

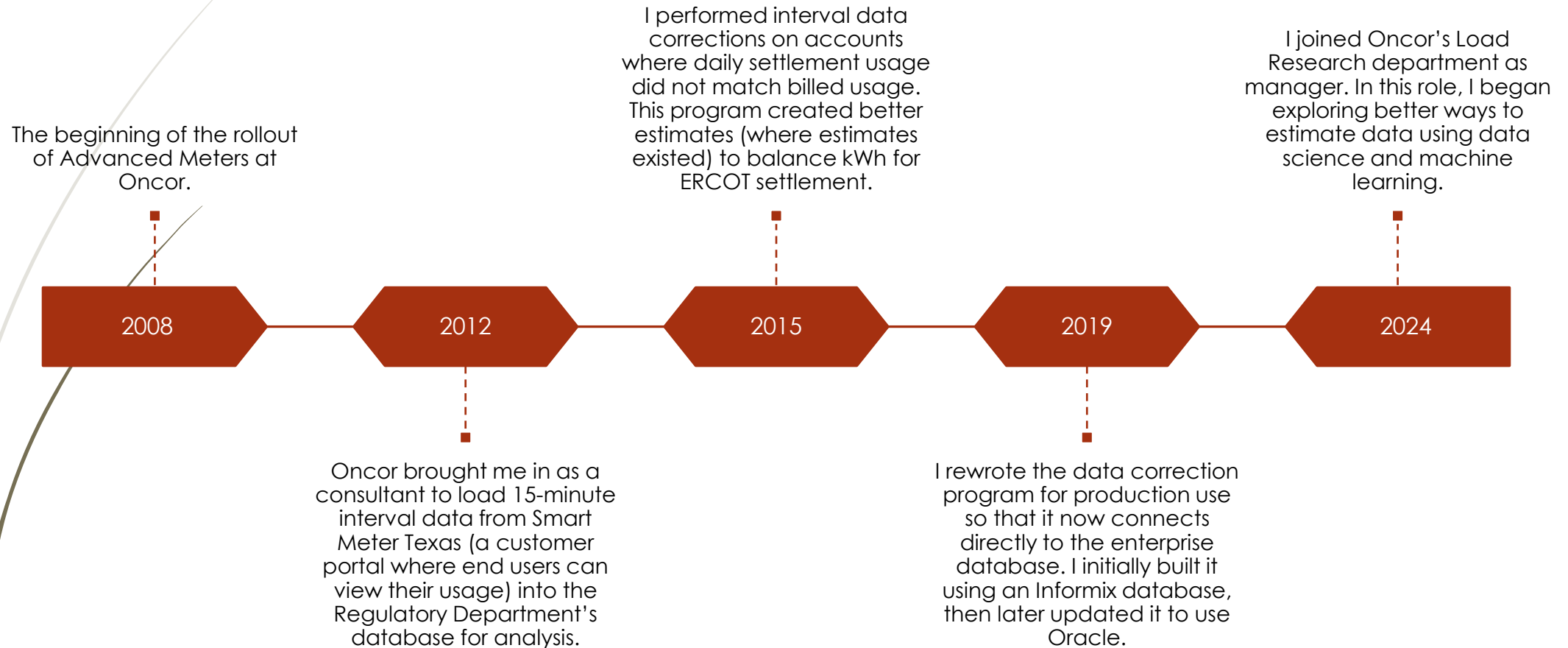


A Hitchhiker's Guide to Machine Learning

Jay Newman: Load Research Manager

Oncor Electric Delivery

My Interval Data Journey



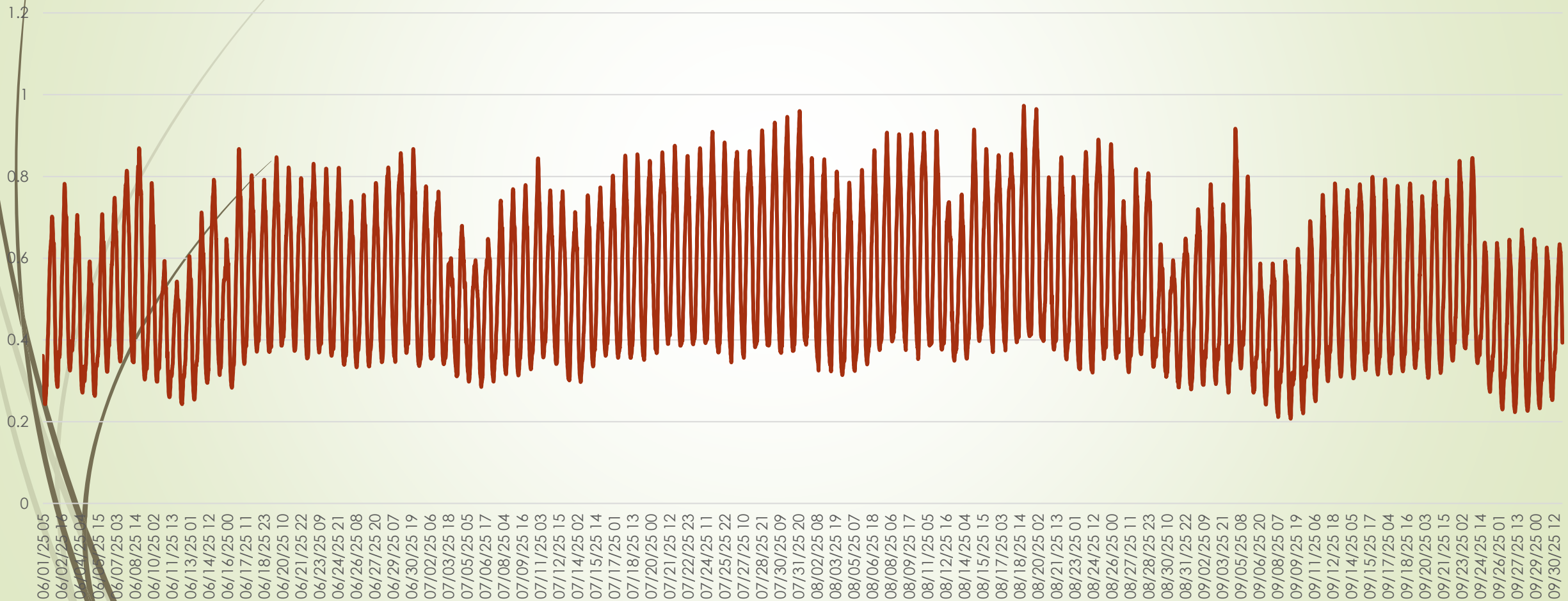


Fix-estimate (version 1)

- In Texas, premises' daily usage is reported to ERCOT every day to settle the wholesale energy market.
- Customers are billed every month using two midnight register reads for the usage and/or demands.
- I was tasked to create an estimation correction to match the daily kWh used for wholesale settlements at ERCOT to Oncor's billed kWh. The program was dubbed Fix-estimate.
- To do the estimations, I was pointed toward ERCOT load profile data. These profiles are available publicly on the site, and kWh values are recorded on a 15-minute interval basis.
- The variation between days is seasonal for almost all of the profiles. For business profiles, usage will also vary on weekdays and weekends.
- Here is a look at a few of them...

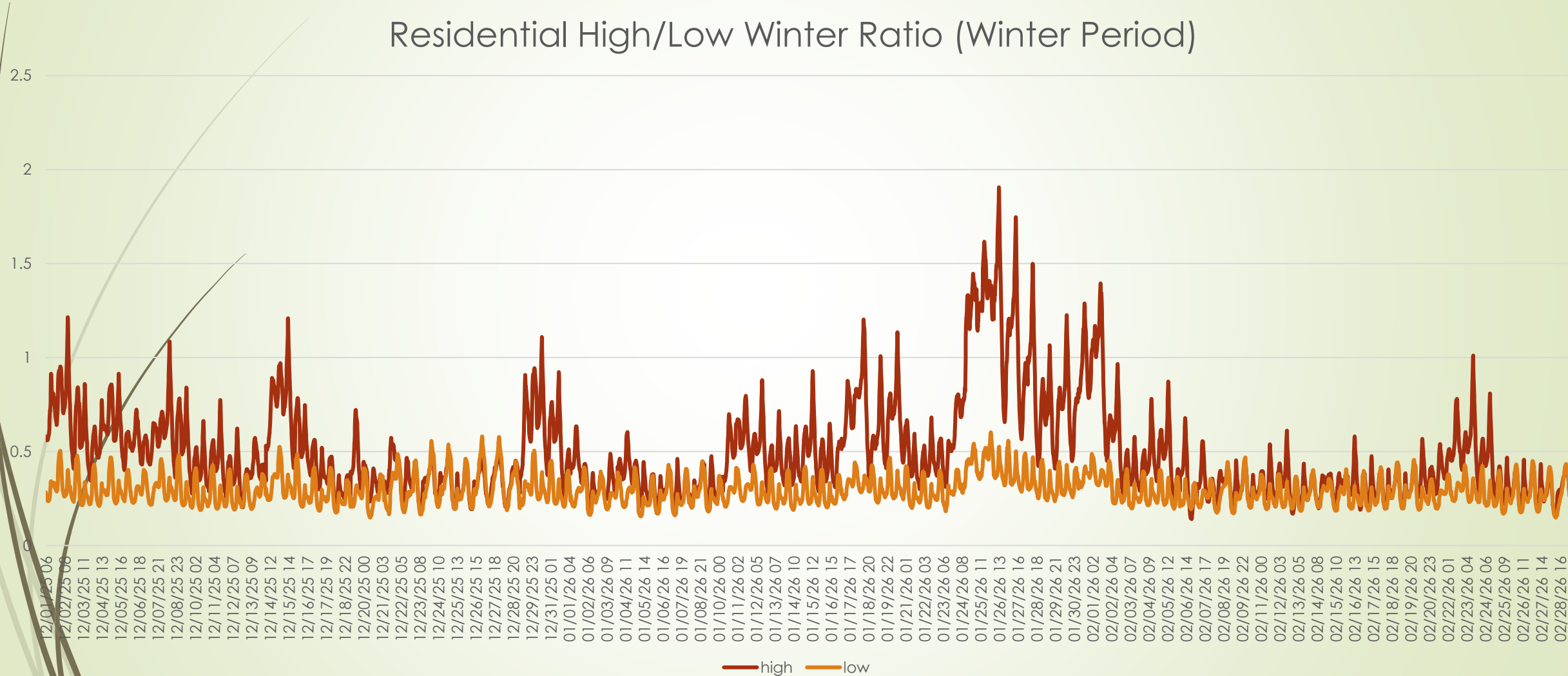
ERCOT Residential Profiles

ERCOT Residential Summer kWh Profile (Same for High and Low Winter Ratio)



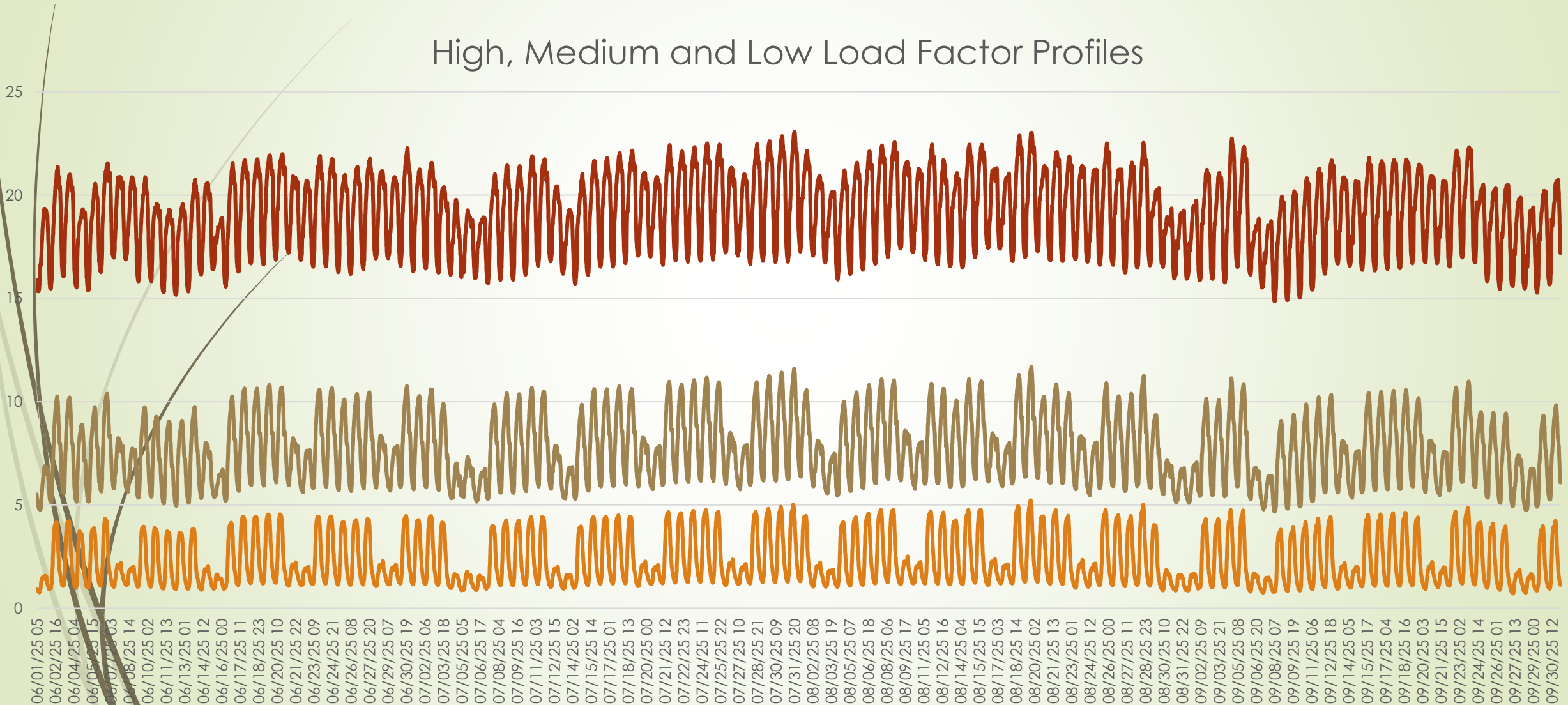
ERCOT Residential Profiles (cont.)

Residential High/Low Winter Ratio (Winter Period)

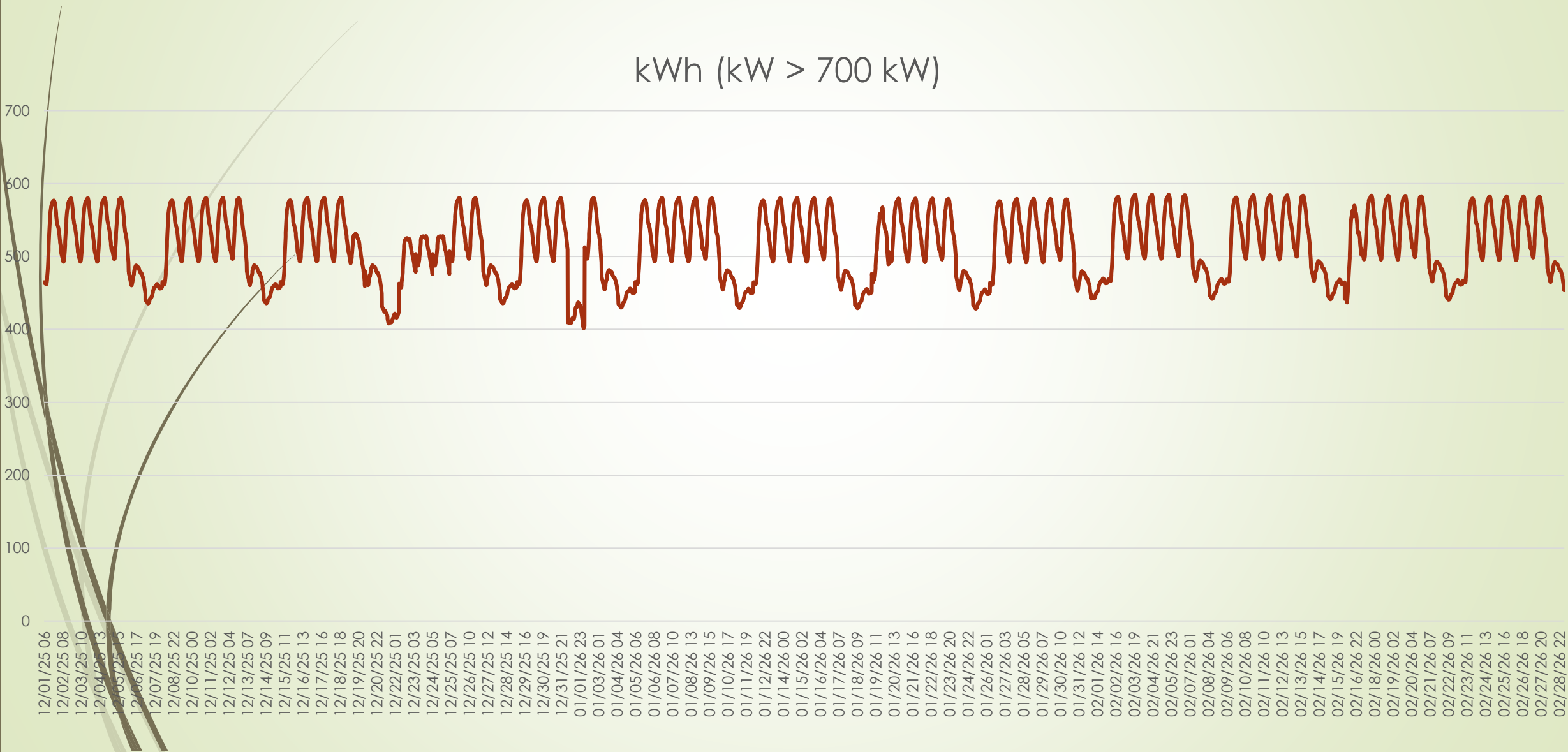


ERCOT Business Profile (11 kW-700 kW)

High, Medium and Low Load Factor Profiles



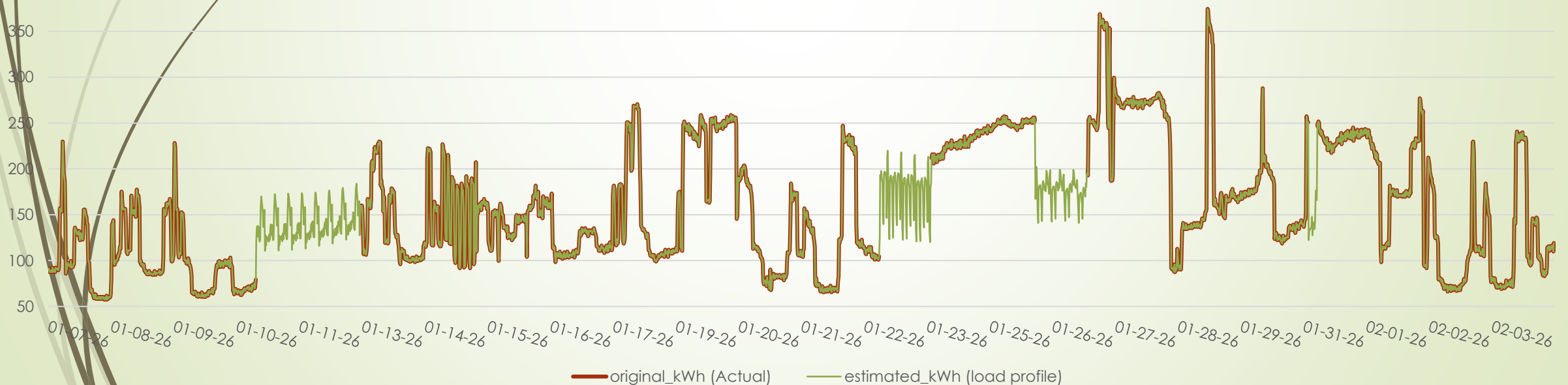
ERCOT Large Business Profile



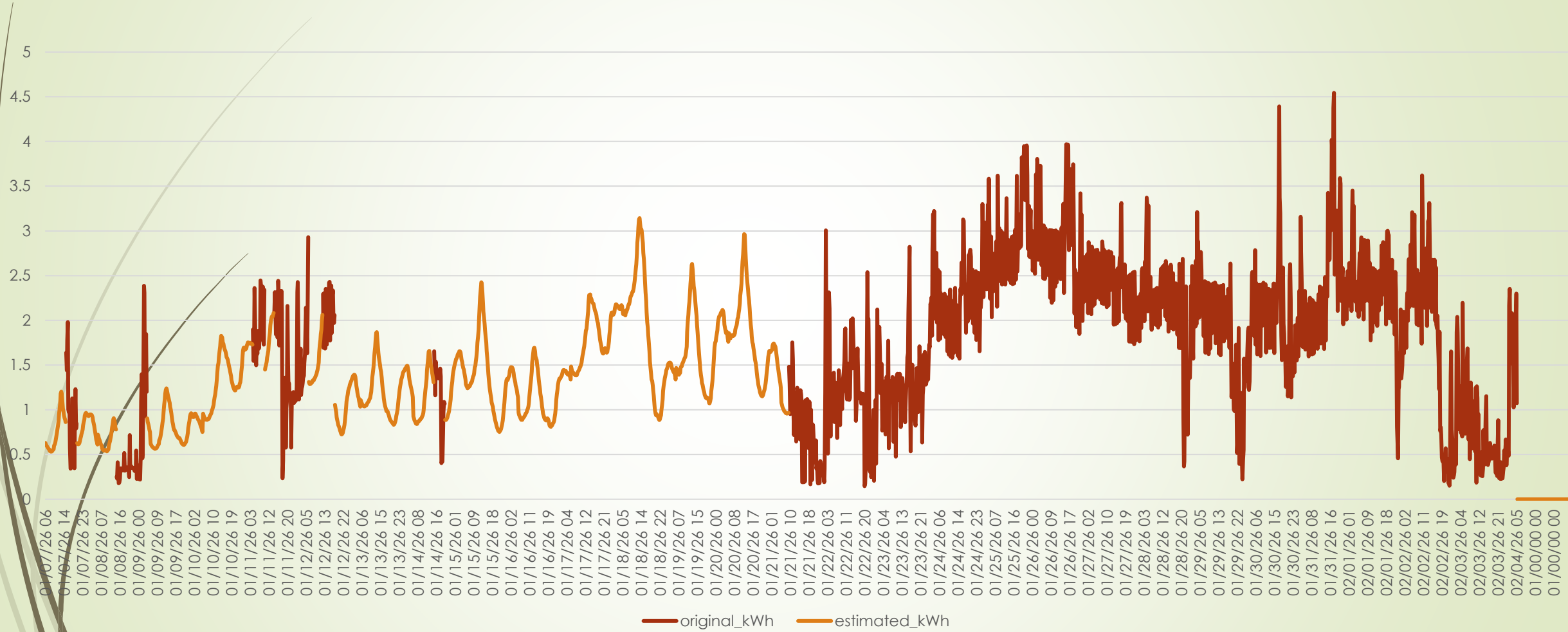
Fix-estimate (ERCOT Load Profile Method)

- I started the project using the ERCOT load profile data to estimate intervals, similar to how our Meter Data Management System did for almost all of our customers. Every day has different values (shown in the example below, a large business profile).

Fix-estimate ERCOT Model Profile Method

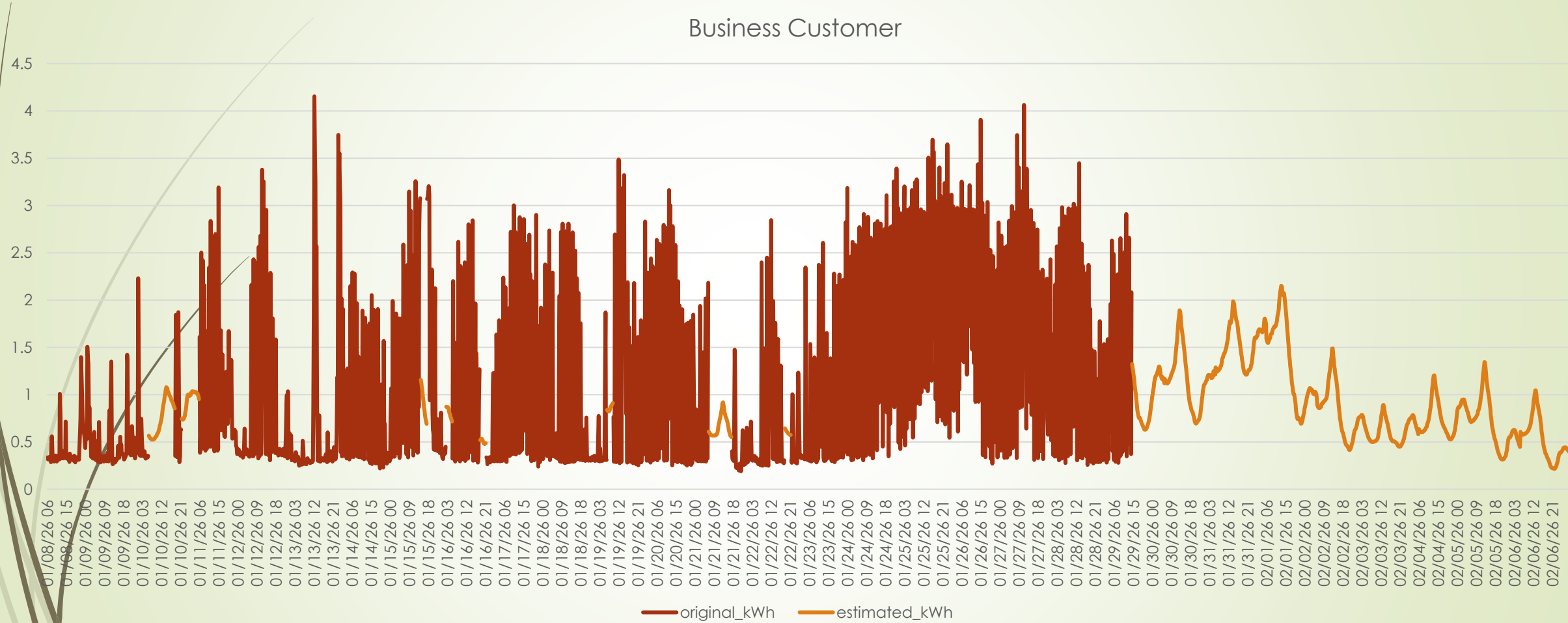


ERCOT Load Profile Example 2

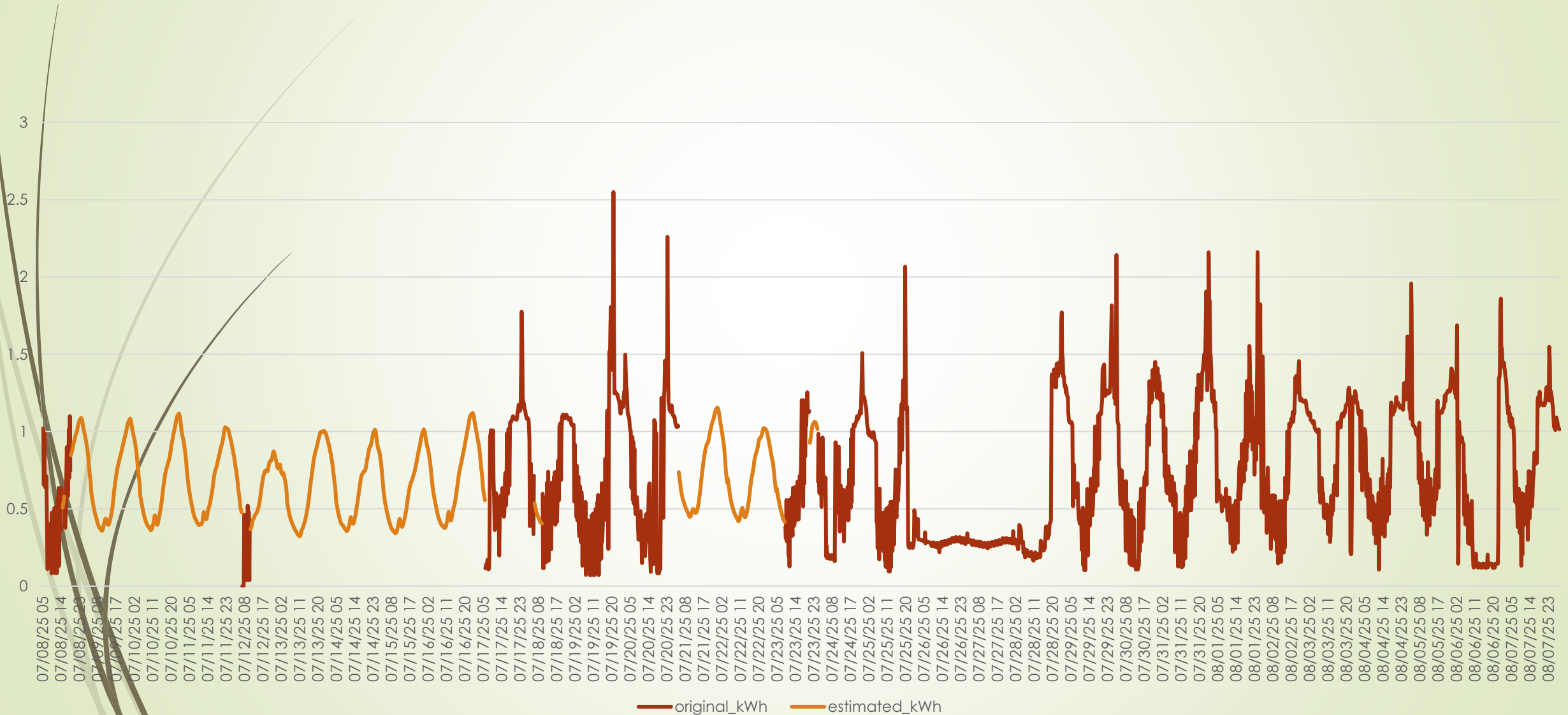


ERCOT Load Profile Example 3

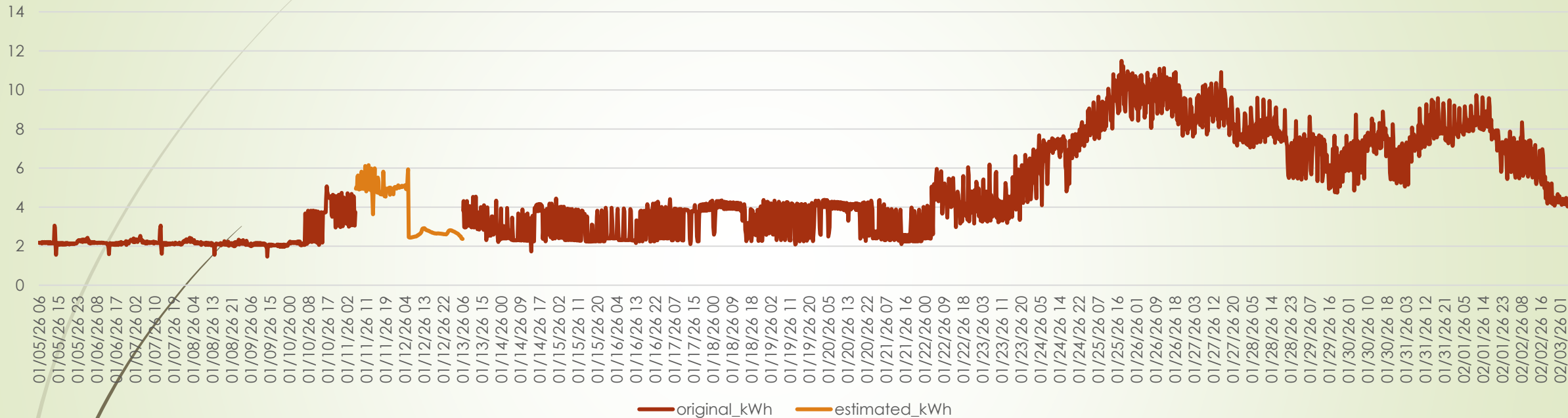
Business Customer



ERCOT Load Profile Example 4



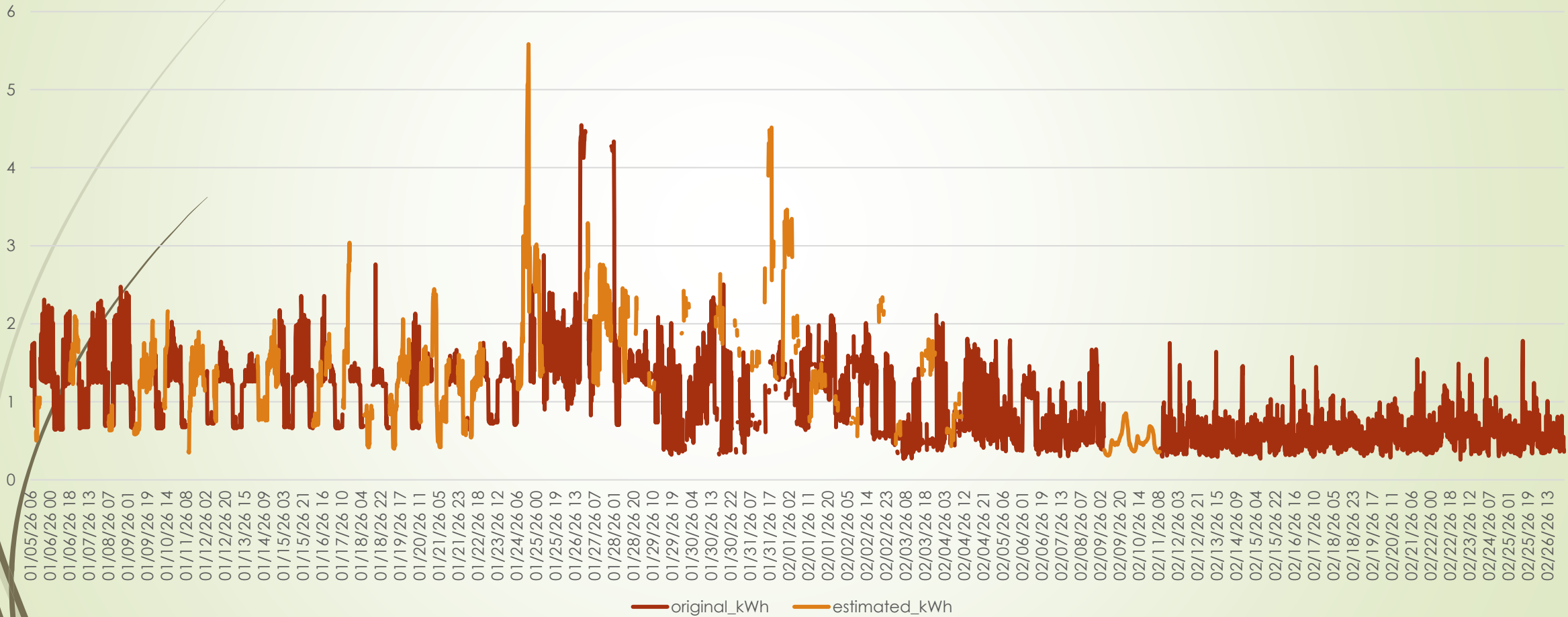
Fix-estimate (Custom Load Profile)



- The ERCOT load profile method, while technically valid, didn't completely satisfy the request. Management wanted me to write a program that would use the customer's actual load profile for a period, and then take that profile and replace values for a like day and 15-minute interval. This process required a lot of additional data and computational power to complete in a relatively short timeframe using multi-threaded tasks. Once completed, the estimated values were increased proportionally up or down until the desired total usage was accomplished.

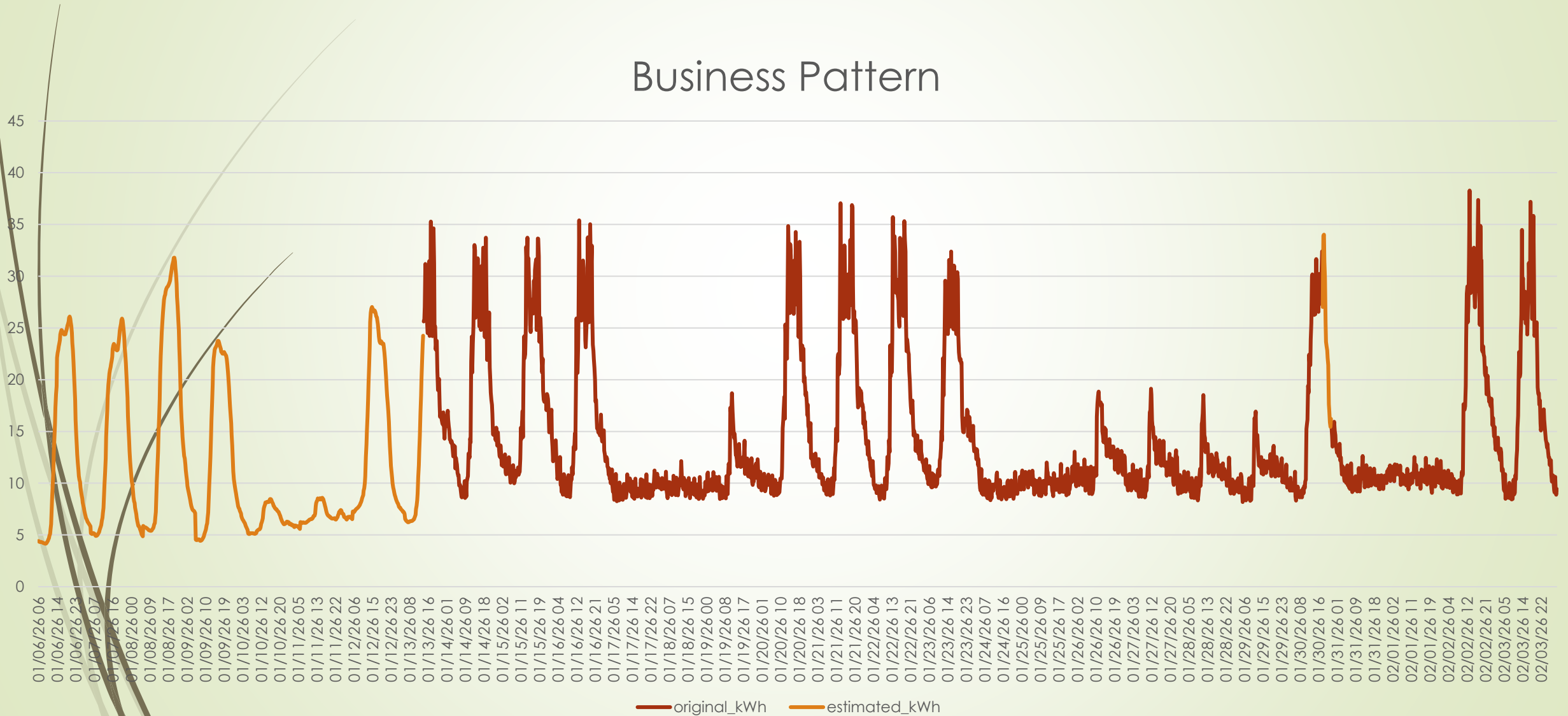
Custom Profile Example 2

Fix-estimate (Sporadic Data Estimation)



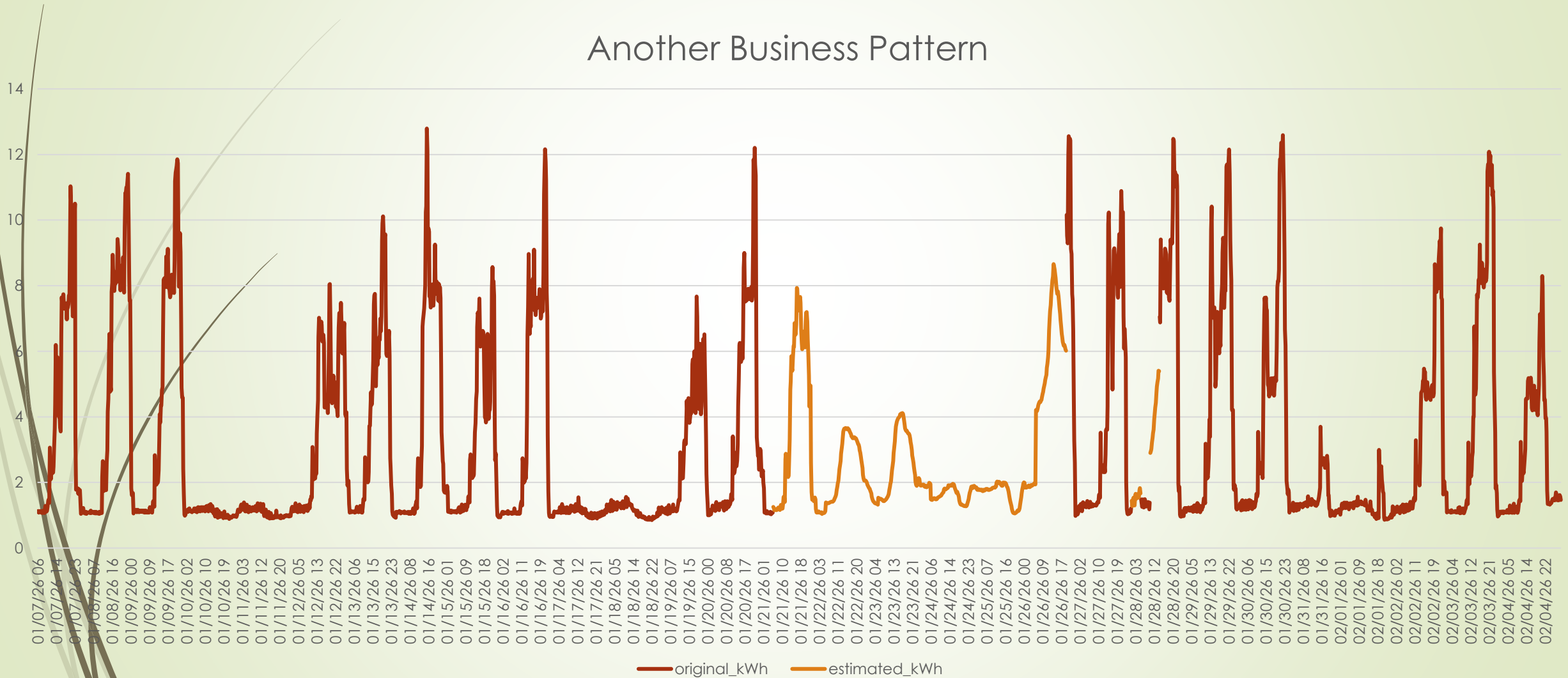
Custom Profile Example 2

Business Pattern




Custom Profile Example 3

Another Business Pattern



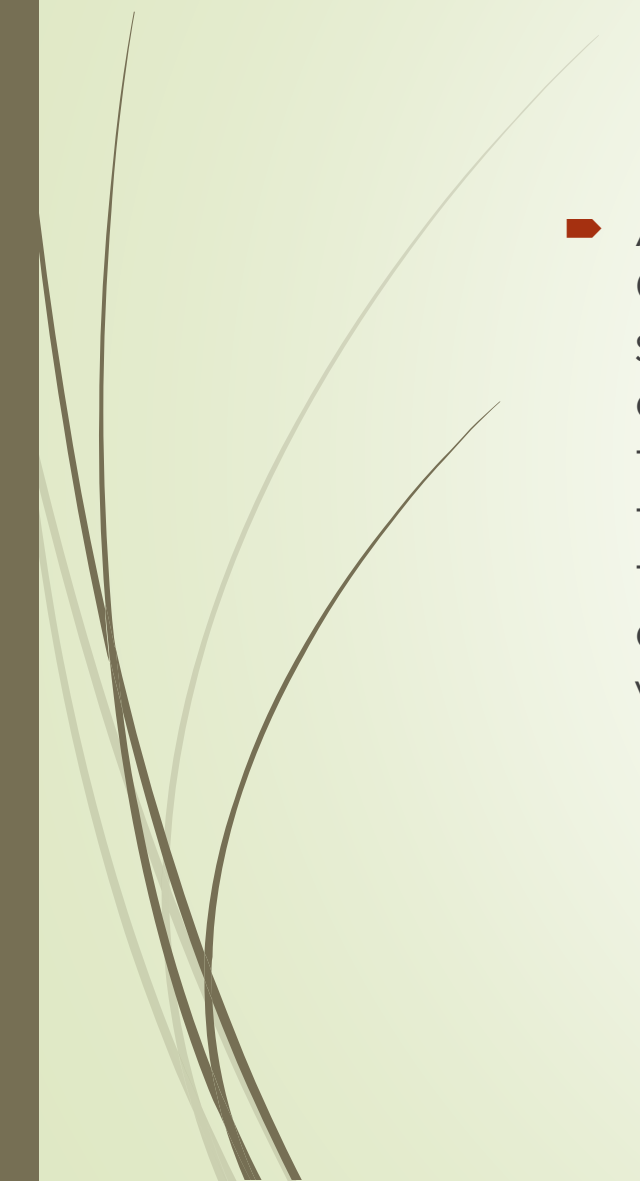


The Move to the Production Product

- Information Technology and our metering group decided to push my code into a production application server and into their daily workflow. Many challenges were addressed to use the database to calculate the estimates using complicated DDLs to do so. While the first project was done in Informix (an extremely buggy product that required a two-stage program sandwiched by database stored procedures), the Oracle version was relatively easy to convert, and in a single program that could execute as many times as required to make up time during any database outages or updates (the old program had to be fired off manually in the workflow for each day missed). It was a frustrating thing trying to match what the old program did to the new one. It took a lot out of me at the time. Finally, my work was done! Or was it?
- 




Load Research Manager

- At the end of 2024, I was hired to be the Load Research manager at Oncor. Before then, I was a firefighter who worked with many departments solving problems. I wrote many routines and interfaces used around the company, and now things were about to change. I picked up Python at that time and haven't looked back, for the most part. I have a team of three load researchers, including a data scientist, who helped me produce this program for you today. This has been some of the most fruitful work to emerge from my department since my arrival. I look forward to discussing it with you and my team.
- 



Sampling and the Trouble with Data

- 
- So here I was again, trudging through bad data, basically where I left off a few years earlier. But this time the world was very different for me. Thinking back to Fix-estimate, I could see the error of my ways. I had never even thought about conserving demand in the estimates I was creating. I was told to never change actual values, and yet there are times when that should be done. But today we have the tools to do all of this with relatively much better, more realistic-looking estimations. Today, we have Machine Learning to assist us with dazzling calculations that would be difficult to replicate otherwise.
 - I would start with the simplest tools and progress from there.

Machine Learning and Statistical Terms

► Training

A set of data used to train the model. Usually, 80% of the total data set.

► Testing

A set of data used to test the model created in testing. Usually, 20% of the total data set.

► R² Value

A score from 0 to 1 that tells us the proportion of a dependent variable's variance that is explained by independent variable(s) in a regression model.

► Mean Absolute Error (MAE)

The average absolute error between expected values and actual values.

► Root Mean Square Error (RMSE)

Similar to the MAE, but outliers are treated more harshly with this approach.

► Mean Decrease in Impurity (MDI)

Unit that determines which features perform best in a model. All features add to 1.

► Hyperparameter

These are external overarching settings, like the model's learning rate or the neural network's design depth, used to tweak the training processes used in machine learning.

Machine Learning and Statistical Terms

► Decision Tree

Used for classification and regression tasks, structured like a flowchart to model decisions and their possible consequences. It breaks down datasets into smaller subsets based on feature values reaching a leaf node (final prediction).

► Bagging

Random data subsets.

► Random Forest

A popular, versatile ensemble machine learning method that builds multiple decision trees using bagging and feature sampling, combining their results for higher accuracy and reduced overfitting.

► Boosting

An ensemble learning method that sequentially combines multiple "weak learners"- models that are only slightly better than random guessing- to create a single "strong learner" with high predictive accuracy.

► Gradient

An iterative optimization algorithm used in machine learning to minimize a loss function by updating model parameters (weights and bias) in the opposite direction of the gradient.

► Neural Network


Computational systems inspired by the human brain, using interconnected layers of nodes (neurons) to learn patterns, process data, and make predictions.

► Recurrent Neural Network

A class of neural networks designed to process sequential or time-series data by using internal memory to feed previous outputs back as inputs.



The Tools



I would like to explore with you four powerful machine learning algorithms: **K-Nearest Neighbors (KNN)**, **Random Forest**, **eXtreme Gradient Boosting (xgBoost)**, and **Long Short-Term Memory imputation (LSTM)**. I will describe them in order from the simplest to the most complex. Along the way, we will compare their strengths, limitations, and real-world suitability. Hopefully by the end, you'll see how these tools represent an evolutionary ladder in machine learning – from intuitive, rule-of-thumb methods to highly engineered ensembles and finally to deep neural architectures capable of capturing intricate patterns over time.

Stage 1: KNN

The idea is straightforward. When you're missing a value at, say, 10:30 AM on Wednesday, KNN looks for the most similar past intervals in your data and borrows their values.

By "similar" I mean things like time of day and day of the week, or whether it's a weekday or weekend. Also, $\text{COS}()$ and $\text{SIN}()$ values for cyclical numerical features (like hours, days, or months) splits into two continuous variables (i.e., Sunday to Monday gap where Monday is the first day in Python).

You turn those into features, and for each missing spot, the algorithm finds the K closest matches (like $K = 5$ to 10) and takes a weighted average of their actual kWh readings.

After KNN imputation, the load shapes look smooth and natural. Morning and evening/afternoon peaks remain realistic in the shoulder and winter months, and when I tested it on known gaps, the error was much lower than with simple averaging or linear interpolation.

It's extremely easy to implement. With Python's scikit-learn package, you can do it in just a few lines of code. You can receive good results without the need for neural networks or huge computational power.

Of course, it's no panacea. If you have a long stretch of missing data (like several days in a row), KNN struggles because it runs out of good neighbors to select from. But for the random scattered gaps that we usually see in 15-minute meter data, it's one of the quickest and most effective fixes.

So next time your electrical interval data has holes, DON'T PANIC! Let KNN ask, with the building's own history, "what were conditions like this before, what was the power usage?"

KNN

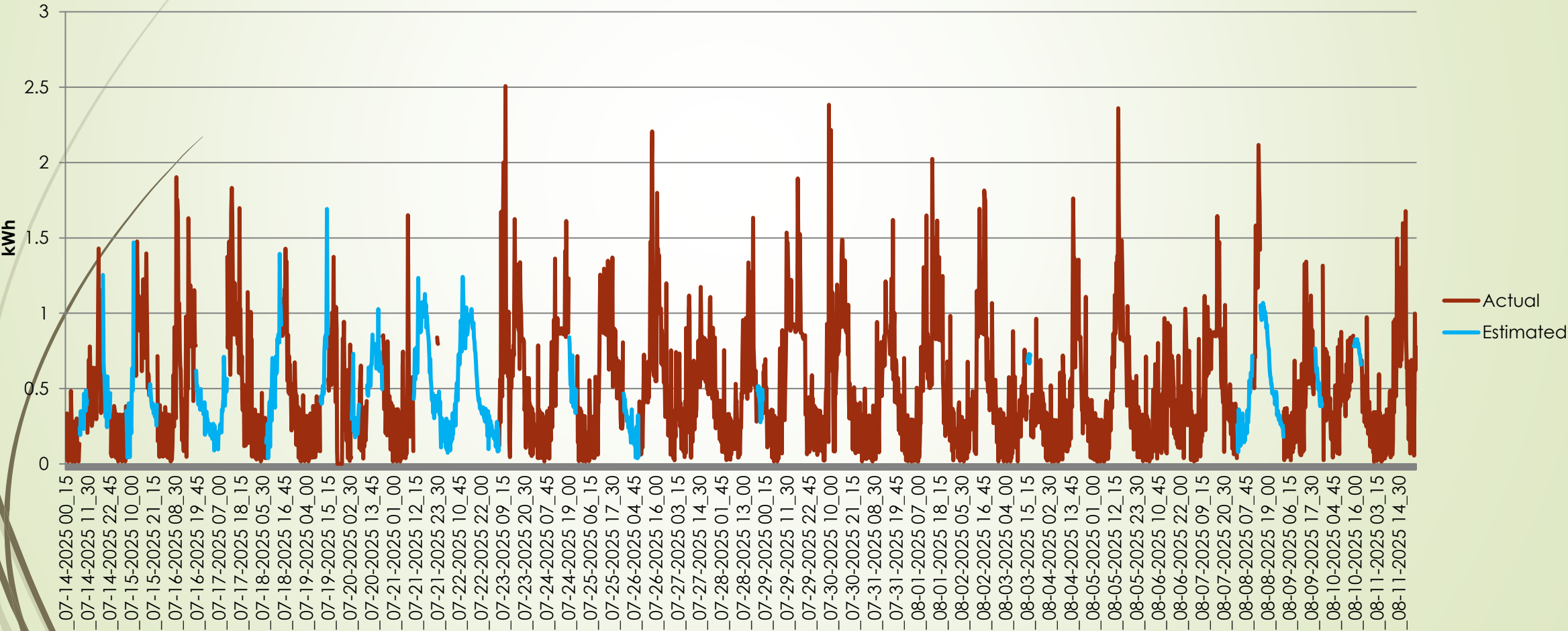
Advantages/Disadvantages/Comparison

- **Why is it the simplest for 15-min data?**
- Minimal training: It's lazy learning—just stores the data.
- Only one key hyperparameter: K (number of neighbors).
- Easy to implement and interpret: You can visualize similar load patterns.
- **Strengths in electricity forecasting:** It works reasonably well as a quick baseline for short-term predictions when you engineer good features (cyclical time encodings like hour sine/cosine). Some studies show KNN delivering solid R^2 scores (around 0.87–0.98 in certain energy datasets) because similar conditions (e.g., hot afternoon weekdays) tend to produce similar 15-min loads.
- **Weaknesses for 15-min intervals:**
- Computationally expensive at prediction time—calculating distances to millions of 15-min records is slow for real-time grid use.
- Struggles with rapid, non-stationary changes or long-term temporal dependencies; it treats each 15-min point somewhat independently.
- Sensitive to noise and scaling of features common in raw smart meter data.
- **Comparison to the others:** KNN serves as an intuitive starting point but lacks the robustness and pattern-learning depth of the other methods or the sequential memory of LSTM. It is often outperformed by tree-based ensemble models (Random Forest and XGBoost) on noisy, high-frequency electricity data.

KNN Example 1

Mode	MAE	R²	MAE_REL	RMSE_REL	RMSE	k Count
test	20.686%	30.762%	56.706%	79.478%	28.993%	65
train	0.683%	98.939%	1.457%	8.727%	4.093%	

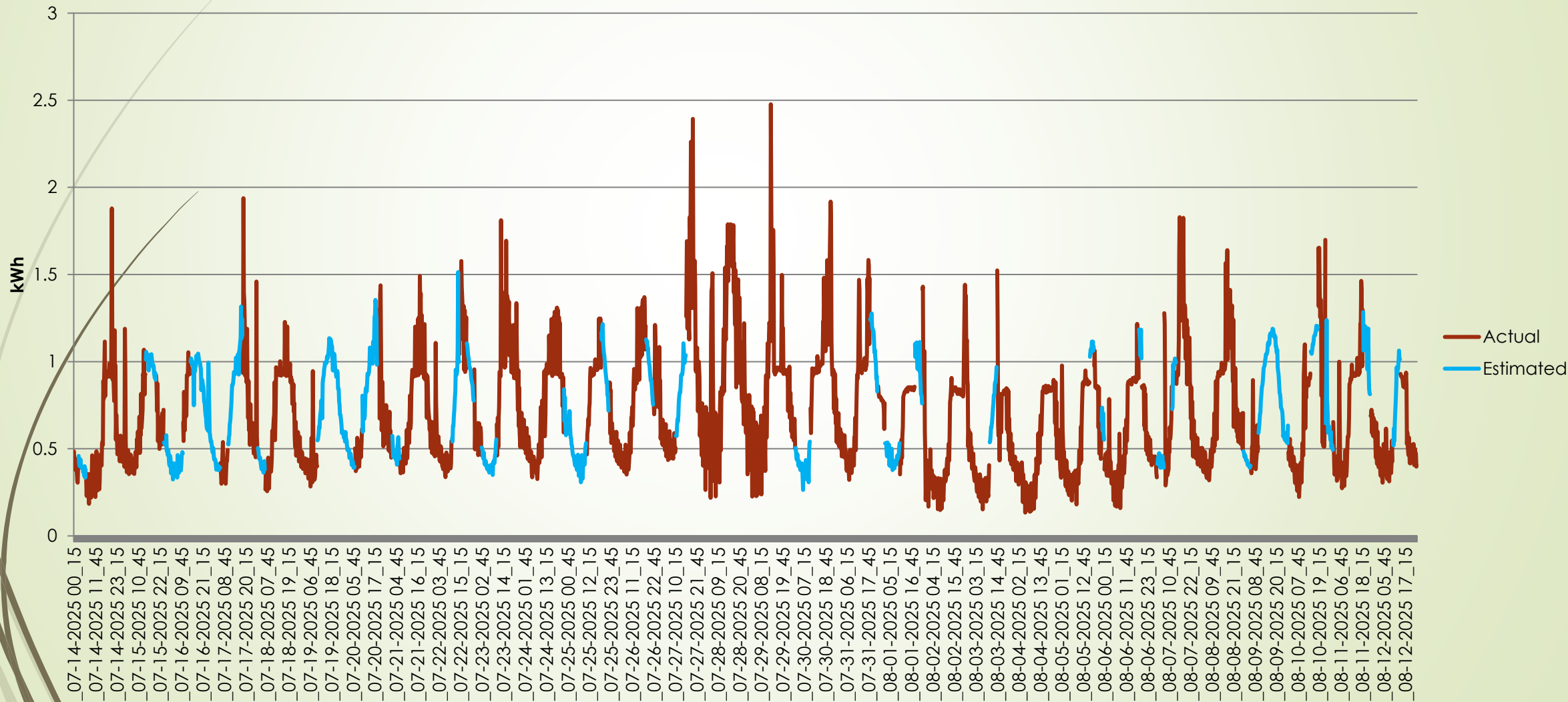
Usage



KNN Example 2

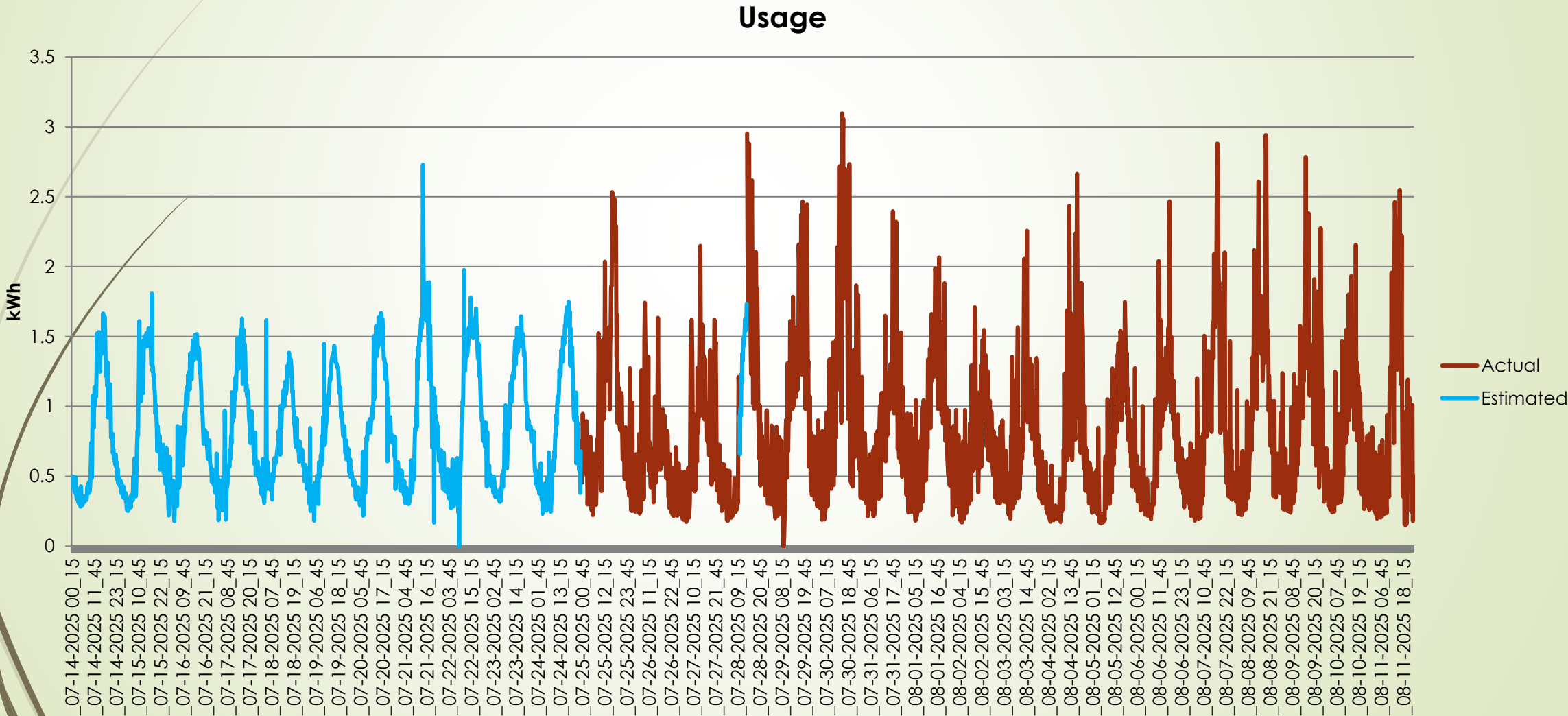
Mode	MAE	R ²	MAE_REL	RMSE_REL	RMSE	k Count
test	17.614%	37.659%	38.234%	52.971%	24.403%	80
train	0.233%	99.443%	0.362%	3.755%	2.414%	

Usage



KNN Example 3

Mode	MAE	R ²	MAE_REL	RMSE_REL	RMSE	k Count
test	27.337%	29.427%	41.948%	58.571%	38.170%	50
train	0.374%	99.664%	0.480%	4.067%	3.176%	



Stage 2: Random Forest

Random Forest is a great machine learning tool for tackling the problem of missing or estimating intervals. It learns complex patterns without making strict assumptions about the data.

Here's how it works in simple steps:

- The model makes a quick initial guess for the missing values (like the median).
- Train a Random Forest regressor using features like hour, $\sin(\text{hour})$, day of week, $\cos(\text{day of the week})$.
- Model predicts the missing or estimated 15-minute interval data.
- This process is repeated until the predictions begin to stabilize (residual losses do not fall enough to justify further training).

In Python, like KNN, it's surprisingly straightforward to put to work with the scikit-learn library.

This approach often beats traditional methods on 15-minute electrical data because it preserves realistic seasonal daily curves, morning/afternoon(in summer)/evening times, and weekend patterns, particularly seen in commercial and industrial customers.

Next time your dataset has missing intervals, skip the crude patches and grow your own Random Forest instead. You'll get cleaner data and make better decisions.

Random Forest

Advantages/Disadvantages/Comparison

How it applies to 15-min data: You typically create a tabular dataset with engineered features: previous loads (previous 1–96 intervals for a full day), rolling statistics (mean/max over last hour), time embeddings, and exogenous variables like temperature or solar irradiance. The forest then learns non-linear interactions, such as how high temperature combined with evening hours spikes residential AC load.

Key advantages over KNN:

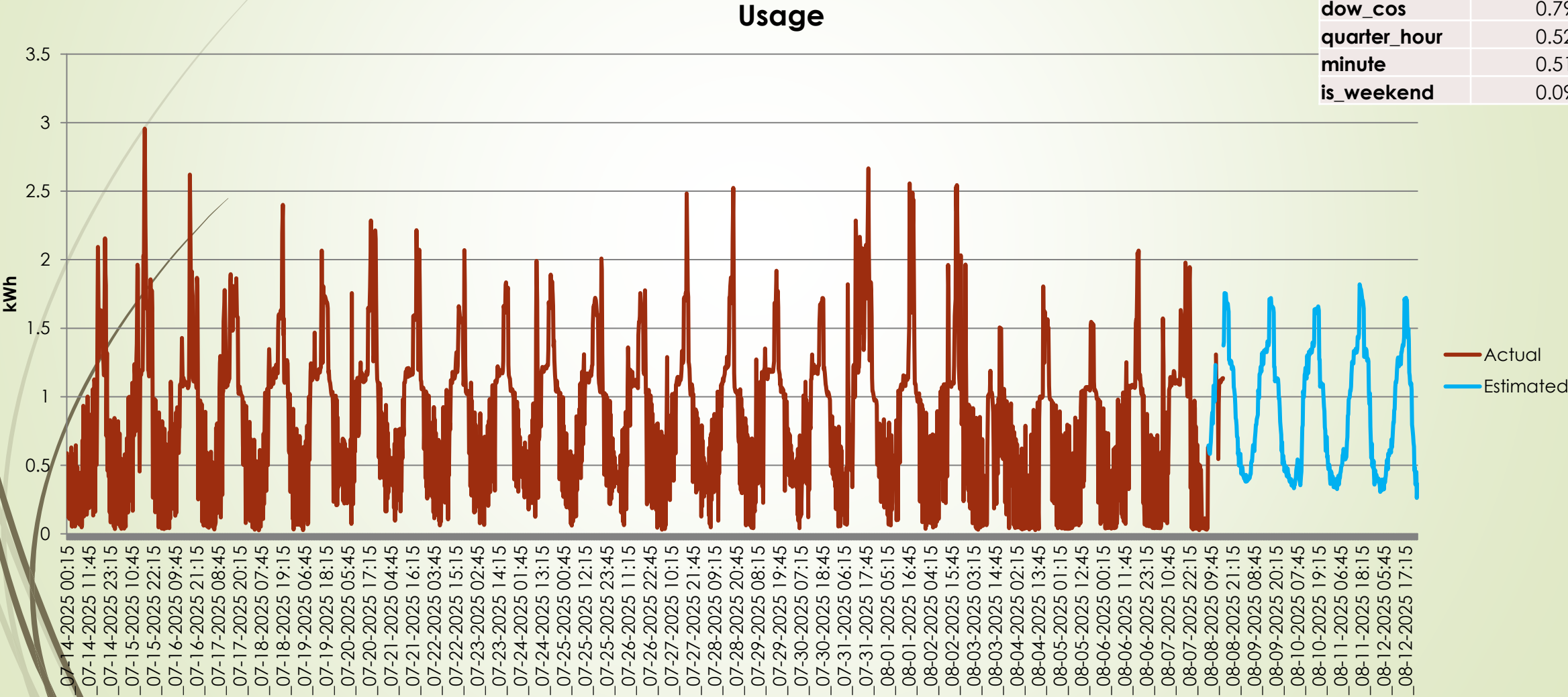
- Much lower variance and better generalization- averaging many trees reduces overfitting to noisy 15-min spikes.
- Built-in feature importance: You can identify that “lag-1 load” and “temperature” dominate predictions.
- Handles missing values and mixed data types well; robust to outliers common in meter readings.
- Parallel training makes it scalable for large smart-meter datasets.

Limitations compared to later tools: It still builds trees independently (bagging), so it doesn't explicitly correct sequential errors. While strong on tabular features, it may miss subtle long-range daily or weekly cycles better captured by sequential models. In electricity studies, Random Forest often achieves good accuracy (high R^2 , low MAE) but can be edged out by boosting methods on complex load patterns.

Random Forest is a dependable “workhorse” for 15-min load forecasting- fast enough for operational use and far more reliable than single trees or simple neighbors.

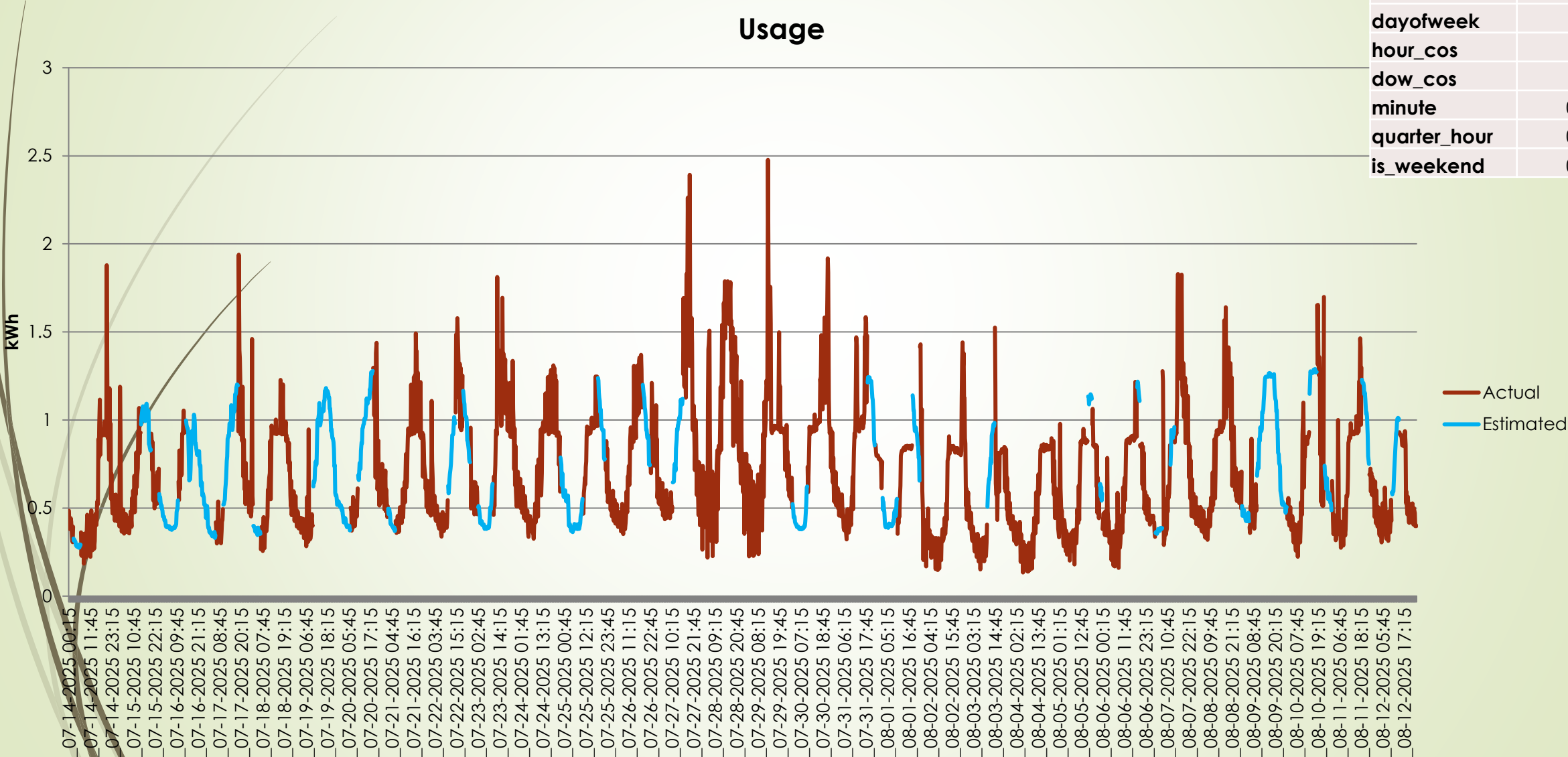
Random Forest Example 1

Mode	MAE	R ²	RMSE	Feature	MDI
test	23.734%	64.296%	30.718%	profile_kwh	72.670%
				hour	8.133%
train	14.719%	82.148%	22.263%	hour_sin	6.592%
				month	6.179%
				dow_sin	1.806%
				hour_cos	1.391%
				dayofweek	1.294%
				dow_cos	0.793%
				quarter_hour	0.527%
				minute	0.519%
				is_weekend	0.095%



Random Forest Example 2

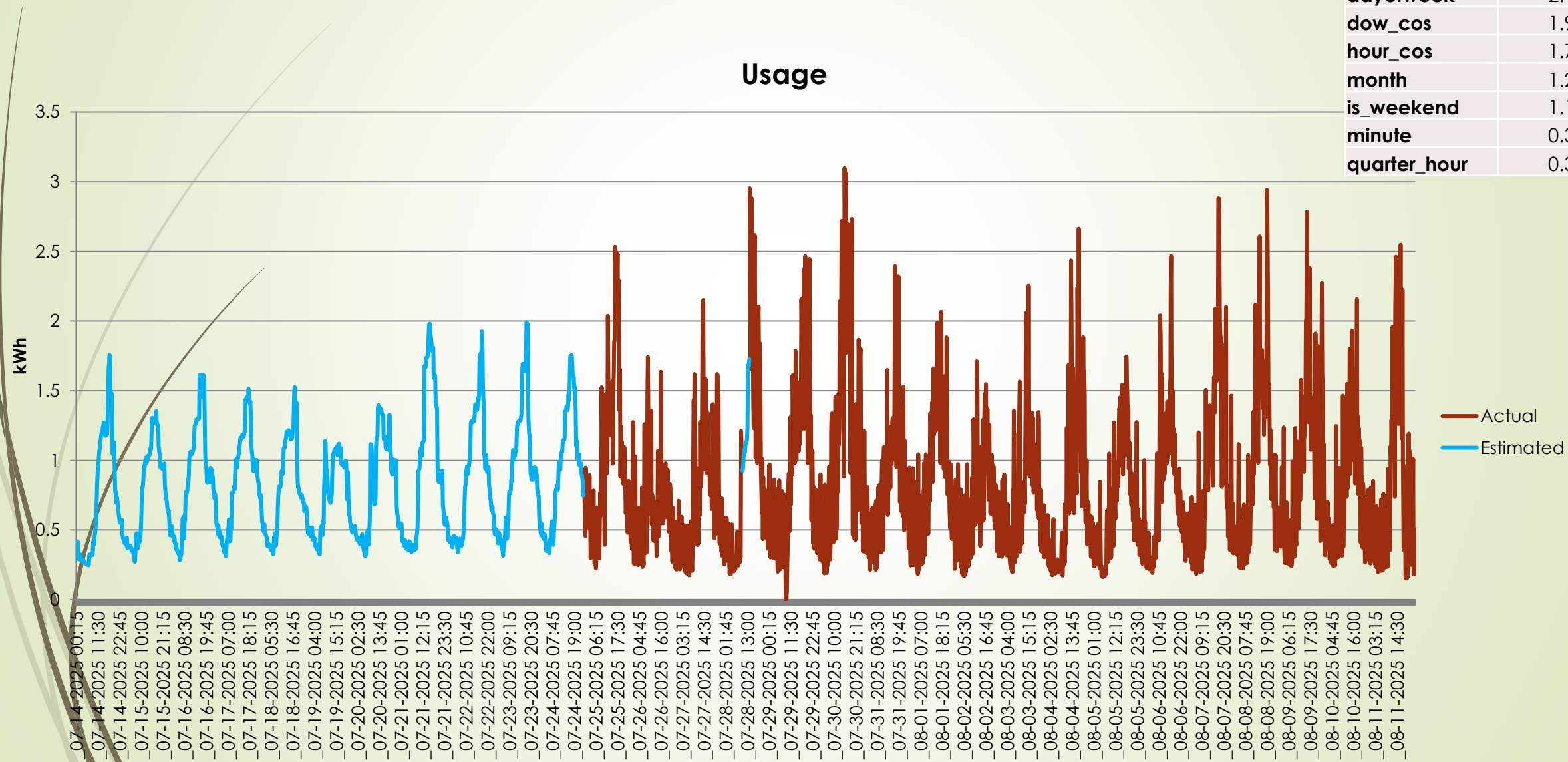
Mode	MAE	R ²	RMSE	Feature	MDI
test	14.664%	64.022%	21.309%	profile_kwh	85.337%
train	9.657%	74.705%	14.878%	hour	3.650%
				hour_sin	2.843%
				dow_sin	1.907%
				month	1.770%
				dayofweek	1.414%
				hour_cos	1.223%
				dow_cos	1.053%
				minute	0.343%
				quarter_hour	0.338%
				is_weekend	0.124%




Random Forest Example 3

Mode	MAE	R ²	RMSE	Feature	MDI
test	29.032%	48.241%	41.133%	profile_kwh	74.934%
				hour_sin	6.255%
train	20.967%	71.356%	28.847%	hour	5.608%
				dow_sin	4.009%
				dayofweek	2.400%
				dow_cos	1.918%
				hour_cos	1.797%
				month	1.217%
				is_weekend	1.134%
				minute	0.385%
				quarter_hour	0.342%

Usage





Stage 3: XGBoost

- XGBoost is another tree-based tool. XGBoost imputation treats the missing gaps like a prediction problem. If we train an XGBoost model on all the good data, feeding it useful features like time of day, day-of-week, recent load trends, we are likely to get much better results than the other 2 methods (the model learns the real patterns and fills the missing information with much more realistic values).
- It learns from its previous errors by treating them as gradients and fitting new trees to correct those mistakes. Training continues until adding more trees stops improving performance on validation data, preventing overcorrection.
- It's fast, handles complex relationships automatically, and usually beats basic methods by a good margin. The result is cleaner data that makes a lot more sense.

eXtreme Gradient Boosting (XGBoost)

Advantages/Disadvantages/Comparison

Tailored to 15-min forecasting: XGBoost excels with carefully engineered features: multiple intervals, time-based variables, weather forecasts, holiday flags, and even interactions. It handles the high volume of 15-min records efficiently thanks to histogram-based splitting and sparsity awareness. Studies on short-term load forecasting frequently show optimized XGBoost delivering low MAPE (around 2-3% in some cases) and outperforming Random Forest and linear models, especially for horizons up to several hours ahead.

Comparison snapshot:

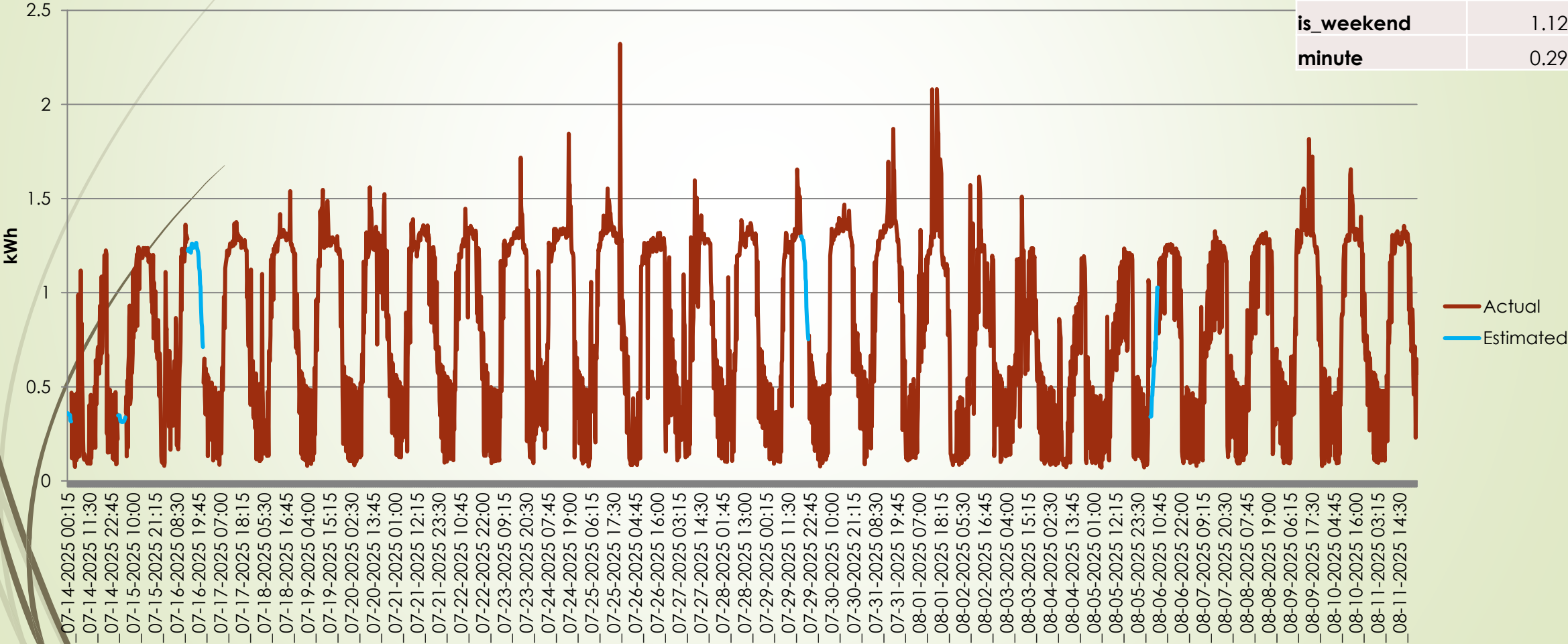
- Versus KNN: Dramatically better at capturing complex non-linear interactions and interactions between features without suffering from distance-based issues.
- Versus Random Forest: Usually more accurate and faster due to error-correcting boosting, built-in regularization (preventing overfitting on volatile 15-min data), and advanced split-finding. It often ranks among the top performers in energy forecasting benchmarks.
- It offers an excellent balance: state-of-the-art results on tabular time-series data with reasonable training time and interpretability via SHAP values (e.g., quantifying how a sudden temperature drop affects the next 15-min load).
- XGBoost powers many production systems for very short-term (next 6-24 hours at 15-min resolution) electricity forecasting because it delivers high accuracy without the heavy computational cost of deep learning- ideal when data is abundant but not massively sequential in raw form.

XGBoost Example 1

Mode	MAE	R ²	MAE_REL	RMSE_REL	RMSE
test	17.944%	73.336%	22.158%	28.936%	23.432%
train	17.822%	75.506%	22.748%	29.218%	22.891%

Feature	MDI
hour	45.157%
time_of_day	28.360%
profile_kwh	9.677%
hour_cos	9.146%
dayofweek	4.244%
hour_sin	1.992%
is_weekend	1.128%
minute	0.296%

Usage

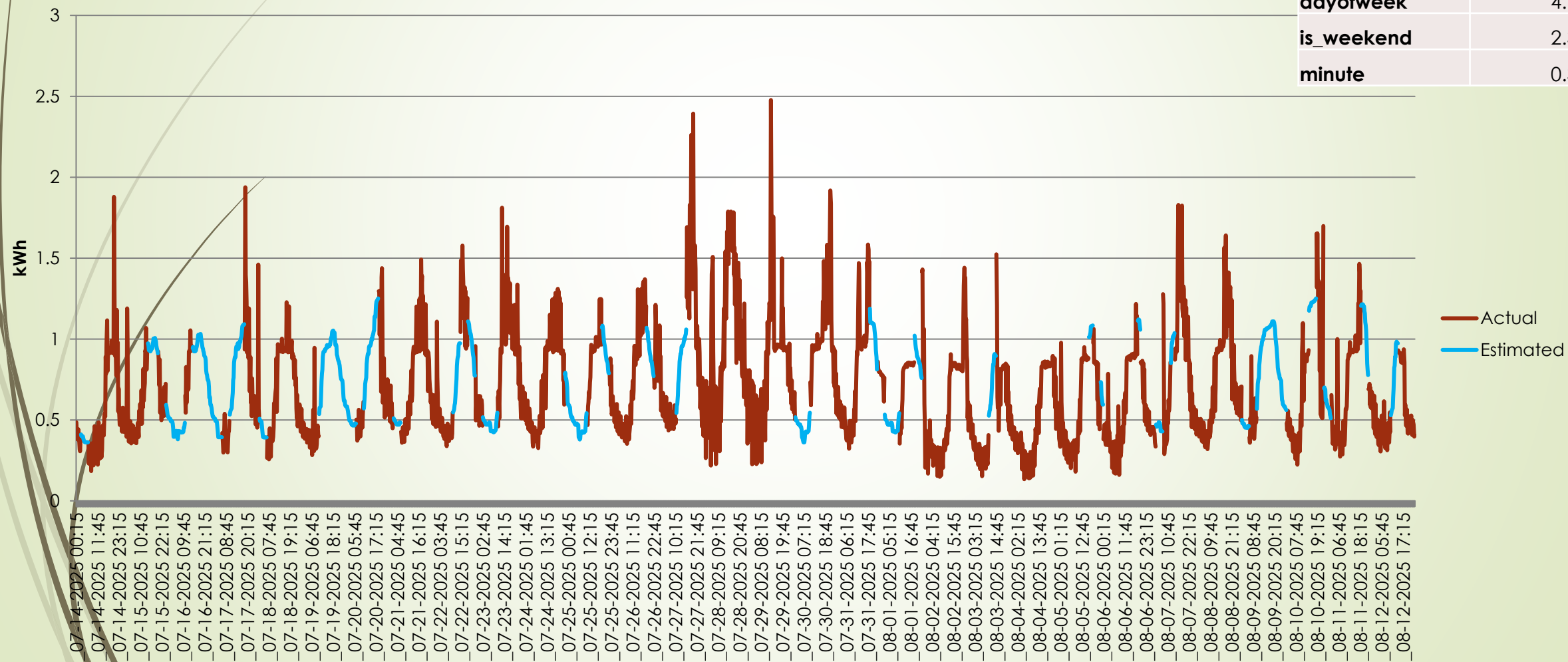


XGBoost Example 2

Mode	MAE	R ²	MAE_REL	RMSE_REL	RMSE
test	12.877%	67.434%	18.888%	27.173%	18.526%
train	13.108%	69.794%	18.479%	26.843%	19.042%

Feature	MDI
profile_kwh	49.158%
time_of_day	22.666%
hour	7.663%
hour_cos	6.995%
hour_sin	6.193%
dayofweek	4.627%
is_weekend	2.324%
minute	0.374%

Usage

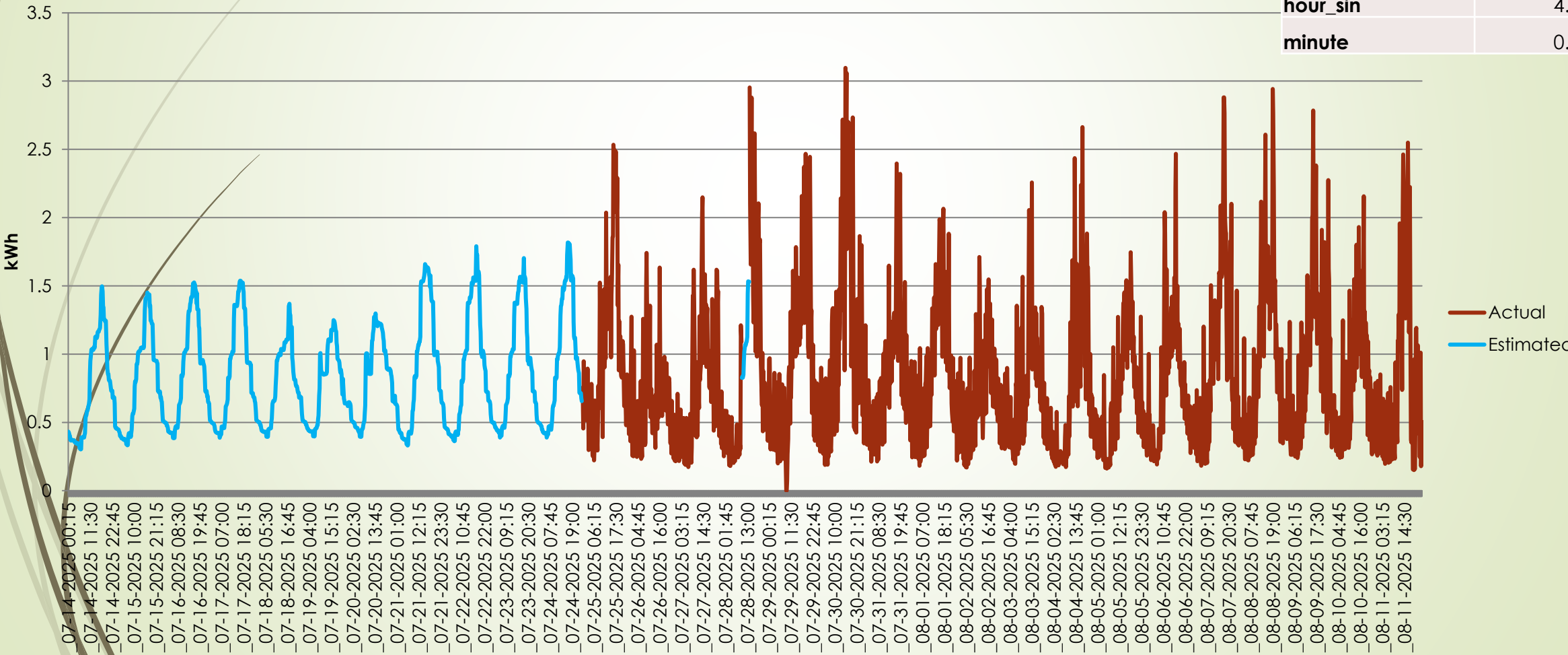



XGBoost Example 3

Mode	MAE	R ²	MAE_REL	RMSE_REL	RMSE
test	30.508%	46.061%	33.420%	46.631%	42.569%
train	24.631%	62.638%	30.379%	40.964%	33.213%

Feature	MDI
profile_kwh	36.007%
time_of_day	18.447%
hour	15.349%
is_weekend	10.054%
hour_cos	8.748%
dayofweek	5.859%
hour_sin	4.651%
minute	0.884%

Usage





Stage 4: Long Short-term Memory (LSTM)

LSTM can be defined as a specific type of Recurrent Neural Network (RNN) designed to model sequential data like time-series, speech, and text by learning their long-term dependencies. LSTMs are designed to remember information over long periods of time.

We train LSTM models to predict what the next 15-minute load value will be using the previous hours or days of actual data. When gaps show up, the model fills them in using the context it knows.

It's very effective. For short gaps I've found it very accurate, and for longer ones it follows the natural load curve better than simpler methods.

It's been said that LSTM doesn't just connect dots, it understands the rhythm of electricity usage.

If you've run out of options with missing interval data, you should give LSTMs a try. They are relatively slow, but if needed they can solve the toughest of problems.



Long Short-term Memory (LSTM)

Advantages/Disadvantages/Comparison

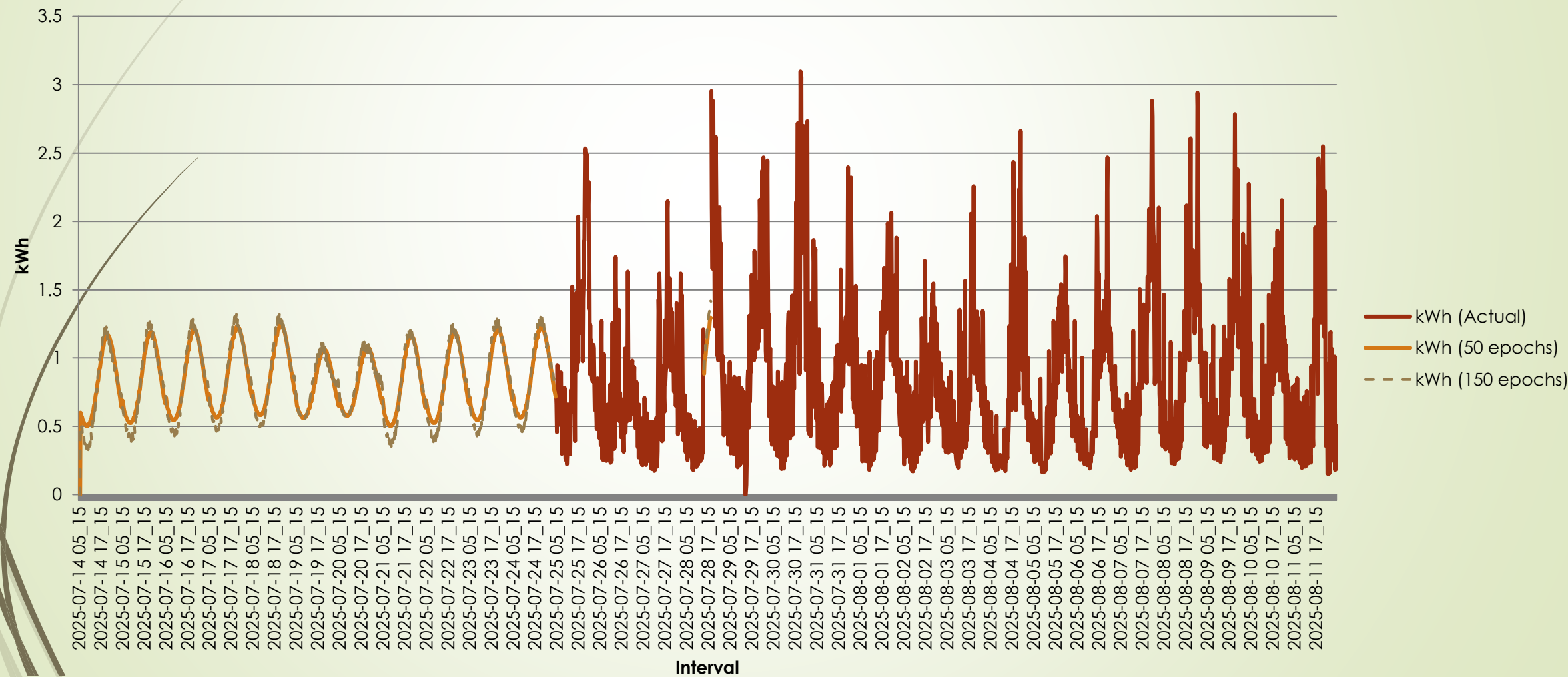
Direct comparisons in 15-min electricity context:

- KNN, Random Forest, and XGBoost treat data primarily as tabular features; LSTM natively models order and long-range dependencies, often outperforming tree ensembles on multi-hour or multi-day horizons where temporal context matters most.
- Tree methods train quickly and work well with moderate data; LSTM demands more data and time but can achieve superior accuracy on complex, non-stationary 15-min series (i.e., capturing sudden EV charging spikes or weather-driven changes).
- In practice, hybrids (XGBoost for feature extraction + LSTM for sequencing, or CNN-LSTM) frequently deliver the best results for smart grid applications.
- LSTM shines in scenarios with rich sequential structure- such as forecasting tomorrow's 96 15-min load profile- where remembering patterns over hours or days provides an edge.

LSTM Example 1

epochs	Mode	MAE	R ²	MAE_REL	RMSE_REL	RMSE
50	test	28.862%	59.737%	31.617%	40.288%	36.778%
50	train	26.921%	61.269%	32.527%	41.266%	34.154%
150	test	26.355%	64.595%	28.871%	37.780%	34.488%
150	train	24.207%	66.094%	29.248%	38.610%	31.956%

LSTM Usage Comparison





Conclusion

- The future looks incredibly exciting. When I first started diving into this a few months ago, it felt like an overwhelming challenge. But the more I worked with it, the easier it became to understand and apply.
 - This is just the beginning of an exciting journey for me and my team. I plan to build pipelines that will automatically update my sample and census data whenever needed, delivering strong results. I'll also use these tools to check peaks and usage patterns, ensuring my load curves are as realistic and accurate as possible.
 - On top of that, I'm looking forward to applying these tools for weather normalization over the coming year. The possibilities really are endless.
 - Just remember... **DON'T PANIC**. Machine learning isn't nearly as scary as it seems.
- 