Power consumption prediction using Neural Networks, Decision Trees, Forest & Boosting Algorithms

MECE 4520 Final Project: Data Science for Mechanical Systems

Columbia University

By

Ali Danish, Avikal Sagar, Pei-Yu Chang, Peter Wu, Qixiao Zhang, Shevi Wolvovsky

Introduction

1. Power Consumption (PC) plays a critical role in a global economy due to the imbalance between energy production and demand.

2. Machine Learning and Deep Learning models have been widely used in PC predictions to help energy managers to control power systems better and improve energy usage.

3. In our project, we used **LSTMs (Neural Networks)**, **Decision Trees** and **Random Forests** and **Boosting algorithms** in order to achieve the same.

4. Power consumption from 3 power zones/stations were considered in **Tetouan (Morocco)**

Data Preprocessing

1. Imported the dataset from UCI Machine Learning repository [Link]

2. Converted and broke down ‘Datetime’ column (time series index) into - hours, minutes, quarter, day of week, day of month, day of year and month (time series features).

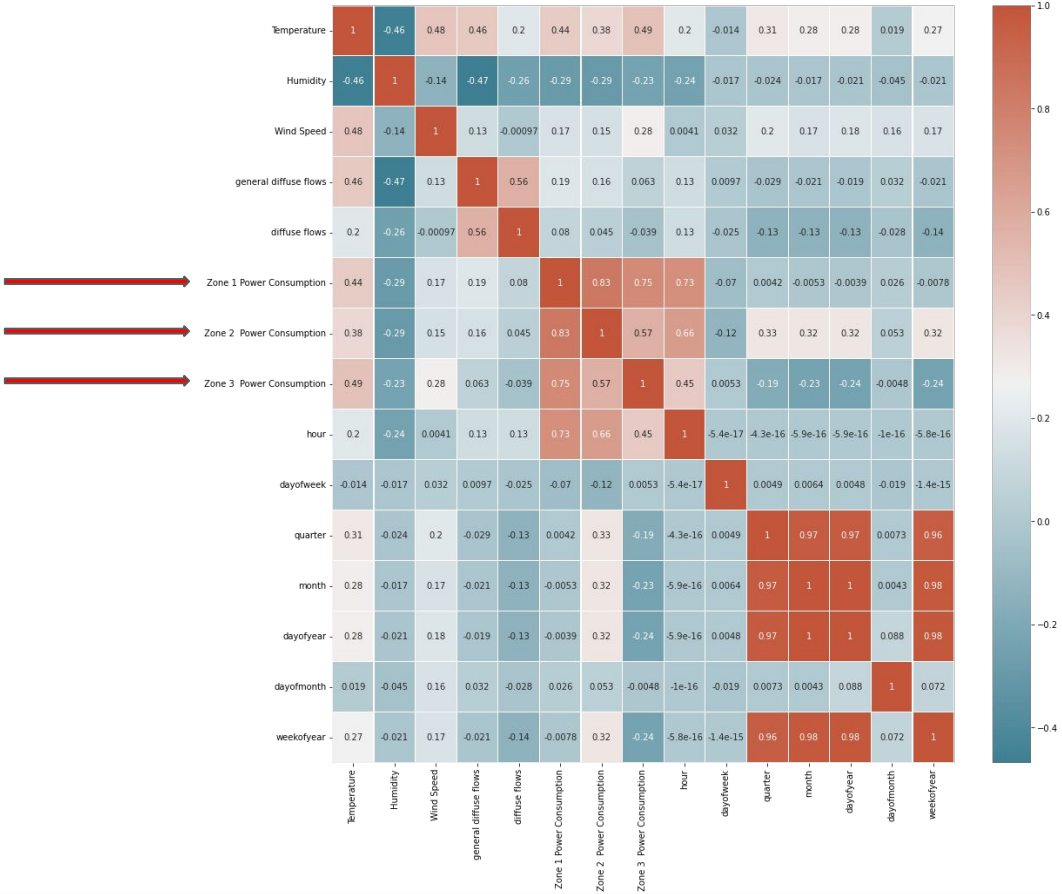
3. Plotted correlation matrix, boxplots and scatterplots (data visualization) to understand feature importance.

Our initial dataset

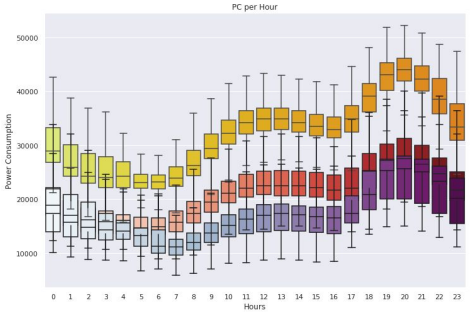
	DateTime	Temperature	Humidity	Wind Speed	general diffuse flows	diffuse flows	Zone 1 Power Consumption	Zone 2 Power Consumption	Zone 3 Power Consumption
0	1/1/2017 0:00	6.559	73.8	0.083	0.051	0.119	34055.69620	16128.87538	20240.96386
1	1/1/2017 0:10	6.414	74.5	0.083	0.070	0.085	29814.68354	19375.07599	20131.08434
2	1/1/2017 0:20	6.313	74.5	0.080	0.062	0.100	29128.10127	19006.68693	19668.43373
3	1/1/2017 0:30	6.121	75.0	0.083	0.091	0.096	28228.86076	18361.09422	18899.27711
4	1/1/2017 0:40	5.921	75.7	0.081	0.048	0.085	27335.69620	17872.34043	18442.40964
...	...	...	...	...	...	...	...	...	...
52411	12/30/2017 23:10	7.010	72.4	0.080	0.040	0.096	31160.45627	26857.31820	14780.31212
52412	12/30/2017 23:20	6.947	72.6	0.082	0.051	0.093	30430.41825	26124.57809	14428.81152
52413	12/30/2017 23:30	6.900	72.8	0.086	0.084	0.074	29590.87452	25277.69254	13806.48259
52414	12/30/2017 23:40	6.758	73.0	0.080	0.066	0.089	28958.17490	24692.23688	13512.60504
52415	12/30/2017 23:50	6.580	74.1	0.081	0.062	0.111	28349.80989	24055.23167	13345.49820

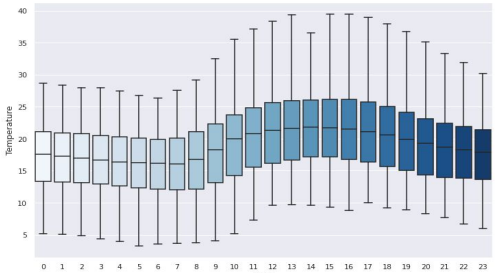
52416 rows × 9 columns

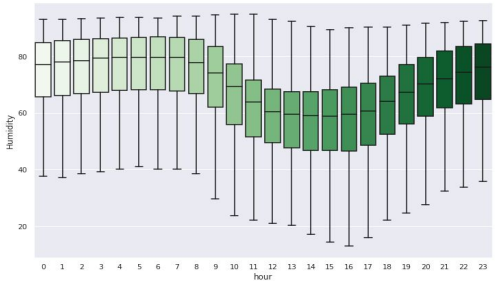
Feature importance: Correlation Matrix

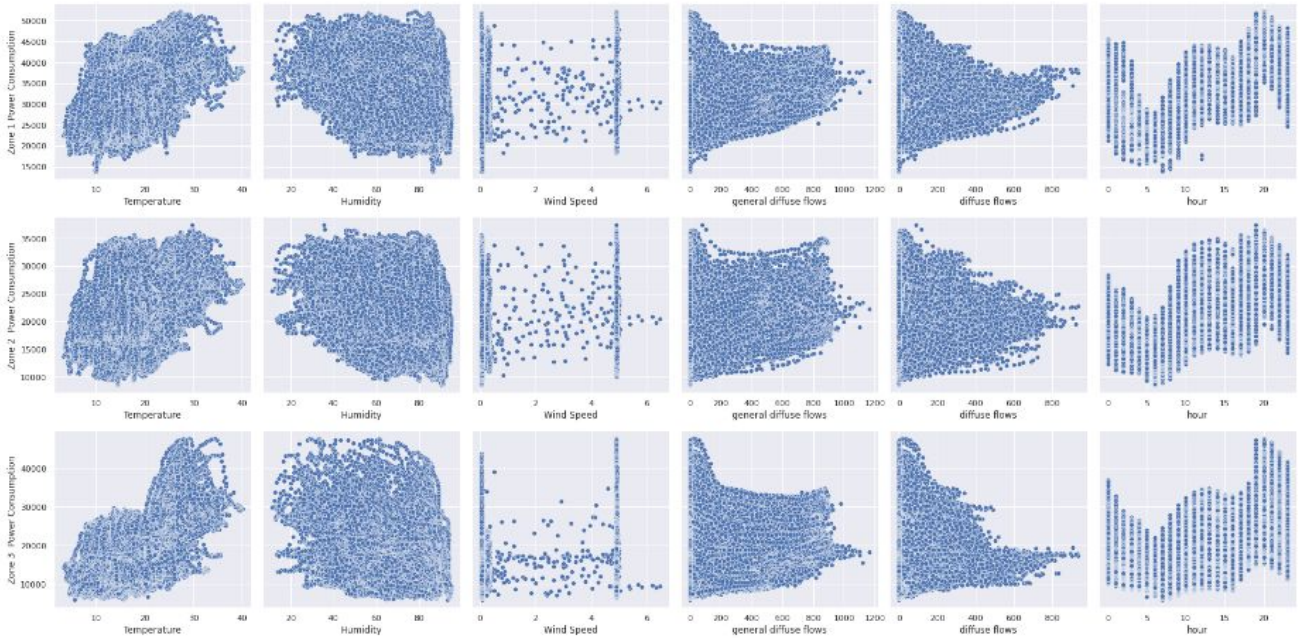


Feature importance: Data Visualization



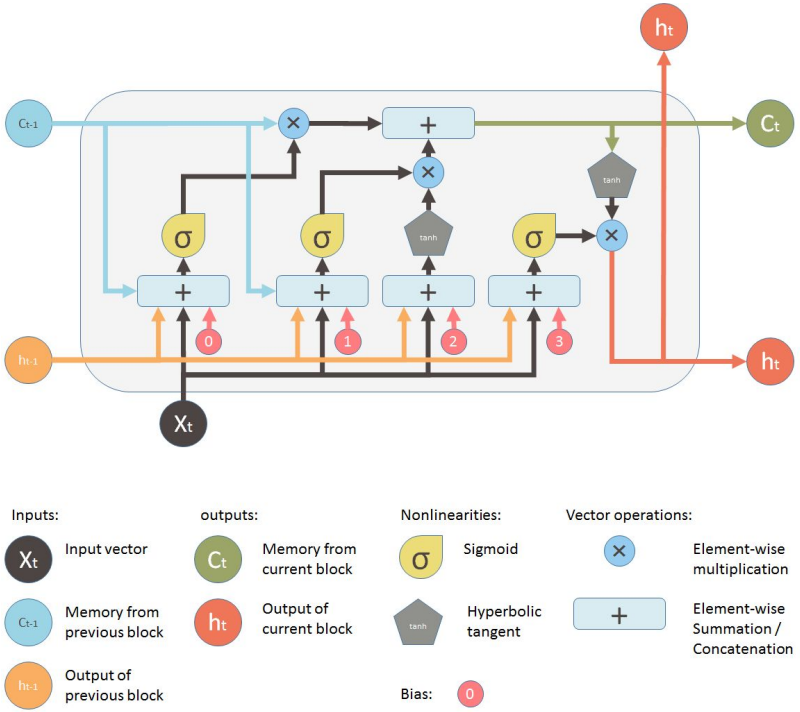






Model #1: Stacked LSTM (Neural Network)

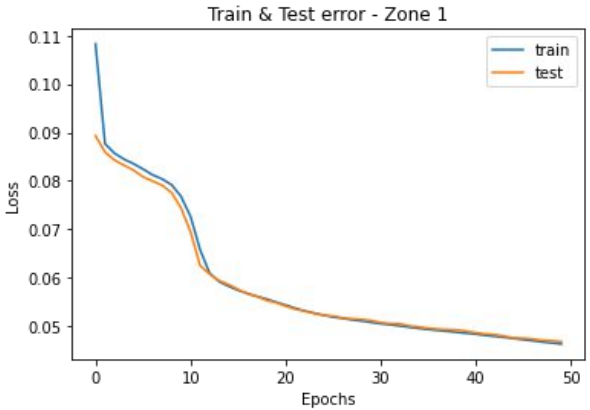
1. Neural Network structure used: Stacked LSTM
   a. Input Layer
   b. Hidden Layers:
      i. **LSTM with 128 blocks**
      ii. **LSTM with 64 blocks**
   c. Output Layer: Dense layer with 1 neuron
2. Model Hyperparameters:
   a. Batch Size: 256
   b. Epochs: 50
   c. Train-test split: 70-30
   d. Optimizer: Adam
   e. Loss for training and validation: MAE
   f. Dropout: 0.2
   g. Activation for the hidden layers: ReLU
   h. Activation for the output layer: Sigmoid

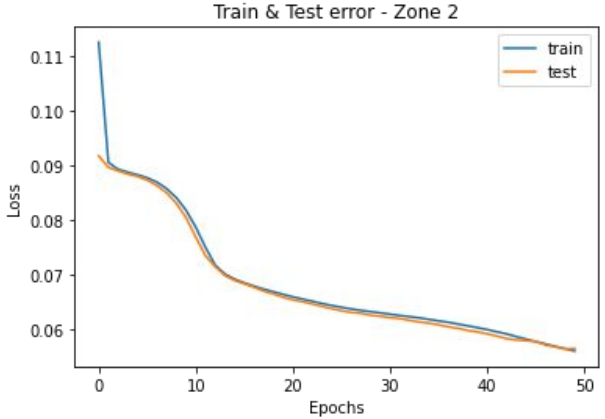


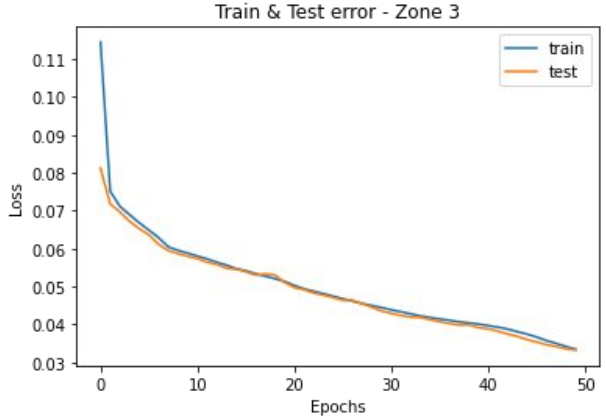
Model #1: Stacked LSTM (Neural Network)

Accuracy measurement:

Zone	Mean Absolute Error (MAE)	Root Mean Square Error (RMSE)	R^2
1	1953.011	2715.579	0.878
2	4206.401	5126.158	0.533
3	2291.988	3051.765	0.811





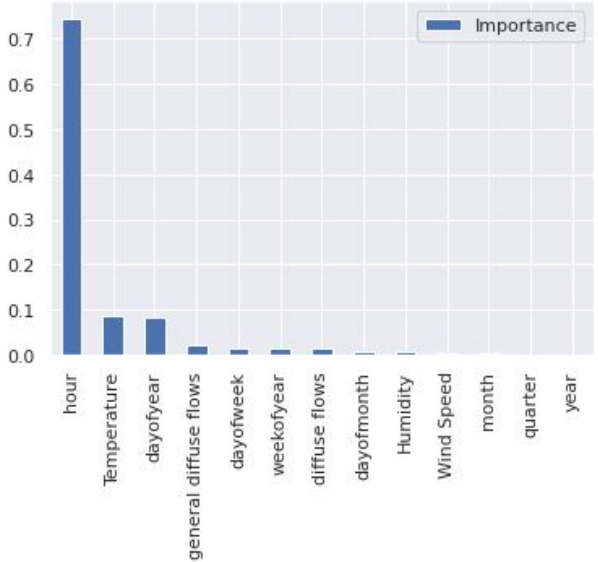


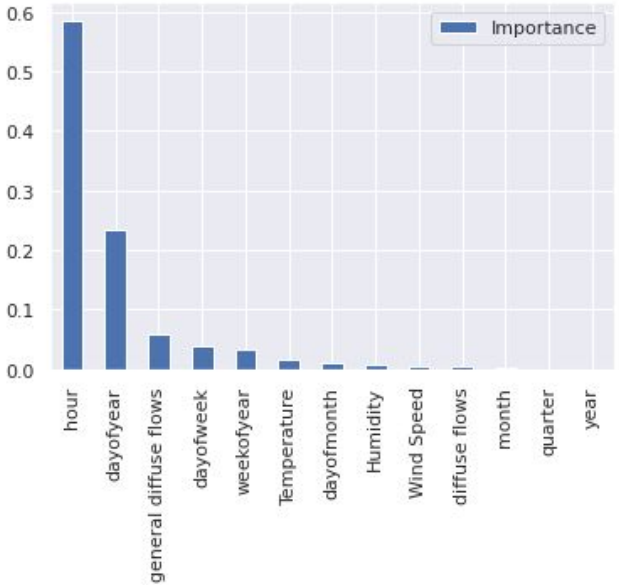
Model #2: Decision Trees

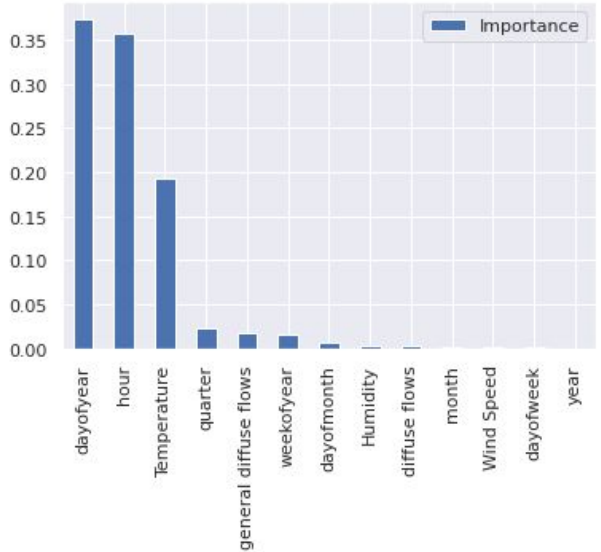
Hyperparameters used:

1. Criterion: Squared Error
2. Max Depth: 5

Feature importance of Zone 1, 2, and 3 (From left to right):







Model #2: Decision Trees

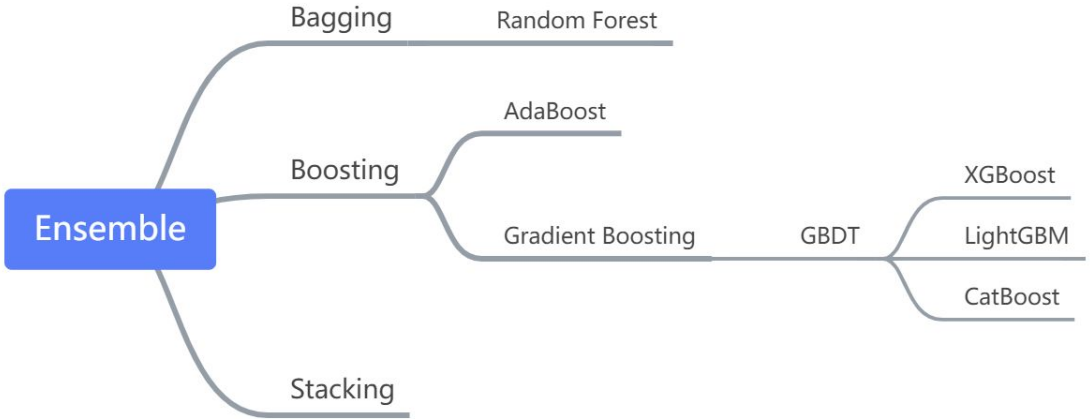
Accuracy measurement:

Zone	MAE	RMSE	R^2
1	2031.663	2733.539	0.854
2	1702.129	2246.524	0.812
3	1722.088	2429.634	0.867

Model #3: Random Forests & Boosting Algorithms

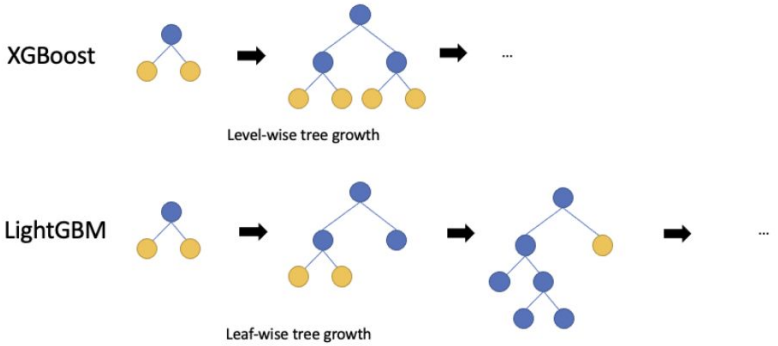
Ensemble methods:

1. Random Forest: Bagging
2. XGBoost: boosting(Gradient-boosting decision tree)
3. LightGBM: boosting(Gradient-boosting decision tree)
4. CatBoost: boosting(Gradient-boosting decision tree)



Difference within boosting models:

1. XGBoost: level-wise
2. LightGBM: leaf-wise
3. CatBoost: Symmetric trees



Model #3: Random Forests & Boosting Algorithms

Hyperparameters tuning (grid search)

1. **XGBoost:** learning_rate=0.015, n_estimators=1000, silent=True, nthread=1, max_depth=16
2. **Random Forest:** n_estimators = 1000, oob_score = True, n_jobs = 1, random_state =1
3. **Light GBM:** max_depth=16, min_child_samples=0, n_estimators=4000, subsample_for_bin=2000000
4. **Catboost:** loss_function='RMSE', iterations=600, learning_rate=0.5, depth=8

Accuracy measurement (for zone-1, which can also be extended to other zones):

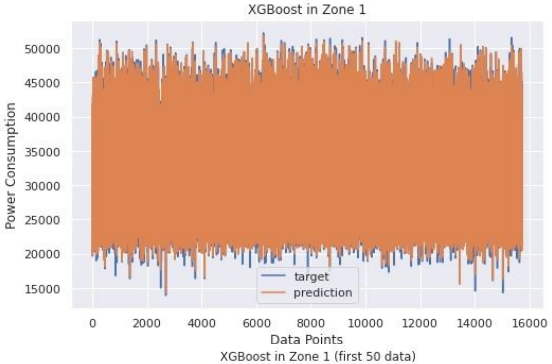
Model	RMSE	R^2
XGBoost	950.953	0.982
Random Forest	1045.304	0.978
Light GBM	1062.366	0.977
CatBoost	1108.793	0.975

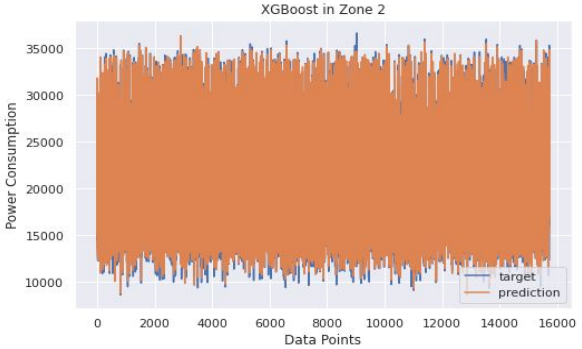
Summary Table

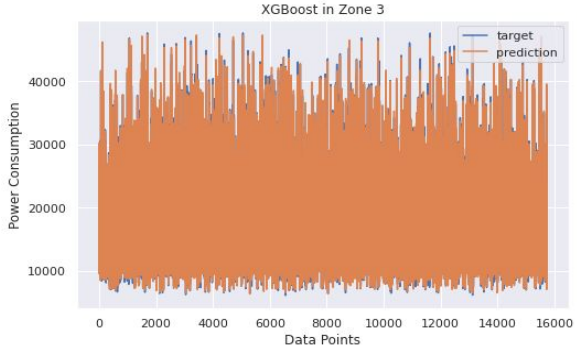
Methods	Neural Network	Decision Tree	Ensemble Learning
Best R^2	0.878	0.867	0.982

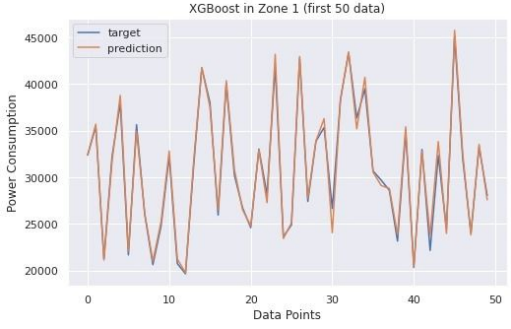
Hence, our best fit model is: **XGBoost**

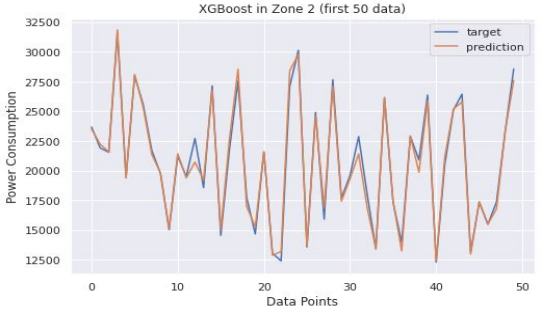
Final Predictions

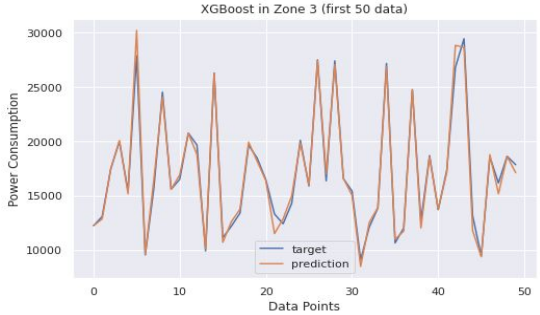












Conclusion

- Boosting algorithms (like XGBoost) are certainly more accurate than deep learning algorithms.

➔ We suspect that deep learning works better on homogenous datasets, but since ours is heterogeneous, boosting algorithms perform better as tree-based methods treat features independently of each other and build rules based on the values for those features.

Future Development

- Every Data Scientists' nightmare: Is our model overfit?
- LSTM: Implement other architectures like vanilla/bidirectional LSTM or even GRUs
- Trees: Play around with hyper parameters such as criterion and max depth for improvement
- Unsupervised learning: Try k-means algorithms

THANK YOU!

ANY QUESTIONS?