess
```

```python
## define huber loss - minimizing it means maximizing its negative
def huber_loss(preds, train):
    d = preds - train
    h = 1
    return -1 * np.sum(np.sqrt(1 + (d/h)**2) - 1)
```

XGBoost Model’s First Decision Tree

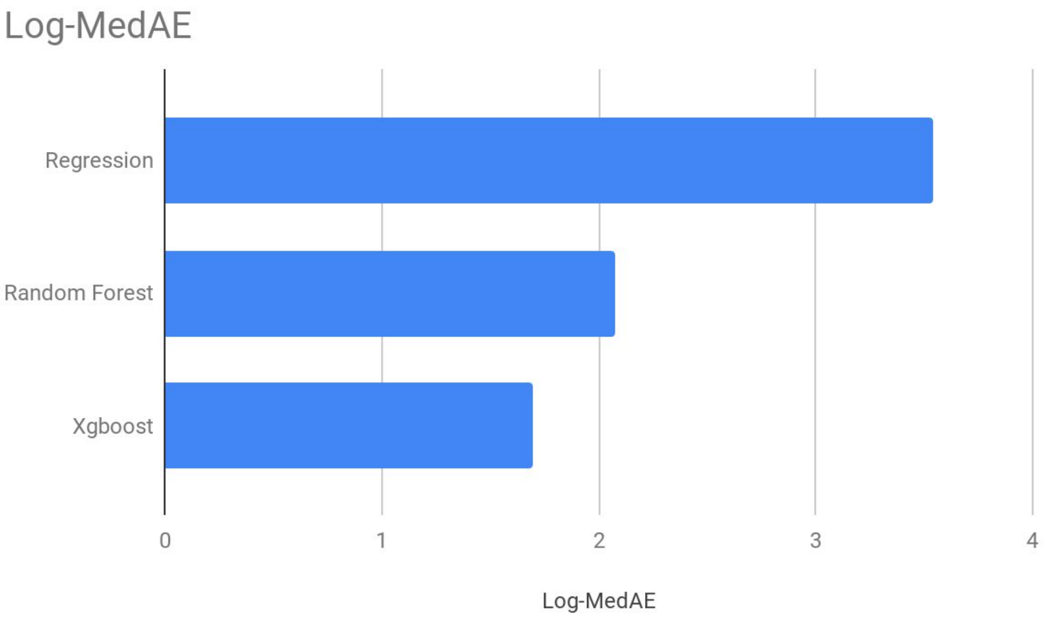


Permutation Feature Importance



Model Performance

Metrics	XGB	RandomForest	Linear Regression
MedAE	52	136	3640

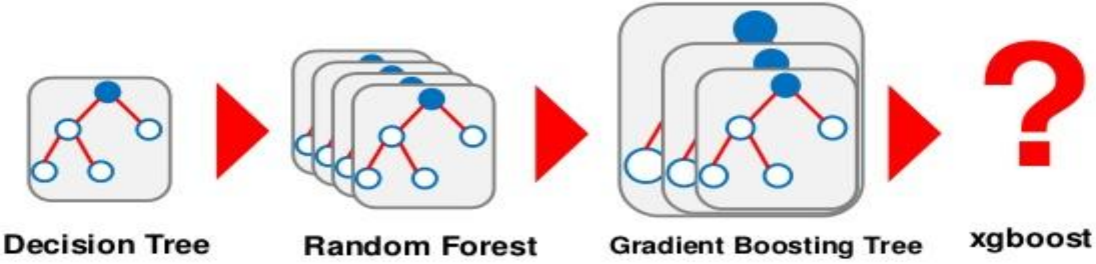


Model - Selection

We chose XGBoost as our final model

Why is XGBoost better than RF?

Gradient boosting builds trees sequentially, where each new tree helps to reduce errors made by previously trained tree



Impact

- Our predictions will allow for more accurate energy payments from buildings under the “pay for performance” payment setup

- Building managers will have a better idea of the energy consumed by their buildings. Seeing the huge cost associated with it, managers will look towards energy efficient devices.

Limitations

- Granularity : We only predict mean hourly power consumption by aggregating hourly level data at a daily level
- Scalability: difficult to scale for new building ids

Summary

- We were able to predict the mean hourly energy consumption of a building, off by 52 Kwh most of the times.

Next Steps

- Calculate lag based features (ex. precipitation last 24 hours) and see if they have additional predictive power
- Improve accuracy by trying out more advanced models (ex. deep learning models)