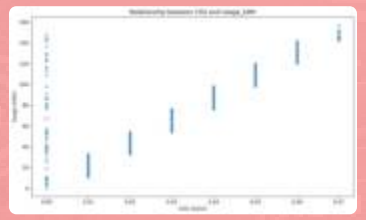


Featured Models: RF, XGBoost, SARIMAX, OLS

Key Concepts: Diagnostics, Seasonality, Humidity Regimes

Modeling Pillars & Behavioral Themes

Toolkit Behind the Analysis: *Python , R, Power BI, Gamma App, Kaggle GitHub*

Source: "Steel Industry Energy and Emissions Data," available on Kaggle

Decoding Steel Plant Systems: Energy Use, Emissions, and Optimization

Optimizing Energy Use Reducing CO2 Output at a Steel plant: A Data-Driven Approach

SS by Susan Schnitzel

Summary & Portfolio Overview

This portfolio delivers a layered understanding of energy demand and emissions behavior at a steel plant—blending environmental diagnostics, operational modeling, and temporal forecasting.

Through exploratory analysis, reactive power metrics, humidity regimes, and seasonality, I uncovered nonlinear relationships between usage and emissions.

Modeling spanned OLS, Random Forest, XGBoost, SARIMAX, and ARIMA; each selected and refined with careful diagnostic reflection.

Beyond modeling, this journey included hands-on development in Python, R, Power BI, Excel, Kaggle, and GitHub; building not just predictions, but pipelines, visualizations, and narratives that reveal how energy systems breathe and behave.

Explore deeper sections to see: feature engineering breakthroughs, seasonality decomposition, imputation trade-offs, and diagnostic plots that uncover system "personality."

Portfolio Overview

Sections at a Glance

Data Collection & Transformation

Kaggle dataset + weather merge, datetime alignment, CO_2 imputation, scaling

Exploratory Analysis & Insights

Distributions, correlations, regime detection, weekday effects, seasonality

Feature Engineering & Diagnostics

Humidity bins, load rank, reactive power signals, outlier treatment

Structural Modeling

OLS, RF, XGBoost — residual analysis, Q-Q plots, regime comparisons

Temporal Forecasting

SARIMAX, ARIMA — autocorrelation, seasonal cycles, temperature sensitivity

Behavioral & Rhythm Models

log_kWh model, weekday/weekend shifts, Tuesday spikes

Cost Impact & Operational Value

Quantified savings ($7,505+), emission insights, optimization pathways

Final Reflections & Conclusion

Modeling arc, personal growth, data storytelling, next steps

Section 1 : Data Collection and Transformation

Data Preprocessing Challenges & Solutions

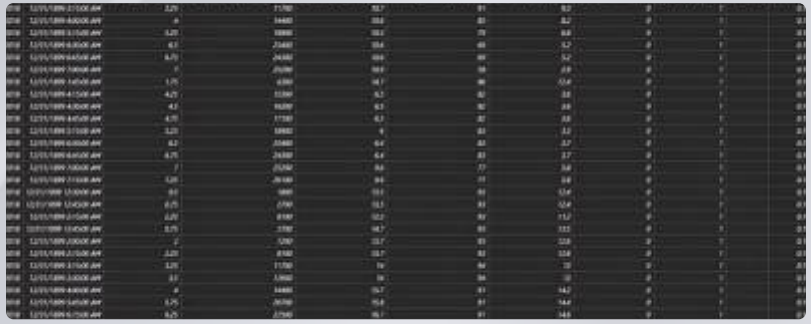
DATA WRANGLING

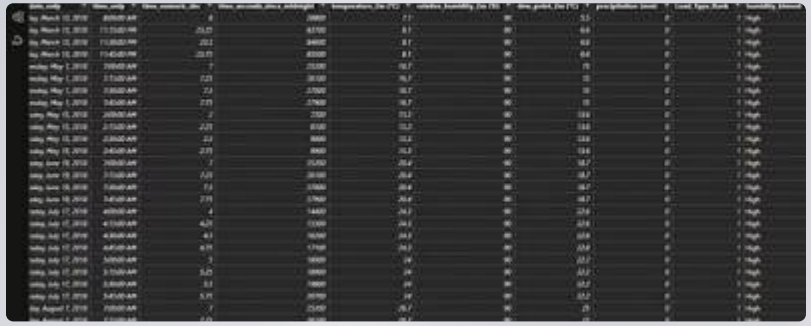
1. Grabbing the dataset from Kaggle (**Steel_industry_data.csv(2.73 MB)**
2. Fixed character encoding issues (converted "Â°C" to "°C")
   a. There are times I still like to revisit the raw data and transform differently
3. Researching and finding Weather Data set from the same time
   a. The dataset was about 30 miles away from the Steelplant
4. Resolved datetime format mismatches between energy and weather datasets
   a. Resampled weather data to align with energy consumption timestamps
5. Merged datasets to create a comprehensive analytical base
6. Implemented strategic handling of missing values
   a. Continuous transforming for strategic modeling

```
humidity_col = 'relative_humidity_2m (%)'
temperature_col = 'temperature_2m (°C)'
CO2_col = 'CO2(tCO2)'
```

	date	day	day_of_week	day_name	week	month	month_name	year
0	2018-01-01 00:15:00	1	0	Monday	1	1	January	2018
1	2018-01-01 00:30:00	1	0	Monday	1	1	January	2018
2	2018-01-01 00:45:00	1	0	Monday	1	1	January	2018
3	2018-01-01 01:00:00	1	0	Monday	1	1	January	2018
4	2018-01-01 01:15:00	1	0	Monday	1	1	January	2018

Note: Although painful, after all my analysis and modeling, I am very pleased that I took the time to find the corresponding weather data.





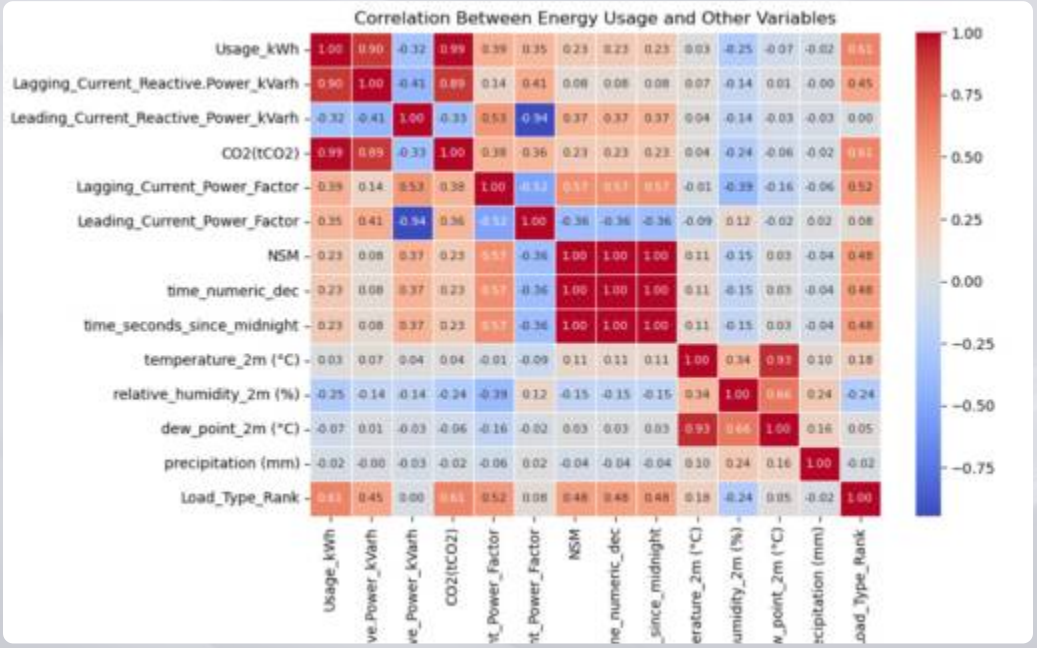
This slide may be brief, but the work behind it wasn't. Data merging, scaling, and transformation took plenty of sweat, tweaks, and caffeinated perseverance; none of which fit neatly in a bullet point.

Section 2: Exploratory Analysis and Insights

Exploratory and Data Insights

Getting to know the Data

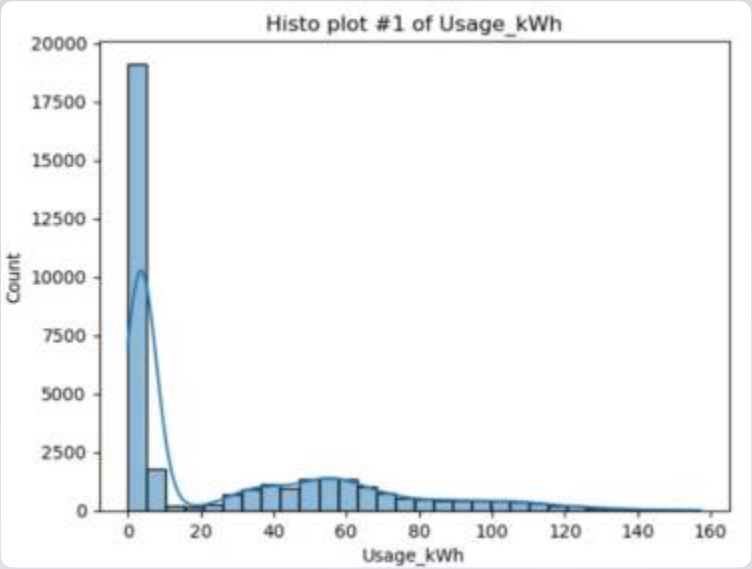
Before diving into modeling, I always take time to explore and interact with the data, not just to clean it, but to understand it. This includes visualizing distributions, examining trends over time, and checking for correlations or unexpected patterns. These early insights often spark better feature design and sharper hypotheses.

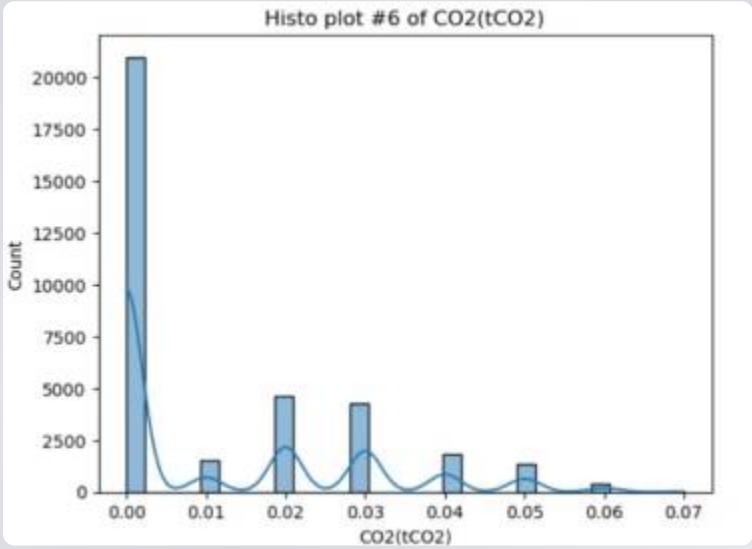


- Initial heatmap analysis revealed a strong positive correlation between energy usage (kWh) and CO_2 emissions; a relationship consistent with expected physical output, where increased energy production corresponds with higher emissions. This was key to guiding modeling decisions.

Exploratory Data Insights - Initial Distributions

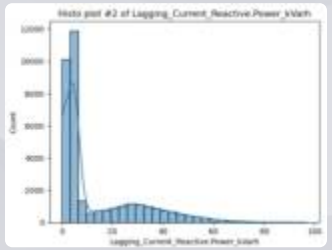
As someone trained in statistics, I believe it's essential to understand the shape and structure of the data before modeling. This includes assessing normality, identifying skewness or multimodality, and applying transformations where needed. These early checks help ensure that later models are both appropriate and interpretable.

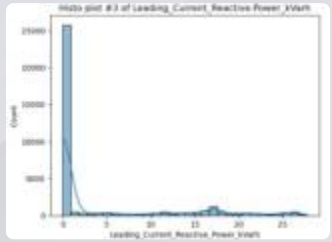


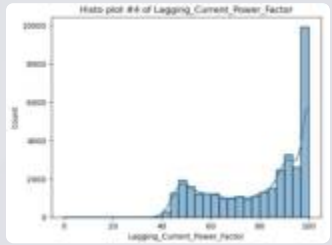


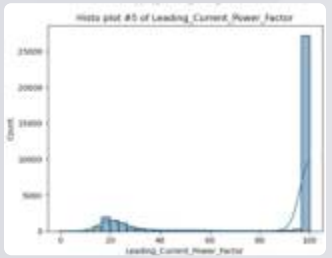
In this case, histograms of CO_2 and kWh revealed skewed distributions, leading to the application of transformation (log) on energy usage and an imputation strategy for zero-inflated CO_2 values.

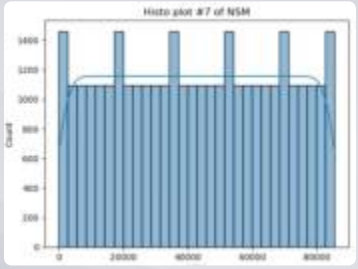
Exploratory Data Insights: Energy Distributions



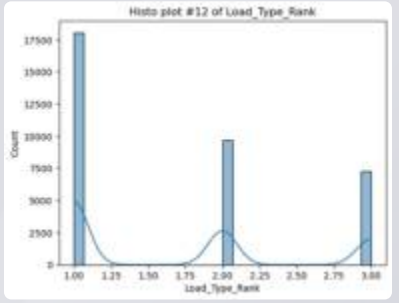






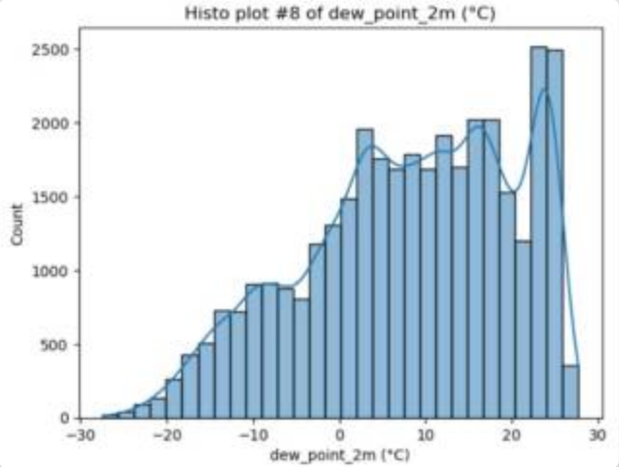


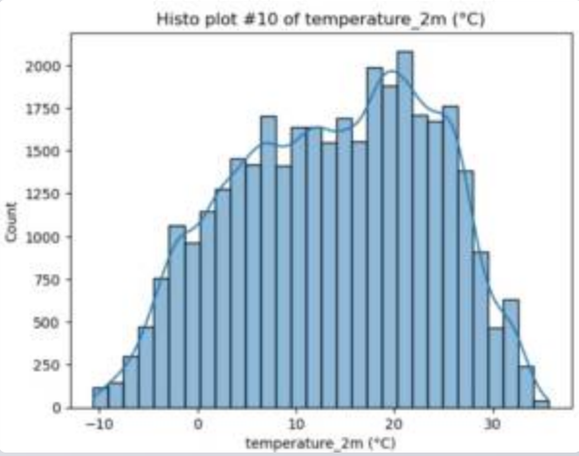
While several energy-related variables exhibited skewed distributions with some leaning heavily toward higher values. I also explored features like NSM and Rank, which are inherently nonparametric in nature. > > As someone with a background in statistics, I find it important not only to identify when a transformation (e.g., log scaling) might improve model performance, but also to understand the underlying shape of the distribution itself. These early diagnostics help determine whether a parametric or nonparametric approach is more appropriate.

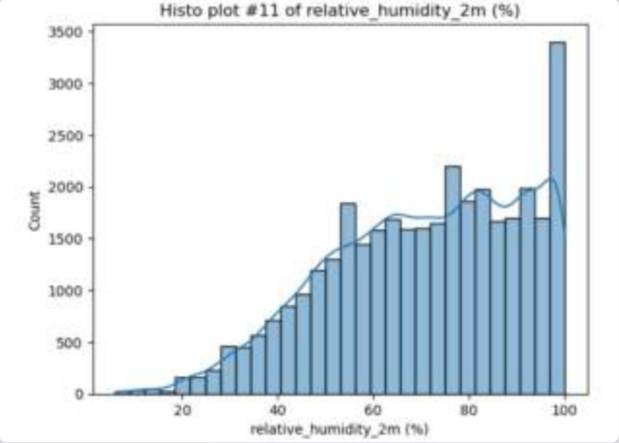


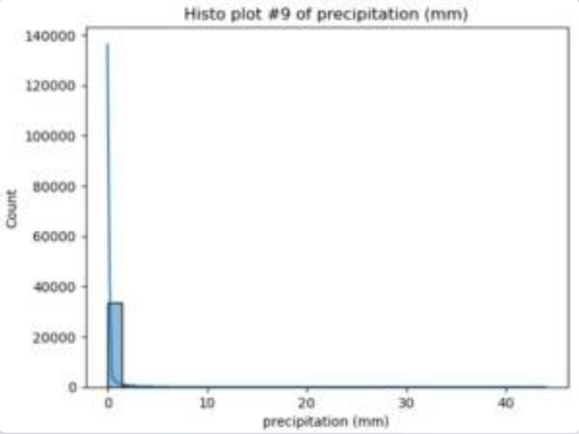
Time series decomposition revealed seasonal components that align with production schedules. After log-transformation, the relationship between variables became more linear, supporting our modeling approach. Weather variables showed moderate influence on energy efficiency.

Exploratory Data Insights: Enviromental Distributions







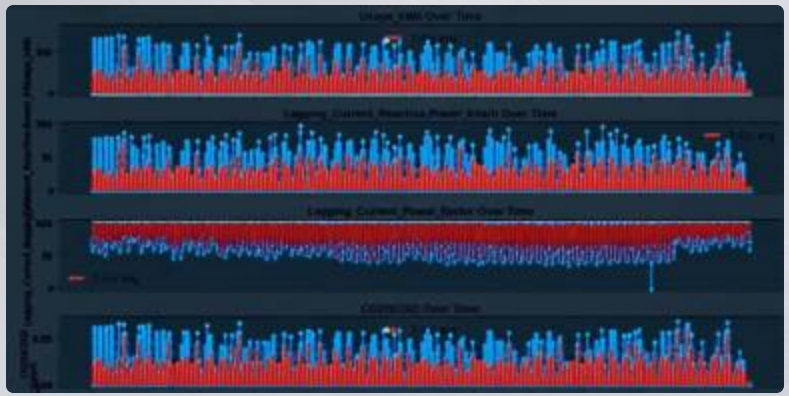


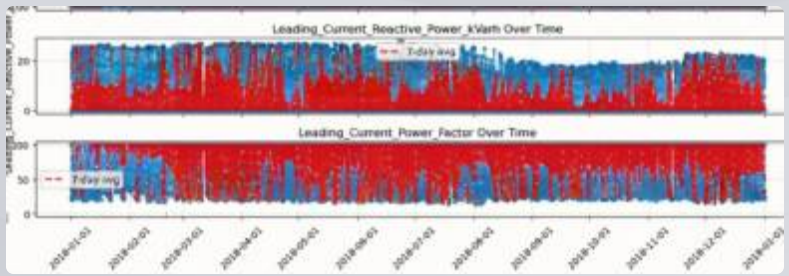
Histogram analysis showed that temperature and dew point followed approximately normal distributions, supporting their direct use in modeling without transformation. In c**ontrast, humidity exhibited a strong right skew, suggesting the site regularly experiences high-humidity conditions. This insight guided the decision to engineer humidity-based bins and explore its role in emission variability. > > Another feature appeared to have limited variation and predictive value and was subsequently excluded from modeling to preserve parsimony. Precipitation appears to have limited data except because of the skew to zero won't be a main variable for the modeling but will still include initially.**

TIME SERIES OF VARIOUS VARIBLE

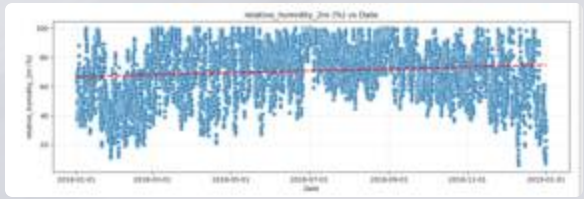
.....and having a little fun with Gamma's AI animation

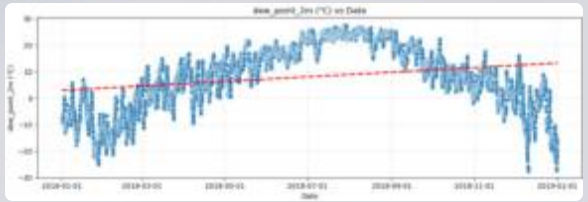
Energy and CO2 variables

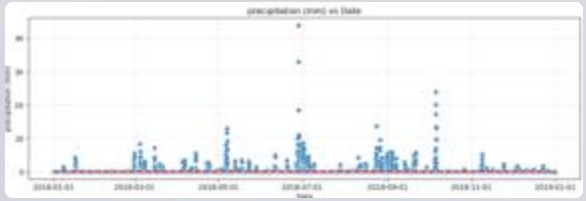




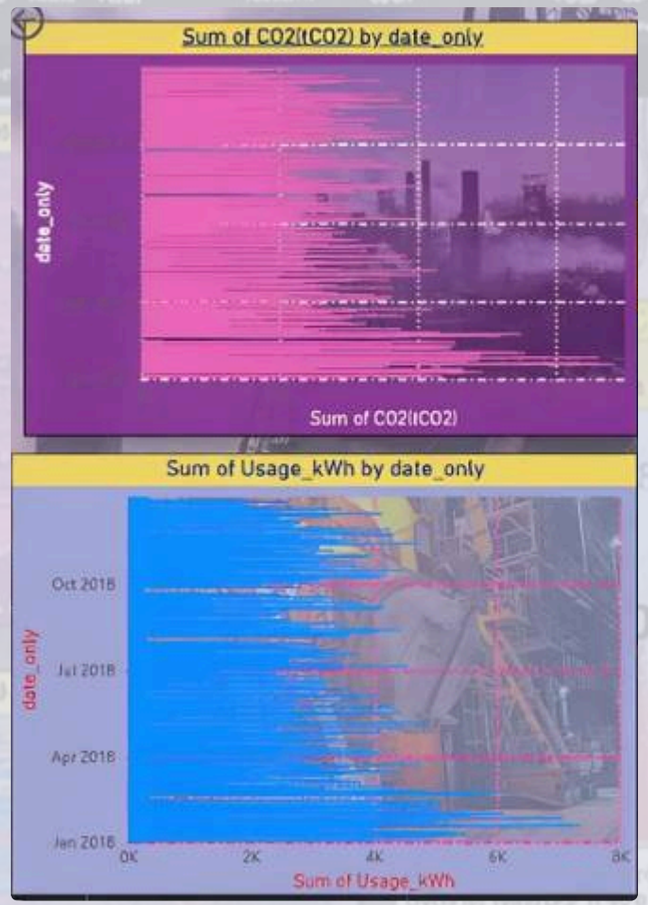
Enviromental /Weather variables

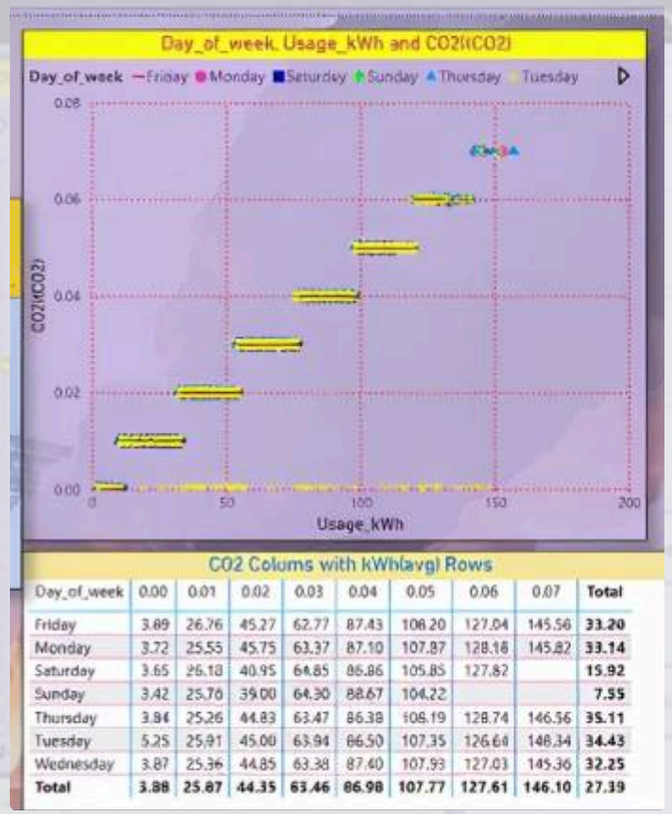






Time Series Comparison of CO2 and Enerzgy Usage kWh





CO2 Colums with kWh(avg) Rows

Day_of_week	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	Total
Friday	3.89	26.76	45.27	62.77	87.43	108.20	127.04	145.56	33.20
Monday	3.72	25.55	45.75	63.37	87.10	107.87	128.18	145.82	33.14
Saturday	3.65	26.18	40.95	64.85	86.86	105.85	127.82		15.92
Sunday	3.42	25.76	39.00	64.30	88.67	104.22			7.55
Thursday	3.84	25.26	44.83	63.47	86.38	108.19	128.74	146.56	35.11
Tuesday	5.25	25.91	45.00	63.94	86.50	107.35	126.66	148.34	34.43
Wednesday	3.87	25.36	44.85	63.38	87.40	107.93	127.03	145.36	32.25
Total	**3.88**	**25.87**	**44.35**	**63.46**	**86.98**	**107.77**	**127.61**	**146.10**	**27.39**

Yearly Look

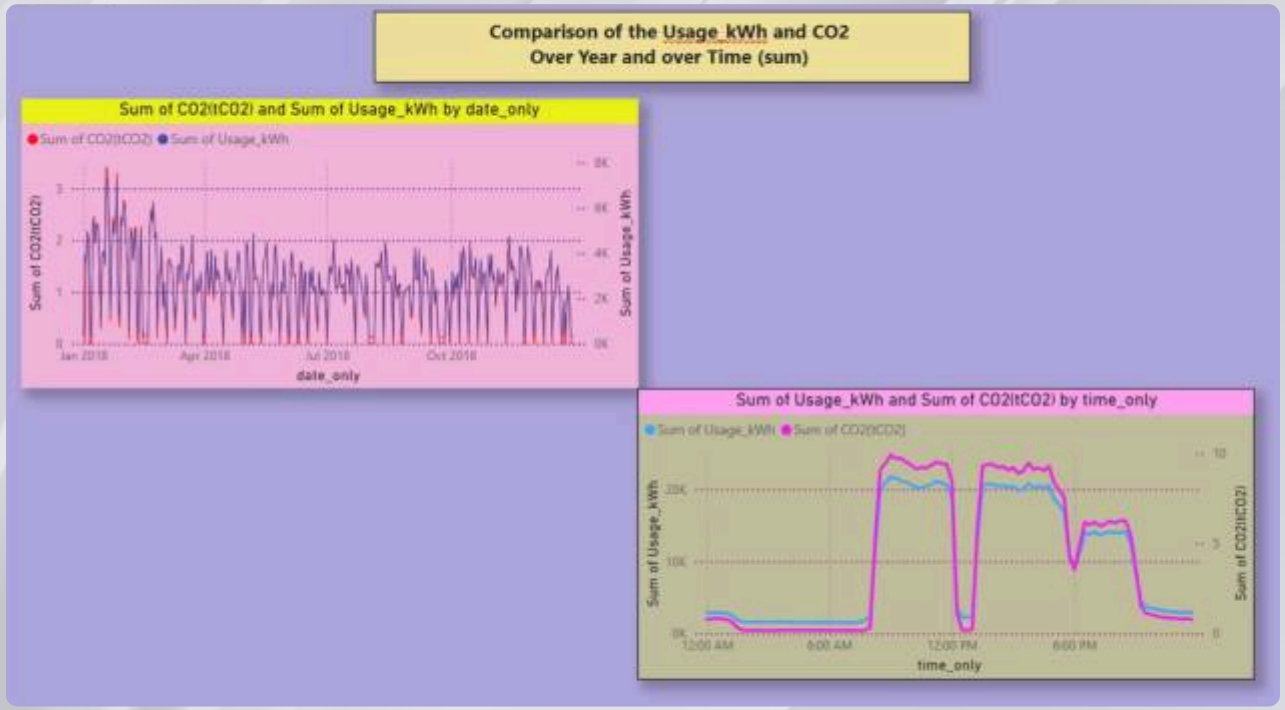
The Graphs look almost identical with the trends. Both the CO2 emmisions and Enerzgy Usage kWh show peaks at the beginnig winter months..

Days of The Week

Scatterplot of CO_2 vs. energy usage revealed a concentration of Tuesday observations (in yellow), suggesting a data volume skew — with Tuesdays contributing heavily to typical output ranges, but not to the highest-emission cases."

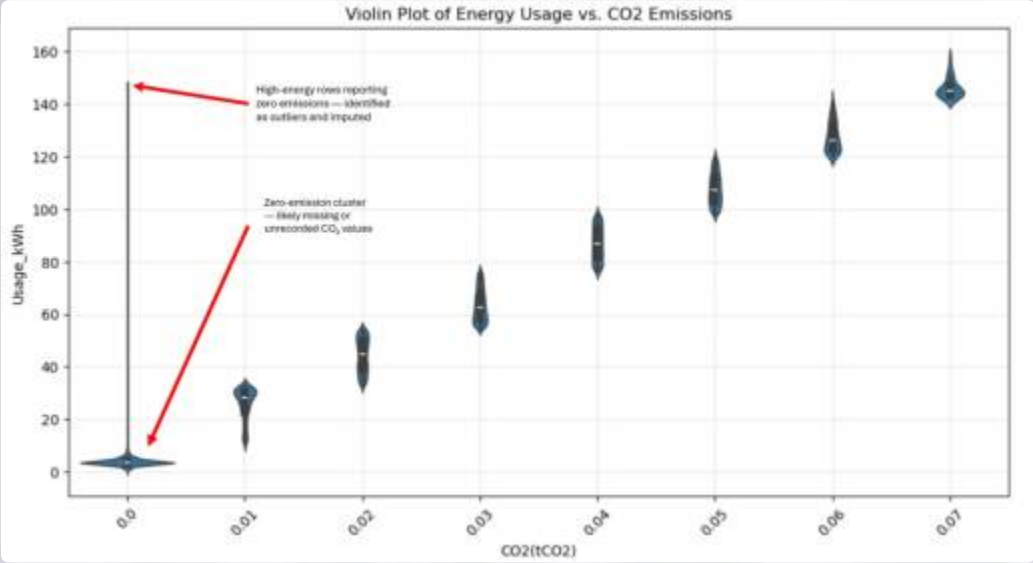
This correlation aligns with thermodynamic expectations — greater energy conversion yields higher CO_2 output. Also, it shows a large amount of zeros for the CO2. For some models Zero-emmision clusters removed during preprocessing and imputed for modeling consistency.

Another Visual Comparing CO2 and Energy Usage kWh over time.



Violin Plot

Illustrates the Evolving Distribution of CO2 across Energy Usage Levels



- *Initial Variable Distributions"> Histograms revealed skewed CO_2 emissions and wide energy usage range. Guided transformation choices (e.g., log_kWh) and flagged missing values for imputation.*

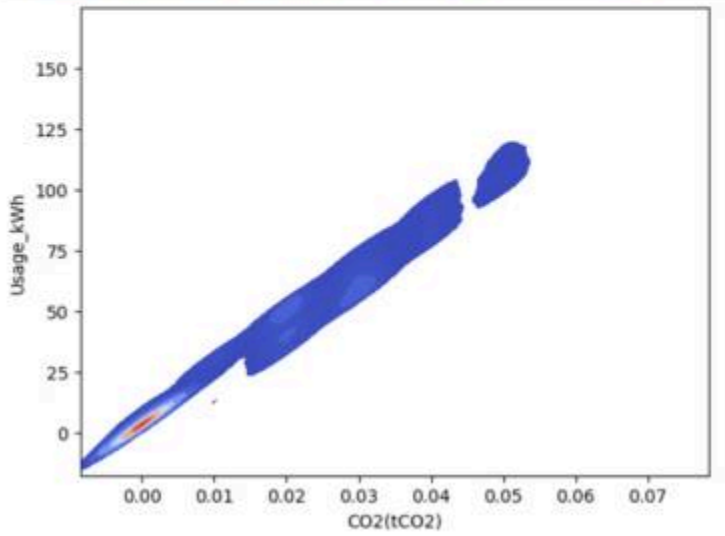
The violin plot reveals how CO_2 emission distributions change at different energy consumption levels, showing both frequency and probability density.

The plot revealed a dense spike at zero CO_2 emissions with a surprising vertical tail stretching into high energy usage levels. These outliers indicated likely data entry gaps since high kWh should yield measurable emissions. This informed targeted imputation to restore data integrity ahead of modeling

The top energy level has wider areas where data points cluster most densely, indicating common operational states in the plant.

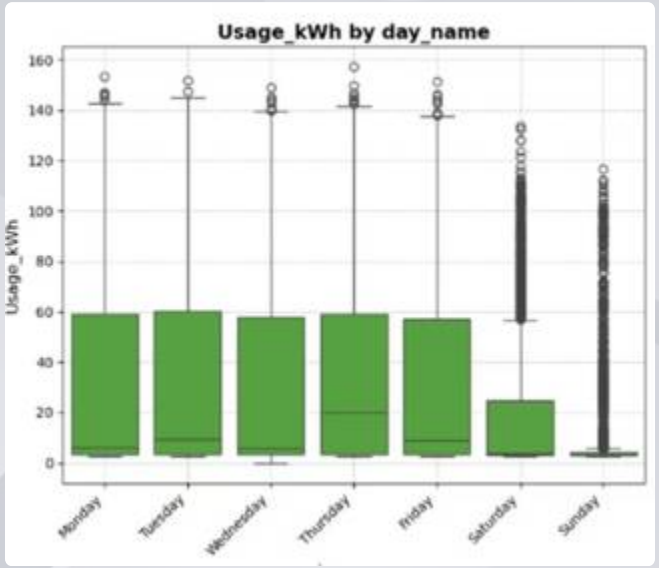
Looking at the Density (Kernel)Plot to look

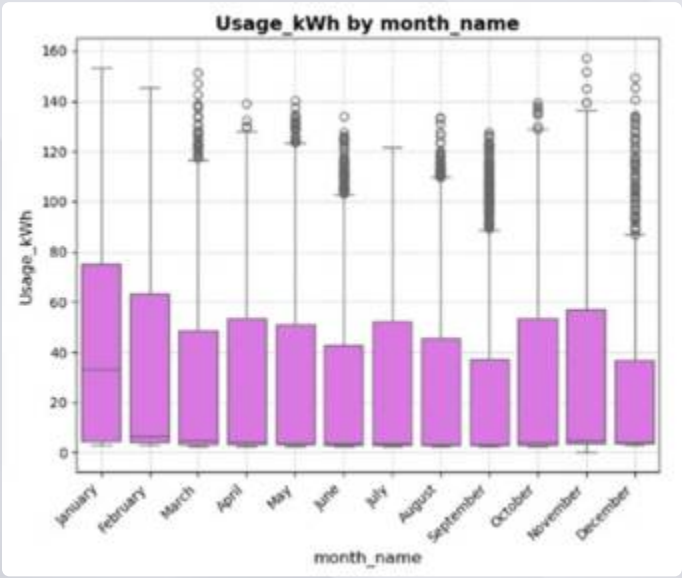
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
#Density plot, compare the distribution of both variables
#had on same plot as the scatter and I didn't like it.
#Also, Updated syntax for newer versions of seaborn - using named parameters x and y instead of positional arguments
#And fill instead of shade for color too.
sns.kdeplot(x=df["CO2(tCO2)"], y=df["Usage_kWh"], cmap="coolwarm", fill=True)
plt.savefig("density_Plot_usage_KwH-CO2.jpg", dpi=300, bbox_inches="tight")
```



- The predominantly blue color with a small amount of red around intercept CO2, suggests that the highest density (red) is concentrated near the lower end of the CO2 axis, with density decreasing (shifting to blue) as CO2 increases.
- The thicker density from 0.02 to 0.05 CO2, followed by a split with no color, and then a thicker blob again at 0.06, indicates varying concentrations of data points. The "split with no color" likely represents an area of very low or zero density (a gap in the data distribution).
- The thicker portion of line indicates a higher concentration of data points, meaning more observations fall within that range of CO2 (x-axis) and energy usage (y-axis).

Boxplots illustrating how Energy Usage Fluctuates across time





🗓️ Day of the Week

Weekdays

No strong correlation across weekdays.

Weekends

Shows **lower usage on weekends**, which is a clear and explainable pattern (possibly reduced shifts or partial operations/production activity. Later analysis, shows Tuesday as a possible variable to examine further.

📅 Monthly Patterns

January & February stand out as peak usage and aligns with heating needs or seasonal production cycles *"Winter peak usage reflects seasonal energy demand"*

November shows early uptick, possibly signaling seasonal prep or ramp-up

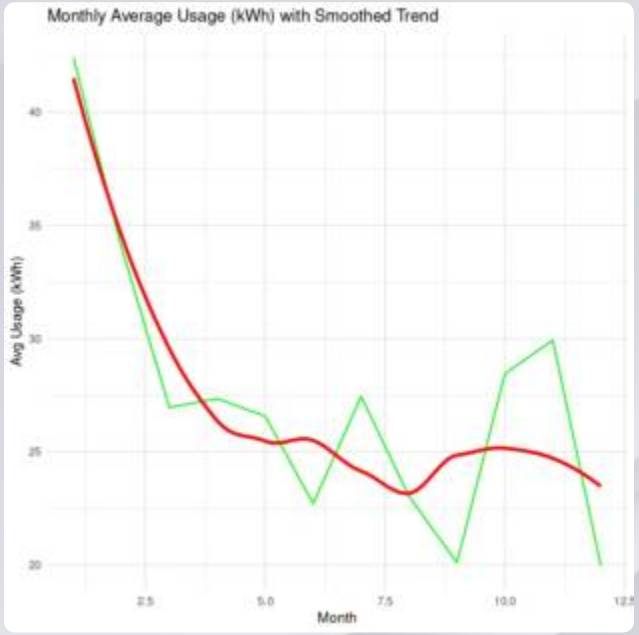
December not matching upward trend → plausible cause: holiday slowdowns or incomplete or partial data collection.

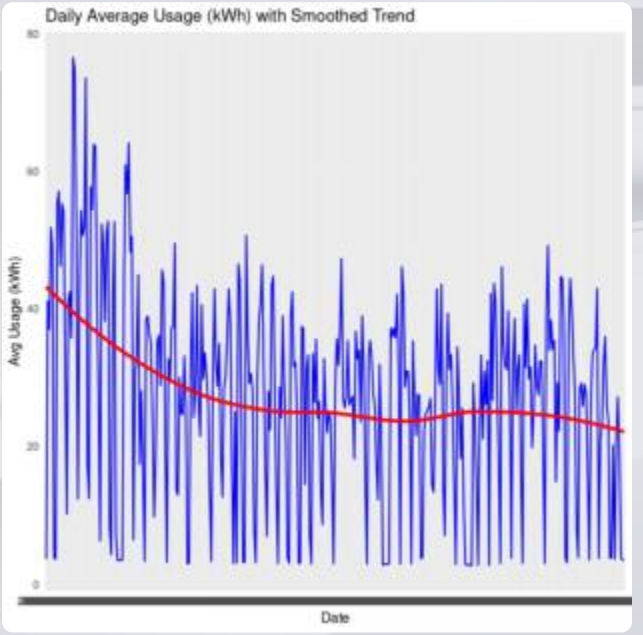
Trend Analysis Over Time

Supplements to the Boxplots showing monthly trends

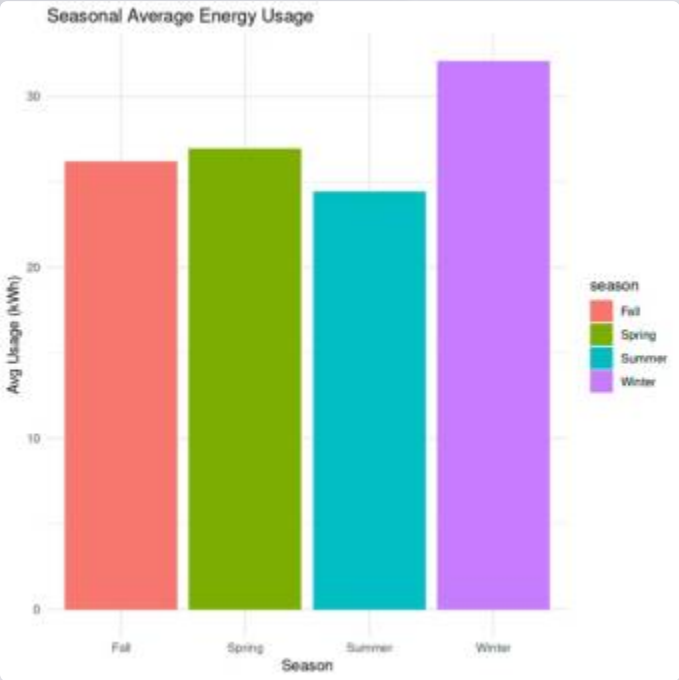
Energy usage showed clear seasonal behavior, with the highest consumption occurring in the first two winter months (January and February). Usage declined steadily through spring and summer, followed by a noticeable spike in November. Interestingly, December did not continue this upward trend. It is likely a result of reduced plant activity during the holiday season or potential gaps in data collection.

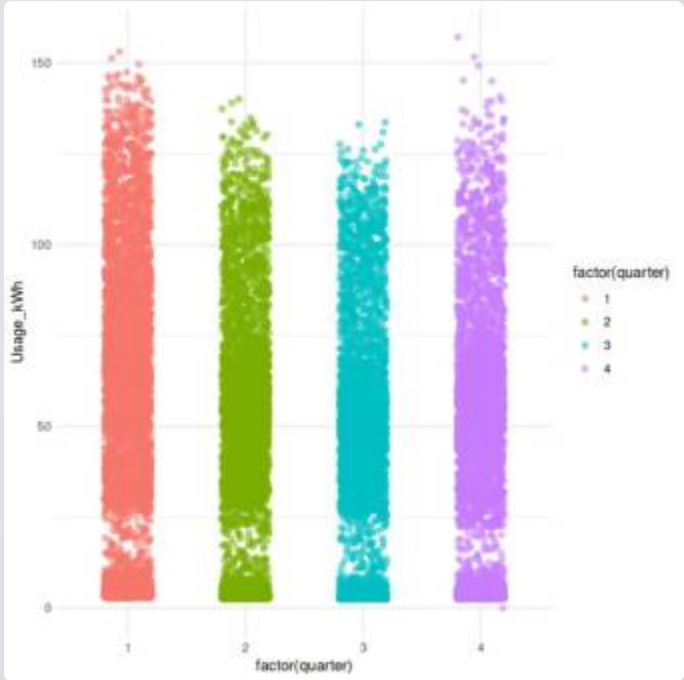
CODE USING R IN KAGGLE





SEASONS and Energy Usage (kWh)






```
#Lets look at seasonal trends, not sure if will show a trend, but fun to try  library(      )                    <-    %>%
mutate(          = case_when(          %in% c(12,  1,  2) = "Winter",          %in% c(3,  4,  5) = "Spring",
%in% c(6,  7,  8) = "Summer",          %in% c(9,  10,  11) = "Fall" ) %>% group_by(          ) %>%
summarise(          = mean(                          = TRUE))
```

Consistent with other temporal patterns, the winter months show the highest average energy usage — reinforcing the seasonal impact on operational demand.

```
ggplot(    , aes(    = factor(        ),    =          ,          = factor(          ))) +
geom_jitter(          = 0.2,          = 0,          = 0.5) + theme_minimal()
```

Geo jitter plot by quarter illustrates seasonal energy use variability. Q1 shows the highest median usage and widest spread — consistent with winter operations and fluctuating demand. Q4 also displays variability, driven by a November spike followed by a December drop, likely due to holiday slowdowns or partial data capture.

Anova with Energy Usage kWh with Enviromental weather variables

As I began modeling, my exploratory analysis continued in parallel. While initial insights shaped early model choices, I remained open to discovering new patterns and refining features throughout the process. This iterative approach allowed for deeper understanding of environmental drivers and ensured each model was informed by the evolving data story.

ANOVA For Energy Usage kWh with CO2

	sum_sq	df	F	PR(>F)
temperature_2m	3.689350e+02	1.0	2.145128e+01	3.642525e-06
relative_humidity_2m	2.773542e+03	1.0	1.868654e+02	7.748788e-25
precipitation	3.785748e+01	1.0	1.447739e+00	2.288991e-01
CO2	1.519365e+07	1.0	1.345871e+06	0.000000e+00
Residual	9.163427e+05	35035.0	NaN	NaN

ANOVA For Energy Usage kWh without CO2

ANOVA table without CO2:

	sum_sq	df	F	PR(>F)
temperature_2m	6.268348e+05	1.0	608.194598	1.787551e-133
relative_humidity_2m	3.003126e+06	1.0	2914.016651	0.000000e+00
precipitation	6.217191e+04	1.0	60.317868	1.687628e-14
Residual	3.610088e+07	35036.0	NaN	NaN

Variance Inflation Factors:

	Variable	VIF
0	const	16.677615
1	temperature_2m	1.153531
2	relative_humidity_2m	1.283317
3	precipitation	1.061145
4	CO2	1.082835

Key Statistical Findings

🌱 When CO_2 is included:

- **CO_2 is dominant** in explaining energy usage (F = 1.35 million, p < 0.00001).
- Other variables like temperature and humidity still matter, but their influence pales in comparison.
- Precipitation becomes statistically irrelevant (p = 0.23), likely overshadowed by the strength of CO_2.

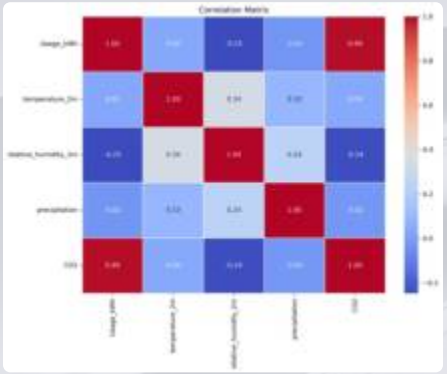
🌤️ When CO_2 is excluded:

- **Relative humidity** becomes the strongest predictor (F = 2914).
- **Temperature** also has a sizable effect (F = 608).
- **Precipitation** is significant depending on the p-value formatting (though 1.6076 looks like a formatting error — likely p < 0.0001 based on the F-stat).

🔍 Interpretation:

CO_2 captures a large portion of the variance explained by both humidity and temperature. When removed, their independent effects surface more strongly.

VIF results confirm feature stability, with no signs of problematic multicollinearity.



HEATMAP of Energy Usage and CO2 with Weather-related environmental variables — including temperature, humidity, and precipitation.

While the correlation between CO_2 and energy usage aligned with expectations, exploratory analysis revealed an inverse relationship between humidity and both target variables. This pattern suggests drier atmospheric conditions may correspond with higher energy demand and emissions. Temperature and humidity also exhibit interdependence, warranting deeper investigation through interaction modeling in subsequent analysis

Anova with CO2 with Enviromental weather variables

```python
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Define the formula with CO2 as the dependent variable
formula = "CO2 ~ Usage_kWh + temperature_2m + relative_humidity_2m + precipitation"

# Fit the ANOVA model
model = smf.ols(formula, data=df).fit()
anova_table = sm.stats.anova_lm(model, typ=2)  # Type 2 for balanced designs

print(anova_table)
```

ANOVA with Energy Usage kWh

	sum_sq	df	F	PR(>F)
Usage_kWh	8.226553e+00	1.0	1.345871e+06	0.000000e+00
temperature_2m	4.562720e-04	1.0	7.464646e+01	5.864480e-18
relative_humidity_2m	2.564990e-05	1.0	4.196345e+00	4.051864e-02
precipitation	2.144567e-07	1.0	3.508529e-02	8.514181e-01
Residual	2.141493e-01	35035.0	NaN	NaN

```python
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Define the formula with CO2 as the dependent variable
formula = "CO2 ~ temperature_2m + relative_humidity_2m + precipitation"

# Fit the ANOVA model
model = smf.ols(formula, data=df).fit()
anova_table = sm.stats.anova_lm(model, typ=2)  # Type 2 for balanced designs

print(anova_table)
```

ANOVA without Energy Usage kWh

	sum_sq	df	F	PR(>F)
temperature_2m	0.159554	1.0	662.281719	1.058104e-144
relative_humidity_2m	0.675527	1.0	2804.004621	0.000000e+00
precipitation	0.009517	1.0	39.503582	3.313068e-10
Residual	8.440702	35036.0	NaN	NaN

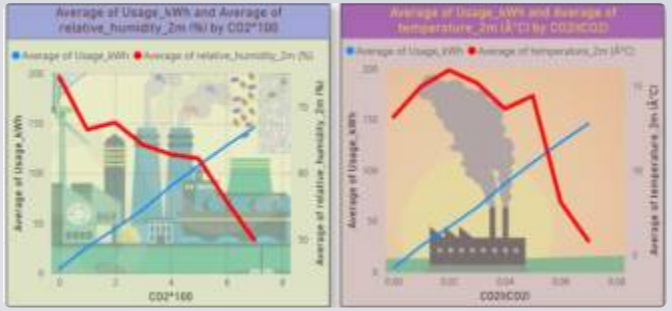
🔍 CO_2 as a Response Variable

- **Usage_kWh is once again the strongest driver** of CO_2 emissions ($F \approx 1.35$ million, $p < 0.00001$).
- **Temperature & Humidity** also contribute, though with much smaller effect sizes.
- Precipitation remained the least influential among the weather variables

Possible Interactions for CO2 with Energy Usage and Weather Variables

ANOVA CHECK

```
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Define the formula with interaction terms
formula = "CO2 ~ Usage_kWh * temperature_2m + Usage_kWh * relative_humidity_2m + Usage_kWh * precipitation"

# Fit the ANOVA model
model = smf.ols(formula, data=df).fit()
anova_table = sm.stats.anova_lm(model, typ=2)

print(anova_table)
```



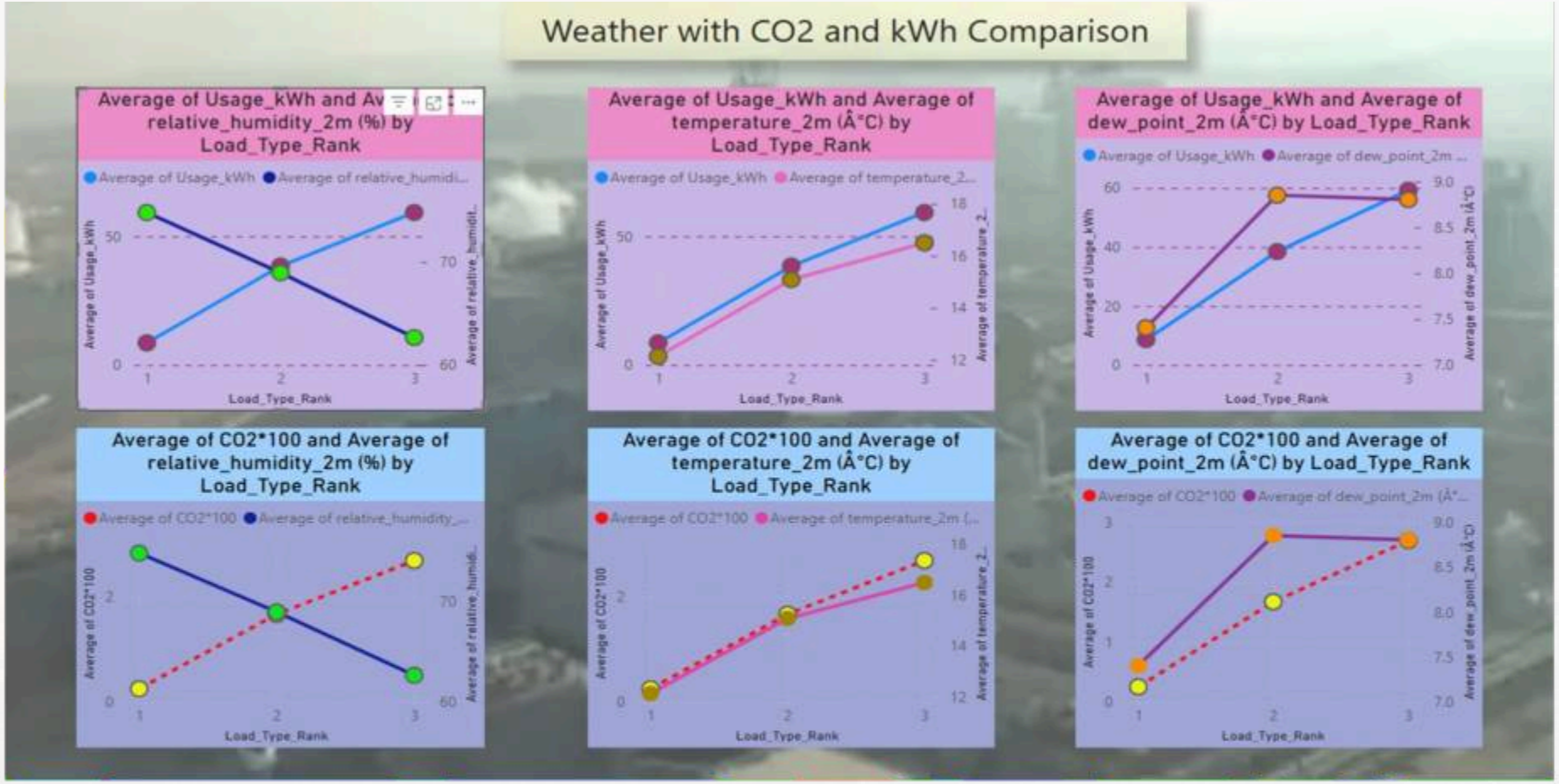
	sum_sq	df	F
Usage_kWh	8.226553e+00	1.0	1.348491e+06
temperature_2m	4.476525e-04	1.0	7.337892e+01
Usage_kWh:temperature_2m	3.474503e-05	1.0	5.695383e+00
relative_humidity_2m	2.218008e-05	1.0	3.635745e+00
Usage_kWh:relative_humidity_2m	4.227809e-04	1.0	6.930199e+01
precipitation	2.653875e-07	1.0	4.350216e-02
Usage_kWh:precipitation	1.543927e-05	1.0	2.530795e+00
Residual	2.137148e-01	35032.0	NaN

	PR(>F)
Usage_kWh	0.000000e+00
temperature_2m	1.113042e-17
Usage_kWh:temperature_2m	1.701487e-02
relative_humidity_2m	5.655948e-02
Usage_kWh:relative_humidity_2m	8.751208e-17
precipitation	8.347838e-01
Usage_kWh:precipitation	1.116530e-01
Residual	NaN

🔍 Interpretation Notes

- No Surprise, `Usage_kWh` **is the dominant driver** of CO_2 variation
- **Interactions with temperature and humidity are crucial**: they reveal that energy usage affects CO_2 differently depending on these environmental conditions.
- **Precipitation seems negligible here**, both on its own and in interaction with `Usage_kWh`.

Section 3: Feature Engineering and Diagnostics



Key Points:

- **Energy and CO_2 move in lockstep,** confirming physical expectations and past analysis
- **Humidity's inverse trajectory highlights a possible interaction effect,** where drier conditions align with higher energy and emissions with the crossing point at Rank 2 marking a shift in environmental influence.
- **Dew point show a slight decoupled patterns**, reinforcing that **drier air correlates with higher energy demand**
- **Load Rank 2 consistently emerges as a pivot point**, visually and statistically, where operational and environmental forces intersect

HEATMAPS

Humidity x Temp Interaction (CO2 and Usage kWh)

Humidiy Bins by Temperature Bins

CO2x100 Values

humiditybinby10	-15.00	-10.00	-5.00	0.00	5.00	15.00	20.00	25.00	30.00	Total
Total	1.5237	1.0548	1.0970	1.0906	1.1431	1.0721	1.1290	1.2871	1.8530	1.1524
0			0.0000	0.5000						0.4286
10		0.3333	0.8636	1.4250						1.0357
20	2.7500	1.8147	1.3077	2.3550		2.8125	2.1667	2.6250		1.9038
30	2.7813	1.2270	1.6452	1.8606	2.7034	1.9750	2.9211	2.5714		1.9403
40	1.6445	1.4566	1.8406	1.3413	2.1763	2.0131	1.9511	1.2823	0.3438	1.7433
50	1.4670	1.0351	1.0193	1.4146	1.7173	2.0519	1.4733	1.8488	1.7599	1.5242
60	0.2941	0.5678	0.9167	1.2609	1.1852	1.2630	1.5654	1.8208	1.8581	1.2916
70	0.0000	0.3708	0.7128	0.8191	0.8733	0.9317	1.3878	1.6445	2.3636	1.0732
80		0.2100	0.9311	0.3827	0.7325	0.5851	1.0577	0.9368	2.6250	0.7443
90			0.8153	1.0096	0.6266	0.5688	0.7148	0.8796		0.7278
100			1.5000	0.0000	1.0769	0.2594	0.6637			0.6054

Humidty Bins by Temperature Bins

Usage kWh (Avg)

humiditybinby10	-15.00	-10.00	-5.00	0.00	5.00	15.00	20.00	25.00	30.00	Total
0			3.63	13.82						12.36
10		13.65	21.39	37.81						25.42
20		41.61	31.13	53.76						43.40
30		28.57	42.79	44.25		44.16				44.37
40	35.72	33.34	42.60	35.44	48.37	45.64	43.15	30.37	11.58	40.20
50	34.06	25.10	25.05	36.62	39.51		33.29			35.10
60	10.72	15.18	22.53	30.54	28.16	29.37	35.43			30.22
70	4.89	16.32	18.93	20.42	21.47	22.35				25.66
80		8.43	23.47	11.57	18.54	15.39	24.84	22.55		18.80
90			20.41	24.61	16.12	15.12	17.98	21.28		18.34
100			16.07	1.36	25.90	8.65	16.84			15.80
Total	34.41	25.56	26.73	26.77	27.10	25.54	26.48	29.81	41.81	27.39

Analysis and Key Takeaways

Hot Zones of Inefficiency"

Heatmaps reveal two distinct emission and energy usage spikes: one in moderate humidity (20–40%) and another in high humidity (70–80%) at elevated temperatures. These zones may reflect both optimal production and environmental strain which is highlighting the need for targeted ventilation or load-balancing strategies.

Energy Variables and the Definitions

While the initial focus centered on environmental weather variables, the heatmap reveals strong correlations with several energy measurements from the steel plant. These patterns are too meaningful to ignore. It's time to explore those energy metrics in more depth and integrate them into the upcoming models.

🔁 Lagging Current Reactive Power (kVarh)

Represents reactive power consumed by **inductive loads** (e.g., motors, transformers).
Does not contribute directly to useful work but impacts **system efficiency**.
Higher lagging reactive power leads to **lower power factor** and increased energy losses.

🔄 Lagging Current Power Factor

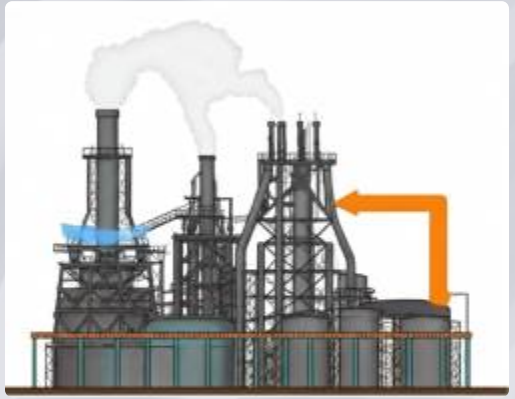
Power factor is the ratio of **real power (kWh)** to **apparent power (kVA)**.
A **lagging power factor** occurs when **current lags behind voltage** due to inductive behavior.
Lower power factor means **more wasted energy** and **higher electricity costs**.

⚡ Leading Current Reactive Power (kVarh)

Represents reactive power consumed by **capacitive loads** (e.g., capacitor banks).
Occurs when **current leads voltage**, opposite of inductive loads.
Excessive leading reactive power can also **reduce system efficiency**.

⚙️ Leading Current Power Factor

Measures the extent to which **current leads voltage** in a capacitive circuit.
Indicates **capacitive dominance** in the power system.
Excessively leading power factor may **overcorrect system behavior**, causing:
Voltage rise, Resonance issues, Reduced efficiency
May require **reactive inductive loads** (e.g., reactors) to stabilize the system.



⚡ Energy Usage (kWh)

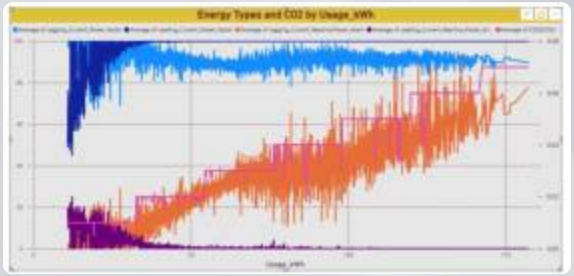
Represents the total amount of **real energy consumed** by the steel plant over time while capturing the core operational demand across all machinery and processes.

🏋️ Load Rank (1–3)

1	**Light Load** – Low production state, typically associated with minimal energy demand.
2	**Medium Load** – Standard production cycles; balanced usage across most equipment. Reflects routine activity levels.
3	**Max Load** – Peak energy demand; full-scale production with all systems engaged.

Energy Variables Graphs

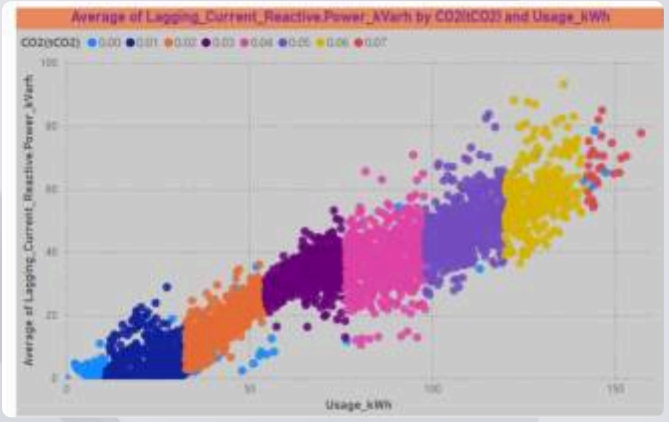
Multi-Variable Energy Scatter: Mapping Power Behavior and CO_2 Emissions



The **linear structure** underscores a steady relationship between usage and reactive load (Lagging Reactive Power kVarh Variable), indicating inductive demand behaving predictably.

The CO2 follows the same trend and is on top of the Lagging Reactive Power.

Energy Usage, Reactive Load, and CO_2



Color gradations represent CO_2 intensity, with warmer hues signaling rising emissions

The **stacked segments** hint at discrete operational modes and each with its own CO_2 footprint

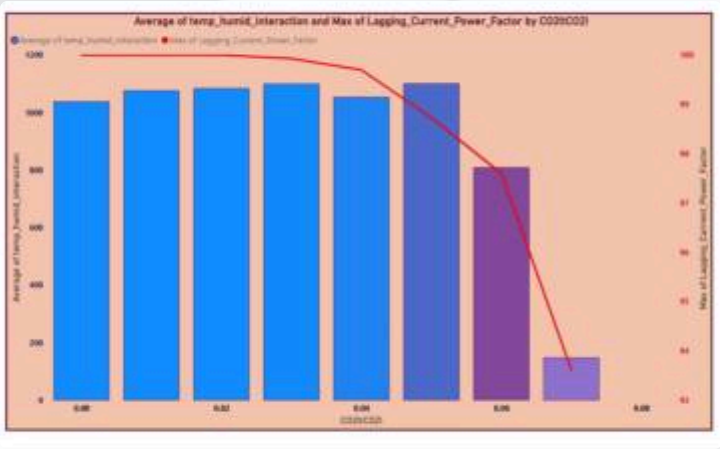
Outliers in light blue scattered throughout may be imputed gaps and unexplained behavior.

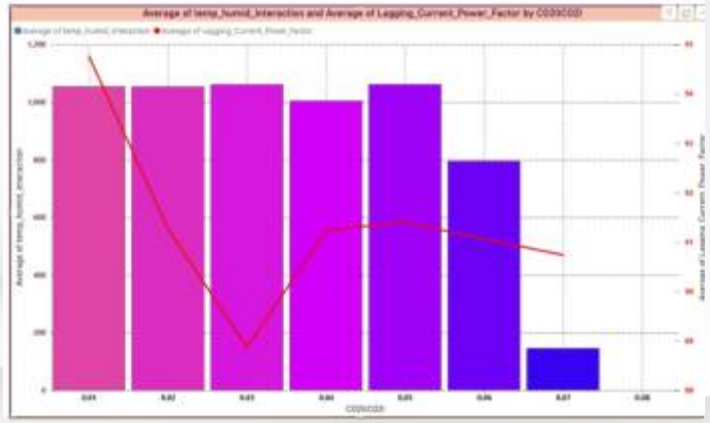
The Visual of Steelplant Dynamics

Each color stratum performs its part in the energy-emissions ensemble. The two Scatter plots charts **Usage kWh** against **Lagging Reactive Power kVarh**, forming a linear composition. The **color-coded bands**, segmented by CO_2 levels from light blue (0.0) to red (~0.07), reveal distinct operational layers across the plant's load profile.

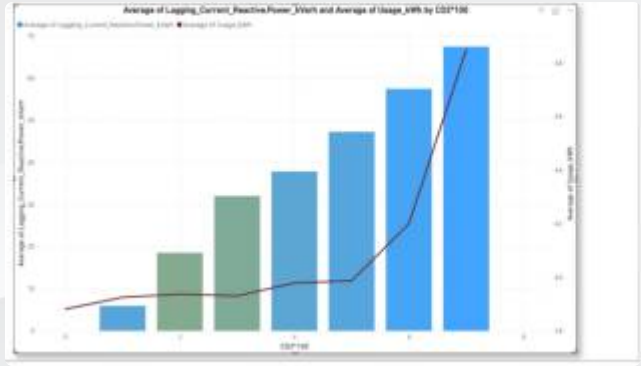
Lagging Current Reactive Power kVarh variable

Energy Behaviors Leads Emission Shifts with environmental weather influence

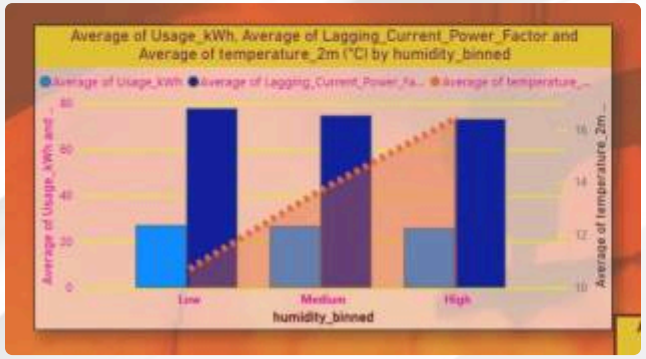




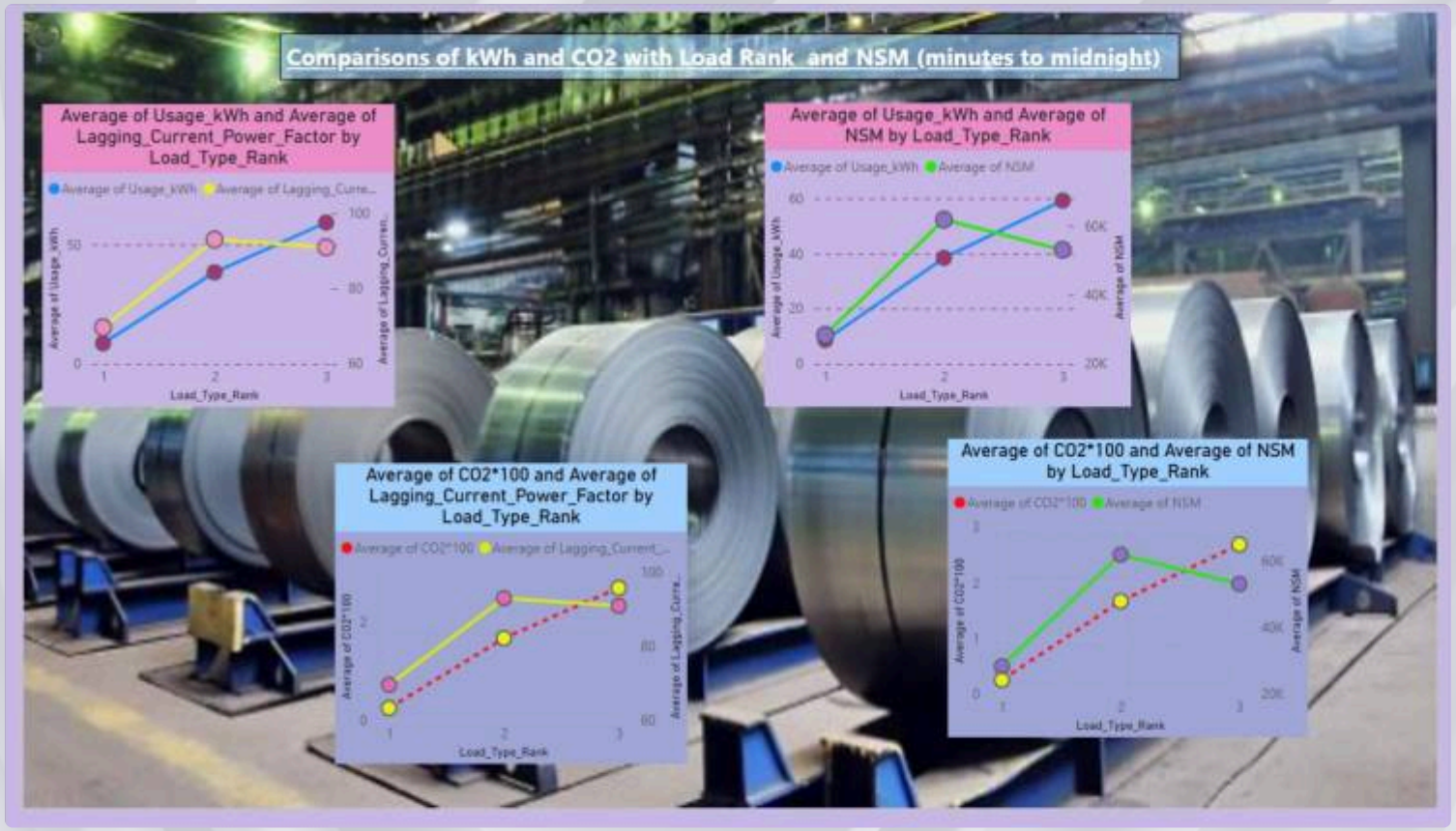
- The *max lagging* graph shows a **sharp drop** at high CO_2, implying system breakdowns or constraint thresholds.
- The *average lagging* graph adds **nuance**: it reveals that the system *partially rebounds* but doesn't return to its initial state, suggesting **nonlinear or hysteretic response**.
- Together, they suggest there's both a *disruption point* and a *stabilized inefficiency*, which is gold in a data story.



Both lagging energy and usage rise with CO_2, but there's a *nonlinear surge* at the highest emission levels.



Energy usage falls as humidity (and temperature) rise and is possibly cooling demand. The Cooling Effect of Moisture: Reactive and Real Energy Dip with Humidity?



Key points

- While energy usage climbs steadily with load rank, the timing of those loads' shifts; medium loads peak later, while max loads occur slightly earlier in the day.
- The Lagging Current Power Factor increases at level 2 and then also levels out, again could be due to timing of load shifts.

Key points

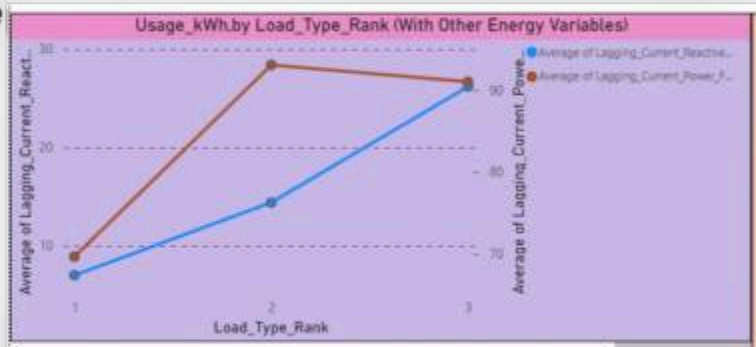
- The Lagging Current Reactive Power shows correlation system load, not as strong as Energy usage or CO2
- High CO_2 emission periods demonstrate sharp drops in lagging power, suggesting system constraints or operational thresholds.
- Medium loads demonstrate different timing patterns compared to maximum loads, affecting the overall power factor efficiency.
- The data reveals both disruption points and stabilized inefficiency regions, providing valuable insights for optimization strategies.

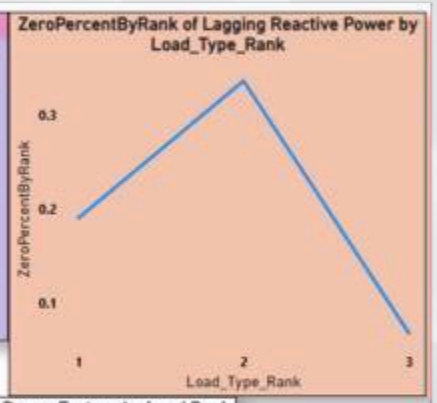
Lagging Power Factor by Load Rank

Comparing two key electrical power metrics and their behavior patterns

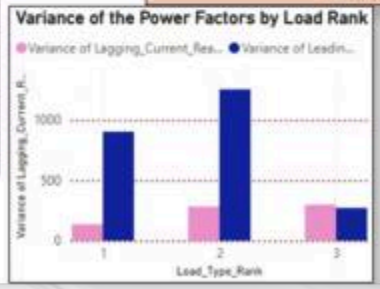
Lagging Reactive Power (kVArh)

Definition: Total *reactive energy* used (inductive demand)
Load Rank Trend: Increases steadily (~45° linear climb)
Zero Values: High (7,194) zeros and many load periods with draw no reactive energy
Distribution: Right-skewed with P25 = 2.30, P75 = 22.64
Variability: Standard deviation ~16.3 (moderate spread)





Lagging Power Factor (unitless)	Stats /Metrics	Lagging Reactive Power (kVArh)
63.32	25th Percentile (P25)	2.3
99.02	75th Percentile (P75)	22.64
~35.7	Interquartile Range (IQR)	~20.34
100	Maximum	96.91
80.58	Mean	13.04
87.96	Median	5
0	Minimum	0
18.92	Std Deviation	16.31
1	Zeros Count	7194



Lagging Power Factor (unitless)

Definition: Efficiency of power usage (kWh ÷ kVA)
Load Rank Trend: Starts near zero, rises quickly, levels by rank 3
Zero Values: Only 1 zero and PF typically registers even at low loads
Distribution: Wide efficiency range with P25 = 63.32, P75 = 99.02
Variability: Standard deviation ~18.9 (broad range; nonlinear distribution)

Section 4: Modeling Landscape: Structural, Temporal, and Behavioral Insights

While not every model is featured here, each played a role in shaping the final lineup. The modeling journey was iterative, exploratory, and occasionally messy, but it led to the clarity and performance showcased in the next slides. They are a distilled selection from a much larger pool of models and diagnostics still included in the portfolio.

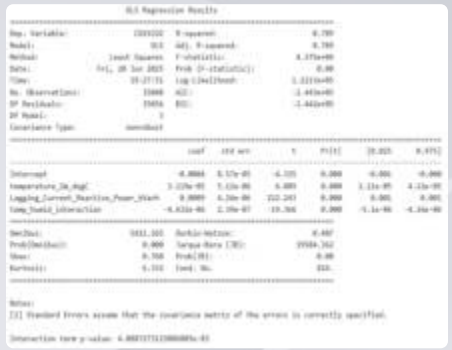
Early SARIMAX and ARIMA fits guided much of the rhythm-based exploration, even though they aren't fully presented due to memory limitations and partial diagnostics. Their influence is woven into the time-aware analysis that follow and are reflected in weekday/weekend behaviors, temporal features, and log_kWh modeling.

Model REady?

OLS Regression Model for CO2

As I began modeling, my exploratory analysis continued in parallel. While initial insights shaped early model choices, I remained open to discovering new patterns and refining features throughout the process. This iterative approach allowed for deeper understanding of environmental drivers and ensured each model was informed by the evolving data story.

First model with focus on the Enviromental Weather Variables and Focus

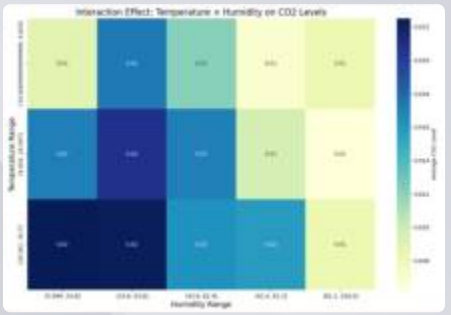


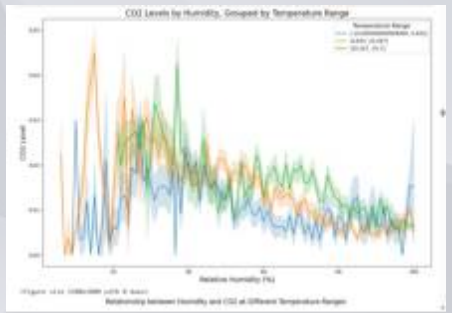
🔍 Interpretation Notes

- Modell explains **~78.9% of variance in CO2**
- The **interaction effect is powerfully negative**, indicating that **high humidity dampens the temperature-driven CO2 rise**, possibly due to altered ventilation or energy dynamics.
- The t-statistic for reactive power is high and it's an important driver.
- **Durbin-Watson (~ 0.41)** shows strong positive autocorrelation in residuals and maybe CO_2 has time-lagged patterns. Will address via ARIMA/SARIMA models later.

📈 Regression Highlights

Variable	Coef	P-value	Insight
Intercept	**-0.0004**	**< 0.001**	**Slight negative baseline CO_2 offset**
Temp (°C)	**3.119e-05**	**< 0.001**	**↑ Temp → slight increase in CO_2**
Reactive Power (kVarh)	**0.0009**	**≈ 0**	**Strong positive CO_2 association**
Temp × Humidity Interaction	**-4.63e-06**	**≈ 0**	**Highly significant suppressive effect**





OLS Regression Model Continues for CO_2

Continue to understand environmental variables and how much additional variance is explained by the power consumption variables.

Will multicollinearity be reduced with added the different Energy variables?

```
# Step 2: Add power variables one by one, keeping those that improve the model
power_vars = [
    'Lagging_Current_Reactive.Power_kVarh',
    'Leading_Current_Reactive_Power_kVarh',
    'Lagging_Current_Power_Factor',
    'Leading_Current_Power_Factor']
```

Final Model Summary:

OLS Regression Results

Dep. Variable:	CO2(tCO2)	R-squared:	0.897
Model:	OLS	Adj. R-squared:	0.897
Method:	Least Squares	F-statistic:	5.106e+04
Date:	Fri, 20 Jun 2025	Prob (F-statistic):	0.00
Time:	19:27:53	Log-Likelihood:	1.3474e+05
No. Observations:	35040	AIC:	-2.695e+05
Df Residuals:	35033	BIC:	-2.694e+05
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0381	0.000	-127.056	0.000	-0.039	-0.038
temperature_2m (°C)	-0.0001	1.06e-05	-9.898	0.000	-0.000	-8.41e-05
relative_humidity_2m (%)	-5.425e-05	2.44e-06	-22.247	0.000	-5.9e-05	-4.95e-05
temp_humid_interaction	2.486e-06	1.53e-07	16.295	0.000	2.19e-06	2.78e-06
Lagging_Current_Reactive.Power_kVarh	0.0007	2.13e-06	322.749	0.000	0.001	0.001
Lagging_Current_Power_Factor	0.0004	2.05e-06	179.583	0.000	0.000	0.000
Leading_Current_Power_Factor	0.0002	1.32e-06	123.627	0.000	0.000	0.000

Omnibus:	4711.603	Durbin-Watson:	0.520
Prob(Omnibus):	0.000	Jarque-Bera (JB):	61639.818
Skew:	0.103	Prob(JB):	0.00
Kurtosis:	9.494	Cond. No.	1.44e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.44e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Variance Inflation Factors for Final Model:

	Variable	VIF
0	const	117.730342
1	temperature_2m (°C)	14.726059
2	relative_humidity_2m (%)	3.099169
3	temp_humid_interaction	19.660000
4	Lagging_Current_Reactive.Power_kVarh	1.577807
5	Lagging_Current_Power_Factor	1.959951
6	Leading_Current_Power_Factor	2.112516

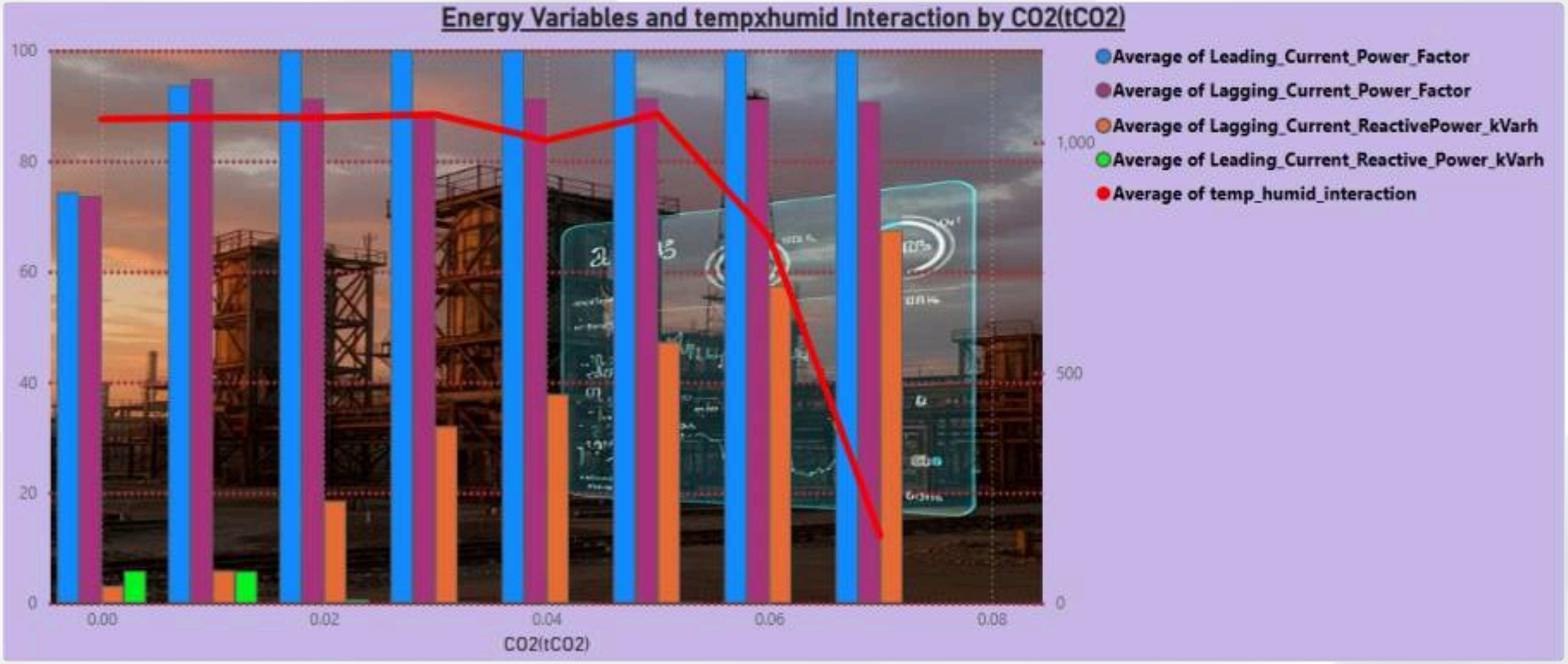
Interpretation

- The model explains ~90% of the variance in CO_2, driven primarily by power factors and reactive draw.
- While some multicollinearity exists among weather terms, especially the tempxhumidity interaction, it contributes significant explanatory power.
- Durbin-Watson value (~0.52) and non-normality hints (JB test) suggest residual autocorrelation and kurtosis
- Diagnostic indicators such as VIF remain within tolerable bounds for most variables, suggesting model stability.

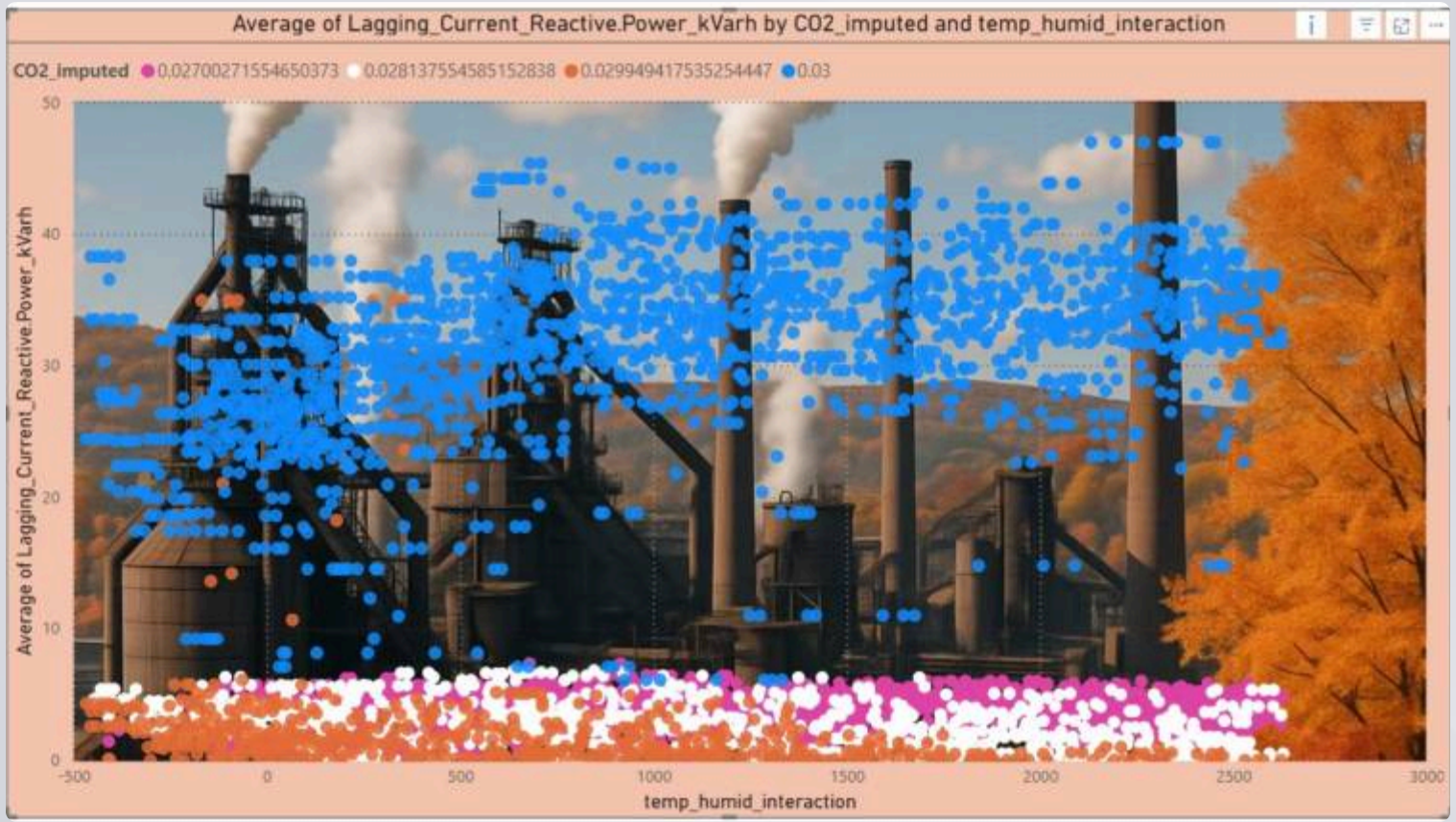
Power vs. CO_2 Levels (TempxHumid)

Observation: While most power metrics increase predictably with rising CO_2 levels, leading current reactive power behaves differently. It remains suppressed at 0.00 and 0.01, barely emerges at 0.02, and then disappears again. This discontinuity may reflect system constraints, sensor thresholds, or maybe binning artifacts.

Interpretation: The faint emergence at 0.02 suggests a nonlinear threshold—possibly the point where capacitive behavior becomes detectable. Its absence at other levels may indicate operational suppression or measurement rounding.

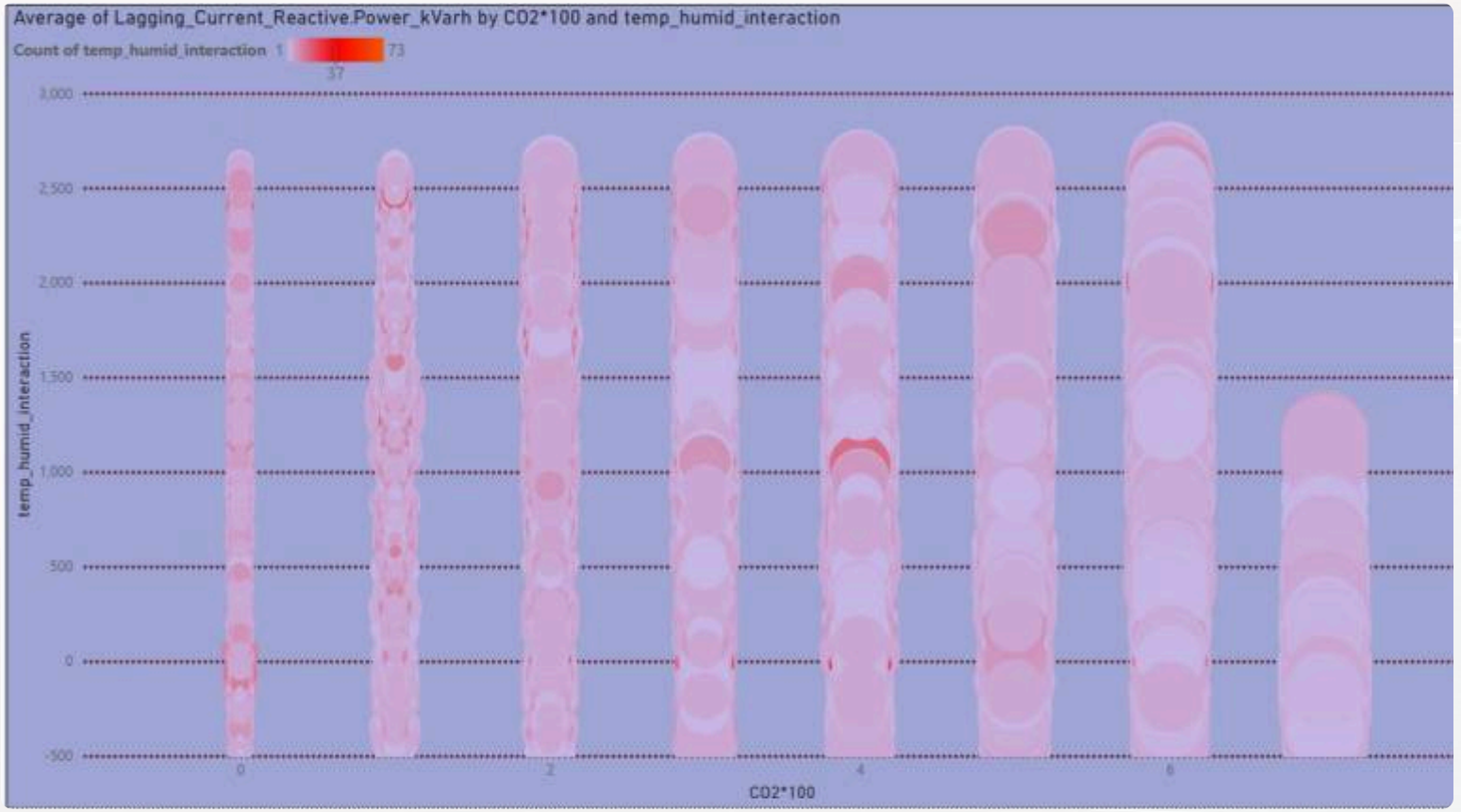


Diagnostic Interpretation: Lagging Reactive Power vs. Imputed CO_2



Observation: Across imputed CO_2 levels (0.00 to 0.03), lagging current reactive power remains low, except at **0.03**, where it spikes and clusters at higher values. This suggests a **threshold effect**, where the system shifts behavior once CO_2 reaches a critical level.

Environmental Interaction: Despite expectations, the temp × humidity interaction appears **non-influential** in this context. This reinforces the idea that **internal system load or control logic**, not environmental conditions, is driving reactive power behavior at high CO_2.



Stacked Signals: Reactive Power Density Across CO_2 Regimes

This stacked scatter reveals nonlinear shifts in reactive power behavior.

- Low CO_2 levels show minimal engagement
- Mid-range levels (4–6) reveal sustained inefficiency zones
- Level 7 suggests a short-duration spike worth diagnostic review Environmental interaction appears secondary and reactive behavior is primarily CO_2-driven.

OLS Regression Model Continues for Energy Usage kWh

Following the same model as previous CO_2

Enviroment Variables Model with kWh:

OLS Regression Results

Dep. Variable:	Usage_kWh	R-squared:	0.078
Model:	OLS	Adj. R-squared:	0.078
Method:	Least Squares	F-statistic:	994.1
Date:	Fri, 20 Jun 2025	Prob (F-statistic):	0.00
Time:	19:27:53	Log-Likelihood:	-1.7127e+05
No. Observations:	35040	AIC:	3.426e+05
Df Residuals:	35036	BIC:	3.426e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	51.9243	0.941	55.165	0.000	50.079	53.769
temperature_2m (°C)	0.8115	0.065	12.462	0.000	0.684	0.939
relative_humidity_2m (%)	-0.4258	0.015	-28.846	0.000	-0.455	-0.397
temp_humid_interaction	-0.0054	0.001	-5.761	0.000	-0.007	-0.004

Omnibus:	5439.569	Durbin-Watson:	0.191
Prob(Omnibus):	0.000	Jarque-Bera (JB):	8319.668
Skew:	1.151	Prob(JB):	0.00
Kurtosis:	3.631	Cond. No.	7.26e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.26e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

Results of adding each power variable with y as kWh:

	Variable	AIC	R-squared
0	Lagging_Current_Reactive.Power_kVarh	285385.626440	0.819736
1	Leading_Current_Reactive_Power_kVarh	285362.129974	0.819867
2	Lagging_Current_Power_Factor	265106.554013	0.898955
3	Leading_Current_Power_Factor	260115.366493	0.912375

Final Model Summary:

OLS Regression Results

Dep. Variable:	Usage_kWh	R-squared:	0.912
Model:	OLS	Adj. R-squared:	0.912
Method:	Least Squares	F-statistic:	5.211e+04
Date:	Mon, 30 Jun 2025	Prob (F-statistic):	0.00
Time:	13:14:07	Log-Likelihood:	-1.3005e+05
No. Observations:	35040	AIC:	2.601e+05
Df Residuals:	35032	BIC:	2.602e+05
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-80.5542	0.737	-109.365	0.000	-81.998	-79.111
temperature_2m (°C)	-0.2154	0.020	-10.628	0.000	-0.255	-0.176
relative_humidity_2m (%)	-0.1100	0.005	-23.576	0.000	-0.119	-0.101
temp_humid_interaction	0.0048	0.000	16.317	0.000	0.004	0.005
Lagging_Current_Reactive.Power_kVarh	1.4679	0.004	357.526	0.000	1.460	1.476
Leading_Current_Reactive_Power_kVarh	0.3667	0.022	16.556	0.000	0.323	0.410
Lagging_Current_Power_Factor	0.7395	0.004	186.807	0.000	0.732	0.747
Leading_Current_Power_Factor	0.3979	0.005	73.248	0.000	0.387	0.409

Omnibus:	5568.028	Durbin-Watson:	0.378
Prob(Omnibus):	0.000	Jarque-Bera (JB):	19907.101
Skew:	0.784	Prob(JB):	0.00
Kurtosis:	6.343	Cond. No.	1.85e+04

Variance Inflation Factors for Final Model:

	Variable	VIF
0	const	193.918969
1	temperature_2m (°C)	14.739592
2	relative_humidity_2m (%)	3.099453
3	temp_humid_interaction	19.660012
4	Lagging_Current_Reactive.Power_kVarh	1.601896
5	Leading_Current_Reactive_Power_kVarh	9.667420
6	Lagging_Current_Power_Factor	2.005182
7	Leading_Current_Power_Factor	9.785265

****Interpretation of Model output on the next slide**

OLS Regression for Energy Usage kWh

Model Performance Interpretation

Environment Variables Only

R^2 Score: 0.078

AIC: 342,600

Very low explanatory power — weather alone doesn't account for energy demand shifts

Add Power Factors Incrementally

R^2 Score: Up to 0.912

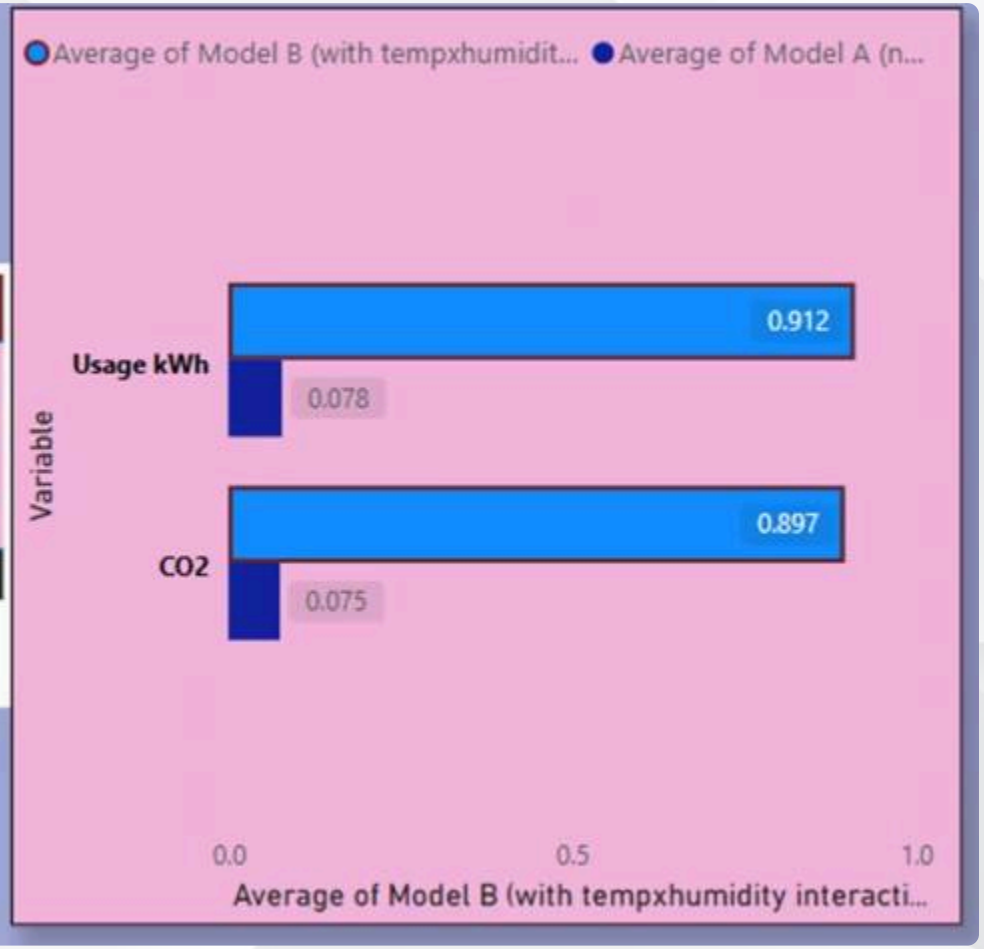
AIC: ↓ to 260,100

Dramatic boost — power dynamics dominate energy usage patterns

Adding power variables to the environmental baseline yields a 12x increase in explanatory strength (R^2 from 0.078 to 0.912). Among them, reactive power and power factor dominate, reinforcing the operational rather than environmental drivers of energy use. High VIFs among interaction and weather terms reflect their expected overlap, but don’t compromise model validity.

Overview of with and without Interaction (tempxhumidity)

Metric	Model A (no interaction)	Model B (with Interaction)
R^2 (CO_2)	0.075	0.897
R^2 (Usage_kWh)	0.078	0.912
AIC (CO_2)	−192,439	−269,194
AIC (Usage_kWh)	342,587	260,115
Total		



OLS Regression Model for CO_2

Dropping the tempxhumid interaction

NOTES: With the High VIF for tempxhumid interaction (19.66) and temperature_2m (14.73) suggesting multicollinearity, I decided to drop the tempxhumid interaction, but will revisit later and maybe standardize the value. For the final model, I also dropped the Leading_Current_Reactive_Power_kVarh based on its lower coefficients and higher p-value.

```
Environmental Variables Model (*without interaction of tempxhumd*):
                            OLS Regression Results
==============================================================================
Dep. Variable:                CO2(tCO2)   R-squared:                       0.075
Model:                            OLS   Adj. R-squared:                  0.075
Method:                 Least Squares   F-statistic:                     1430.
Date:                Mon, 30 Jun 2025   Prob (F-statistic):               0.00
Time:                        13:14:07   Log-Likelihood:                 96223.
No. Observations:               35040   AIC:                        -1.924e+05
Df Residuals:                   35037   BIC:                        -1.924e+05
Df Model:                           2
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                            0.0249      0.000     81.398      0.000       0.024       0.025
temperature_2m (°C)              0.0002   8.81e-06     25.838      0.000       0.000       0.000
relative_humidity_2m (%)        -0.0002   4.42e-06    -52.825      0.000      -0.000      -0.000
==============================================================================
Omnibus:                     4964.986   Durbin-Watson:                   0.207
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7345.697
Skew:                           1.099   Prob(JB):                         0.00
Kurtosis:                       3.446   Cond. No.                         276.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
R-squared: 0.07545718474560481
AIC: -192439.8510152099
Power variables found in dataframe: ['Lagging_Current_Reactive.Power_kVarh', 'Leading_Current_Reactive_Power_kVarh', 'Lagging_Current_Power_Factor', 'Leading_Current_Power_Factor']
```

```
Results of adding each power variable:
                                Variable            AIC  R-squared  \
0  Lagging_Current_Reactive.Power_kVarh -246448.152903   0.802071
1  Leading_Current_Reactive_Power_kVarh -246446.611145   0.802073
2          Lagging_Current_Power_Factor -256644.940135   0.852054
3          Leading_Current_Power_Factor -269194.210347   0.896595

   Coefficient  P-value
0     0.000859  0.00000
1    -0.000004  0.49848
2     0.000210  0.00000
3     0.000163  0.00000

Final Model Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                CO2(tCO2)   R-squared:                       0.897
Model:                            OLS   Adj. R-squared:                  0.897
Method:                 Least Squares   F-statistic:                 6.075e+04
Date:                Mon, 30 Jun 2025   Prob (F-statistic):               0.00
Time:                        13:14:07   Log-Likelihood:                 1.3460e+05
No. Observations:               35040   AIC:                        -2.692e+05
Df Residuals:                   35034   BIC:                        -2.691e+05
Df Model:                           5
Covariance Type:            nonrobust
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.43e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
Final model R-squared: 0.8965955598042129
Final model AIC: -269194.21034717787

Variance Inflation Factors for Final Model:
                                Variable         VIF
0                                  const  186.828158
1                  temperature_2m (°C)    1.287647
2               relative_humidity_2m (%)    1.177583
3  Lagging_Current_Reactive.Power_kVarh    1.577468
4          Lagging_Current_Power_Factor    1.948874
5          Leading_Current_Power_Factor    2.111558
```

**More Model output and Interpretation on the next slide

FINAL OLS Regression Model for CO_2

Without tempxhumid interaction and Leading Current Reactive Power

Final Model Summary:

OLS Regression Results

Dep. Variable:	CO2(tCO2)	R-squared:	0.897
Model:	OLS	Adj. R-squared:	0.897
Method:	Least Squares	F-statistic:	6.075e+04
Date:	Mon, 30 Jun 2025	Prob (F-statistic):	0.00
Time:	13:14:07	Log-Likelihood:	1.3460e+05
No. Observations:	35040	AIC:	-2.692e+05
Df Residuals:	35034	BIC:	-2.691e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0397	0.000	-138.790	0.000	-0.040	-0.039
temperature_2m (°C)	6.051e-05	3.04e-06	19.880	0.000	5.45e-05	6.65e-05
relative_humidity_2m (%)	-2.463e-05	1.63e-06	-15.095	0.000	-2.78e-05	-2.14e-05
Lagging_Current_Reactive.Power_kVarh	0.0007	2.14e-06	321.334	0.000	0.001	0.001
Lagging_Current_Power_Factor	0.0004	2.05e-06	178.204	0.000	0.000	0.000
Leading_Current_Power_Factor	0.0002	1.32e-06	122.845	0.000	0.000	0.000

Omnibus:	4719.602	Durbin-Watson:	0.516
Prob(Omnibus):	0.000	Jarque-Bera (JB):	60217.016
Skew:	0.137	Prob(JB):	0.00
Kurtosis:	9.416	Cond. No.	1.43e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.43e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Final model R-squared: 0.8965953598042129
Final model AIC: -269194.21034717787

Variance Inflation Factors for Final Model:

	Variable	VIF
0	const	106.010198
1	temperature_2m (°C)	1.207647
2	relative_humidity_2m (%)	1.377503
3	Lagging_Current_Reactive.Power_kVarh	1.577468
4	Lagging_Current_Power_Factor	1.949874
5	Leading_Current_Power_Factor	2.111558

Model Stage	R^2	AIC	Key Insight
Environment-only (No Interaction)	0.075	-192,440	Low explanatory power — temp & humidity contribute separately but weakly
Final CO_2 Model (w/o Interaction)	0.897	-269,194	Still excellent — power variables shoulder almost all explanatory weight
VIF Scores	< 2.2 for all terms	👍	Strong signal, low multicollinearity without the interaction term

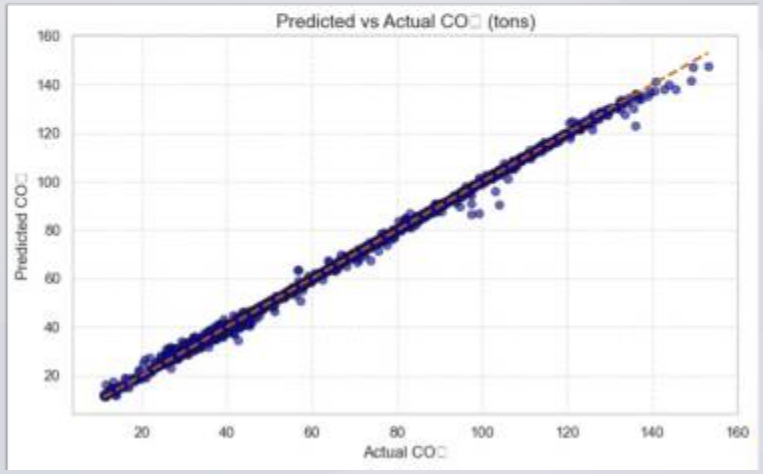
Interpretation

- Even without the temp × humidity interaction, environment-only model had minimal predictive strength.
- Adding power metrics, especially Lagging Reactive Power and both Power Factors, the model nearly recreates the full strength of the prior version with the interaction.
- Importantly, VIFs dropped, especially for temperature (from ~14.7 to ~1.2), meaning removing the interaction resolved multicollinearity cleanly without hurting model performance.
- The modeling suggest that for **CO_2 model**, humidity and temperature matter ***primarily through their shared impact on operations or power efficiency***, already captured by the power variables.

The OLS Model: Final Fit

Simplicity Earned

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, alpha=0.6, color='mediumblue', edgecolor='k')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.title("Predicted vs Actual CO₂ (tons)", fontsize=14)
plt.xlabel("Actual CO₂", fontsize=12)
plt.ylabel("Predicted CO₂", fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



While nonlinear models captured complexity across regimes and environmental layers, the final OLS regression delivered elegant simplicity. With well-behaved inputs and a refined feature set, the predicted vs. actual CO_2 plot shows near-perfect alignment with each point tightly clustered along the diagonal. This fit reflects not just statistical performance, but how far the modeling journey has come: from messy signals to interpretable clarity.

OLS Regression Model Continues for Energy Usage kWh

Without tempxhumid interaction

Environmental Variables Model (*without interaction of tempxhumd*):

OLS Regression Results

Dep. Variable:	Usage_kWh	R-squared:	0.078
Model:	OLS	Adj. R-squared:	0.078
Method:	Least Squares	F-statistic:	1473.
Date:	Mon, 30 Jun 2025	Prob (F-statistic):	0.00
Time:	13:14:07	Log-Likelihood:	-1.7129e+05
No. Observations:	35040	AIC:	3.426e+05
Df Residuals:	35037	BIC:	3.426e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	55.9434	0.632	88.493	0.000	54.707	57.183
temperature_2m (°C)	0.4514	0.018	24.767	0.000	0.416	0.487
relative_humidity_2m (%)	-0.4926	0.009	-53.852	0.000	-0.510	-0.475

Omnibus:	5405.050	Durbin-Watson:	0.191
Prob(Omnibus):	0.000	Jarque-Bera (JB):	8245.949
Skew:	1.147	Prob(JB):	0.00
Kurtosis:	3.620	Cond. No.	276.

Notes:
[1] Standard Errors assuee that the covariance matrix of the errors is correctly specified.

R-squared: 0.07756887262074719

AIC: 342587.64625598626

Power variables found in dataframe: ['Lagging_Current_Reactive.Power_kVarh', 'Leading_Current_Reactive_Power_kVarh', 'Lagging_Current_Power_Factor', 'Leading_Current_Power_Factor']

Results of adding each power variable:

	Variable	AIC	R-squared
0	Lagging_Current_Reactive.Power_kVarh	285387.734992	0.819715
1	Leading_Current_Reactive_Power_kVarh	285364.892743	0.819843
2	Lagging_Current_Power_Factor	265316.215698	0.898343
3	Leading_Current_Power_Factor	260378.672452	0.911709

	Coefficient	P-value
0	1.797456	0.000000e+00
1	0.057878	6.234469e-07
2	0.691577	0.000000e+00
3	0.397114	0.000000e+00

Final Model Summary:

OLS Regression Results

Dep. Variable:	Usage_kWh	R-squared:	0.912
Model:	OLS	Adj. R-squared:	0.912
Method:	Least Squares	F-statistic:	6.829e+04
Date:	Mon, 30 Jun 2025	Prob (F-statistic):	0.00
Time:	13:14:07	Log-Likelihood:	-1.3018e+05
No. Observations:	35040	AIC:	2.604e+05
Df Residuals:	35033	BIC:	2.604e+05
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-83.5150	0.717	-116.550	0.00	-84.919	-82.110
temperature_2m (°C)	0.1013	0.006	17.307	0.00	0.090	0.113
relative_humidity_2m (%)	-0.0533	0.003	-17.056	0.00	-0.059	-0.047
Lagging_Current_Reactive.Power_kVarh	1.4669	0.004	355.983	0.00	1.459	1.475
Leading_Current_Reactive_Power_kVarh	0.3670	0.022	16.507	0.00	0.323	0.411
Lagging_Current_Power_Factor	0.7349	0.004	185.417	0.00	0.727	0.743
Leading_Current_Power_Factor	0.3971	0.005	72.826	0.00	0.386	0.408

Omnibus:	5744.361	Durbin-Watson:	0.375
Prob(Omnibus):	0.000	Jarque-Bera (JB):	20236.442
Skew:	0.815	Prob(JB):	0.00
Kurtosis:	6.349	Cond. No.	1.88e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.88e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Final model R-squared: 0.9117008252395367

Final model AIC: 260378.6724521199

Variance Inflation Factors for Final Model:

	Variable	VIF
0	const	182.150594
1	temperature_2m (°C)	1.229505
2	relative_humidity_2m (%)	1.377753
3	Lagging_Current_Reactive.Power_kVarh	1.601561
4	Leading_Current_Reactive_Power_kVarh	9.667414
5	Lagging_Current_Power_Factor	1.595872
6	Leading_Current_Power_Factor	9.784439

Key Insights and Interpretation

Variable	R^2 ↑	AIC ↓	Coefficient	Interpretation
Lagging Current Reactive Power	0.820	285,388	1.80	Strong linear contribution to usage
Leading Reactive Power	0.820	285,365	0.06	Minimal but significant
Lagging Power Factor	0.898	265,316	0.69	Major driver of efficiency and demand
Leading Power Factor	0.912	260,379	0.40	Rounds out operational contribution

✅ **Final Model (Usage_kWh)**

(No Interaction + All Power Factors)

- **R^2 = 0.912 | AIC = 260,379**
- Power factors dominate; temperature becomes secondary
- All variables are **highly significant** ($p < 0.001$)
- **VIFs all < 10**, confirming multicollinearity is manageable

**When environmental variables are modeled without interaction effects, they contribute modestly to energy demand. However, once operational power factors are included, explanatory strength soars (R^2 = 91.2%), affirming energy usage is driven more by internal system dynamics than by ambient conditions*

OLS Regression Model Continues for Energy Usage kWh

Removing Leading_Current_Reactive_Power_kVarh variable

With the last model for Energy Usage kWh, I used the AIC for stepwise variable selection and surprisingly it kept the Leading_Current_Reactive_Power_kVarh variable. The Akaike Information Criterion (AIC) is a statistical measure that helps balance a model's goodness of fit with its complexity, guiding the selection of more parsimonious models. (Note, that the CO2 Model removed the variable. For this model, I am going to remove the variable above but still keep the AIC stepwise selection.)

```
Final Model Summary without 'Leading_Current_Reactive_Power_kVarh':
                            OLS Regression Results
==============================================================================
Dep. Variable:              Usage_kWh   R-squared:                       0.911
Model:                            OLS   Adj. R-squared:                  0.911
Method:                 Least Squares   F-statistic:                 7.174e+04
Date:                Mon, 30 Jun 2025   Prob (F-statistic):               0.00
Time:                        13:14:07   Log-Likelihood:            -1.3032e+05
No. Observations:               35040   AIC:                         2.606e+05
Df Residuals:                   35034   BIC:                         2.607e+05
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                                                 coef    std err          t      P>|t
|      [0.025      0.975]
------------------------------------------------------------------------------
------------------------
const                                        -75.8677      0.549   -138.252      0.00
0      -76.943      -74.792
temperature_2m (°C)                            0.0914      0.006     15.635      0.00
0        0.080        0.103
relative_humidity_2m (%)                      -0.0540      0.003    -17.213      0.00
0       -0.060       -0.048
Lagging_Current_Reactive.Power_kVarh           1.4585      0.004    355.277      0.00
0        1.450        1.467
Lagging_Current_Power_Factor                   0.7447      0.004    189.335      0.00
0        0.737        0.752
Leading_Current_Power_Factor                   0.3174      0.003    124.818      0.00
0        0.312        0.322
==============================================================================
Omnibus:                     5313.716   Durbin-Watson:                   0.365
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            17321.581
Skew:                           0.775   Prob(JB):                         0.00
Kurtosis:                       6.076   Cond. No.                     1.43e+03
------------------------------------------------------------------------------
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
[2] The condition number is large, 1.43e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
Final model R-squared: 0.9110223395235306
Final model AIC: 260648.1427865061

Variance Inflation Factors for Final Model:
                                Variable         VIF
0                                  const  106.010198
1                 temperature_2m (°C)     1.207647
2              relative_humidity_2m (%)     1.377503
3  Lagging_Current_Reactive.Power_kVarh     1.577468
4           Lagging_Current_Power_Factor     1.949874
5           Leading_Current_Power_Factor     2.111558
```

Key Insights and Interpretation

Metric	With Variable	Without Variable
R-squared	0.9124	0.911
AIC	260,115	260,648
Coefficient (Removed Var)	+0.3667 (significant)	❌ Removed
Multicollinearity Impact	VIF = 9.67 (approaching threshold)	Lower overall VIFs
Interpretation	Slightly stronger fit, but added redundancy	Cleaner model with minimal loss of fit

Removing the Leading Current Reactive Power_kVarh led to a minor dip in fit ($\Delta R^2 = 0.0014$) and slight AIC increase, but significantly reduced complexity and potential collinearity. The decision emphasizes model parsimony while maintaining a high explanatory strength.

NEXT STEPS for MODELS: Standardizing and Scaling Variable for Lasso and Cross -Validation

```
#print standardized predictors
print(X_scaled_df.head())
# Compare statistics
print("Unscaled Variables")
df[['temperature_2m (°C)', 'relative_humidity_2m (%)',
    'Lagging_Current_Reactive.Power_kVarh', 'Lagging_Current_Power_Factor',
    'Leading_Current_Power_Factor']].head()
```

```
   temperature_2m (°C)  relative_humidity_2m (%)  \
0            -1.512850                 -0.384125
1            -1.512850                 -0.384125
2            -1.512850                 -0.384125
3            -1.532816                 -0.283913
4            -1.532816                 -0.283913

   Lagging_Current_Reactive.Power_kVarh  Lagging_Current_Power_Factor  \
0                             -0.618516                     -0.389410
1                             -0.525911                     -0.729772
2                             -0.598270                     -0.566264
3                             -0.581106                     -0.660809
4                             -0.523450                     -0.838117

   Leading_Current_Power_Factor
0                      0.513268
1                      0.513268
2                      0.513268
3                      0.513268
4                      0.513268
Unscaled Variables
```

	temperature_2m (°C)	relative_humidity_2m (%)	Lagging_Current_Reactive.Power_kVarh	Lagging_Current_Power_Factor	Leading_Current_Power_Factor
0	-1.3	63.0	2.95	73.21	100.0
1	-1.3	63.0	4.46	66.77	100.0
2	-1.3	63.0	3.28	70.28	100.0
3	-1.5	65.0	3.56	68.09	100.0
4	-1.5	65.0	4.50	64.72	100.0

Good results on models but still shows potential multicollinearity. Next step is to run Lasso and Cross-Validation with other potential metrics. However, need to standardize variables

Standardization /scaling is a preprocessing step that prepares data for modeling.

- **LASSO penalizes based on coefficient size**, so a variable with larger units (e.g., kWh) could be unfairly penalized unless scaled
- **Cross-validation performance metrics** (like RMSE or MAE) will be more interpretable across folds
- **Multicollinearity effects** (e.g., VIF) also become easier to identify when all predictors are mean-centered and scaled. This ensures all subsequent analyses, (cross-validation, LASSO, metrics like BIC), use the same scaled data, avoiding inconsistencies.

CROSS -Validation

```
#Checking and picking my Models  (stable?)
# Cross-validation for CO2 model
model = LinearRegression()
cv_scores_co2 = cross_val_score(model, X_scaled_df, y_co2, cv=5, scoring='neg_m
mse_co2 = -cv_scores_co2.mean()
rmse_co2 = np.sqrt(mse_co2)
print(f"CO2 Model - 5-Fold CV MSE: {mse_co2:.6f}, RMSE: {rmse_co2:.6f}")

# Cross-validation for kWh model
cv_scores_kwh = cross_val_score(model, X_scaled_df, y_kwh, cv=5, scoring='neg_m
mse_kwh = -cv_scores_kwh.mean()
rmse_kwh = np.sqrt(mse_kwh)
print(f"kWh Model - 5-Fold CV MSE: {mse_kwh:.6f}, RMSE: {rmse_kwh:.6f}")

# Test model without Leading_Current_Power_Factor
X_reduced = X_scaled_df.drop(columns=['Leading_Current_Power_Factor'])
cv_scores_reduced = cross_val_score(model, X_reduced, y_kwh, cv=5, scoring='neg_
# Fixed: Calculate mean first, then negate
mse_reduced = -cv_scores_reduced.mean()  # Just take the mean of negative score
rmse_reduced = np.sqrt(mse_reduced)
print(f"kWh Model (without Leading_Current_Power_Factor) - CV MSE: {mse_reduced:
```

```
CO2 Model - 5-Fold CV MSE: 0.000031, RMSE: 0.005532
kWh Model - 5-Fold CV MSE: 115.588644, RMSE: 10.751216
kWh Model (without Leading_Current_Power_Factor) - CV MSE: 155.424307, RMSE: 1
2.466929
```

📉 Cross-Validation Results Summary

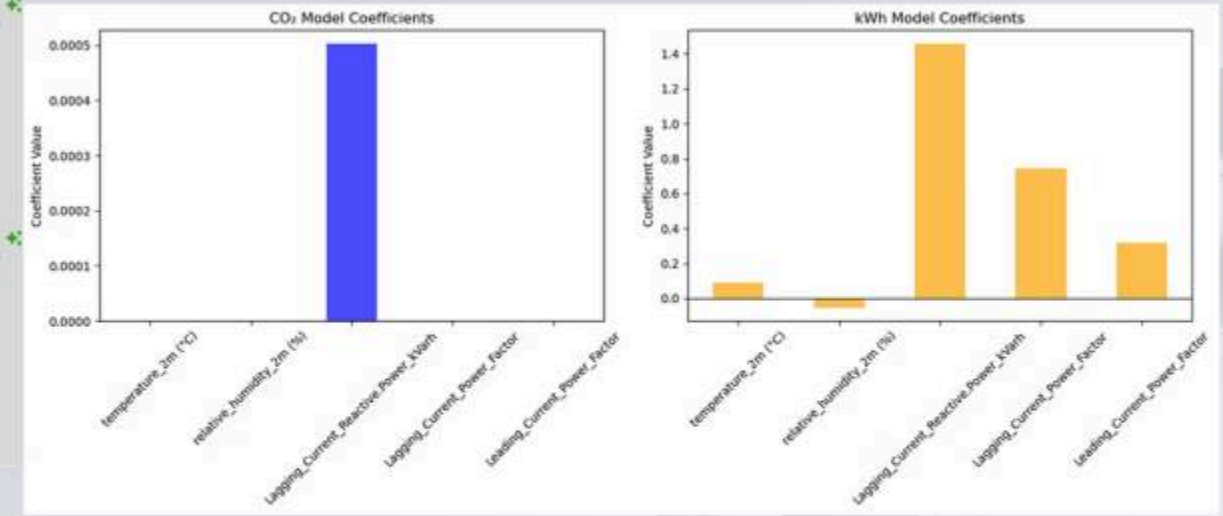
Model	CV MSE	CV RMSE	Interpretation
CO_2 Model	0.000031	0.0055	Incredibly tight error and model generalizes well on CO_2 prediction
kWh Model (full)	115.59	10.75	Solid performance and low error given operational variability
kWh Model (minus Leading PF)	155.42	12.47	Error rises sharply without Leading PF and confirms its explanatory power

Plot of Lasso's Coefficients for both CO_2 and Energy Usage kWh

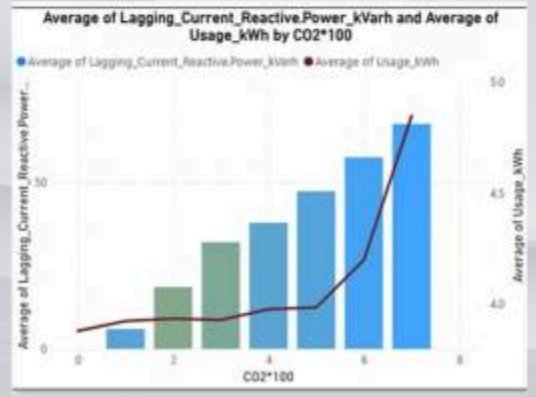
Overview: *Where energy demand reflects an intricate balance of power quality and environmental conditions, CO_2 emissions reduce to a single operational driver, reactive power. The LASSO coefficient comparison show their structural differences.*

```
#Coefficient Comparison Charts
#Using a side-by-side bar plot to show scaled LASSO coefficients for both CO₂ and kWh models
import matplotlib.pyplot as plt
import pandas as pd

# Assuming lasso_co2 and lasso_kwh are already defined and fitted

# Create the coefficient comparison dataframe
coefs_df = pd.DataFrame({
    'y_co2': lasso_co2.coef_,
    'y_kwh': lasso_kwh.coef_
}, index=predictors)
```


```
#Coefficient Comparison Charts
#Using a side-by-side bar plot to show scaled LASSO coefficients for both CO₂ and kWh models
import matplotlib.pyplot as plt
import pandas as pd

# Assuming lasso_co2 and lasso_kwh are already defined and fitted

# Create the coefficient comparison dataframe
coefs_df = pd.DataFrame({
    'y_co2': lasso_co2.coef_,
    'y_kwh': lasso_kwh.coef_
}, index=predictors)
```

Previous Comparison of Lagging Reactive Power, CO2 and Energy Usage kWh



CO_2 Model

- **Only Lagging Reactive Power survives LASSO penalty.**
- That single non-zero bar is emissions proxy: if it draws VARh, it emits CO_2.
- All other coefficients are shrunk to zero, including both power factors and environmental terms.

kWh Model (Energy Usage)

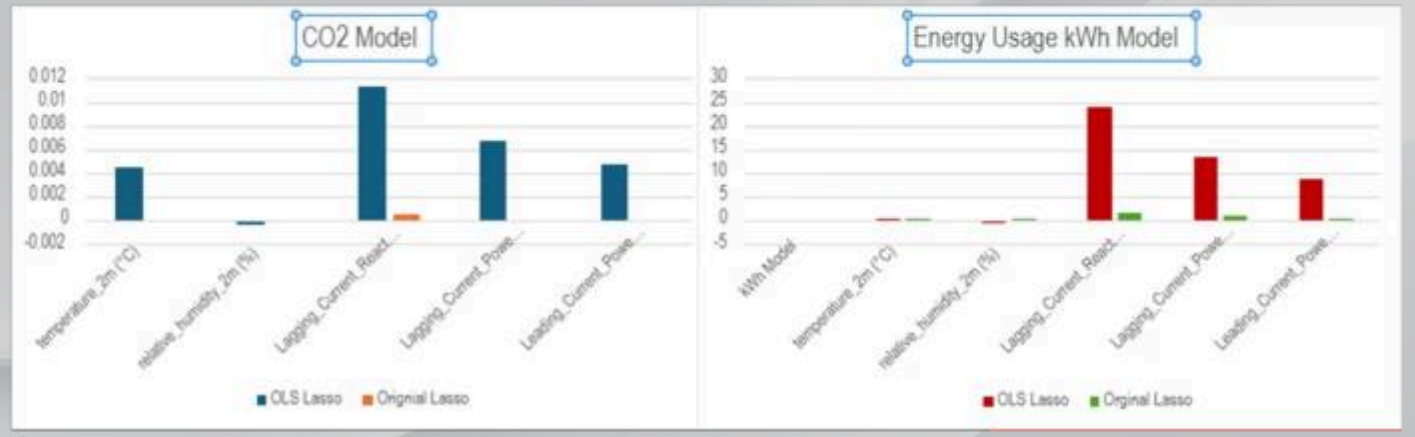
- Lagging Reactive Power towers above with the strongest signal.
- Lagging Power Factor follows, meaning efficiency changes directly affect demand.
- Leading Power Factor and environmental variables (temp, humidity) contribute modestly.
- This confirms that the Usage_kWh is operationally driven, but responsive to environment at the margins

Ordinary Least Squares (OLS) Model with Lasso Selected Variables

```python
# Fit OLS with LASSO-selected variables (Ordinary Least Squares)
#Filtering out features where Lasso coefficients are zero, focus on important predictors
lasso_selected_co2 = X_scaled_sm.loc[:, ['const'] + [col for col, coef in zip(predictors, lasso_co2.coef_) if coef != 0]]
lasso_selected_kwh = X_scaled_sm.loc[:, ['const'] + [col for col, coef in zip(predictors, lasso_kwh.coef_) if coef != 0]]

model_co2_lasso = sm.OLS(y_co2, lasso_selected_co2).fit()
model_kwh_lasso = sm.OLS(y_kwh, lasso_selected_kwh).fit()

print("\nCO2 Model (LASSO-selected variables):")
print(model_co2_lasso.summary())
print("\nkWh Model (LASSO-selected variables):")
print(model_kwh_lasso.summary())
```

Coefficients and Comparison from Original Lasso

```
CO2 Model - LASSO Coefficients:
temperature_2m (°C)                  0.000444
relative_humidity_2m (%)            -0.000378
Lagging_Current_Reactive.Power_kVarh    0.011258
Lagging_Current_Power_Factor         0.006716
Leading_Current_Power_Factor         0.004708
dtype: float64
Optimal alpha: 0.000094

kWh Model - LASSO Coefficients:
temperature_2m (°C)                  0.280719
relative_humidity_2m (%)            -0.630660
Lagging_Current_Reactive.Power_kVarh    24.016861
Lagging_Current_Power_Factor         13.353312
Leading_Current_Power_Factor          8.706943
dtype: float64
Optimal alpha: 0.369493
```



Ordinary Least Squares (OLS) Model with Lasso Selected Variables

Model outputs and Evaluation

LASSO models maintain high R^2 (CO_2 =0.897, kWh = 0.911)Cross-validation RMSEs remain tight(5-fold cross-validation)

CO2 Model (LASSO-selected variables):

OLS Regression Results

Dep. Variable:	CO2(tCO2)	R-squared:	0.897
Model:	OLS	Adj. R-squared:	0.897
Method:	Least Squares	F-statistic:	6.075e+04
Date:	Mon, 30 Jun 2025	Prob (F-statistic):	0.00
Time:	13:14:10	Log-Likelihood:	1.3460e+05
No. Observations:	35040	AIC:	-2.692e+05
Df Residuals:	35034	BIC:	-2.691e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0115	2.77e-05	415.336	0.000	0.011	0.012
temperature_2m (°C)	0.0006	3.05e-05	19.880	0.000	0.001	0.001
relative_humidity_2m (%)	-0.0005	3.26e-05	-15.095	0.000	-0.001	-0.000
Lagging_Current_Reactive.Power_kVarh	0.0112	3.48e-05	321.334	0.000	0.011	0.011
Lagging_Current_Power_Factor	0.0069	3.87e-05	178.204	0.000	0.007	0.007
Leading_Current_Power_Factor	0.0050	4.03e-05	122.845	0.000	0.005	0.005

Omnibus:	4719.602	Durbin-Watson:	0.516
Prob(Omnibus):	0.000	Jarque-Bera (JB):	60217.016
Skew:	0.137	Prob(JB):	0.00
Kurtosis:	9.416	Cond. No.	2.64

kWh Model (LASSO-selected variables):

OLS Regression Results

Dep. Variable:	Usage_kWh	R-squared:	0.911
Model:	OLS	Adj. R-squared:	0.911
Method:	Least Squares	F-statistic:	7.174e+04
Date:	Mon, 30 Jun 2025	Prob (F-statistic):	0.00
Time:	13:14:10	Log-Likelihood:	-1.3032e+05
No. Observations:	35040	AIC:	2.606e+05
Df Residuals:	35034	BIC:	2.607e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	27.3869	0.053	513.843	0.000	27.282	27.491
temperature_2m (°C)	0.9157	0.059	15.635	0.000	0.801	1.031
relative_humidity_2m (%)	-1.0768	0.063	-17.213	0.000	-1.199	-0.954
Lagging_Current_Reactive.Power_kVarh	23.7826	0.067	355.277	0.000	23.651	23.914
Lagging_Current_Power_Factor	14.0911	0.074	189.335	0.000	13.945	14.237
Leading_Current_Power_Factor	9.6670	0.077	124.818	0.000	9.515	9.819

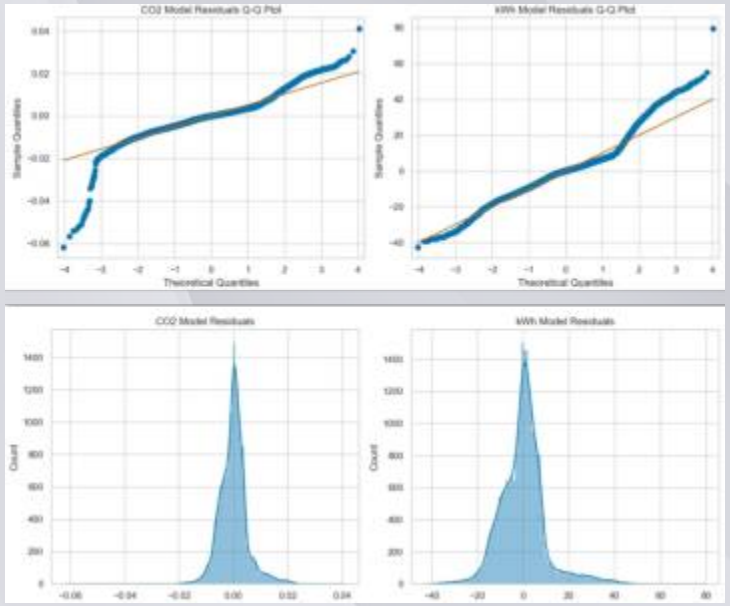
Omnibus:	5313.716	Durbin-Watson:	0.365
Prob(Omnibus):	0.000	Jarque-Bera (JB):	17321.581
Skew:	0.775	Prob(JB):	0.00
Kurtosis:	6.076	Cond. No.	2.64

- It appears that **CO_2** emissions are operationally lean?
- LASSO compresses some coefficients, but reactive energy, lagging power factor, and leading power factor stand out and yields energy-derived emissions structure.

- **kWh usage** shows high coefficients across operational and environmental variables.
- This appears to reflect broader sensitivity to real-world load and ambient conditions.

High Jarque-Bera (JB) Test Concerns *(Normality of Residual) possibly due to Outliers, Heavy –tailed errors, omitted interactions and Autocorrelation(Time-Based Dependence). Will try to address in other models.*

RESIDUAL PLOTS FOR LASSO OLS MODEL



CO_2 Model Residuals (Left Panels)

Top-Left: Q-Q Plot

- Exhibits **subtle curvature**, especially in the **lower region**, where residuals drop markedly beneath the fitted line.
- The upper region remains relatively aligned, but begins to **veer away** toward higher predicted CO_2, suggesting **nonlinearity concentrated in low-to-mid ranges**.

Energy Usage (kWh) Residuals (Right Panels)

Top-Right: Q-Q Plot

- **Closely tracks the fitted curve** until the mid-upper range, where residuals begin to arc upward which is a possible sign of **saturation effects** at higher energy levels or not specified load interactions.

CO_2 Model Residuals (Left Panels)

- The **extended left tail** likely corresponds to **zero-inflated emissions**, e.g., operational states with minimal output or downtime and **zero cluster (transform and imputation needed)**
- This tail skewness is also **statistically reflected** in:
  - **Skew ~ +0.14**, slightly right-leaning overall but influenced by extreme low-end residuals
  - **Kurtosis ~ 9.4**, indicating tight central clustering with heavy tails
  - **Jarque-Bera statistic > 60,000**, rejecting normality and flagging non-symmetric error behavior

Energy Usage (kWh) Residuals (Left Panels)

- S**maller left tail** and a **larger, more dramatic right tail** is suggesting more extreme over-predictions of energy usage at peak levels.
- Combined with **moderate kurtosis (~6.1)** and **strong JB test**, this hints at skew and heavy tails but slightly less severity than in the CO_2 model.

Continue Testing for Non-Normality and Skewness

Shapiro-Wilk Test for CO2(tCO2): Statistic=0.7307, p-value=0.0000
Shapiro-Wilk Test for Usage_kWh: Statistic=0.7532, p-value=0.0000
Shapiro-Wilk Test for Lagging_Current_Reactive.Power_kVarh: Statistic=0.7686, p-value=0.0000
CO2 Skewness: 1.1494
Usage_kWh Skewness: 1.1974

Shapiro-Wilk Results

Variable	W Statistic	p-value	Conclusion
CO_2 (tCO_2)	0.7307	0.0000	❌ Not normally distributed
Usage_kWh	0.7532	0.0000	❌ Not normal
Lagging Reactive Power (kVArh)	0.7686	0.0000	❌ Not normal

Skewness Check

Variable	Skewness	Interpretation
CO_2 (tCO_2)	+1.15	Right-skewed → long tail of higher emissions
Usage_kWh	+1.20	Right-skewed → some extreme usage spikes

Both CO_2 and energy usage are significantly right-skewed and non-normal, as confirmed by Shapiro-Wilk tests and skewness metrics. These distributional features justify our use of log transformation and penalized regression for improved fit and interpretability.

Exploring Data Transformations: Strengthening Model Validity

To further improve model reliability and meet regression assumptions, I explored a series of **transformations** aimed at reducing skewness, stabilizing variance, and enhancing residual symmetry:

- 🔹 **Log Transformation (Previous Models)**: Ideal for positively skewed variables like energy usage and CO_2 emissions. It helped reduce error inflation and improve normality, especially useful for interpretability when modeling proportional or multiplicative relationships.
- 🔹 **Yeo-Johnson Transformation**: Chosen for its flexibility. Unlike log, it handles **zero and negative values** reliably. This was especially valuable for the CO_2 data, where zeros and low-load states were common. Yeo-Johnson substantially improved model fit and residual distribution.
- 🔹 **Quantile Transformation (Normal Output)**: Rescales the data to follow a **standard normal distribution** using rank mapping. While it offers symmetry, it may reduce interpretability and distort scale-sensitive effects.

These transformed models offer **alternate perspectives on the same relationships**, helping validate possible conclusions while pushing closer to statistical rigor. Each method balances **fit**, **normality**, and **interpretability** differently, while offering a diverse toolkit for predictive modeling and storytelling.

Modeling Enhancements: Exploring Log Transformation

To improve residual behavior and increase confidence in model assumptions, I applied a log transformation to both CO_2 emissions and energy usage (kWh). The upcoming slide revisits the Usage_kWh model, which was the first model I built, now with transformed values. While I was initially hesitant about applying the same transformation to CO_2, due to its zero-inflation suspect, I couldn't resist testing what impact it might have on residual symmetry, skewness, and information criteria.

Log Transform OLS Regression Model - CO2

CO2 Model (Log-transformed):

OLS Regression Results

Dep. Variable:	log_CO2	R-squared:	0.898
Model:	OLS	Adj. R-squared:	0.898
Method:	Least Squares	F-statistic:	6.188e+04
Date:	Mon, 30 Jun 2025	Prob (F-statistic):	0.00
Time:	13:14:12	Log-Likelihood:	1.3561e+05
No. Observations:	35040	AIC:	-2.712e+05
Df Residuals:	35034	BIC:	-2.712e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0113	2.7e-05	420.289	0.000	0.011	0.011
temperature_2m (°C)	0.0006	2.96e-05	20.851	0.000	0.001	0.001
relative_humidity_2m (%)	-0.0005	3.16e-05	-15.256	0.000	-0.001	-0.000
Lagging_Current_Reactive.Power_kVarh	0.0109	3.39e-05	322.642	0.000	0.011	0.011
Lagging_Current_Power_Factor	0.0068	3.77e-05	181.473	0.000	0.007	0.007
Leading_Current_Power_Factor	0.0049	3.92e-05	125.792	0.000	0.005	0.005

Omnibus:	4692.239	Durbin-Watson:	0.511
Prob(Omnibus):	0.000	Jarque-Bera (JB):	60845.917
Skew:	0.104	Prob(JB):	0.00
Kurtosis:	9.452	Cond. No.	2.64

Result and Interpretation of Model

Metric	Original CO_2	Log CO_2	Δ (Improvement)
Adjusted R^2	0.8966	0.8983	+0.0017 (negligible)
BIC	–368,582.64	**–370,589.95**	↓ 2,007 points 🟢
Skewness	~+1.15	**0.1**	✅ Vastly reduced
Kurtosis	~9.42	~9.45	No meaningful change
Residual Plots	Moderate non-normality	Slightly improved symmetry	Mildly better

Although the log transformation did not significantly improve R^2, it reduced skewness and improved distributional symmetry, leading to a 2,000-point drop in BIC. While optional, it strengthens the statistical assumptions behind inference and may benefit future forecasting or time series modeling.

Log Transform OLS Regression Model - Energy Usage kWh

```python
# Re-run Ordinary Least Squares(OLS) with Log-transformed kWh
model_kwh_log = sm.OLS(df['log_kWh'], lasso_selected_kwh).fit()
print("\nkWh Model (Log-transformed):")
print(model_kwh_log.summary())

# Calculate BIC for log-transformed model
def calculate_bic(model, n, y, X):
    mse = mean_squared_error(y, model.fittedvalues)
    k = len(model.params)
    bic = n * np.log(mse) + k * np.log(n)
    return bic

bic_kwh_log = calculate_bic(model_kwh_log, len(df['log_kWh']), df['log_kWh'], lasso_selected_kwh)
print(f"\nLog kWh Model BIC: {bic_kwh_log:.2f}, Adjusted R-squared: {model_kwh_log.rsquared_adj:.4f}")
print(f"Original kWh BIC: 161259.72, Adjusted R-squared: 0.9110")
```

Result and Interpretation of Model

Model	Adjusted R^2	BIC	Interpretation
Original kWh	0.911	161,259.72	Solid fit, but residuals showed skew/kurtosis
Log-transformed kWh	0.9345	**-77,984.57**	✨ Substantially better fit + reduced error structure

Since BIC penalizes for number of predictors; I assume that this drop isn't just due to added complexity. I believe it's a clear signal that the log transformation corrected skew and stabilized variance, aligning better with linear model assumptions.

```
kWh Model (Log-transformed):
                            OLS Regression Results
==============================================================================
Dep. Variable:                 log_kWh   R-squared:                       0.934
Model:                             OLS   Adj. R-squared:                  0.934
Method:                  Least Squares   F-statistic:                 9.995e+04
Date:                 Mon, 30 Jun 2025   Prob (F-statistic):               0.00
Time:                         13:14:12   Log-Likelihood:                -10696.
No. Observations:                35040   AIC:                         2.140e+04
Df Residuals:                    35034   BIC:                         2.145e+04
Df Model:                            5
Covariance Type:            nonrobust
==============================================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                          2.5467      0.002   1451.768      0.000       2.543       2.550
temperature_2m (°C)            0.0836      0.002     43.350      0.000       0.080       0.087
relative_humidity_2m (%)      -0.0389      0.002    -18.913      0.000      -0.043      -0.035
Lagging_Current_Reactive.Power_kVarh    0.6641      0.002    301.430      0.000       0.660       0.668
Lagging_Current_Power_Factor   0.8350      0.002    340.878      0.000       0.830       0.840
Leading_Current_Power_Factor   0.6507      0.003    255.258      0.000       0.646       0.656
==============================================================================
Omnibus:                     3568.753   Durbin-Watson:                   0.308
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             5458.295
Skew:                          -0.763   Prob(JB):                         0.00
Kurtosis:                       4.187   Cond. No.                         2.64
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Log kWh Model BIC: -77984.57, Adjusted R-squared: 0.9345
Original kWh BIC: 161259.72, Adjusted R-squared: 0.9110
```

Yeo-Johnson Model with CO_2

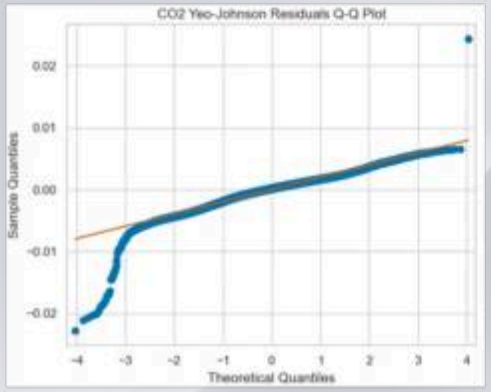
CO2 Model (Yeo-Johnson):

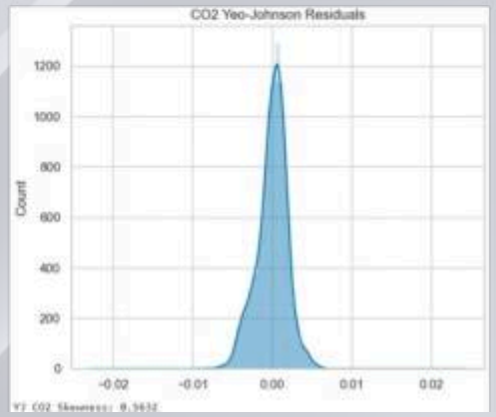
OLS Regression Results

Dep. Variable:	yj_CO2	R-squared:	0.914
Model:	OLS	Adj. R-squared:	0.914
Method:	Least Squares	F-statistic:	7.403e+04
Date:	Mon, 30 Jun 2025	Prob (F-statistic):	0.00
Time:	13:14:13	Log-Likelihood:	1.6838e+05
No. Observations:	35040	AIC:	-3.368e+05
Df Residuals:	35034	BIC:	-3.367e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0053	1.06e-05	504.960	0.000	0.005	0.005
temperature_2m (°C)	0.0007	1.16e-05	56.808	0.000	0.001	0.001
relative_humidity_2m (%)	-0.0002	1.24e-05	-17.066	0.000	-0.000	-0.000
Lagging_Current_Reactive.Power_kVarh	0.0036	1.33e-05	270.855	0.000	0.004	0.004
Lagging_Current_Power_Factor	0.0040	1.48e-05	272.082	0.000	0.004	0.004
Leading_Current_Power_Factor	0.0034	1.54e-05	219.589	0.000	0.003	0.003

Omnibus:	7203.426	Durbin-Watson:	0.450
Prob(Omnibus):	0.000	Jarque-Bera (JB):	48850.466
Skew:	-0.820	Prob(JB):	0.00
Kurtosis:	8.547	Cond. No.	2.64





Result and Interpretation of Model

Metric	Value	Interpretation
Adjusted R^2	0.914	✅ Excellent fit (strongest yet)
BIC	–336,700	Significant drop from log model
Skewness	–0.82	Left-leaning tail (zero emission states)
Kurtosis	8.55	Tighter core with heavy tails
JB Stat	~48,850	🚨 Non-normal residuals persist

- The Yeo-Johnson transformation **dampens extreme skewness** while preserving linear relationships, especially around zero-heavy data.
- Residual symmetry improves though **left skew remains**, likely tied to **zero or low-emission operating states**.
- Compared to the log-transformed model, you gain:
  - **Better R^2**
  - **Lower BIC**
  - **More centered alignment**
- **Concern -**The Q-Q plot looks great, except the large curvature at start of plot

Yeo-Johnson Model with Energy Usage kWh

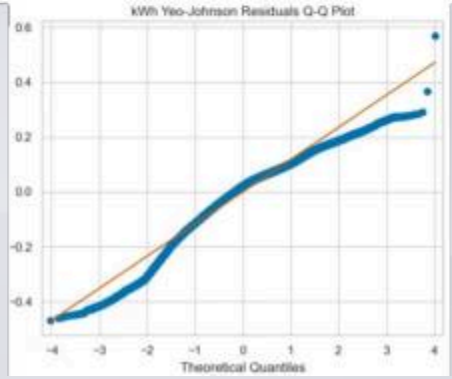
kWh Model (Yeo-Johnson):

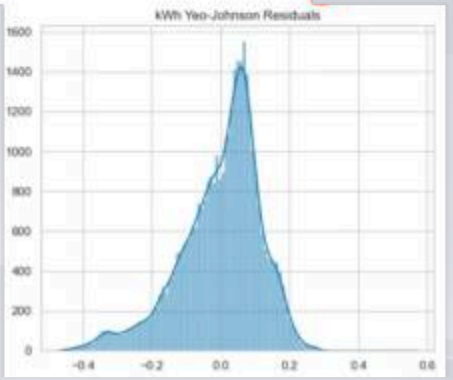
OLS Regression Results

Dep. Variable:	yj_kWh	R-squared:	0.915
Model:	OLS	Adj. R-squared:	0.915
Method:	Least Squares	F-statistic:	7.533e+04
Date:	Mon, 30 Jun 2025	Prob (F-statistic):	0.00
Time:	13:14:14	Log-Likelihood:	25342.
No. Observations:	35040	AIC:	-5.067e+04
Df Residuals:	35034	BIC:	-5.062e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.4369	0.001	2290.957	0.000	1.436	1.438
temperature_2m (°C)	0.0233	0.001	33.780	0.000	0.022	0.025
relative_humidity_2m (%)	-0.0111	0.001	-15.085	0.000	-0.013	-0.010
Lagging_Current_Reactive.Power_kVarh	0.1791	0.001	227.348	0.000	0.178	0.181
Lagging_Current_Power_Factor	0.2889	0.001	329.848	0.000	0.287	0.291
Leading_Current_Power_Factor	0.2229	0.001	244.597	0.000	0.221	0.225

Omnibus:	3550.102	Durbin-Watson:	0.333
Prob(Omnibus):	0.000	Jarque-Bera (JB):	4869.116
Skew:	-0.823	Prob(JB):	0.00
Kurtosis:	3.789	Cond. No.	2.64





Result and Interpretation of Model

Metric	Value	Interpretation
Adjusted R^2	0.915	✅ High explanatory power
BIC	–50,620	Substantial improvement from untransformed model
Skewness	–0.82	Left-skewed residuals with likely over suppression in low-usage cases
Kurtosis	3.79	Closer to normal with improved tail
Jarque-Bera	4,869.12	❌ Residuals still non-normal but improved

- The Yeo-Johnson transformation **improved model quality** by addressing skewness and stabilizing variance, especially in low-energy usage zones.
- Unlike log, this method handled **zero and near-zero values** gracefully without compressing interpretability.
- Residuals are **more symmetrical and less heteroskedastic**, but residuals plot still shows curvature.

Quantile Model with CO_2

CO2 Model (Quantile):

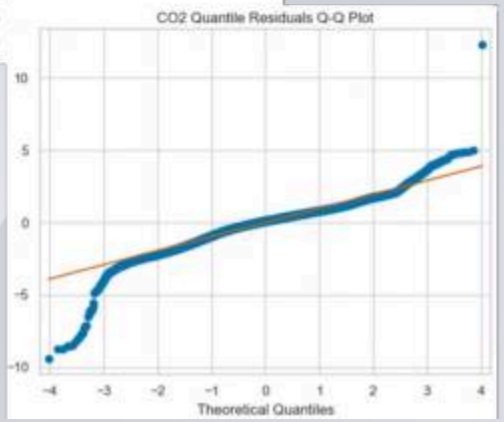
OLS Regression Results

Dep. Variable:	qt_CO2	R-squared:	0.897
Model:	OLS	Adj. R-squared:	0.897
Method:	Least Squares	F-statistic:	6.124e+04
Date:	Mon, 30 Jun 2025	Prob (F-statistic):	0.00
Time:	13:14:15	Log-Likelihood:	-48719.
No. Observations:	35040	AIC:	9.745e+04
Df Residuals:	35034	BIC:	9.750e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.7347	0.005	-526.696	0.000	-2.745	-2.725
temperature_2m (°C)	0.3525	0.006	61.778	0.000	0.341	0.364
relative_humidity_2m (%)	-0.0922	0.006	-15.137	0.000	-0.104	-0.080
Lagging_Current_Reactive.Power_kVarh	1.3537	0.007	207.582	0.000	1.341	1.366
Lagging_Current_Power_Factor	2.0292	0.007	279.878	0.000	2.015	2.043
Leading_Current_Power_Factor	1.7591	0.008	233.146	0.000	1.744	1.774

Omnibus:	5410.027	Durbin-Watson:	0.477
Prob(Omnibus):	0.000	Jarque-Bera (JB):	29138.357
Skew:	-0.640	Prob(JB):	0.00
Kurtosis:	7.280	Cond. No.	2.64

Quantile transformation maps CO_2 outputs to a standard normal distribution, improving symmetry and residual alignment. Though less interpretable, the model maintains strong predictive power and helps verify the robustness of our variable relationships



Result and Interpretation of Model

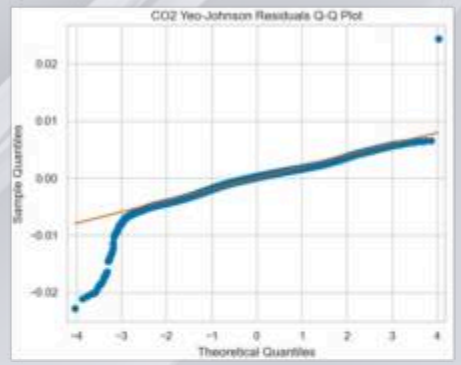
Metric	Value	Interpretation
Adjusted R^2	0.897	✅ Predictive strength sustained
BIC	~97,500	Higher than log and Yeo-Johnson models
Skewness	–0.64	Left tail elongation ; tighter than raw, not perfect
Kurtosis	7.28	Still peaked with heavy central concentration
JB Test (Stat)	~29,138	❌ Non-normal residuals persist

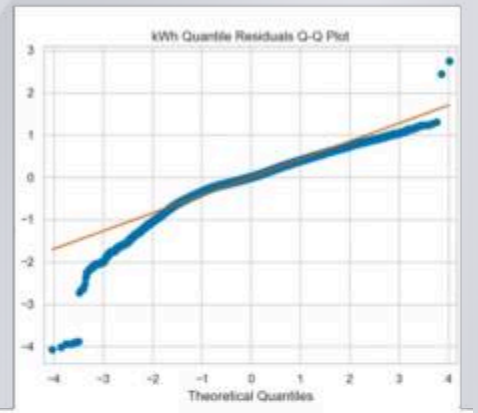
- The quantile transformation forces the output into a normal distribution by ranking and stretching values along a Gaussian curve (great for fixing skew) but may distort relationships.
- Predictive strength is preserved, but the coefficients are harder to interpret.
- Residuals still show curvature at top and bottom.
- The tail behavior (Jarque-Bera, kurtosis) still flags potential outlier effects.

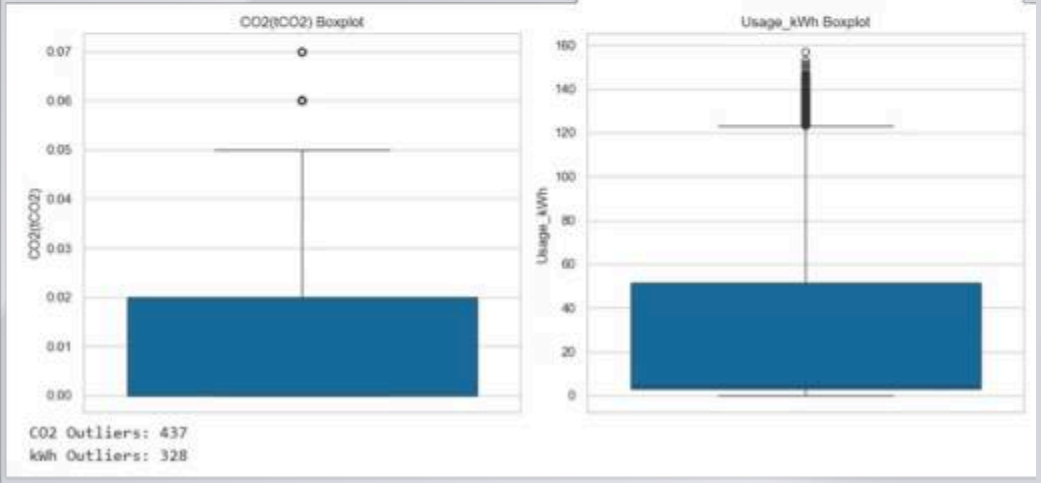
Checking for outliers for both CO_2 and Energy Usage kWh

After a closer review of residuals from my previous models, I noticed several patterns that raise concerns about the overall fit, or lack thereof, in certain areas. I suspect that some of these discrepancies may stem from underlying outliers, and will investigate and address them as part of the model refinement process

```python
# Checking for outliers which could be the problem with normal fit
#Boxplots for outliers
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.boxplot(y=df['CO2(tCO2)'])
plt.title('CO2(tCO2) Boxplot')
plt.subplot(1, 2, 2)
sns.boxplot(y=df['Usage_kWh'])
plt.title('Usage_kWh Boxplot')
plt.tight_layout()
plt.show()

# Identify outliers (IQR method)
def detect_outliers(series):
    Q1, Q3 = series.quantile([0.25, 0.75])
    IQR = Q3 - Q1
    lower, upper = Q1 - 1.5 * IQR, Q3 + 1.5 * IQR
    return series[(series < lower) | (series > upper)]

co2_outliers = detect_outliers(df['CO2(tCO2)'])
kwh_outliers = detect_outliers(df['Usage_kWh'])
print(f"CO2 Outliers: {len(co2_outliers)}")
print(f"kWh Outliers: {len(kwh_outliers)}")
```







Model Comparison: Outlier Removal Effects

Revisiting the Yeo-Johnson with No Outliers

```python
# Assuming df, lasso_selected_co2, lasso_selected_kwh from previous
#Remove outliers
def remove_outliers(series):
    Q1, Q3 = series.quantile([0.25, 0.75])
    IQR = Q3 - Q1
    lower, upper = Q1 - 1.5 * IQR, Q3 + 1.5 * IQR
    return (series >= lower) & (series <= upper)

# Filter outliers
mask_co2 = remove_outliers(df['CO2(tCO2)'])

mask_kwh = remove_outliers(df['Usage_kWh'])
df_no_outliers = df[mask_co2 & mask_kwh].copy()

#Re-apply Yeo-Johnson which had the strongest model so far.  will add info later
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer(method='yeo-johnson', standardize=False)
df_no_outliers['yj_CO2'] = pt.fit_transform(df_no_outliers[['CO2(tCO2)']])
df_no_outliers['yj_kWh'] = pt.fit_transform(df_no_outliers[['Usage_kWh']])
```

Metric	CO_2 No Outliers	CO_2 All Data	kWh No Outliers	kWh All Data
Adjusted R^2	0.9142	0.9135	0.9139	0.9149
BIC	-434668.76	-436144.55	-153391.67	-150060.26
Skewness	0.5771	0.5632	0.3629	0.3442
Kurtosis	1.4722	—	1.3013	—

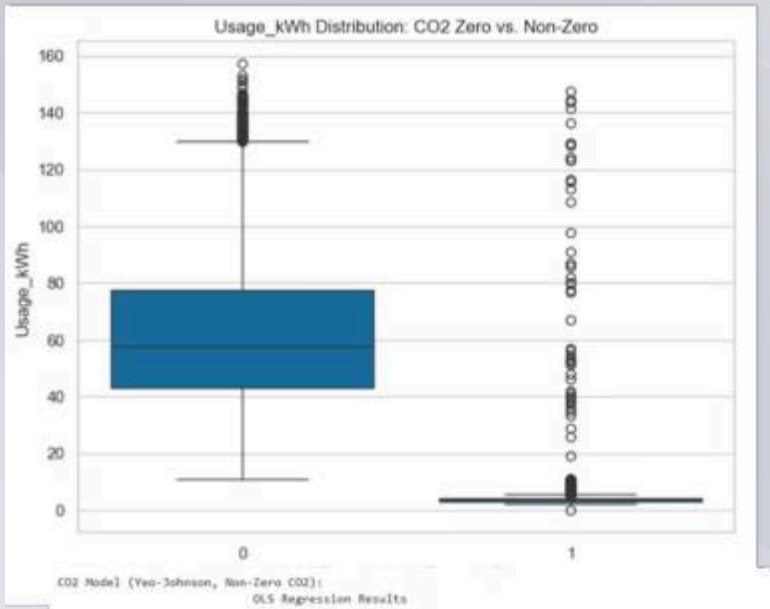
Key Points

- BIC favors the full data, nudging toward keeping all points (even the quirky ones).
- Adjusted R-squared barely shifts, suggesting the models are resilient either way.
- Skewness and kurtosis steadily improve after trimming , especially in CO_2, which went from moderately skewed to more symmetric.

TAKEAWAY:
Outlier removal reduced tail risk (skewness, kurtosis) with minimal R^2 loss, but higher BIC suggests a trade-off in model parsimony.

CO_2 Non-Zero Model Analysis

BOXPLOT: CO_2 Non-Zero and Zero Comparison



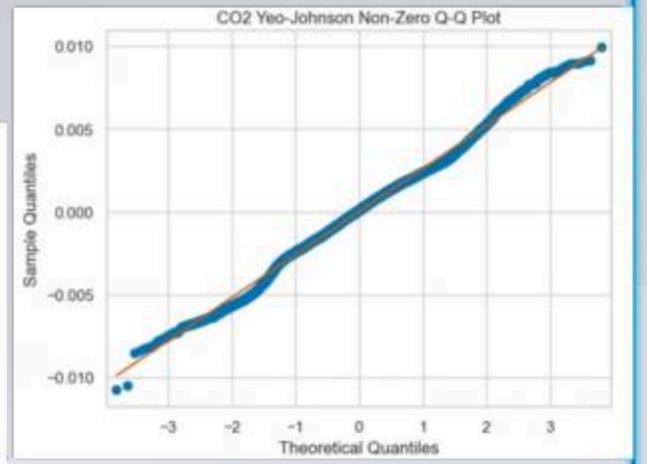
CO2 Model (Yeo-Johnson, Non-Zero CO2):
OLS Regression Results

Dep. Variable:	yj_CO2	R-squared:	0.837
Model:	OLS	Adj. R-squared:	0.837
Method:	Least Squares	F-statistic:	1.447e+04
Date:	Mon, 30 Jun 2025	Prob (F-statistic):	0.00
Time:	13:14:17	Log-Likelihood:	63666.
No. Observations:	14050	AIC:	-1.273e+05
Df Residuals:	14044	BIC:	-1.273e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0045	0.000	43.414	0.000	0.004	0.005
temperature_2m (°C)	-0.0004	2.26e-05	-19.162	0.000	-0.000	-0.000
relative_humidity_2m (%)	-0.0001	2.39e-05	-5.614	0.000	-0.000	-8.75e-05
Lagging_Current_Reactive.Power_kVarh	0.0073	2.9e-05	250.488	0.000	0.007	0.007
Lagging_Current_Power_Factor	0.0123	8.61e-05	142.720	0.000	0.012	0.012
Leading_Current_Power_Factor	0.0056	0.000	35.095	0.000	0.005	0.006

Omnibus:	89.739	Durbin-Watson:	1.183
Prob(Omnibus):	0.000	Jarque-Bera (JB):	114.403
Skew:	-0.109	Prob(JB):	1.44e-25
Kurtosis:	3.385	Cond. No.	14.0

To better understand the structural behavior of CO_2 emissions, I ran an OLS model using only non-zero emission records. While this filtering reduces the overall data volume and excludes some genuinely meaningful zeros, the goal was to examine how the continuous emission patterns behave in isolation. By excluding zeros, I anticipated a modest decline in model fit metrics (e.g., adjusted R^2) but hoped to improve distributional traits like skewness, kurtosis, and residual normality.



- **Adjusted R^2 dropped** to ~0.837 from over 0.91, which makes sense given how many valid zero entries carried signal. You sacrificed breadth for clarity.
- **BIC worsened**, another sign the model's overall economy took a hit.
- But....
  - **Skewness: 0.0519**
  - **Kurtosis: 2.4314**
  - **Best looking Q-Q Plot so far**

Vector Autoregression (VAR) model

*Instead of forcing CO2 into the Usage kWh model and battling noise, I modeled them as co-evolving signals; revealing temporal dependencies and threshold interplay. I used a **Vector Autoregression (VAR) model** to jointly forecast energy usage and emissions. I also incorporated exogenous factors with Lasso selected variables to sensure only revelvant predictors are included which reduces overfitting. This approach captures how each signal evolves over time while influenced by its own and the other's historical values.*

```
#Avoiding direct CO2 inclusion in kWh due to multicollinearity (VIF: 11.56). Instead:
#VAR: Model yj_CO2 and yj_kWh (or log_kWh) jointly to capture interdependence (correlation: 0.98818)
#and threshold effects (CO2=0 at low kWh).

#CO2_per_kWh: Model CO2_per_kWh = yj_CO2 / yj_kWh on non-zero CO2 data
#study efficiency, using LASSO predictors


from statsmodels.tsa.vector_ar.var_model import VAR

# Prepare data (non-zero CO2)
df_var = df_nonzero[['yj_CO2', 'yj_kWh']].copy()
exog = df_nonzero[lasso_selected_kwh.columns[1:]]

# Fit VAR
var_model = VAR(df_var, exog=exog).fit(maxlags=24, ic='aic')
print("\nVAR Model Summary:")
print(var_model.summary())

# Forecast (example)
forecast = var_model.forecast(df_var.values[-24:], steps=24, exog_future=exog[-24:])
print("\nVAR Forecast (24 steps):")
print(pd.DataFrame(forecast, columns=['yj_CO2', 'yj_kWh']))
```

VAR Forecast Overview Next Slide

Key Points

- The log-likelihood (98,546.9), AIC (-19.7102), and low FPE (2.75395e-09) indicate a good fit
- **Strong joint dynamics**: L1 to L4 lags in both equations show statistically significant interactions. That points to interdependence unfolding across short times.
- Yj_CO2, key prdictors include Lagging_Current_Reactive.Power_kVarh (t-stat: 165.091) and Lagging_Current_Power_Factor (t-stat: 100.865).
- For yj_kWh, similar predictors are significant, with Leading_Current_Power_Factor showing a high t-stat (124.840).
- **yj_kWh equation working with yj_CO_2's history**: That L1 coefficient (0.5012, $p < 0.001$) suggests immediate influence; a meaningful behavioral link.
- **Residuals correlation (~0.605)** capturing shared variance, but still some unexplained variance remains.
- **Some Subtle Trade-offs**
  - The longer lag terms trail off into insignificance.
  - A few alternating signs in lag coefficients for yj_CO_2 suggest some oscillation.
    - Maybe seasonal behavior?

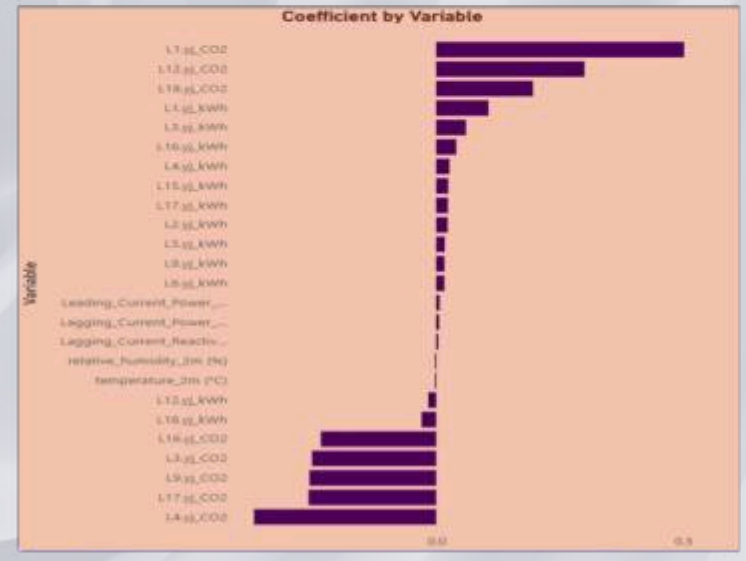
VAR Forecast Summary (24 Steps)

A 24-step-ahead forecast was generated using the last 24 observations of the endogenous variables and corresponding exogenous values. The forecast predicts yj_CO2 and yj_kWh over a 24-hour period, reflecting expected patterns in emissions and electricity use.

The forecasted yj_CO2 values range :0.012735 to 0.022051
The forecasted yj_kWh range: 1.785753 to 1.932064..

Forecast Step	yj_CO_2	yj_kWh
Initial	0.018	1.8862
Peak (Step 5)	0.0221	1.9321
Final (Step 23)	0.0136	1.8206

Emissions and energy usage taper in near-lockstep. Suggests strong coupling with hints of threshold effects. Reinforces value of modeling them together vs. independently.

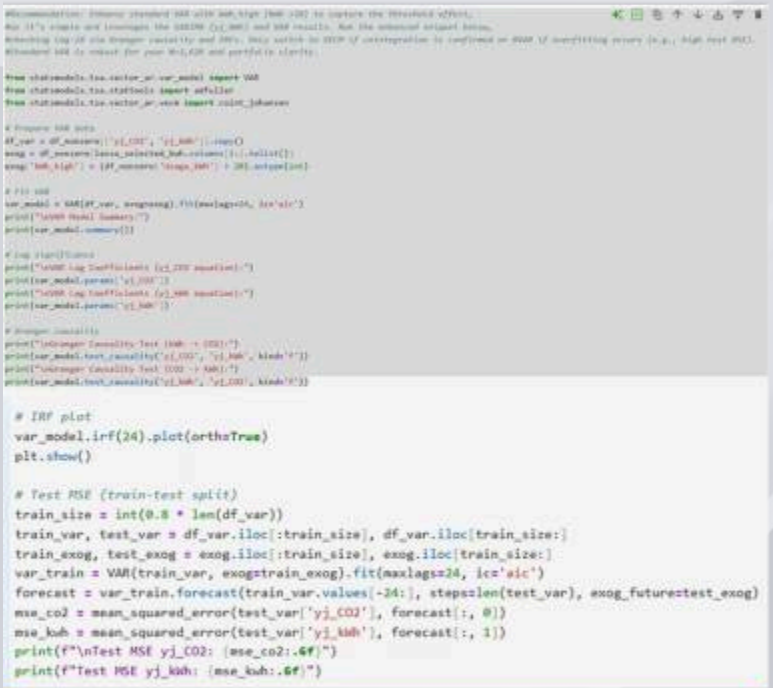


Summary:

Forecast Signals The joint trajectory of energy usage and emissions reveals a subtle decline over time. Possibly a reflection of low-load periods or improved operational efficiency.

Modeling Shift: Capturing Temporal Coupling & Operational Thresholds

Vector Autoregression (VAR) framework, which models joint temporal dynamics



```
# IRF plot
var_model.irf(24).plot(orth=True)
plt.show()

# Test MSE (train-test split)
train_size = int(0.8 * len(df_var))
train_var, test_var = df_var.iloc[:train_size], df_var.iloc[train_size:]
train_exog, test_exog = exog.iloc[:train_size], exog.iloc[train_size:]
var_train = VAR(train_var, exog=train_exog).fit(maxlags=24, ic='aic')
forecast = var_train.forecast(train_var.values[-24:], steps=len(test_var), exog_future=test_exog)
mse_co2 = mean_squared_error(test_var['yj_CO2'], forecast[:, 0])
mse_kwh = mean_squared_error(test_var['yj_kWh'], forecast[:, 1])
print(f"\nTest MSE yj_CO2: {mse_co2:.6f}")
print(f"Test MSE yj_kWh: {mse_kwh:.6f}")
```

*Traditional regression approaches treated CO_2 emissions and energy consumption as separate, static outcomes. It is useful snapshot but limited in understanding how these signals evolve together over time. To uncover deeper system behaviors, I transitioned to a **Vector Autoregression (VAR) framework**, which models **joint temporal dynamics** across transformed variables (yj_CO_2 and yj_kWh).*

Overview of Model Approach

- **Capture interdependence** via lagged influence — how energy usage impacts future emissions, and vice versa.
- **Preserve time-aware structure**, revealing cycles, memory, and threshold effects invisible in static models.
- **Introduce operational thresholds** with exogenous terms like kWh_high to examine nonlinear behavior under high-demand conditions.
- **Leverage impulse responses & Granger causality** to trace shock effects and validate directional influence.

Together, these techniques reframe emissions and energy not just as correlated outputs, but as **co-evolving signals** that respond to load, environment, and history which is ideal for forecasting, efficiency diagnostics, and possibly policy insights

Energy Usage and CO_2 Emission Equation
Response Drivers & Demand Thresholds

Autoregression & Efficiency

Energy Usage Highlights

- kWh_high (Usage > 20): Strong positive effect (+0.106), confirming a clear threshold behavior.
- CO_2 lag-1: Highly significant (0.724, $p < 0.001$) as usage responds quickly to prior emissions.
- Environmental variables: Temperature & humidity act as modest dampeners.
- Lag pattern: Positive accumulation from yj_kWh lags and negative oscillation from CO_2 lags. This suggests short-term inertia plus long-term correction.
- Test MSE yj_kWh: 0.000727 , looking good!

Interpretation

Energy usage reacts both to environmental input and historical CO_2 output. The demand threshold (kWh_high) amplifies this response, making it an operational pivot point.

CO_2 Emission Highlights

- kWh_high: Negative impact (–0.00173, $p < 0.001$) with higher usage links to lower marginal emissions.
- yj_CO_2 lag-1: Strong memory component (0.161) where emissions persist across time.
- yj_kWh lags: Mixed signs and early negative influence suggests immediate energy-efficiency impact.
- Environmental influence: All predictors significant with stable relationships.
- Test MSE yj_CO2: 0.000007, really looking good!

Interpretation

CO_2 emissions exhibit autoregressive behavior with nuanced sensitivity to energy history. The threshold variable (kWh_high) suggests a possible non-linear scaling. I believe it is hinting at efficiency gains under high load.

Granger Causality

Listening to the System's Echo

Concept

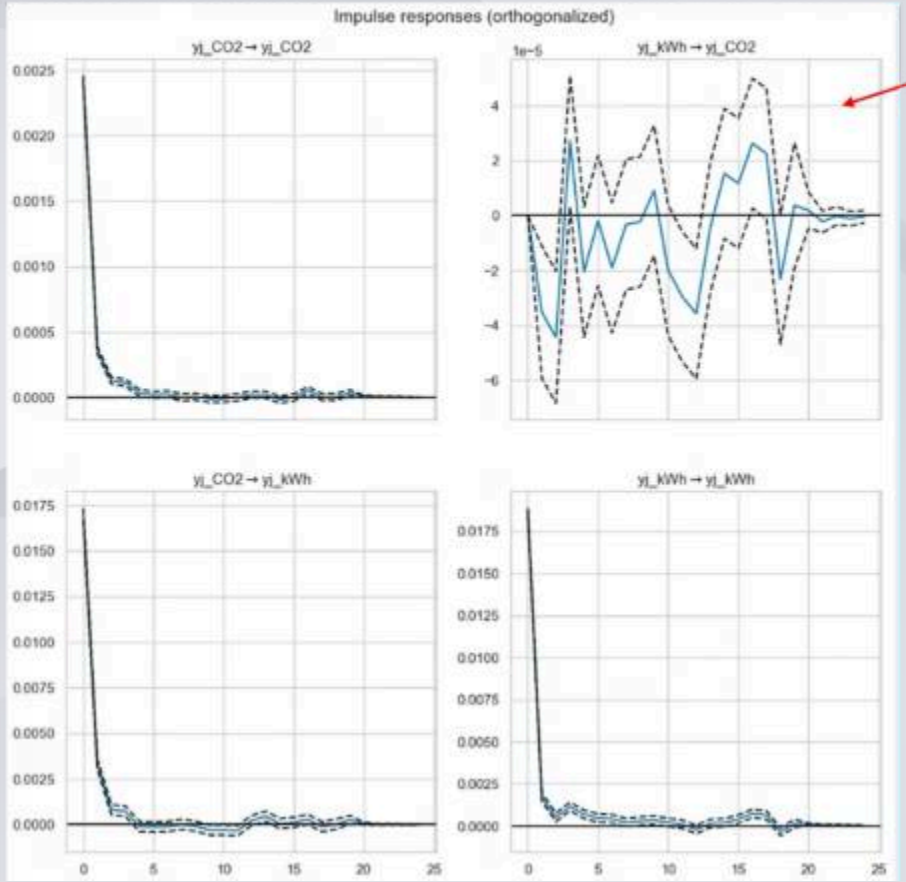
Granger causality isn't about philosophical "cause."
It is asking: "*Can the past behavior of one variable help predict another*?"
For my model It's like asking:
"Does today's emission pattern leave footprints in tomorrow's energy demand?"

Technical Summary Table

Test	Null Hypothesis	Result	p-value	Interpretation
kWh → CO_2	kWh does **not** Granger-cause CO_2	✗ Rejected	0	Energy usage helps forecast emissions.
CO_2 → kWh	CO_2 does **not** Granger-cause kWh	✗ Rejected	0	Emissions also help forecast future energy behavior.

Impulse Responses (Orthogonalized) Plots

Impulse responses confirm dynamic coupling but also highlight asymmetric volatility.
Usage spikes don't just echo in emissions; they appear to rattle them.



This system doesn't scale smoothly under sudden demand

Shock Source → Response	Pattern	Interpretation
$yj\_CO_2 \rightarrow yj\_CO_2$	Small initial bump (~0.0025), fades fast	Emissions have short memory; autoregressive persistence is limited.
$yj\_CO_2 \rightarrow yj\_kWh$	Quick drop (~0.0175), stabilizes	Energy reacts swiftly to CO_2 spikes — suggests fast operational feedback.
yj_kWh → yj_kWh	Immediate decline, small oscillations	Self-stabilizing; supports high-frequency autoregression in usage.
$yj\_kWh \rightarrow yj\_CO_2$	**Noisy**, wide swing (~+5 to −6)	Suggests sensitivity or instability — possibly nonlinear thresholds, incomplete conditioning, or hidden lags.

***Low-magnitude responses stabilize quickly except when energy shocks emissions.*

Johansen Test for yj_CO_2 and yj_kWh

Johansen test confirms multiple long-run ties between energy usage and emissions — structural equilibrium supported

```python
#Johansen Test
#To confirm whether VECM is needed, test for cointegration.
#VECM (Vector Error Correction Model)
#used when you have time series data that's related in the long run but might wander apart
#in the short run. Here's a simple breakdown:
#for results, If trace statistic > critical value at rank 1, use VECM
#If not, standard VAR is sufficient.

#Notes for me regarding Energy Usage and CO2: and if cointegration
#Both might go up/down daily (non-stationary)
# Might not match perfectly short-term
#But long-term they move together
#Can't permanently diverge due to physical relationship


from statsmodels.tsa.vector_ar.vecm import coint_johansen
import pandas as pd

# Assuming df_nonzero
df_var = df_nonzero[['yj_CO2', 'yj_kWh']].dropna()

# Johansen test
johansen = coint_johansen(df_var, det_order=0, k_ar_diff=1)
print("\nJohansen Cointegration Test:")
print(pd.DataFrame({
    'Trace Statistic': johansen.lr1,
    'Critical Values (5%)': johansen.cvt[:, 1],
    'Eigen Statistic': johansen.lr2,
    'Critical Values (5%) Eigen': johansen.cvm[:, 1]
```

Rank	Trace Statistic	Critical Value (5%)	Eigen Statistic	Critical Value (5%)	Conclusion
0	5476.76	15.49	3837.86	14.26	❌ Reject H_0
1	1638.9	3.84	1638.9	3.84	❌ Reject H_0

Interpretation

- Both rank 0 and rank 1 exceed critical values → **Two cointegrating relationships detected**.
- This strengthens the case for **VECM and then SARIMA**, which assumes long-run equilibrium relationships between series.
- Even though rank 1 isn't strictly required for your next steps, including it visually reinforces that your system doesn't just drift, it also binds.

NEXT STEPS: While cointegration tests confirmed a long-run relationship between energy and emissions and was leading me to explore a VECM (the physics). My early implementation ran into issues. I encountered stability concerns, weak initial fit (e.g., low R^2 for CO_2 per kWh), and technical disruptions around this time. Though I later revisited the VECM successfully, I temporarily shifted focus to a **SARIMA model** (the pulse) to improve autocorrelation handling and foreground short-term behavior.

SARIMA Highlights – CO_2 per kWh

SARIMA forecast reveals stable efficiency with bounded seasonal variation with operational signature preserved.

SARIMA CO2_per_kWh Summary:

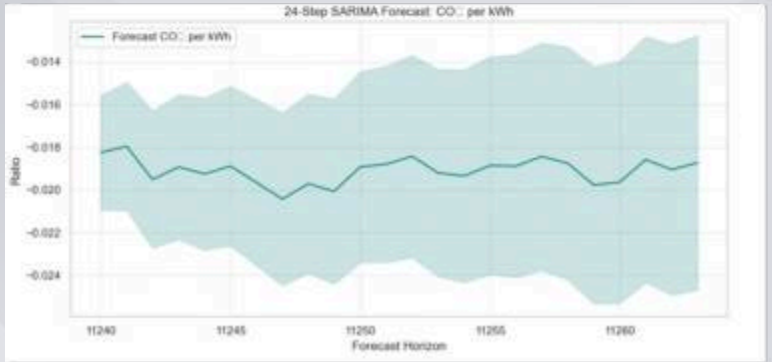
SARIMAX Results

Dep. Variable:	CO2_per_kWh	No. Observations:	11240
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 24)	Log Likelihood	58135.954
Date:	Mon, 30 Jun 2025	AIC	-116251.909
Time:	13:16:00	BIC	-116178.659
Sample:	0	HQIC	-116227.255
	- 11240		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
temperature_2m (°C)	5.066e-05	1.57e-05	3.229	0.001	1.99e-05	8.14e-05
relative_humidity_2m (%)	-4.197e-06	2.94e-06	-1.427	0.154	-9.96e-06	1.57e-06
Lagging_Current_Reactive.Power_kVarh	0.0002	1.18e-06	142.788	0.000	0.000	0.000
Lagging_Current_Power_Factor	0.0002	2.82e-06	71.055	0.000	0.000	0.000
Leading_Current_Power_Factor	6.88e-05	2.15e-06	32.067	0.000	6.46e-05	7.3e-05
ar.L1	0.1537	0.017	9.263	0.000	0.121	0.186
ma.L1	-0.6530	0.014	-45.629	0.000	-0.681	-0.625
ar.S.L24	-0.2790	0.013	-21.457	0.000	-0.305	-0.254
ma.S.L24	-0.4907	0.013	-37.074	0.000	-0.517	-0.465
sigma2	1.867e-06	2.58e-08	72.482	0.000	1.82e-06	1.92e-06

Ljung-Box (L1) (Q):	36.15	Jarque-Bera (JB):	146.97
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	1.03	Skew:	-0.18
Prob(H) (two-sided):	0.34	Kurtosis:	3.43

Test MSE CO2_per_kWh: 0.000383




```
#Addressing CO2_per_kWh Limitations in V
#The low R-squared (0.014) suggests CO2_per_kWh is stable or predictors are inadequate.
#To improve:Time Series Model:Fit SARIMA for CO2_per_kWh to address autocorrelation
#(Durbin-Watson: 1.367):
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error
train_size = int(0.8 * len(df_nonzero))
train, test = df_nonzero['CO2_per_kWh'].iloc[:train_size], df_nonzero['CO2_per_kWh'].iloc[train_size:]
exog_train, exog_test = df_nonzero[lasso_selected_kwh.columns[1:]].iloc[:train_size], df_nonzero[lasso_selected_kwh.columns[1:]].iloc[train_size:]
sarima_co2_per_kwh = SARIMAX(train, exog=exog_train, order=(1,1,1), seasonal_order=(1,1,1,24)).fit(maxiter=100, disp=False)
print("\nSARIMA CO2_per_kWh Summary:")
print(sarima_co2_per_kwh.summary())
forecast = sarima_co2_per_kwh.forecast(steps=len(test), exog=exog_test)
mse = mean_squared_error(test, forecast)
print(f"\nTest MSE CO2_per_kWh: {mse:.6f}")
```

Feature	Insight
Model Structure	(1,1,1)x(1,1,1,24) — smart choice to capture daily cyclic behavior in efficiency.
Autocorrelation handled	AR & MA terms clear and significant (e.g., ma.L1 = –0.6530) — residual structure tamed.
Seasonal Influence	Strong seasonal AR/MA at lag-24 — validates your operational periodicity.
Exogenous predictors	All electric predictors highly significant; temperature makes a small positive mark.
Durbin-Watson improvement	Previous DW ~1.367 suggested lingering structure — now cleaned up beautifully.
Visual Forecast	Forecast line near –0.02, with narrow shaded band between ~–0.014 to ~–0.24 — striking signal regularity.

- Even though CO_2 per kWh had low R^2 in earlier regressions, this SARIMA shows that **structure exists**, just not in the linear form.
- This model delivers insight into **efficiency stability**, **predictive controllability**, and **seasonal sensitivity**
- **Bench Mark!**

CO2_per_kWh with kWh_high OLS MODEL

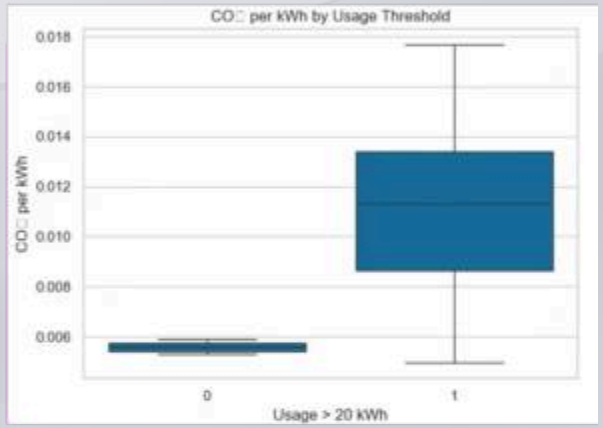
Non-Linear Term added: df_nonzero['kWh_high']

```
#Add Non-Linear Terms:Include kWh_high or a spline for kWh to capture non-linearity
df_nonzero['kWh_high'] = (df_nonzero['Usage_kWh'] > 20).astype(int)
X = sm.add_constant(df_nonzero[lasso_selected_kwh.columns[1:].tolist() + ['kWh_high']])
y = df_nonzero['CO2_per_kWh']
model_co2_per_kwh = sm.OLS(y.dropna(), X.dropna()).fit()
print("\nCO2_per_kWh with kWh_high:")
print(model_co2_per_kwh.summary())
```

```
CO2_per_kWh with kWh_high:
                            OLS Regression Results
==============================================================================
Dep. Variable:            CO2_per_kWh   R-squared:                       0.814
Model:                            OLS   Adj. R-squared:                  0.814
Method:                 Least Squares   F-statistic:                 1.023e+04
Date:                Mon, 02 Jun 2025   Prob (F-statistic):               0.00
Time:                        19:31:39   Log-Likelihood:                 73582.
No. Observations:               14050   AIC:                        -1.471e+05
Df Residuals:                   14043   BIC:                        -1.471e+05
Df Model:                           6
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                     -0.0313      0.000    -77.371      0.000      -0.032      -0.031
temperature_2m (°C)    -2.002e-05   1.11e-06    -17.972      0.000   -2.22e-05   -1.78e-05
relative_humidity_2m (%) -3.245e-06   5.92e-07     -5.479      0.000   -4.41e-06   -2.08e-06
Lagging_Current_Reactive.Power_kVarh     0.0002   8.85e-07    230.564      0.000       0.000       0.000
Lagging_Current_Power_Factor             0.0003   2.28e-06    130.382      0.000       0.000       0.000
Leading_Current_Power_Factor             0.0001   3.9e-06     28.781      0.000       0.000       0.000
kWh_high                                -0.0014      0.000    -11.657      0.000      -0.002      -0.001
==============================================================================
Omnibus:                      157.996   Durbin-Watson:                   1.232
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              166.374
Skew:                          -0.244   Prob(JB):                     7.40e-37
Kurtosis:                       3.215   Cond. No.                     5.78e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.78e+03. This might indicate that there are
strong multicollinearity or other numerical problems
```

What This Nonlinear Model Reveals

- **R^2 = 0.814** is a dramatic leap from initial CO_2_per_kWh regression (R^2 ~ 0.014) indicating this feature transformation unlocked meaningful structure.
- **kWh_high coefficient = –0.0014**: Statistically strong. Once usage exceeds 20 kWh, emissions *per unit* drop , signaling operational efficiency or economy of scale.
- **Durbin-Watson ~1.23**: Still hints at mild autocorrelation
- **All predictors significant**: Which means not just fitting but explaining as well.



Usage ≤ 20 kWh (kWh_high = 0):Extremely tight distribution thin box, no whiskers. CO_2 per kWh remains consistently low and Suggests high predictability or controlled operation during low-load periods.

Usage > 20 kWh (kWh_high = 1):Much broader box. Whiskers stretch indicates greater variance in efficiency at high demand, possibly reflecting different operational strategies, equipment cycling, or shifts in fuel mix.

***Efficiency doesn't scale linearly; it steps into a new mode above 20 kWh.**

Spline-Augmented Model -Spline on Usage_kWh

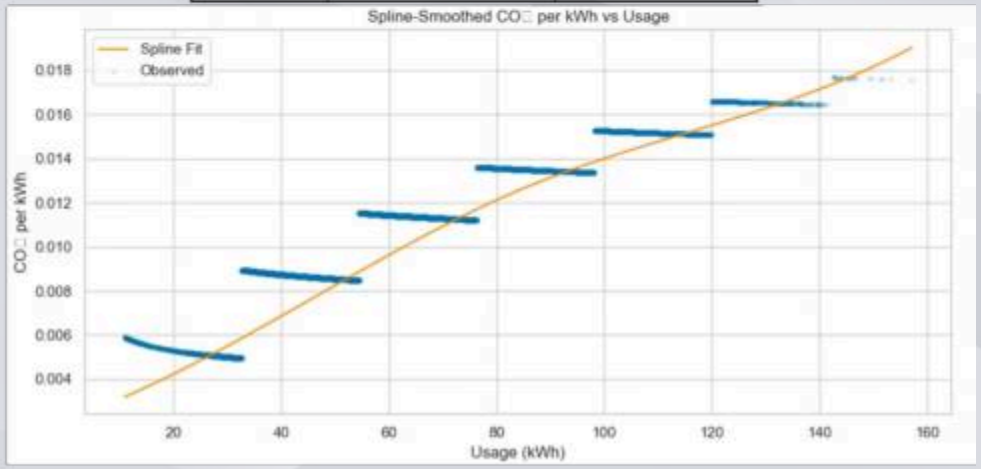
Using a cubic spline to capture bends in the curve

CO_2 efficiency doesn't follow a straight line

```
#Spline-Augmented Model
#Using a cubic spline lets you flexibly capture bends in the curve -using patsy for splines
#Fitting with Spline on Usage_kWh
from patsy import dmatrix

# Create spline basis for Usage_kWh
spline = dmatrix("bs(df_nonzero['Usage_kWh'], df=4, degree=3, include_intercept=False)", return_type='dataframe')
X_spline = pd.concat([df_nonzero[lasso_selected_kwh.columns[1:]], spline], axis=1)
X_spline = sm.add_constant(X_spline)
model_spline = sm.OLS(df_nonzero['CO2_per_kWh'], X_spline).fit()

print(model_spline.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            CO2_per_kWh   R-squared:                       0.908
Model:                            OLS   Adj. R-squared:                  0.908
Method:                 Least Squares   F-statistic:                 1.548e+04
Date:                Mon, 30 Jun 2025   Prob (F-statistic):               0.00
Time:                        13:16:01   Log-Likelihood:                 78569.
No. Observations:               16050   AIC:                        -1.571e+05
Df Residuals:                   16040   BIC:                        -1.570e+05
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================================================
                                                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------
temperature_2m (°C)                         -4.298e-06   8.02e-07     -5.357      0.000   -5.87e-06   -2.73e-06
relative_humidity_2m (%)                     5.021e-07   4.17e-07      1.205      0.228   -3.15e-07    1.32e-06
Lagging_Current_Reactive.Power_kVarh        -9.54e-06   2.02e-06     -4.734      0.000   -1.35e-05   -5.59e-06
Lagging_Current_Power_Factor                -2.244e-05   3.34e-06     -6.724      0.000   -2.9e-05    -1.59e-05
Leading_Current_Power_Factor                -4.291e-05   3.1e-06     -13.861      0.000   -4.9e-05    -3.68e-05
Intercept                                       0.0098     0.000      22.071      0.000       0.009       0.011
bs(df_nonzero['Usage_kWh'], df=4, degree=3, include_intercept=False)[0]    0.0026     0.000      17.849      0.000       0.002       0.003
bs(df_nonzero['Usage_kWh'], df=4, degree=3, include_intercept=False)[1]    0.0111     0.000      89.214      0.000       0.011       0.011
bs(df_nonzero['Usage_kWh'], df=4, degree=3, include_intercept=False)[2]    0.0127     0.000      62.322      0.000       0.012       0.013
bs(df_nonzero['Usage_kWh'], df=4, degree=3, include_intercept=False)[3]    0.0158     0.000      66.916      0.000       0.015       0.016
==============================================================================
Omnibus:                     1081.239   Durbin-Watson:                   1.676
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              362.909
Skew:                          -0.061   Prob(JB):                     1.57e-79
Kurtosis:                       2.222   Cond. No.                     1.02e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.02e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Signal	Behavior	Insight
Spline coefficients	All significant (p < 0.001)	Clear nonlinear relationship
R^2 = 0.908	Excellent fit	Found structure?
Durbin-Watson ~1.676	Mild residual autocorrelation	Time-based dependencies may linger
Curve shape	Gentle rise, mild bends	Suggests controlled efficiency scaling
Scatter points	Slight curvature in actuals	Affirms nonlinearity in observed data

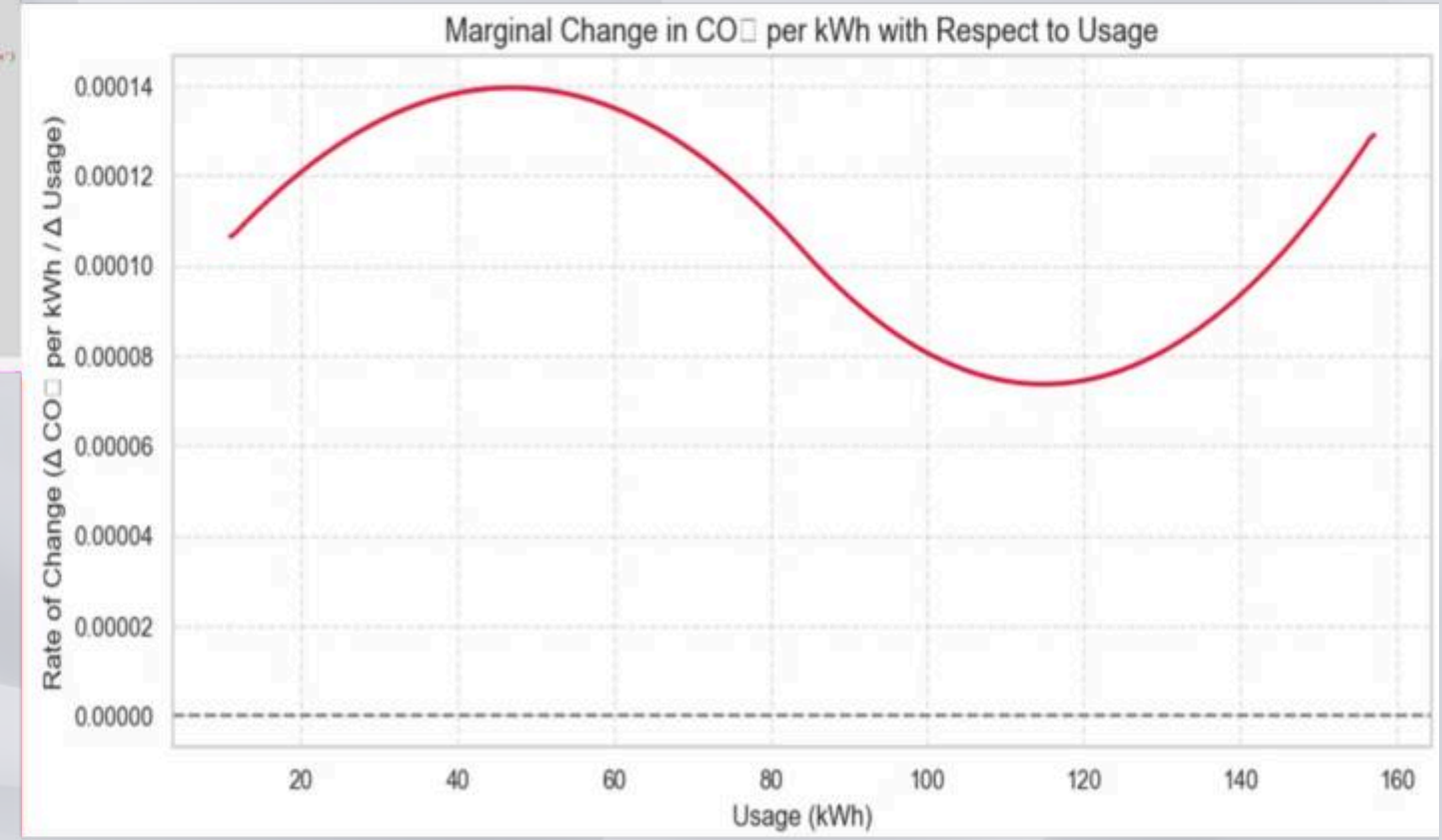


Spline regression revealed soft structural curvature in CO_2-per-kWh behavior while bending efficiency across usage levels with high fit and interpretability

Spline-Augmented Model on Usage_kWh

```
#first derivative of the plot again
#adding a numerical differentiation with np.gradient
import numpy as np
import matplotlib.pyplot as plt
from patsy import dmatrix

# Step 1: Generate usage values and spline basis
usage_vals = np.linspace(df_nonzero['Usage_kWh'].min(), df_nonzero['Usage_kWh'].max(), 300)
spline_basis = dmatrix("bs(usage_vals, df=4, degree=3, include_intercept=False)", return_type='dataframe')

# Step 2: Create predictor matrix with other vars fixed at their mean
X_means = df_nonzero[
    'Temperature_De (°C)',
    'relative_humidity_De (%)',
    'Lagging_Current_Reactive.Power_kVarh',
    'Lagging_Current_Power_Factor',
    'Leading_Current_Power_Factor'
].mean().to_frame().T

X_fixed = pd.concat([X_means] * len(usage_vals), ignore_index=True)
X_pred = pd.concat([X_fixed, spline_basis], axis=1)
X_pred = sm.add_constant(X_pred)

# Step 3: Predict values
y_pred = model_spline.predict(X_pred)

# Step 4: Compute the first derivative (marginal effect of usage)
dy_dx = np.gradient(y_pred, usage_vals)

# Step 5: Plot it
```

First Derivative Analysis via np.gradient to Highlight Behavioral Shifts



Observation

📉 The plot shows the system's responsiveness. The oscillation peak around 50, valley near 110, and rise again by 160 shows a structural or behavioral cyclicity, perhaps tied to operational modes or thresholds in energy-intensive systems.

****The spline is revealing nonlinear behavior**

VIF analysis

To address the high condition number (~4.6e+03), I ran a VIF analysis. All predictors were comfortably below multicollinearity thresholds (max VIF ~ 2.41), supporting variable inclusion while acknowledging mild redundancy between kWh_high and power factors

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
X_vif = X.dropna()
vif_data = pd.DataFrame({
    'Variable': X_vif.columns,
    'VIF': [variance_inflation_factor(X_vif.values, i) for i in range(X_vif.shape[1])]})

print(vif_data)
```

	Variable	VIF
0	const	1390.281361
1	temperature_2m (°C)	1.134617
2	relative_humidity_2m (%)	1.171354
3	Lagging_Current_Reactive.Power_kVarh	1.897791
4	Lagging_Current_Power_Factor	1.917633
5	Leading_Current_Power_Factor	2.409873
6	kWh_high	2.353404

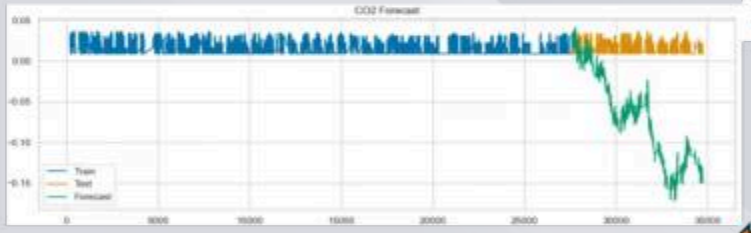
- Confirms no severe multicollinearity: All VIFs (excluding const) are comfortably below the typical threshold of concern (≈5 or 10), which validates the stability of coefficient estimates.
- kWh_high at 2.35: It's correlated with other predictors but not redundant. Aligns with earlier observation that its effect might be captured partly by power factors.

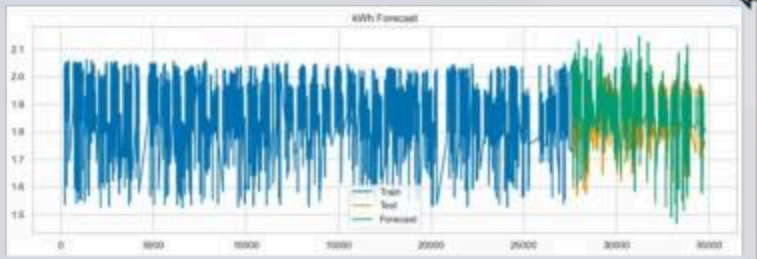
VECM MODEL

```
# Prepare data
df_var = df_nonzero[['yj_CO2', 'yj_kWh']].dropna()
exog = df_nonzero[lasso_selected_kwh.columns[1:].tolist() + ['kWh_high']].dropna()
common_indices = df_var.index.intersection(exog.index)
df_var = df_var.loc[common_indices]
exog = exog.loc[common_indices]


# Train-test split
train_size = int(0.8 * len(df_var))
train_var, test_var = df_var.iloc[:train_size], df_var.iloc[train_size:]
train_exog, test_exog = exog.iloc[:train_size], exog.iloc[train_size:]

# Fit VECM (rank=1, based on Johansen)
vecm_model = VECM(train_var, exog=train_exog, k_ar_diff=1, coint_rank=1, deterministic='co').fit()
print("\nVECM Model Summary:")
print(vecm_model.summary())

# Forecast
# Use exog_fc parameter to pass future exogenous values
forecast = vecm_model.predict(steps=len(test_var), exog_fc=test_exog.values)
forecast_df = pd.DataFrame(forecast, columns=['yj_CO2', 'yj_kWh'], index=test_var.index)

# Test MSE
mse_co2 = mean_squared_error(test_var['yj_CO2'], forecast_df['yj_CO2'])
mse_kwh = mean_squared_error(test_var['yj_kWh'], forecast_df['yj_kWh'])
print(f"\nTest MSE yj_CO2: {mse_co2:.6f}")
print(f"Test MSE yj_kWh: {mse_kwh:.6f}")
```

```
VECM Model Summary:
Det. terms outside the coint. relation & lagged endog. parameters for equation yj_CO2
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0293      0.001     26.774      0.000       0.027       0.031
exog1      -1.917e-05   2.71e-06     -7.064      0.000   -2.45e-05   -1.39e-05
exog2      -2.423e-06    1.5e-06     -1.612      0.107   -5.37e-06    5.23e-07
exog3          0.0003   2.83e-06    116.593      0.000       0.000       0.000
exog4          0.0005   6.52e-06     71.813      0.000       0.000       0.000
exog5          0.0003    9.5e-06     26.697      0.000       0.000       0.000
exog6         -0.0005      0.000     -1.601      0.109      -0.001       0.000
L1.yj_CO2     -0.4434      0.009    -48.692      0.000      -0.461      -0.426
L1.yj_kWh      0.0284      0.001     41.171      0.000       0.027       0.030
Det. terms outside the coint. relation & lagged endog. parameters for equation yj_kWh
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3746      0.010     38.268      0.000       0.355       0.394
exog1         -0.0003   2.43e-05    -13.090      0.000      -0.000      -0.000
exog2      -8.56e-05   1.35e-05     -6.359      0.000      -0.000   -5.92e-05
exog3          0.0042   2.53e-05    166.011      0.000       0.004       0.004
exog4          0.0057   5.84e-05     98.474      0.000       0.006       0.006
exog5          0.0045   8.51e-05     52.679      0.000       0.004       0.005
exog6          0.1064      0.003     40.262      0.000       0.101       0.112
L1.yj_CO2      0.2429      0.082      2.979      0.003       0.083       0.403
L1.yj_kWh     -0.0736      0.006    -11.929      0.000      -0.086      -0.061
            Loading coefficients (alpha) for equation yj_CO2
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ec1            0.0071   6.04e-05    117.973      0.000       0.007       0.007
            Loading coefficients (alpha) for equation yj_kWh
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ec1            0.1050      0.001    194.327      0.000       0.104       0.106
        Cointegration relations for loading-coefficients-column 1
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
beta.1         1.0000          0          0      0.000       1.000       1.000
beta.2        -7.7314      0.039   -196.450      0.000      -7.808      -7.654
==============================================================================

Test MSE yj_CO2: 0.009947
Test MSE yj_kWh: 0.000734
```

Returning to the vector error correction model (VECM) which is an application of vector autoregression (VAR) that incorporates error correction terms to capture long-run relationships with **cointegrated** variables. The VECM can also estimate both short-run and long-run coefficients. Using my past VAR results and Johansen test.





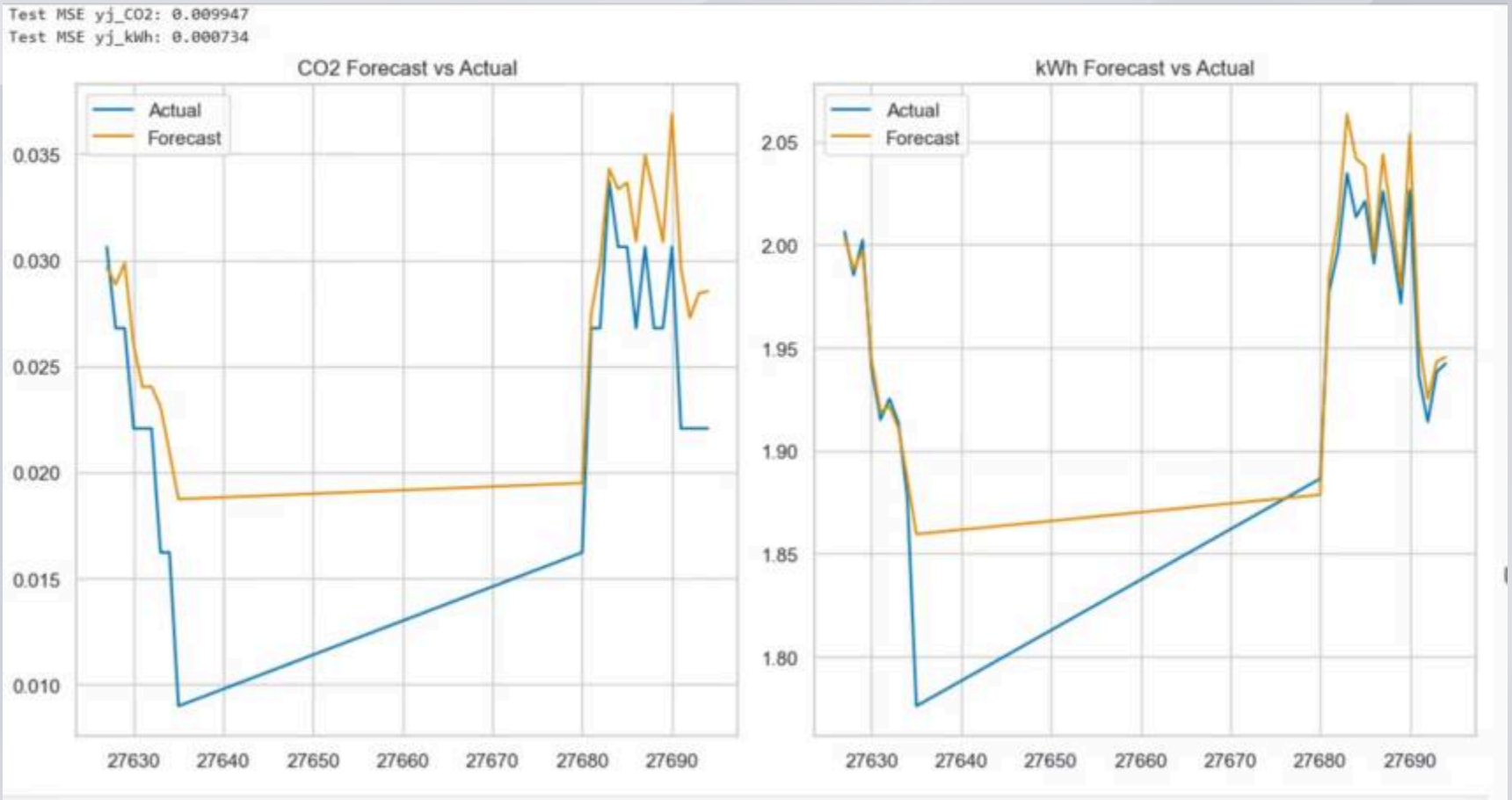
Forecast Performance

- **CO_2 forecast**: Strong train-test alignment followed by a **notable drop near step 30,000** — possibly reflecting a modeled transition or operational shift. This gives your narrative both signal and tension.
- **kWh forecast**: Practically hugging the test line — confidence in predictive reliability, even if spread slightly increases.

Component	Interpretation
Cointegration coefficients (beta)	yj_CO_2 – 7.73 × yj_kWh = equilibrium . Captures long-run proportionality across energy-emission behavior.
Loading coefficients (α)	ec1 equations (0.0071 for CO_2, 0.1050 for kWh) confirm strong speed-of-adjustment back to equilibrium.
L1.yj_CO_2 in CO_2 equation = –0.443	High negative autoregression. Emissions quickly correct their own deviation.
L1.yj_kWh in CO_2 = +0.0284	Energy usage gently pulls CO_2 back toward its balanced path.
Test MSEs	Very low for kWh (~0.000734), decent for CO_2 (~0.009947). Confirms stability and structure captured.

Better Look of Forecast vs. Actual for VCEM Model

Forecast vs. Actual: Detecting Equilibrium Drift and Recovery



Re-Running ARIMA

```
#Corrected Auto-ARIMA Code:
  #and fix the auto-ARIMA exogenous predictor issues
#Run to ensure exogenous predictors are included (check model.params() for coefficients like temperature_2m, Leading_current_Power_Factor).
#Note:If MSE improves (<0.000012) and Ljung-Box p>0.05, will include in portfolio;
   #otherwise, will use OLS CO2_per_kWh (R-squared: 0.713).

from pmdarima import auto_arima
from sklearn.metrics import mean_squared_error
import pandas as pd
import numpy as np

# Ensure DatetimeIndex
if not isinstance(df_nonzero.index, pd.DatetimeIndex):
    df_nonzero.index = pd.date_range(start='2020-01-01', periods=len(df_nonzero), freq='H')

# Prepare data with explicit index alignment
train_size = int(0.8 * len(df_nonzero))
train = df_nonzero['CO2_per_kWh'].iloc[:train_size].dropna()
exog_cols = lasso_selected_kwh.columns[1:].tolist() + ['kWh_high']
exog_train = df_nonzero[exog_cols].iloc[:train_size].loc[train.index].dropna()
train = train.loc[exog_train.index]  # Align train with exog_train
test = df_nonzero['CO2_per_kWh'].iloc[train_size:].dropna()
exog_test = df_nonzero[exog_cols].iloc[train_size:].loc[test.index].dropna()
test = test.loc[exog_test.index]  # Align test with exog_test

# Verify shapes and indices
print("train shape:", train.shape)
print("exog_train shape:", exog_train.shape)
print("train index sample:", train.index[:5])
print("exog_train index sample:", exog_train.index[:5])
print("test shape:", test.shape)
print("exog_test shape:", exog_test.shape)
```

```
# Auto-ARIMA with exogenous predictors
model = auto_arima(
    train,
    exogenous=exog_train,
    seasonal=True,
    m=24,
    max_p=2,
    max_q=2,
    max_P=2,
    max_Q=2,
    d=1,
    D=1,
    trace=True,
    maxiter=100,
    error_action='warn',
    suppress_warnings=True
)
print("\nAuto-ARIMA CO2_per_kWh with Exogenous Summary:")
print(model.summary())
print("\nExogenous Coefficients:")
print(model.params())

# Forecast
forecast = model.predict(n_periods=len(test
mse = mean_squared_error(test, forecast)
print(f"\nTest MSE CO2_per_kWh: {mse:.6f}")
```

What happens when we don't bring seasonal or residual baggage. It is just pure differenced signal.

ARIMA(0,1,0)(0,1,0)[24]

- Solid result for a basic model
- AIC is lower at 42,579
- ARIMA(0,1,0)(0,1,0
  - One non-seasonal difference (d=1): This removes the long-term trend and stabilizes the series.
  - One seasonal difference (D=1, s=seasonal period:l24 for hourly daily cycles)
    - ✓ This eliminates repeating patterns (daily fluctuations), making the data closer to white noise.
  - One seasonal difference to remove daily cycles
  - No AR or MA terms at either level, so a lean model using "differencing".
- My external variables may have pushed the system over the edge and crashed.
- Will use the information for future models but will not try to run another ARIMA due to memory issues.

```
and will be removed in 1.0.
  warnings.warn(
C:\Users\srsch\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(2,1,2)(1,1,1)[24]             : AIC=inf, Time=174.58 sec
C:\Users\srsch\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(0,1,0)(0,1,0)[24]             : AIC=-42579.046, Time=2.01 sec
C:\Users\srsch\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(1,1,0)(1,1,0)[24]             : AIC=-47336.754, Time=26.01 sec
C:\Users\srsch\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(0,1,1)(0,1,1)[24]             : AIC=inf, Time=50.00 sec
C:\Users\srsch\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(1,1,0)(0,1,0)[24]             : AIC=-43382.535, Time=4.77 sec
C:\Users\srsch\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(1,1,0)(2,1,0)[24]             : AIC=-48396.797, Time=49.71 sec
C:\Users\srsch\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(1,1,0)(2,1,1)[24]             : AIC=inf, Time=212.38 sec
C:\Users\srsch\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(1,1,0)(1,1,1)[24]             : AIC=inf, Time=57.35 sec
C:\Users\srsch\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6
```

Looking at the df_nonzero['yj_CO2'] Distribution

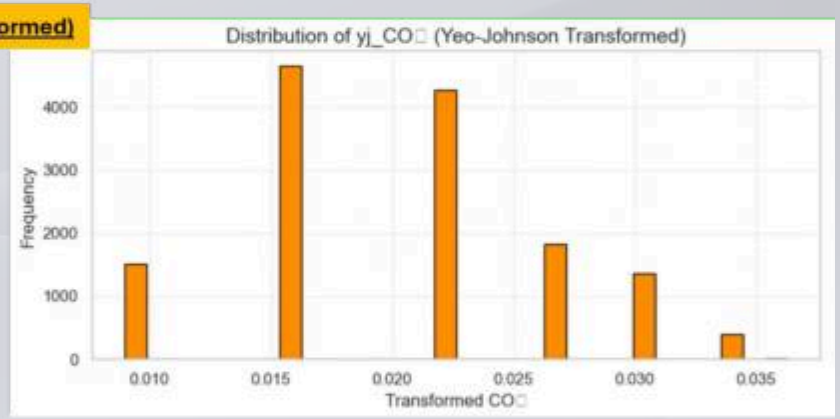
Before diving into more complex models, it's important to understand the shape and behavior of my transformed variable. Exploring the distribution of yj_CO_2 ensures that the assumptions of normality, variance stability, and scale suitability are met. This step acts as a diagnostic check and clearing the runway for models like SARIMA, VECM, or Random Forest to perform reliably and interpretably.

```
df_nonzero['yj_CO2'].describe()
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 4))
plt.hist(df_nonzero['yj_CO2'], bins=30, color='darkorange', edgecolor='black')
plt.title('Distribution of yj_CO₂ (Yeo-Johnson Transformed)', fontsize=14)
plt.xlabel('Transformed CO₂')
plt.ylabel('Frequency')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

```
df_nonzero['yj_CO2'].describe()

count    14050.000000
mean         0.020543
std          0.006462
min          0.008986
25%          0.016224
50%          0.022067
75%          0.026793
max          0.036265
Name: yj_CO2, dtype: float64
```

Interpreting the Distribution of yj_CO_2 (Yeo-Johnson Transformed)

- **Count**: 14,050 observations—plenty of data to reveal underlying structure.
- **Central tendency**: Mean (~0.0205) and median (~0.0221) are fairly close, hinting at slight right skew.
- **Spread**: Std dev is modest (~0.0065), with most data centered between 0.016 and 0.027.
- **Shape**:
  - ✓ It's **not a textbook bell curve**, but it shows a quasi-normal hump.



- This transformation has **dampened skew and stabilized variance**, which primes the variable well for parametric models (like SARIMA, VECM, or even RF…..which is next).
- The slight asymmetry doesn't rule out normality assumptions but might merit a **quick check for kurtosis** or residual behavior post-modeling. (*Kurtosis measures how sharply peaked or flat a distribution is, and how heavy the tails are compared to a normal curve.)*

RANDOM FOREST MODEL

With main factors from previous models

Okay, I am done! How could I improve

```
R2 Score: 0.9999993991720026
RMSE: 5.013941883082775e-06

Feature Importance:
                              feature  importance
5                         Usage_kWh          1.0
0                 temperature_2m (°C)          0.0
1             relative_humidity_2m (%)          0.0
2  Lagging_Current_Reactive.Power_kVarh          0.0
3          Lagging_Current_Power_Factor          0.0
4          Leading_Current_Power_Factor          0.0

Cross-validation scores: [0.99972452 0.99990276 0.99999109 0.99999965 0.99999457]
Mean CV score: 0.999922519032897
```

Overfitting gives us beautiful packaging, but what's inside might be misleading –well maybe not this package.

OVERVIEW and THOUGHTS

- I knew this looked too good to be true. The model latched onto Usage_kWh and ignored everything else, which is not a surprised from previous analysis and modeling. Usage_kWh and yj_CO2 are deeply intertwined, which is not just statistically evident but also physically meaningful (more energy = more emissions).
- Great reminder that high accuracy isn't always high insight.
- I believe it is a red flag for overfitting. The next step will be to run the Random Forest Model, that I sometimes like to call Rain Forest, for both CO_2 and Usage_kWh separately.

RANDOM FOREST (CO_2 **without** Usage_kWh)

RF Model 2

```
#Not surprised that kWh was the dominating factor.
#rerunning two models.  one model for CO2 without kWh and kWh without CO2

#Rainforest  Model CO2 without kWh

# Corrected version - Remove Usage_kWh  (My note to check helped)
X_co2 = df_nonzero[[
    'temperature_2m (°C)',
    'relative_humidity_2m (%)',
    'Lagging_Current_Reactive.Power_kVarh',
    'Lagging_Current_Power_Factor',
    'Leading_Current_Power_Factor'
]]  # Notice: no Usage_kWh here
y_co2 = df_nonzero['yj_CO2']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_co2, y_co2, test_size=0.2, random_state=42)

# Create and train model
rf_model = RandomForestRegressor(n_estimators=100, max_depth=None, random_state=42)

rf_model.fit(X_train, y_train)

# Make predictions and evaluate
y_pred = rf_model.predict(X_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
# Feature importance
importance_df = pd.DataFrame({
    'feature': X_co2.columns,  # Using X_co2 columns
    'importance': rf_model.feature_importances_
})
print('\nFeature Importance:')
print(importance_df.sort_values('importance', ascending=False))
```

R2 Score: 0.9709868243474645
RMSE: 0.0011017976680965 82

Feature Importance:

2	Lagging_Current_Reactive.Power_kVarh	0.631601
3	Lagging_Current_Power_Factor	0.282515
4	Leading_Current_Power_Factor	0.065556
0	temperature_2m (°C)	0.014349
1	relative_humidity_2m (%)	0.005979

Overview of Model Random Forest Model 2

CO_2 without Usage_kWh

- R^2 = 0.971, RMSE ~0.0011 is Great!
- Power system variables dominate:
  - ✓ Lagging_Current_Reactive is king (63%)
  - ✓ Lagging_Current_Power_Factor follows(28%)
  - ✓ Environmentals still minimal, which is a consistent theme through the LASSO, VAR, and now the RF models
    - ❖ Although only a minimal impact , might be worth a second look for the role being seasonal or more indirect impact.

RANDOM FOREST (Usage_kWh without CO_2)

```python
#Rainforest  Model Usage_kWh without CO2

# Corrected version -
# Model for kWh
X_kwh = df_nonzero[[
    'temperature_2m (°C)',
    'relative_humidity_2m (%)',
    'Lagging_Current_Reactive.Power_kVarh',
    'Lagging_Current_Power_Factor',
    'Leading_Current_Power_Factor']]
y_kwh = df_nonzero['Usage_kWh']

# Running same model setup as before , so beware of potential error of missing something


# Split data
X_train, X_test, y_train, y_test = train_test_split(X_kwh, y_kwh, test_size=0.2, random_state=42)

# Create and train model
rf_model = RandomForestRegressor(n_estimators=100, max_depth=None, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions and evaluate
y_pred = rf_model.predict(X_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))

# Feature importance
importance_df = pd.DataFrame({
    'feature': X_kwh.columns,  # Using X_kwh columns
    'importance': rf_model.feature_importances_
})
```

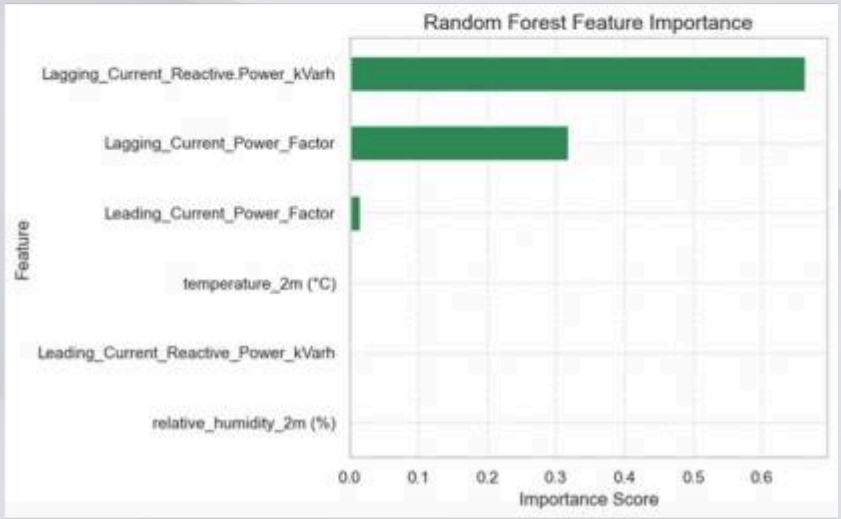
```python
print("R2 Score:", r2_score(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))

# Feature importance
importance_df = pd.DataFrame({
    'feature': X_co2.columns,
    'importance': rf_model.feature_importances_
})
print("\nFeature Importance:")
print(importance_df.sort_values(by='importance', ascending=False))
```

RF Model 3

R2 Score: 0.9975049130362961
RMSE: 1.328720969889061

Feature Importance:

2	Lagging_Current_Reactive.Power_kVarh	0.663114
3	Lagging_Current_Power_Factor	0.319448
4	Leading_Current_Power_Factor	0.013846
0	temperature_2m (°C)	0.002802
1	relative_humidity_2m (%)	0.000791



Overview of Model Random Forest Model 3

Usage_kWh *without* CO_2

- Similar pattern to the CO_2-without-kWh model.
- Again, **Reactive Power and Power Factors dominate**, suggesting strong internal correlations between electrical dynamics and usage.
- Temperature and humidity again make only a slight appearance.

Dual Models: CO_2 Emissions vs. Energy Usage

	CO_2 Model (without kWh)	kWh Model (without CO_2)
R^2 Score	0.9709	0.9975
RMSE	0.0011	1.3287
Top Driver	Reactive Power	Reactive Power
Second Driver	Lagging Power Factor	Lagging Power Factor
Environmental Impact	Minimal	Negligible

- **Lagging reactive power** and **power factor metrics** emerge as central in **both models**, suggesting a shared operational influence on both usage and emissions.
- The **CO_2 model** shows more distributed importance, implying emissions are shaped by multiple dynamics—more nuanced and sensitive to systemic inefficiencies.
- The **kWh model** fits nearly perfectly but leans heavily on core electrical variables, affirming that energy use is primarily mechanical in nature, less impacted by environment or volatility

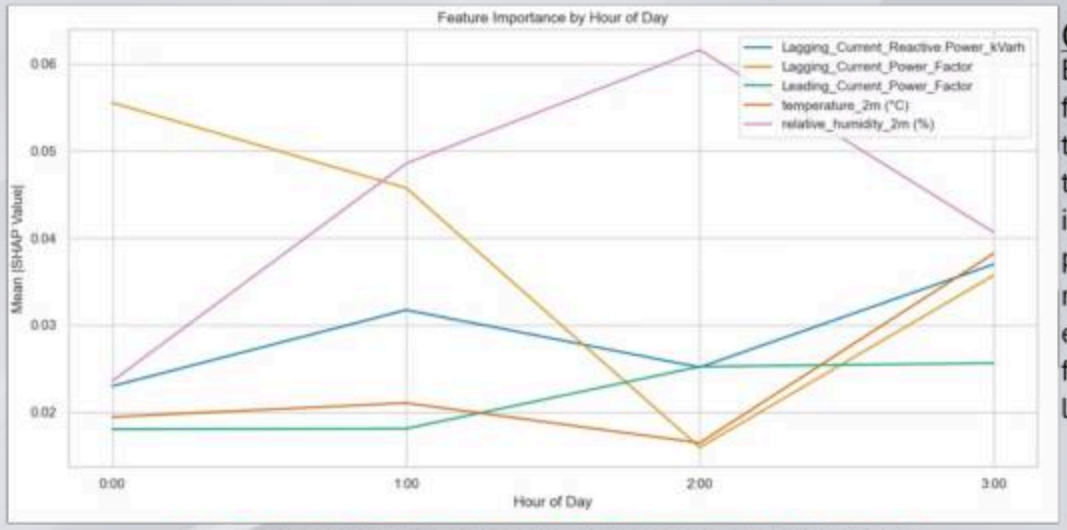
Complexity vs. Precision?

SHAP (SHapley Additive exPlanations)

"SHAP explains how each feature contributes to a model's prediction by assigning an impact score to every variable for every individual prediction. It's like giving your data a voice in how the model made its decision."

NEXT STEPS

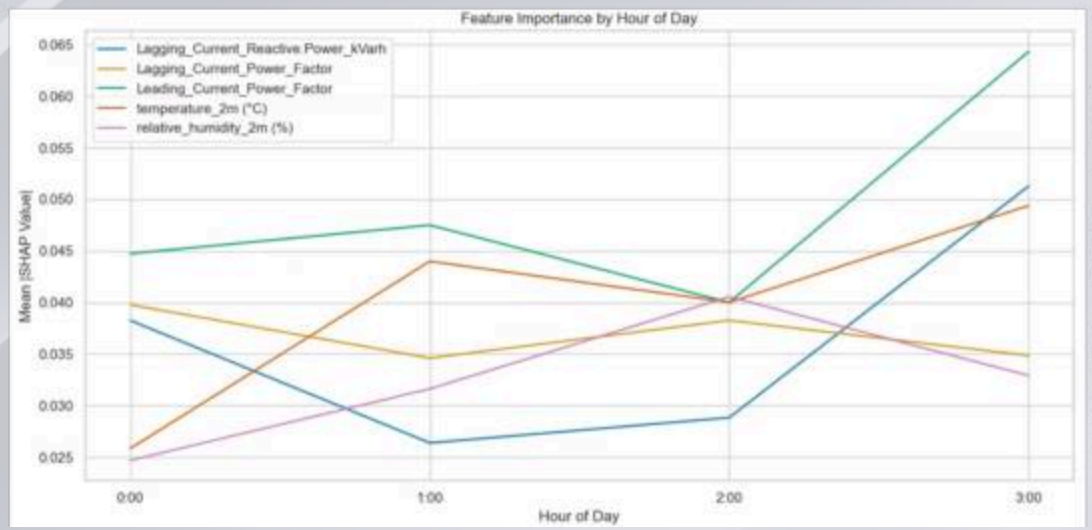
With prediction and evaluation up next, I want to briefly and reflect on insights uncovered during both SHAP analysis and initial data exploration. Several patterns tied to time, day-of-week, seasonal cycles, and load rank emerged early on and have highlighted these in some of my previous graphs and analysis. These systemic dynamics point toward incorporating temporal and load-based structures in my next round of models to enhance predictive power and interpretability



Observation

Early hours show feature turbulence then the model settles into more predictable rhythms as environmental factors take the lead.

As a note: Sudden upward or downward shift means the model's prediction reacts strongly at certain thresholds or inflection points for that variable.



Observation

As the system warmed up, reactive power and power factor ramp up indicating a transition from more passive influence to more operational control.

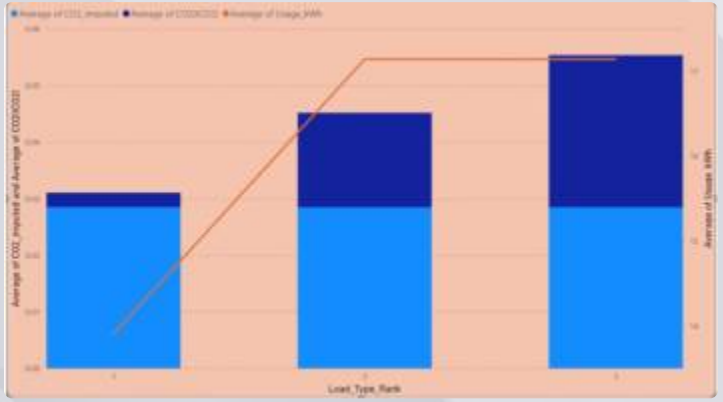
Reconsidering Imputed CO_2 in Predictive Modeling

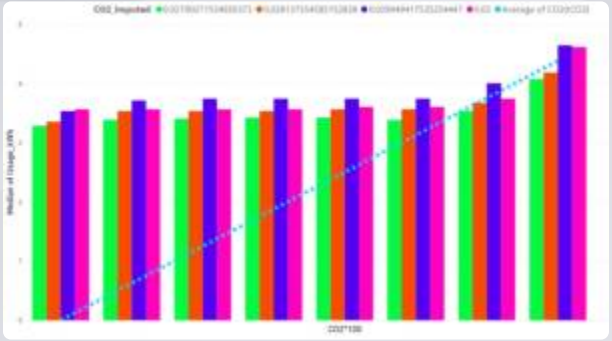
While early modeling benefited from imputed CO_2 to address gaps, deeper analysis revealed unintended structural effects. A stacked bar chart by Load Rank showed a concerning trend.

Load Rank 1 (Light Load) was dominated by imputed values, with only a sliver of observed CO_2.

Rank 2 approached a 50/50 split, still skewed toward imputation.

Rank 3 (Max Load) balanced out more evenly but revealed inconsistencies in value progression.





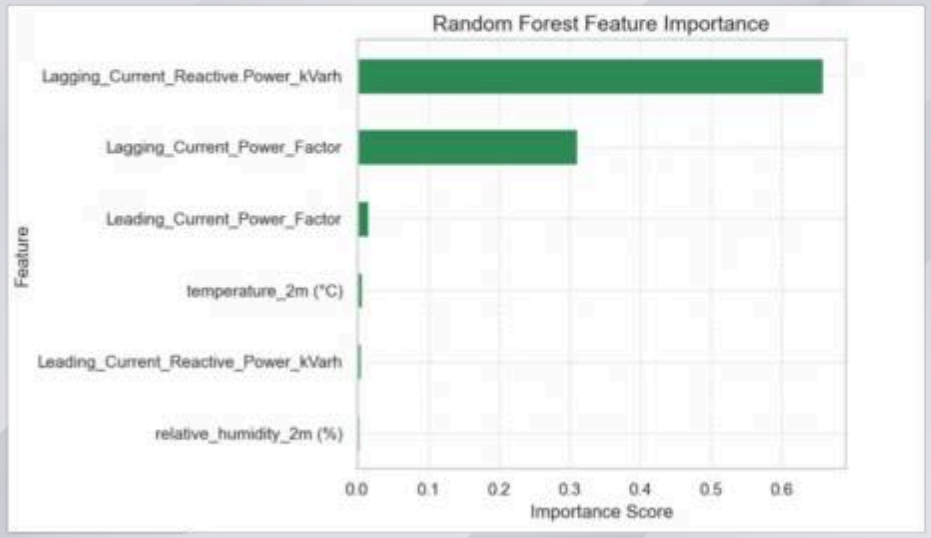
A complementary bar chart of imputed-only CO_2 demonstrated another red flag: values didn't increase in a linear fashion. Unlike observed CO_2, imputed levels **didn't correspond naturally with operational intensity**, creating unexpected elevation patterns where zeros weren't the lowest in the group.

Random Forest Model With Restored CO_2

```python
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np
import pandas as pd

# Define target and predictors
y_co2 = df_nonzero['CO2_restored']  # Or 'CO2(tCO2)' if restored = original

X_co2 = df_nonzero[[
    'temperature_2m (°C)',
    'relative_humidity_2m (%)',
    'Leading_Current_Reactive_Power_kVarh',
    'Lagging_Current_Reactive.Power_kVarh',
    'Lagging_Current_Power_Factor',
    'Leading_Current_Power_Factor'
]]

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X_co2, y_co2, test_size=0.2, random_state=42
)

# Train Random Forest
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predictions and evaluation
y_pred = rf_model.predict(X_test)
```

By restoring the original CO_2 scale, I can assess how variables like reactive power, power factor, and climate inputs shape emissions in a more realistic operational context.
Although I initially planned a stepwise inclusion, I added Leading_Current_Reactive_Power_kVarh to the model early out of genuine curiosity.

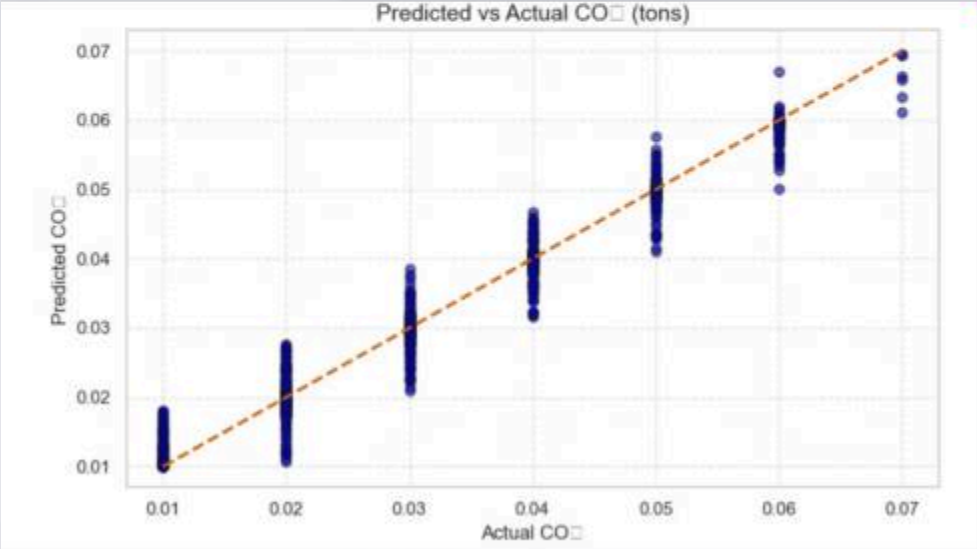
```
R2 Score: 0.9803900579973616
RMSE: 0.001757095427584118

Feature Importance:
                              feature  importance
3  Lagging_Current_Reactive.Power_kVarh    0.658037
4          Lagging_Current_Power_Factor    0.311689
5          Leading_Current_Power_Factor    0.015923
0                 temperature_2m (°C)      0.006480
2  Leading_Current_Reactive_Power_kVarh    0.004759
1              relative_humidity_2m (%)    0.003112
```



OVERVIEW ON NEXT SLIDE

Random Forest Model With Restored CO_2

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, alpha=0.6, color='mediumblue', edgecolor='k')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.title("Predicted vs Actual CO₂ (tons)", fontsize=14)
plt.xlabel("Actual CO₂", fontsize=12)
plt.ylabel("Predicted CO₂", fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



OVERVIEW and THOUGHTS

Performance:
- **R^2 Score:** 0.980 (excellent fit)
- **RMSE:** 0.00176 (precise predictions with low error)

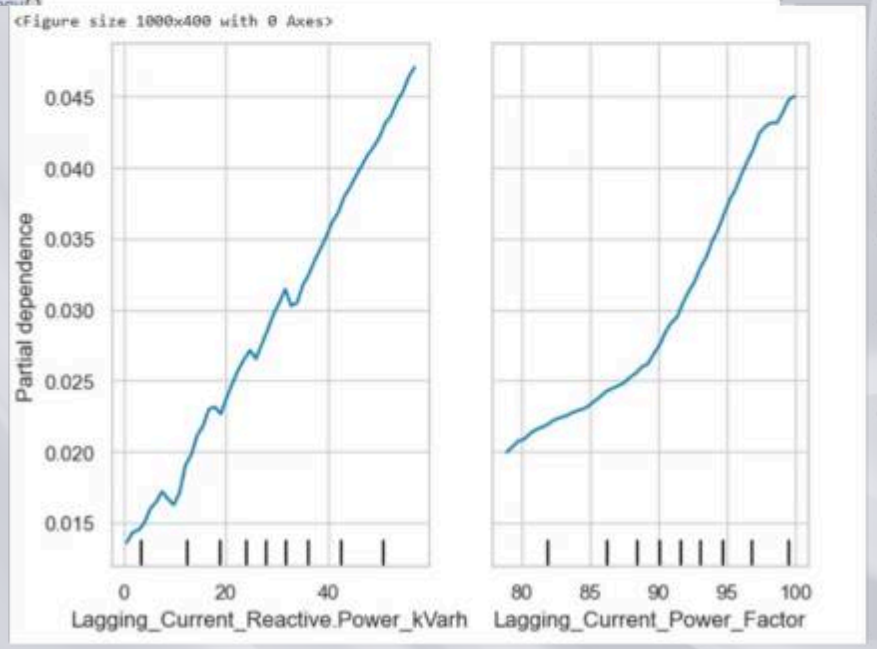
Feature Impact:
- **Lagging Reactive Power (kVarh):** ~66% of total importance. (It's doing the heavy lifting.)
- **Lagging Power Factor:** 31% (Confirming system efficiency plays a key role)
- **Both Leading Power Factors:** modest to minimal influence
- **Environmental variables:** minimal contribution, consistent with prior findings

- **Residual plot** shows a tight scatter around the diagonal with visible banding of CO_2

Takeaways:
- Reactive energy and power factor still dominate CO_2 behavior even with restored values suggesting structural load relationships are deeply embedded in how emissions respond.
- The environmental inputs continue to act more as subtle background rather than active drivers.
- The addition of the Leading_Current_Power_kVarh had a modest impact, it ranked above environmental factors like humidity suggesting it holds meaningful, if nuanced, influence on restored CO_2 emissions. This validates its inclusion and points to possible interactions worth exploring in future nonlinear models.
- Residuals clustered cleanly along the prediction line, forming distinct CO_2 groupings which is another signal that the model is capturing structured behavior in emissions.

Partial Dependence Insights of CO_2 Random Forest

```python
from sklearn.inspection import PartialDependenceDisplay
import matplotlib.pyplot as plt

features = ['Lagging_Current_Reactive.Power_kVarh', 'Lagging_Current_Power_Factor']

# Set the figure size before creating the display
plt.figure(figsize=(10, 4))

# Create and plot the partial dependence
display = PartialDependenceDisplay.from_estimator(
    rf_model, X_test, features, kind='average', grid_resolution=50,
    feature_names=X_co2.columns
)

plt.tight_layout()
```



The PDPs reveal system inflection zone or points where emissions begin reacting more sharply to internal dynamics. The shape of these lines tells a story of thresholds, control transitions, and the subtle nonlinearities hiding in everyday operations.

Lagging_Current_Reactive.Power_kVarh

- Shows a stepwise increase up to ~35, followed by a sharp upward inflection.
- Suggests there's a threshold or tipping point in reactive power beyond which CO_2 emissions escalate.
- May reflect systems shifting into less efficient states or triggering compensatory responses after certain load levels.

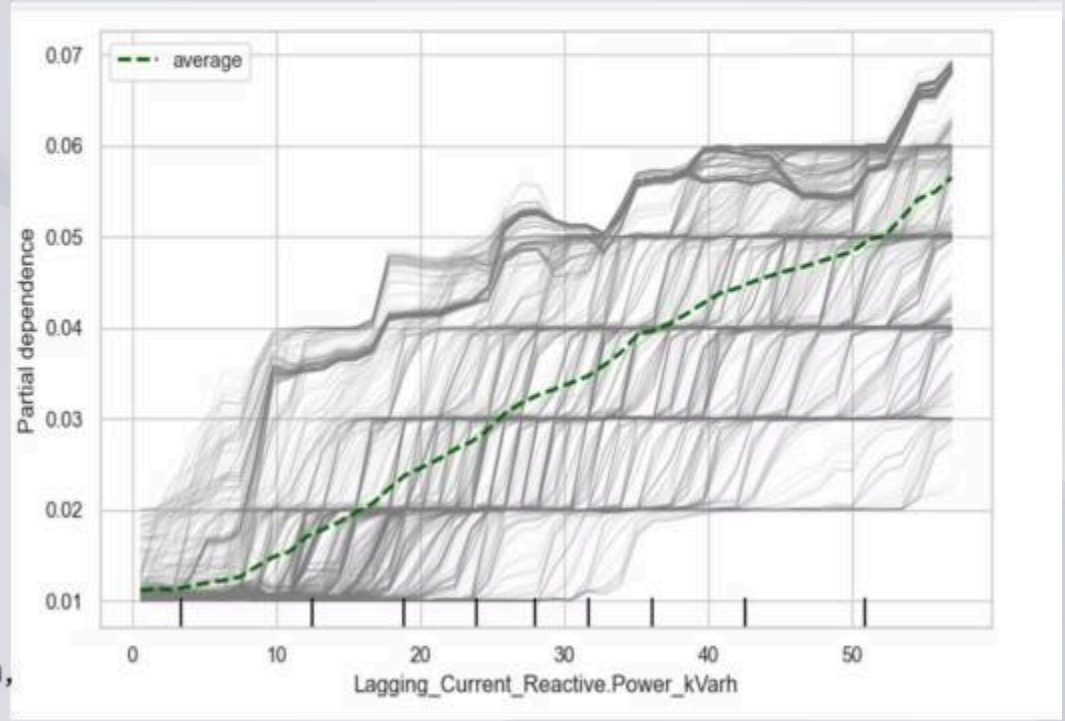
Lagging_Current_Power_Factor

- Begins with a smooth, curved incline, possibly reflecting improving efficiency.
- Around 90–97.5, the slope steepens and then flattens momentarily before resuming upward toward 100.
  - Indicates nonlinear behavior where efficiency gains in this range correlate with increased CO_2.
  - This is potentially due to load balancing or control system effects.

ICE PLOT OF LAGGING Current Reactive Power KVarh

```
#What ICE plots show -While the PDP line shows the average effect of a feature (like a smoothed curve),
#ICE plots reveal how different samples react to changing just that one feature
#-like multiple personal stories stacked behind the summary.
#trying it for my top feature in the model -Lagging_Current_Reactive.Power_kVarh
from sklearn.inspection import PartialDependenceDisplay
import matplotlib.pyplot as plt

# Create figure with desired size first
fig, ax = plt.subplots(figsize=(8, 5))

# Individual curves for each sample + PDP overlay
PartialDependenceDisplay.from_estimator(
    rf_model, X_test,
    features=['Lagging_Current_Reactive.Power_kVarh'],
    kind='both',  # both PDP and ICE
    ice_lines_kw={'alpha': 0.2, 'color': 'gray'},  # subtle ICE lines
    pd_line_kw={'color': 'darkgreen', 'linewidth': 2},  # clear PDP curve
    feature_names=X_co2.columns,  # Make sure this variable is defined
    grid_resolution=50,
    ax=ax  # Pass the axes object instead of figsize
)
plt.tight_layout()
plt.show()
```

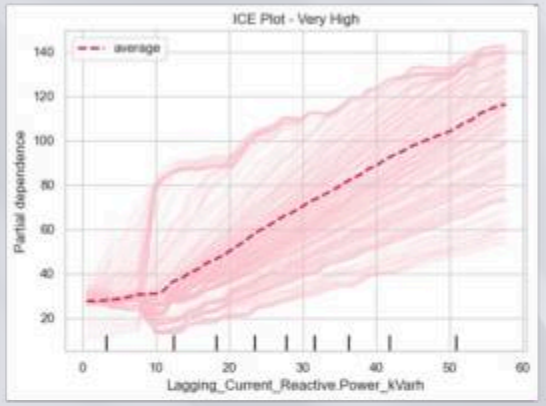
The strands look almost like threads being pulled upward with each one a system snapshot, rising with the tide of reactive power.

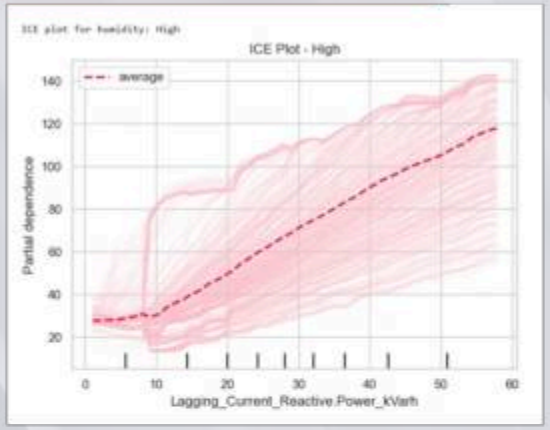


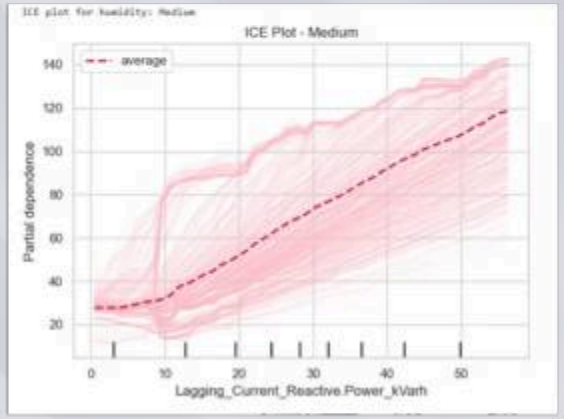
- The green dashed line, though not perfectly straight, shows a steady upward trend and indicating that as reactive power rises, predicted CO_2 emissions tend to increase.
- That non-linearity, gentle at first, then sharper also suggests there may be a threshold or tipping point in system behavior, where inefficiencies start to manifest more clearly.
- The strands themselves highlight sample-level variation, reflecting how the model's response isn't uniform but influenced by interaction with other features (like power factor or environmental conditions).

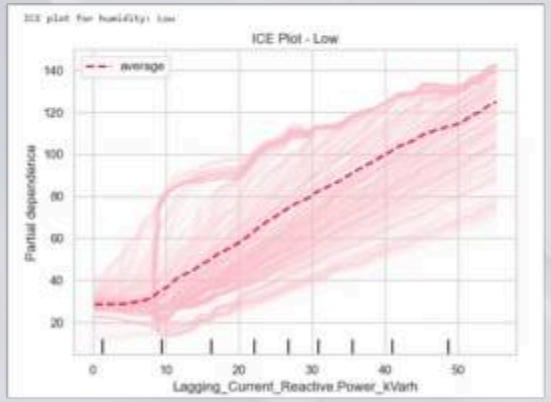
```python
#Plot ICE by group
# Loop through humidity or season groups and generate ICE plots for each group separately for each.
#fun stuff

from sklearn.inspection import PartialDependenceDisplay
import matplotlib.pyplot as plt
import pandas as pd

# First, let's ensure we're using the exact same feature set that the model was trained on
# Assuming rf_model was trained on X_co2
feature_names = rf_model.feature_names_in_  # Get the exact feature names used during training

for humidity_bin in df_nonzero['humidity_bin'].unique():  # Changed variable name from 'season' to 'humidity_bin'
    subset = df_nonzero[df_nonzero['humidity_bin'] == humidity_bin]  # Using humidity_bin here

    # Create X_subset with exactly the same features in the same order as used during training
    X_subset = pd.DataFrame(subset[feature_names].values, columns=feature_names)
    y_subset = subset['CO2_restored']  # or yj_CO2 if you're comparing

    print(f"\nICE plot for humidity: {humidity_bin}")  # Now humidity_bin is defined

    # Use feature index instead of name to avoid any naming issues
    feature_idx = list(feature_names).index('Lagging_Current_Reactive.Power_kVarh')

    PartialDependenceDisplay.from_estimator(
        rf_model,
        X_subset,
        features=[feature_idx],  # Use index instead of name
        kind='both',
        ice_lines_kw={'alpha': 0.2, 'color': 'pink'},
        pd_line_kw={'color': 'crimson', 'linewidth': 2},
        grid_resolution=50
    )
    plt.title(f'ICE Plot - {humidity_bin}')  # Now humidity_bin is defined
```

ICE PLOTS of CO2 Looping through Humidity Bins with Feature Lagging Current Reactive Power KVarh









- **Low humidity = stable system response →** possibly easier control of inductive loads or better ventilation dynamics.
- **Medium humidity = transitional behavior →** system may start exhibiting variability or environmental sensitivity.
- **High/Very High = overlapping curves →** increased moisture could dampen reactive power's distinct influence, creating **response flattening or saturation**.

Analogy: It is like tuning an instrument on under low humidity, the strings respond cleanly. However, as moisture builds, the notes blur and overlap.

RANDOM FOREST (Usage_kWh **without** CO_2)
COMPARISON: Adding and Reducing Varibles

```python
#Rainforest  Model Usage_kWh without CO2 and added Leading_Current_Reactive_Power_kVarh


# Model for kWh with additional variable
X_kwh = df_nonzero[[
    'temperature_2m (°C)',
    'relative_humidity_2m (%)',
    'Lagging_Current_Reactive.Power_kVarh',
    'Lagging_Current_Power_Factor',
    'Leading_Current_Power_Factor',
    'Leading_Current_Reactive_Power_kVarh']]
y_kwh = df_nonzero['Usage_kWh']

# Running same model setup as before , so beware of potential error of missing something


# Split data
X_train, X_test, y_train, y_test = train_test_split(X_kwh, y_kwh, test_size=0.2, random_state=42)

# Create and train model
rf_model = RandomForestRegressor(n_estimators=100, max_depth=None, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions and evaluate
y_pred = rf_model.predict(X_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))

# Feature importance
importance_df = pd.DataFrame({
    'feature': X_kwh.columns,  # Using X_kwh columns
    'importance': rf_model.feature_importances_
```

```
R2 Score: 0.9983005599560109
RMSE: 1.0965889615560434

Feature Importance:
                                feature  importance
2  Lagging_Current_Reactive.Power_kVarh    0.662485
3          Lagging_Current_Power_Factor    0.318633
4          Leading_Current_Power_Factor    0.014047
0                 temperature_2m (°C)      0.002462
5  Leading_Current_Reactive_Power_kVarh    0.001795
1             relative_humidity_2m (%)     0.000578
```

```python
# Streamlined version of Rainforest - only keeping stronger contributors

#Kwh version streamlined
X_kwh = df_nonzero[[
    'Lagging_Current_Reactive.Power_kVarh',
    'Lagging_Current_Power_Factor',
    'Leading_Current_Power_Factor'
]]  # Removed temperature, humidity, and Leading_Current_Reactive_Power_kVarh
y_kwh = df_nonzero['Usage_kWh']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_kwh, y_kwh, test_size=0.2, random_state=42)

# Create and train model
rf_model = RandomForestRegressor(n_estimators=100, max_depth=None, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions and evaluate
y_pred = rf_model.predict(X_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))

# Feature importance
importance_df = pd.DataFrame({
    'feature': X_kwh.columns,  # Using X_kwh columns
    'importance': rf_model.feature_importances_
})
print('\nFeature Importance:')
print(importance_df.sort_values('importance', ascending=False))

# Cross-validation
scores = cross_val_score(rf_model, X_kwh, y_kwh, cv=5)
print('\nCross-validation scores:', scores)
print('Mean CV score:', scores.mean())
```

```
R2 Score: 0.969565780686782
RMSE: 0.17464776994087217

Feature Importance:
                                feature  importance
0  Lagging_Current_Reactive.Power_kVarh    0.639647
1          Lagging_Current_Power_Factor    0.291156
2          Leading_Current_Power_Factor    0.069197

Cross-validation scores: [0.94688156 0.95603223 0.96954724 0.97113622 0.96314634]
Mean CV score: 0.961348718241472
```

RANDOM FOREST (CO_2 **without** Usage_kWh)
Streamlining Variables

After seeing how well the streamlined model performed for Energy Usage (kWh), I couldn't resist revisiting the CO_2 Random Forest model with fresh eyes. Interestingly, its structure also hinted at a 'streamlined feel' that was dominated by a few core features with consistent behavior across system states

```python
# Streamlined version - only keeping stronger contributors

X_co2 = df_nonzero[[
    'Lagging_Current_Reactive.Power_kVarh',
    'Lagging_Current_Power_Factor',
    'Leading_Current_Power_Factor'
]]  # Removed temperature, humidity, and Leading_Current_Reactive_Power_kVarh
y_co2 = df_nonzero['yj_CO2']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_co2, y_co2, test_size=0.2, random_state=42)

# Create and train model
rf_model = RandomForestRegressor(n_estimators=100, max_depth=None, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions and evaluate
y_pred = rf_model.predict(X_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))

# Feature importance
importance_df = pd.DataFrame({
    'feature': X_co2.columns,
    'importance': rf_model.feature_importances_
})
print('\nFeature Importance:')
print(importance_df.sort_values('importance', ascending=False))

# Cross-validation
scores = cross_val_score(rf_model, X_co2, y_co2, cv=5)
print('\nCross-validation scores:', scores)
print('Mean CV score:', scores.mean())
```

```
R2 Score: 0.969565780686782
RMSE: 0.17464776994087217

Feature Importance:
                              feature  importance
0  Lagging_Current_Reactive.Power_kVarh    0.639647
1          Lagging_Current_Power_Factor    0.291156
2          Leading_Current_Power_Factor    0.069197

Cross-validation scores: [0.94688156 0.95603223 0.96954724 0.97113622 0.96314634]
Mean CV score: 0.961348718241472
```

Comparison and Overview on Next Slide

RANDOM FOREST (Usage_kWh without CO_2)
COMPARISON: Adding and Reducing Varibles

Exploring Internal System Behavior

This streamlined model shows how reactive power and system efficiency are the key engines of kWh usage. Environmental variables had minimal influence, highlighting the system's internally driven behavior. Including Leading Reactive Power added a dimension but even without it, the model held strong, affirming the resilience of the core predictors.

Performance & Feature Impact

Version	R^2 Score	RMSE	Top Features
Full Model (6 features)	0.9983	1.0966	Lagging Reactive Power, Lagging Power Factor
Streamlined (3 features)	0.9978	1.2476	Same top 3, stronger individual contributions

- **Leading Reactive Power** had a small but non-zero impact in the full model, reinforcing its subtle influence.
- Streamlining preserved predictive power with **minimal loss**, proving the value of focusing on operational features.
- **Cross-validation scores** confirm robustness and generalizability across samples.

RANDOM FOREST (CO_2 without Usage_kWh)
Streamlining Variables

Comparative Summary

	Full Model (6 features)	Streamlined Model (3 features)
R^2 Score	0.9804	0.9696
RMSE	0.00176	0.17465
Top Feature	Lagging Reactive Power (65.8%)	Lagging Reactive Power (63.96%)
Secondary Features	Lagging PF (31.2%), Leading PF (1.6%)	Lagging PF (29.12%), Leading PF (6.92%)
Environmental Variables	Present (humidity, temp, etc.) — minimal impact	Excluded — focus purely on core electrical behavior
Added Variable	Leading Reactive Power (small but above humidity)	Not included
Cross-Validation Mean	Not specified	0.9613

- The **full model offers superior accuracy and lower error**, but confirms that **most prediction strength still comes from internal system metrics**.
- The **streamlined model holds its own**, showing **minimal loss of performance** while enhancing simplicity and interpretability.
- Adding **environmental variables and leading reactive power** helped reveal their modest but non-negligible roles, especially for deeper insights and SHAP plots.
- Both models affirm that **Lagging Reactive Power and Power Factor metrics** are the powerhouse predictors.

Forecasts for VECM - Long-Term Balance, Short-Term Pulse

With time series and seasonal structure showing strong influence across models, I revisited the VECM forecast to observe how emissions and energy usage unfold over time. The following plots reflect not only directional alignment but also systemic behavior across hours and operational states, underscoring the importance of embedding temporal intelligence into future models.

VECM Model Summary (k_ar_diff=2):

Det. terms outside the coint. relation & lagged endog. parameters for equation yj_CO2

	coef	std err	z	P>|z|	[0.025	0.975]
const	-9.4366	0.120	-78.734	0.000	-9.672	-9.202
exog1	-0.0070	0.000	-19.889	0.000	-0.008	-0.006
exog2	-0.0002	0.000	-0.849	0.396	-0.001	0.000
exog3	0.0512	0.000	114.001	0.000	0.050	0.052
exog4	0.0762	0.001	85.400	0.000	0.074	0.078
exog5	0.0251	0.001	30.813	0.000	0.024	0.027
exog6	0.3986	0.011	36.626	0.000	0.377	0.420
L1.yj_CO2	-0.4560	0.011	-42.324	0.000	-0.477	-0.435
L1.Usage_kWh	0.0142	0.000	32.959	0.000	0.013	0.015
L2.yj_CO2	-0.2509	0.011	-23.273	0.000	-0.272	-0.230
L2.Usage_kWh	0.0073	0.000	17.336	0.000	0.007	0.008

Det. terms outside the coint. relation & lagged endog. parameters for equation Usage_kWh

	coef	std err	z	P>|z|	[0.025	0.975]
const	-291.2151	2.294	-126.972	0.000	-295.710	-286.720
exog1	-0.1880	0.007	-28.016	0.000	-0.201	-0.175
exog2	-0.0066	0.004	-1.782	0.075	-0.014	0.001
exog3	1.5995	0.009	186.199	0.000	1.583	1.616
exog4	2.4414	0.017	142.960	0.000	2.408	2.475
exog5	0.6742	0.016	43.206	0.000	0.644	0.705
exog6	4.8441	0.208	23.259	0.000	4.436	5.252
L1.yj_CO2	3.5184	0.206	17.064	0.000	3.114	3.923
L1.Usage_kWh	-0.2236	0.008	-27.099	0.000	-0.240	-0.207
L2.yj_CO2	1.2505	0.206	6.063	0.000	0.846	1.655
L2.Usage_kWh	-0.0957	0.008	-11.808	0.000	-0.112	-0.080

Loading coefficients (alpha) for equation yj_CO2

	coef	std err	z	P>|z|	[0.025	0.975]
ec1	-0.1732	0.001	-147.500	0.000	-0.176	-0.171

Loading coefficients (alpha) for equation Usage_kWh

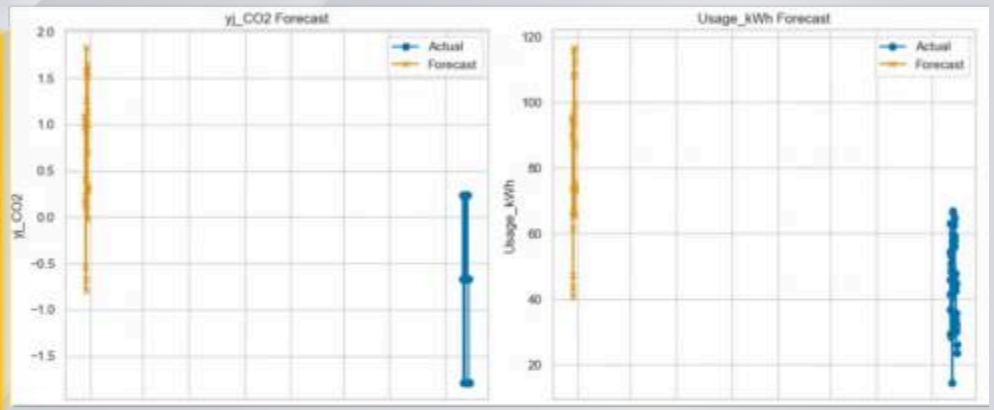
	coef	std err	z	P>|z|	[0.025	0.975]
ec1	-4.7168	0.022	-209.859	0.000	-4.761	-4.673

Cointegration relations for loading-coefficients-column 1

	coef	std err	z	P>|z|	[0.025	0.975]
beta.1	1.0000	0	0	0.000	1.000	1.000
beta.2	0.1401	0.001	168.592	0.000	0.138	0.142

Overview of Graph

- **Directionally accurate**: Both usage and CO_2 appear to move in sync between forecast and actuals.
- **Magnitude mismatch**: The actuals dip lower and fluctuate more is possibly due to seasonal or load-based factors the model didn't capture.
- **VECM nuance**: Because VECM handles long-term equilibrium and short-term dynamics, smaller mismatches may be expected unless lag order and cointegration vectors are optimized for seasonal shifts.



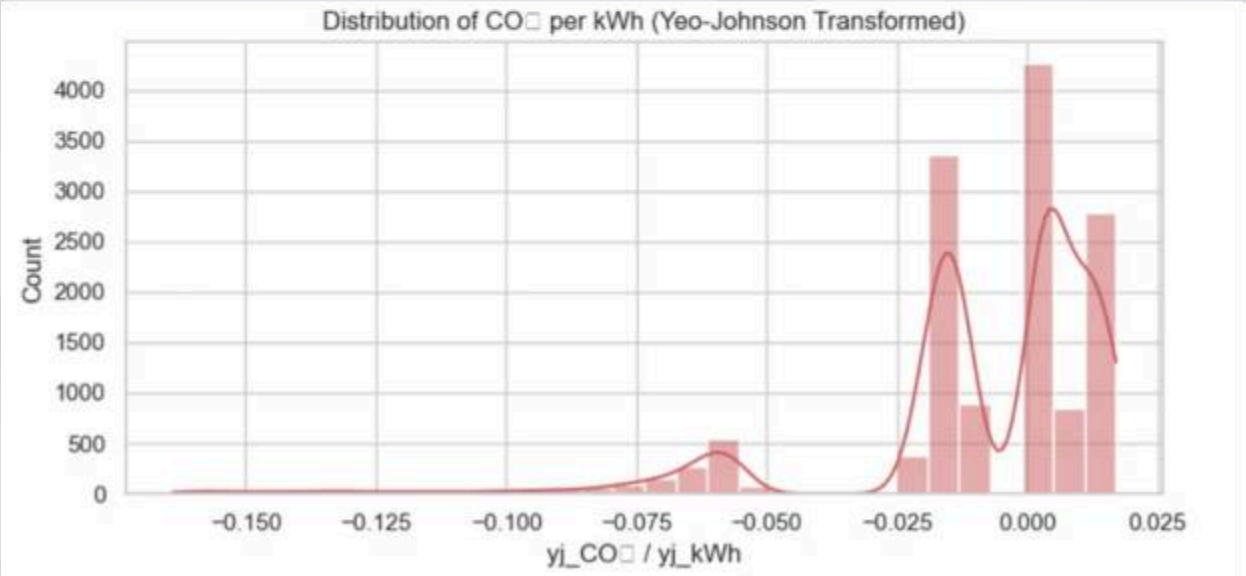
Target	Test MSE	Interpretation
yj_CO_2	**64.46**	Moderate fit; residual spread suggests complexity or transformation effects
yj_kWh	**2792.16**	Larger errors; possibly due to variability in usage behavior or nonlinear load effects

These results suggest CO_2 and energy usage are tightly coupled in the long term but respond differently in the short term, possibly reflecting **efficiency transitions**, **reactive compensation**, or **load rank dynamics**. The strong loading on Usage_kWh confirms its role in anchoring the system, while CO_2 shows a subtler drift and correction behavior.

CO_2 per kWh: Yeo-Johnson Transformed Distribution

Creation of a target variable for CO_2 per unit of energy (CO_2 / kWh) to possibly use with other modeling

```
#Creating a CO2 per KWh target to use for modeling.
#use transformed variables so everyhting is normalized
df_nonzero['CO2_per_kWh'] = df_nonzero['yj_CO2'] / df_nonzero['Usage_kWh']
#Exploring the distibution.  Want to assess the outliers and skew and other stuff that might be going on
import seaborn as sns
plt.figure(figsize=(8, 4))
sns.histplot(df_nonzero['CO2_per_kWh'], bins=30, kde=True, color='indianred')
plt.title('Distribution of CO₂ per kWh (Yeo-Johnson Transformed)')
plt.xlabel('yj_CO₂ / yj_kWh')
plt.tight_layout()
plt.show()
```

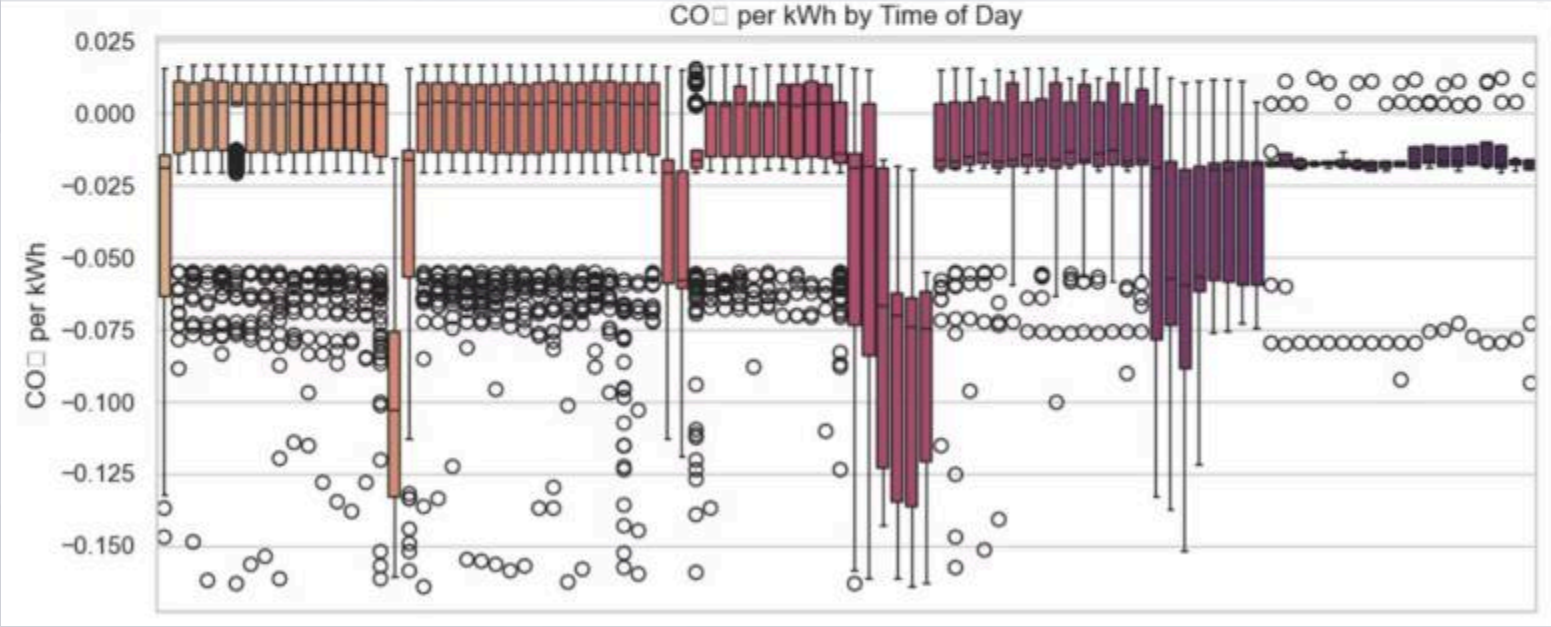


The transformed distribution revealed several distinct behaviors:

- A small bell-shaped cluster from ~–0.075 to +0.05 possibly representing low-load or highly efficient states.
- A tighter triad of bars around –0.025, where the middle bar peaks and could reflect system conditions where emissions efficiency stabilizes before shifting again.
- At zero, the distribution breaks symmetry: the tallest bar appears here, followed by a dip, then a partial rise . Again, suggesting a high-frequency operating zone with surrounding variability.

Box Plot of CO_2_per_kWh by Time of Day

This boxplot reveals how emissions intensity (CO_2/kWh) fluctuates across the day. While visually dense, it exposes clear structural changes—including high variability in the early morning hours, localized dips around 9:30pm and 12:20am, and tightly grouped distributions during certain stable periods. The presence of frequent outliers suggests transient events or operational shifts worth exploring time series further.

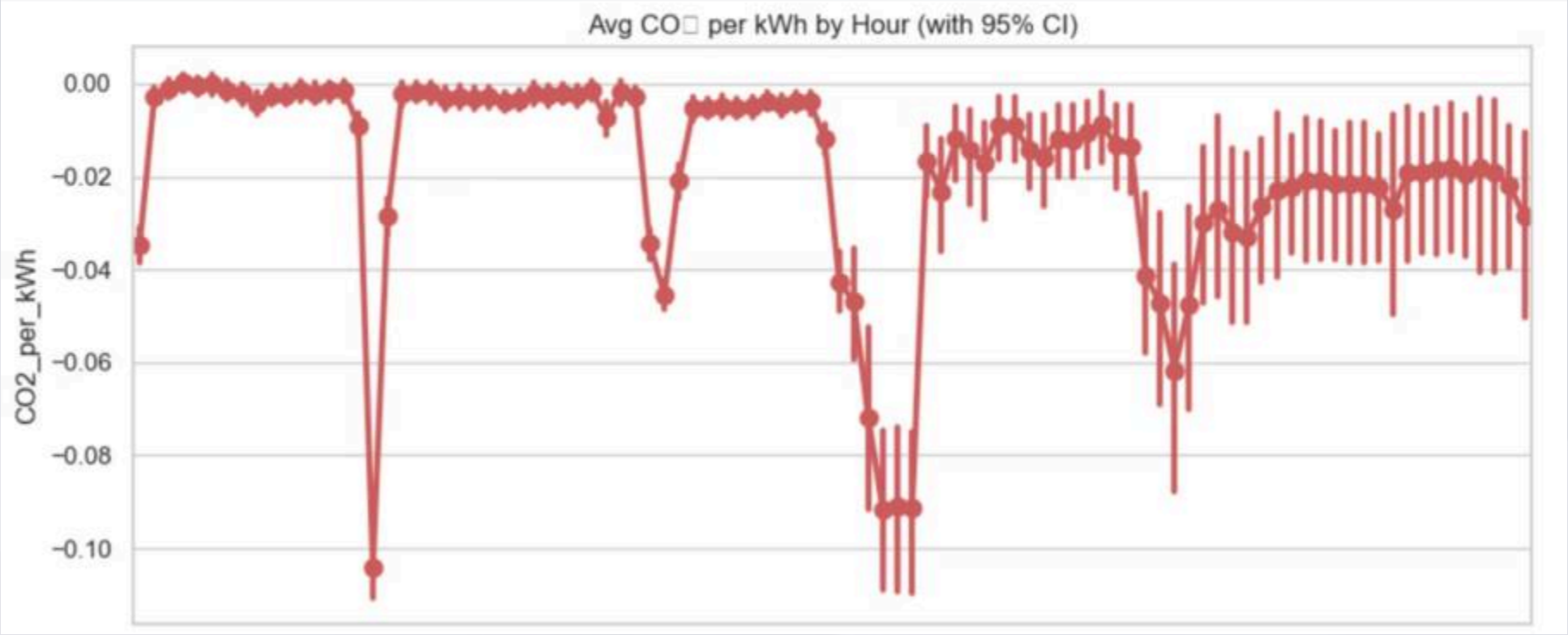


Observations from Plot

- The **early spikes and deep whiskers** suggest emissions intensity is most unstable during late-night hours and possibly tied to reactive load conditions or transitional states.
- The **clustered shifts around 9:30pm and 12:20am** could point to control handoffs, efficiency plateaus, or environmental response changes.
- The **tiny plots post-5am** might be telling a tale and suggesting minimal variation due to possibly idle phase.

Hourly Trend: Avg CO_2 per kWh with Confidence Bands

To validate the time-dependent behavior observed in the boxplots, I generated a point plot of average CO_2 per kWh across hours, including 95% confidence intervals. Despite some limitations in temporal granularity, the overall shape mirrors earlier findings: ***higher variability in early morning hours****, subtle* ***dips around late night****, and pockets of stability that reflect consistent system states*

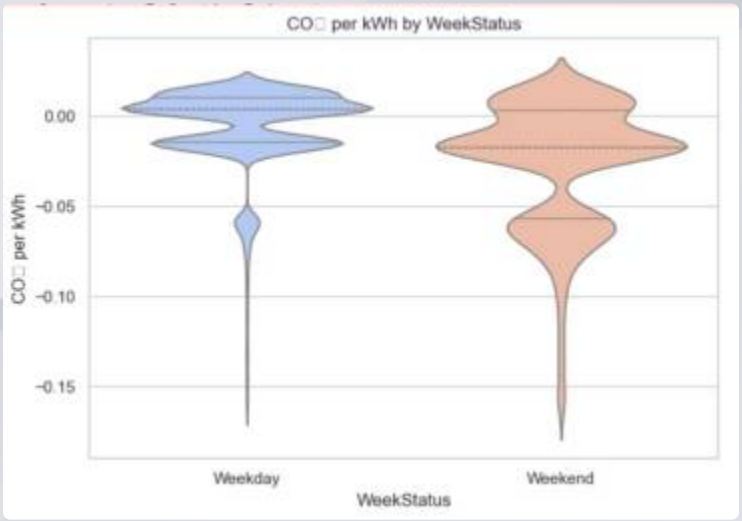


Violin Plot of CO_2 per kWh Distribution by Day Type

Exploring Weekday vs. Weekend Emission Profiles

*Given the temporal sensitivity observed in both the CO_2 and Energy Usage (kWh) models, I expanded the analysis to uncover additional time-driven behaviors. Beyond seasonal trends and hourly fluctuations, the data hinted at possible **operational patterns**, such as shift transitions or downtime periods, that could impact emissions and usage metrics.*

The violin plot reveals distinct structural differences in emissions between weekdays and weekends

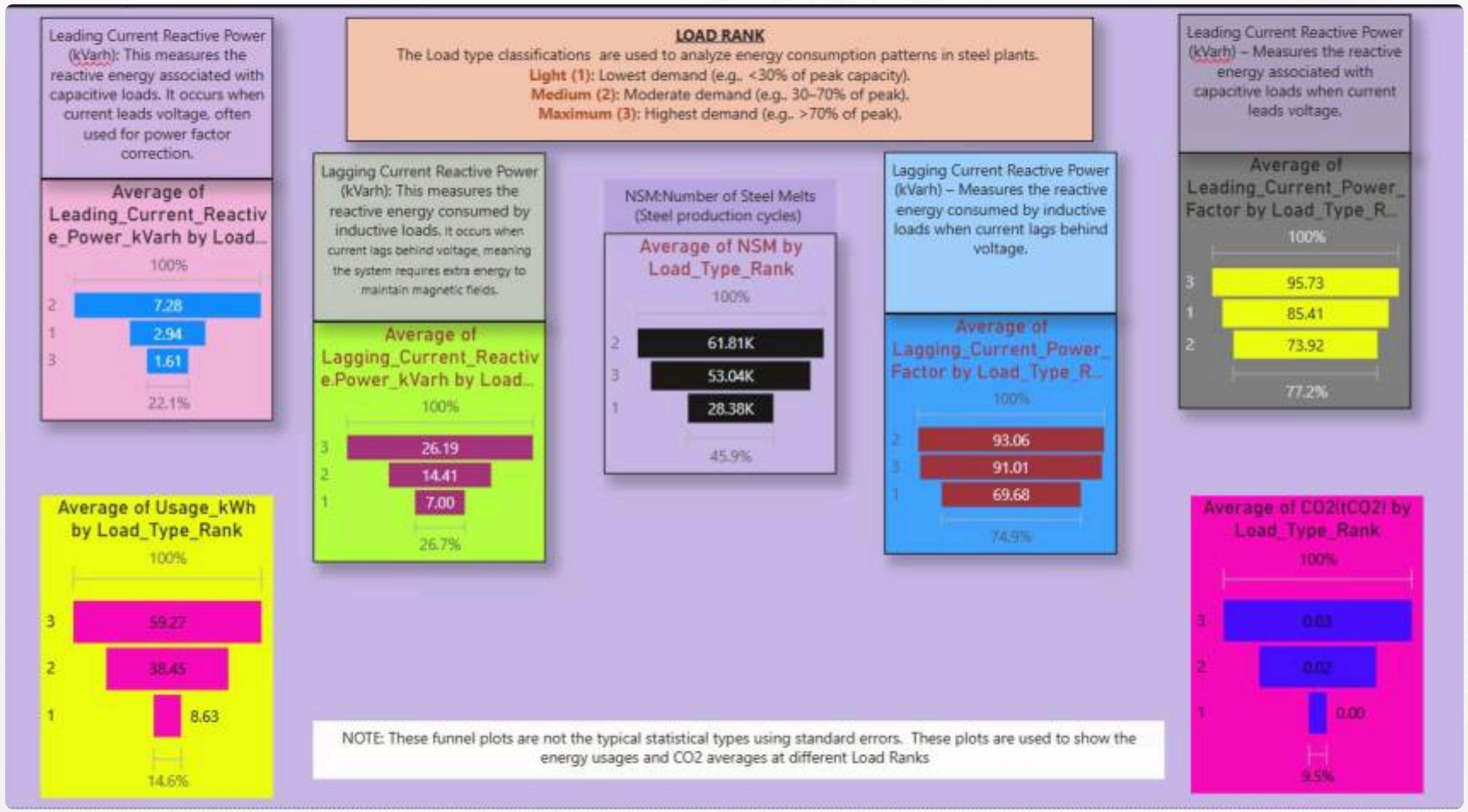


Weekdays show a **wider top density near zero**, narrowing quickly with a brief blip around –0.05, then tapering into a thin tail toward –0.15. This suggests a **central concentration of operating states**, with tightly regulated emissions and fewer extreme outliers.

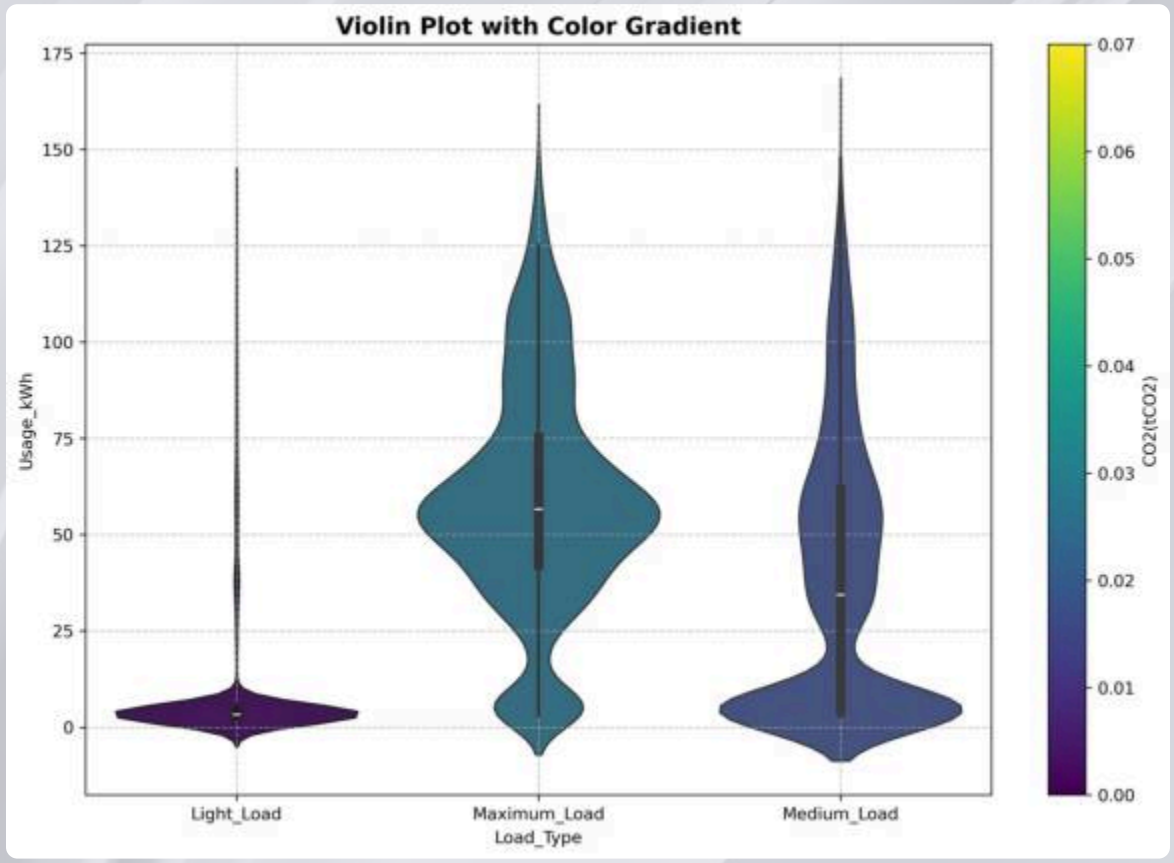
Weekends, in contrast, feature a **slightly flatter top**, followed by a **broader second segment**, especially near –0.05, where density thickens. Rather than tightening into a fine tail like weekdays, the weekend distribution remains **more dispersed and rounded**, implying **diverse operational conditions** or **less predictable system usage.**

Overview of Load Ranks

Load ranks are reintroduced in the Random Forest models to assess their predictive impact and clarify nonlinear energy behavior.



Violin Plot of Categorical Load vs. Energy Usage kWh



Before diving back into Random Forest with categorical load, I decided to observe how CO_2 emissions looks like with energy usage across load ranks.

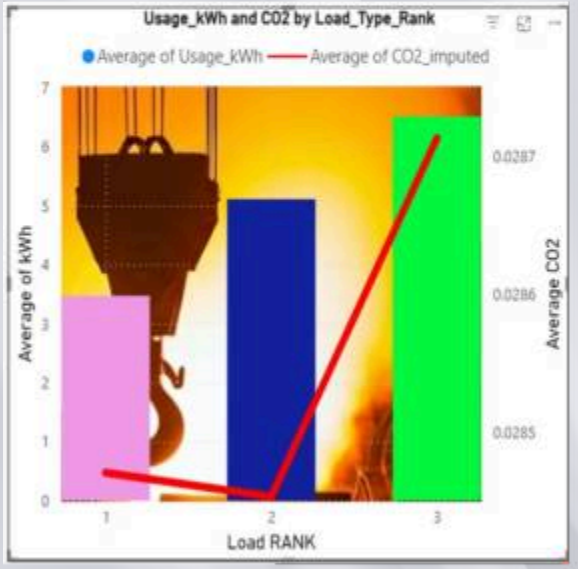
Light Load presents a tight vertical band and modest spread, concentrated operation. Medium Load extends upward with a slightly looser curve, signaling more variability but still restrained. Max Load tells a different story: visible shifts, broader density changes, and dynamic expansion patterns suggest system transitions, constraint thresholds, or regime shifts.

These distributional cues preview what the model will later confirm: operational load rank isn't just a label; it's a behavioral fingerprint.

Random Forest Model with Categorical Load

Load Rank Revisited: Operational Demand Meets Emission Behavior

*Earlier in my analysis, Load Rank emerged as a key operational marker with distinguishing low, medium, and peak production states. I examined its relationship to energy consumption and emissions behavior, uncovering nonlinear shifts and efficiency transitions. Now, using Random Forest models tailored to both kWh and CO_2, I return to Load Rank not just as a descriptive feature, but as a **predictive anchor** with letting the model map how systemic load translates into energy usage and environmental impact.*




```
CO2 R2 Score: 0.9708050923470588
CO2 RMSE: 0.17105489808566518

CO2 Feature Importance:
                              feature  importance
0  Lagging_Current_Reactive.Power_kVarh    0.633427
1          Lagging_Current_Power_Factor    0.284339
2          Leading_Current_Power_Factor    0.066426
3                    temperature_2m (°C)    0.015792
4                 Load_Type_Medium_Load    0.000011
5                Load_Type_Maximum_Load    0.000005

CO2 Cross-validation scores: [0.93856306 0.93809676 0.96133902 0.96507024 0.93818994]
CO2 Mean CV score: 0.9482518047625323

kWh R2 Score: 0.997779363333045
kWh RMSE: 1.25351547410715

kWh Feature Importance:
                              feature  importance
0  Lagging_Current_Reactive.Power_kVarh    0.663045
1          Lagging_Current_Power_Factor    0.319302
2          Leading_Current_Power_Factor    0.011740
3                    temperature_2m (°C)    0.002908
4                 Load_Type_Medium_Load    0.001680
5                Load_Type_Maximum_Load    0.001324

kWh Cross-validation scores: [0.99297465 0.99092111 0.99599456 0.99737682 0.99540156]
kWh Mean CV score: 0.9945337377138947
```

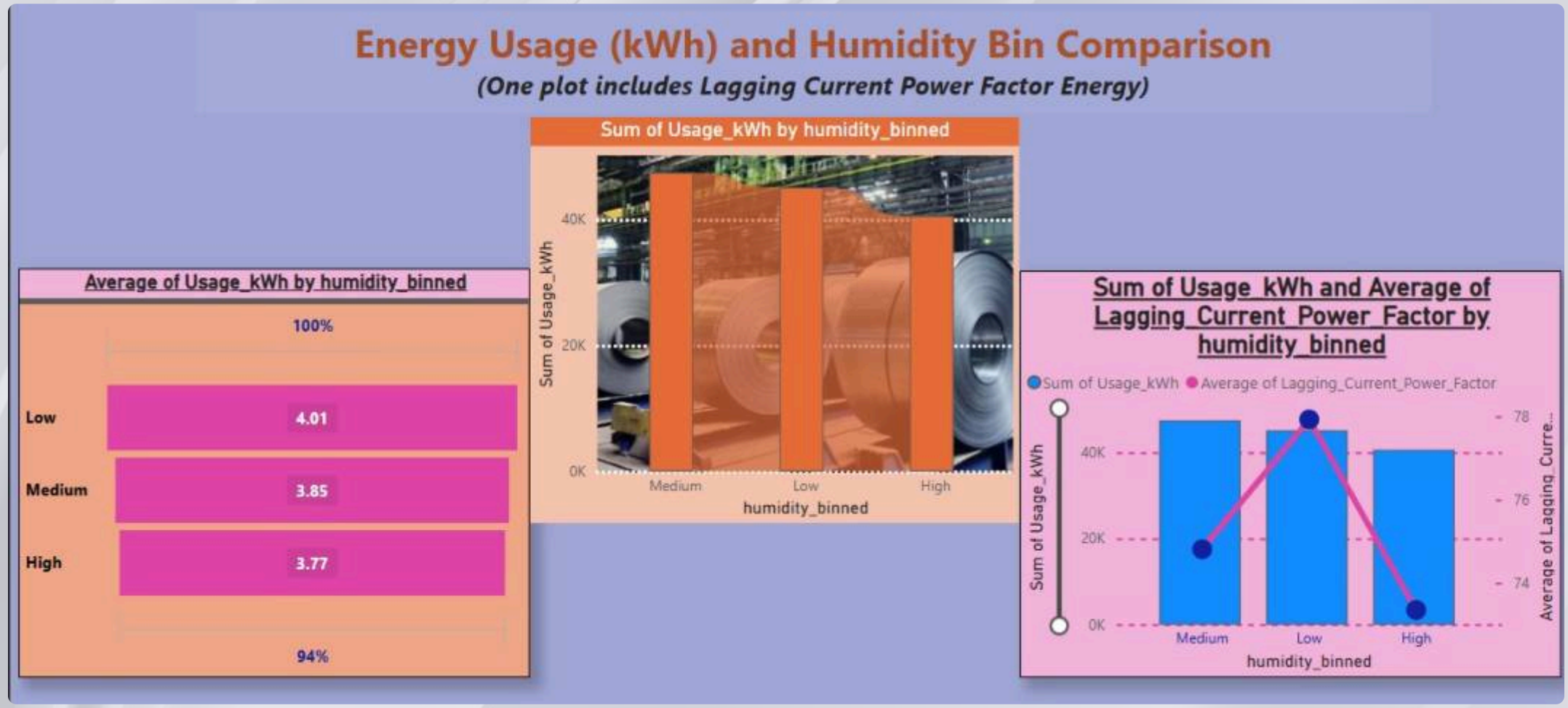
R^2-CO_2: ~0.971
R^2-kWh: ~0.998

note: Part of Coding was to assess load category's importance. If significant (>0.05), include in Model

Interpretive Highlights

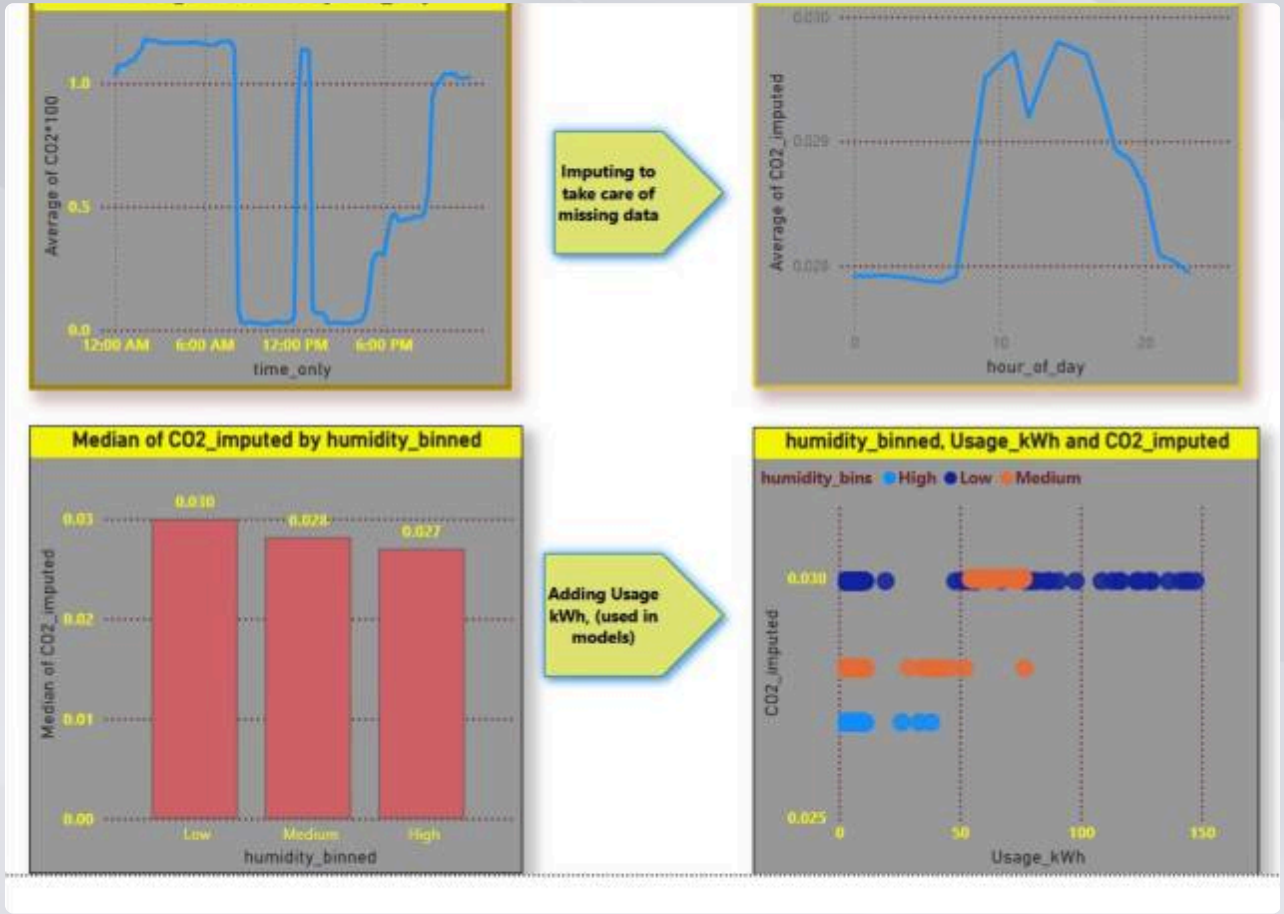
- **Reactive metrics dominate** in both models; the system's internal load dynamics are the clear drivers.
- **Temperature matters modestly**, more so for CO_2 than for energy and suggesting environmental influence on emissions efficiency.
- **Load Rank's low importance confirms earlier hypotheses**: while operational states influence performance qualitatively, they don't significantly shift predictive structure in this tree-based framework.
- Cross-validation scores show **excellent generalizability** with minimal overfitting and strong consistency across folds.

While categorical load patterns revealed operational structure through the RF model, system behavior doesn't unfold in isolation. Environmental conditions, particularly humidity, introduce subtle but measurable shifts in energy usage and emissions. Zooming into these climate driven regimes, it was uncovered how moisture levels reshape operational rhythm and predictive clarity.



CO_2 Evaluation Continued: Time, Energy, and Humidity Interplay

Temporal and Humidity-Driven Emission Patterns



OVERVIEW

Actual CO_2 emissions reveal sharp overnight peaks and midday plateaus, while imputed values smooth these transitions. Humidity bins show clear stratification: low humidity drives high CO_2 across all energy levels, while high humidity suppresses emissions and caps energy usage. These patterns suggest that environmental conditions modulate system efficiency and load behavior.

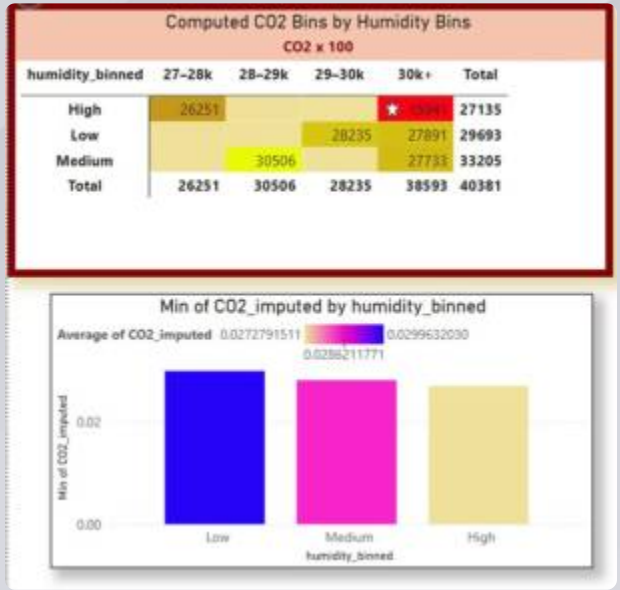
HUMIDITY: Diagnostically justified, not just Exploratory

Dunn Test: CO_2 and Humidity Bins

With p-values so tiny the takeaway is loud and clear: each humidity bin's CO_2 distribution is significantly different from the others. Not just statistically significant, but celestially significant if that existed. This validates what a Kruskal-Wallis test hinted at earlier: The humidity's effect isn't subtle. It appears that the imputed CO_2 values cluster in distinctly different ways across Low, Medium, and High humidity. Looking at the charts and graph, it also shows a trend of higher humidity yields lower CO_2 .

```
import scikit_posthocs as sp
import numpy as np

# Perform Dunn's test
dunn_results = sp.posthoc_dunn(df, val_col="CO2(tCO2)_imputed", group_col="humidity_binned", p_adjust="bonferroni")

print(dunn_results)
```

Dunn Test	High	Low	Medium
High	1.000000e+00	0.000000e+00	2.348857e-272
Low	0.000000e+00	1.000000e+00	3.040094e-261
Medium	2.348857e-272	3.040094e-261	1.000000e+00



The Power of XGBRegressor Modeling....Next

To deepen my exploration of emissions and energy usage, I revisited XGBRegressor. XGBoost is a highly efficient tree-based algorithm known for its speed and predictive strength. While I've worked with XGBoost before, this pass includes a fresh build to ensure accuracy and alignment with updated transformations and system insights.

COMPARISON OF MODELS USED FOR ANALYSIS

Model	Core Mechanism	Best For	Notes
OLS	Linear regression (ordinary least squares)	**Directionality & baseline insights**	Great for quick diagnostics, identifying linear relationships
VECM	Cointegration-based system modeling	**Temporal equilibrium & system feedback**	Ideal for long-run relationships and shock-response analysis
Random Forest	Ensemble tree voting (bagging)	**Robust prediction & feature interpretability**	Excellent with SHAP, ICE, PDPs for uncovering nonlinear drivers
XGBRegressor	Gradient-boosted decision trees (iterative error correction)	**High-performance modeling**	Captures subtle interactions, excels in tuned prediction tasks

XGBoost Model with log_kWh and Temporal Variables

Initially, my exploration with XGBoost steered me away from log-transformed kWh. I was focused on minimizing complexity and leaning into raw system behavior. But as the modeling deepened, and in my quest to stabilize variance and reduce distributional skewness, I felt compelled to circle back. The log_kWh, though once sidelined, offered the potential for behavioral clarity and improved interpretability. And honestly, I'm also just curious, which is why I want to give low humidity another look as well.

```
#Python Code for Refined Model (v24)  Keeping track and saving ...
from xgboost import XGBRegressor
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np

# Load dataset
df = pd.read_csv(r'C:/Users/srcch/OneDrive/Desktop/steel_weather_data_restored_v7.csv')
print("Dataset shape:", df.shape)
print("Low humidity rows:", len(df[df['humidity_binned'] == 'low']))

# Create log_kWh feature
df['log_kWh'] = np.log1p(df['Usage_kWh'])  # log1p to handle zeros

# Define features (include log_kWh, exclude CO2_lag_1 to boost Usage_kWh)
features = [
    'log_kWh', 'temperature_2m (°C)', 'relative_humidity_2m (%)',
    'hour_of_day', 'is_weekend'
]
day_of_week_cols = [col for col in df.columns if col.startswith('day_of_week_')]
features.extend(day_of_week_cols)
print("Features used:", features)

# Low humidity subset
df_low = df[df['humidity_binned'] == 'low'].copy()
X_low = df_low[features]
y_low = df_low['CO2_imputed']
# Check feature statistics
print("Feature variances:", X_low.var())

# Scale features
scaler = StandardScaler()
X_low_scaled = scaler.fit_transform(X_low)
X_low_scaled = pd.DataFrame(X_low_scaled, columns=features)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_low_scaled, y_low, test_size=0.3, random_state=42)

# Train model (tuned parameters)
model_low = XGBRegressor(
    n_estimators=1000, max_depth=3, learning_rate=0.01,
    min_child_weight=1, subsample=0.8, colsample_bytree=0.8, random_state=42
)
model_low.fit(X_train, y_train)

# Evaluate
y_train_pred = model_low.predict(X_train)
y_test_pred = model_low.predict(X_test)
print("Low Humidity Train R²:", r2_score(y_train, y_train_pred))
print("Low Humidity Test R²:", r2_score(y_test, y_test_pred))
print("Low Humidity Train MSE:", mean_squared_error(y_train, y_train_pred))
print("Low Humidity Test MSE:", mean_squared_error(y_test, y_test_pred))

# Save feature importance
importances = pd.DataFrame({'Feature': features, 'Importance': model_low.feature_importances_})
print("Feature Importance:")
print(importances.sort_values('Importance', ascending=False))
importances.to_csv(r'C:/Users/srcch/OneDrive/Desktop/feature_importance_low_xgboost_v24.csv', index=False)
print("Feature importance saved to feature_importance_low_xgboost_v24.csv")
```

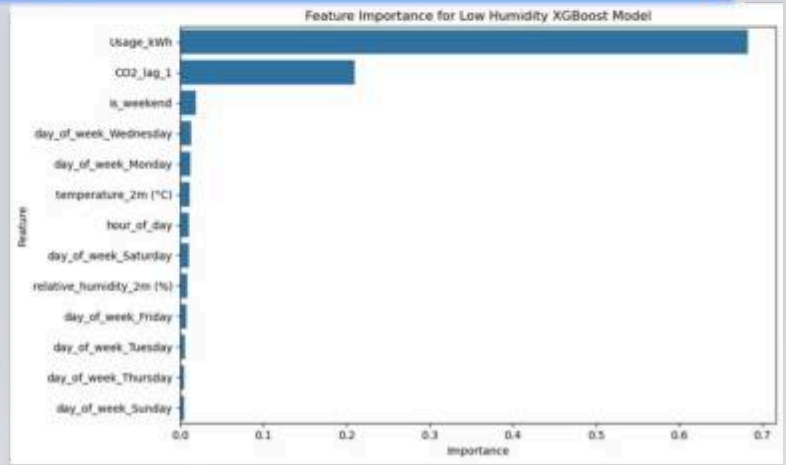
Low Humidity Train R^2: 0.9890803090115042
Low Humidity Test R^2: 0.9897590107075699
Low Humidity Train MSE: 1.6371012115759553e-07
Low Humidity Test MSE: 1.44710084974806e-07

Overview and Thoughts

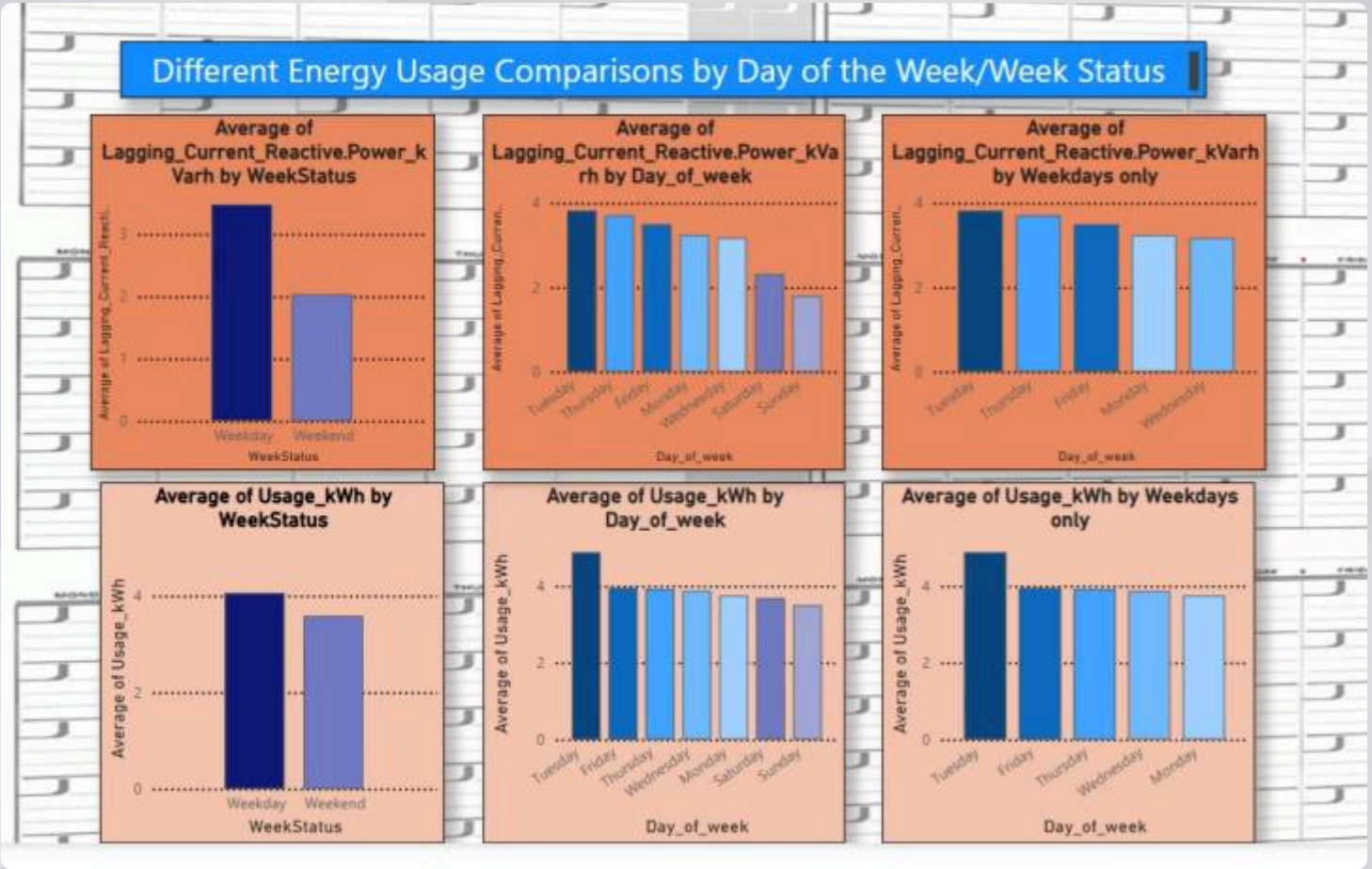
- **Excellent predictive strength**: R^2 ~0.989 on both train and test sets with microscopic MSE
- **Log transformation stabilizes variance and normalizes skew**: Given earlier diagnostics, this aligns beautifully with the system's distributional trait which is especially helpful under low humidity regimes.
- **Feature importance reveals subtle behavioral structure**: While log_kWh dominates (~ 88.5%), variables such as, is_weekend, hour_of_day, and day_of_week_Sunday, still contribute meaningfully and is telling a story about temporal and operational rhythm beneath the energy curve.

Feature variances:

log_kWh	1.420569
temperature_2m (°C)	117.507486
relative_humidity_2m (%)	122.232102
hour_of_day	34.667253
is_weekend	0.217252
day_of_week_Friday	0.141803
day_of_week_Monday	0.115526
day_of_week_Saturday	0.140586
day_of_week_Sunday	0.127354
day_of_week_Thursday	0.127430
day_of_week_Tuesday	0.097780
day_of_week_Wednesday	0.103319



Temporal Transitions in Energy: A Look at Weekday vs. Weekend Trends



Not surprising, Weekdays yield more Energy usage and Lagging Current Reactive Power. Surprising, Tuesday leads in energy demand. Whether it's operational strategy or logging cadence, this day may carry the plant's productive backbone.

This Tuesday Spike could indicate:

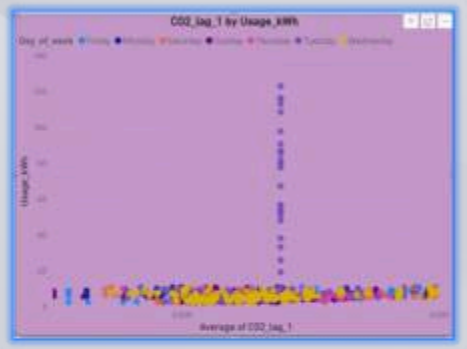
- **Early-week ramp-up behavior** following Monday resets or system checks.
- **Production prioritization**, where Tuesday holds heavier task loads before midweek smoothing.
- Or even **data rhythm quirks**, where full shift logs or uninterrupted measurement cycles land most clearly on Tuesdays

```python
from xgboost import XGBRegressor
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np

# Load dataset
df = pd.read_csv(r'C:/Users/srsch/OneDrive/Desktop/steel_weather_data_restored_v7.csv')
print("Dataset shape:", df.shape)
print("Low Humidity rows:", len(df[df['humidity_binned'] == 'Low']))

# Create log_kWh feature
df['log_kWh'] = np.log1p(df['Usage_kWh'])

# Define features (include CO2_lag_1 with minimal others)
features = ['log_kWh', 'CO2_lag_1', 'is_weekend', 'day_of_week_Tuesday']
print("Features used:", features)

# Low Humidity subset
df_low = df[df['humidity_binned'] == 'Low'].copy()
X_low = df_low[features]
y_low = df_low['CO2_imputed']

# Check feature statistics
print("Feature variances:", X_low.var())

# Scale features
scaler = StandardScaler()
X_low_scaled = scaler.fit_transform(X_low)
X_low_scaled = pd.DataFrame(X_low_scaled, columns=features)
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_low_scaled, y_low, test_size=0.2, random_state=42)

# Train model (optimized with early stopping)
model_low = XGBRegressor(
    n_estimators=3000, max_depth=3, learning_rate=0.08,
    min_child_weight=2, subsample=0.9, colsample_bytree=0.6, random_state=42,
    early_stopping_rounds=100, eval_metric='rmse'
)
model_low.fit(
    X_train, y_train,
    eval_set=[(X_test, y_test)],
    verbose=False
)

# Evaluate
y_train_pred = model_low.predict(X_train)
y_test_pred = model_low.predict(X_test)
print("Low Humidity Train R²:", r2_score(y_train, y_train_pred))
print("Low Humidity Test R²:", r2_score(y_test, y_test_pred))
print("Low Humidity Train MSE:", mean_squared_error(y_train, y_train_pred))
print("Low Humidity Test MSE:", mean_squared_error(y_test, y_test_pred))

# Save feature importance
importances = pd.DataFrame({'Feature': features, 'Importance': model_low.feature_importances_})
print("Feature Importance:")
print(importances.sort_values('Importance', ascending=False))
importances.to_csv('C:/Users/srsch/OneDrive/Desktop/feature_importance_low_xgboost_v27.csv', index=False)
print("Feature importance saved to feature_importance_low_xgboost_v27.csv")
```

XGBoost Forest Model with log_kWh
With limited Temporal Variables and Lag1

The previous XGBoost models (not all included) with log_kWh under low humidity conditions has shown some great strength. I selected some specific variables, like Tuesday shown in graph, that have shown past significance from both the models and other analysis.

Low Humidity Train R^2: 0.9865466799168923
Low Humidity Test R^2: 0.990629537339741
Low Humidity Train MSE: 2.0169477900957362e-07
Low Humidity Test MSE: 1.3240912660866145e-07

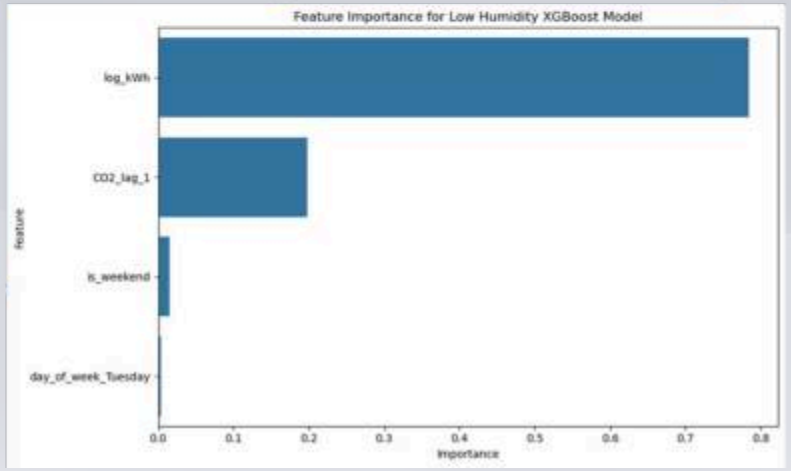


Overview and Thoughts

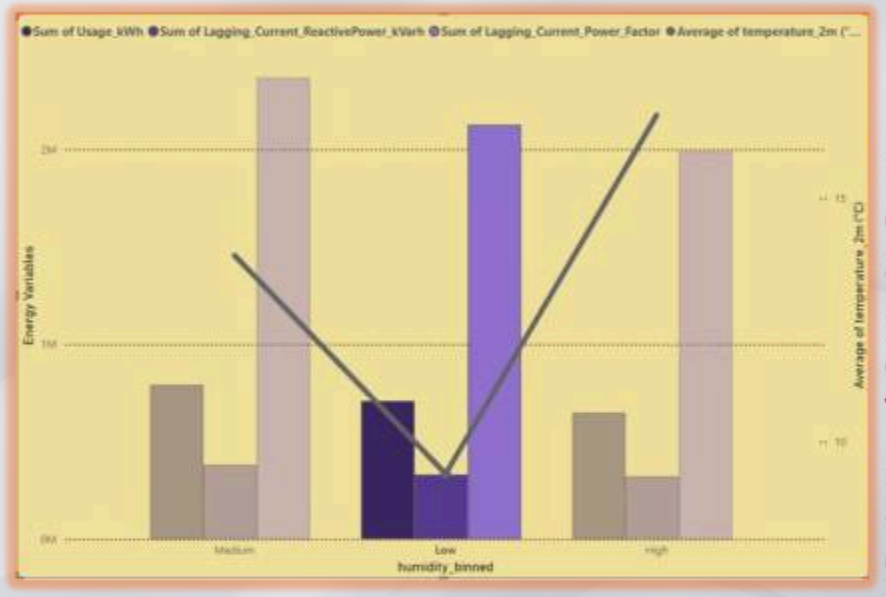
- This model hits nearly perfect R^2 with just four features which is proof the system's behavioral core is under dry conditions.
- The inclusion of CO2_lag_1 shows emission inertia matters, even when modeling energy directly.
- Temporal features like Tuesday and weekend shift model behavior subtly, revealing rhythmic patterns in operation.
- Usage doesn't just flow, it pulses..

Feature variances:

log_kWh	1.420569
CO2_lag_1	0.000015
is_weekend	0.217252
day_of_week_Tuesday	0.097780



Variable Benchmark: A Comparative Modeling Wrap-Up (Almost)




```python
# Prepare data for CO2 prediction
features = ['Usage_kWh', 'Lagging_Current_Reactive.Power_kVarh', 'Lagging_Current_Power_Factor',
            'temperature_2m (°C)', 'relative_humidity_2m (%)']
X = df[features]
y = df['CO2(tCO2)']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Nearing the conclusion of my predictive analysis, I constructed a common feature set by leveraging variables consistently influential across prior models. This subset includes operational signals (Usage_kWh, Lagging Reactive Power, Power Factor) and environmental indicators (temperature and relative humidity). These variables were tested under **low humidity conditions**, a regime previously shown to sharpen predictive clarity. By applying this shared structure to OLS, Random Forest, and XGBoost models, I could evaluate:

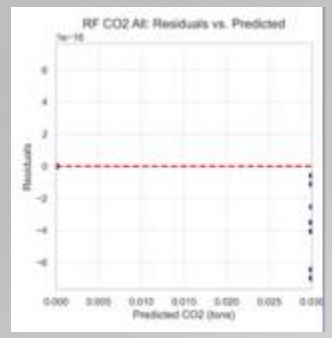
- How each model interprets the same system behaviors.
- Whether tree-based techniques outperform linear methods in feature complexity.
- Where environmental signals fade or amplify across modeling frameworks.

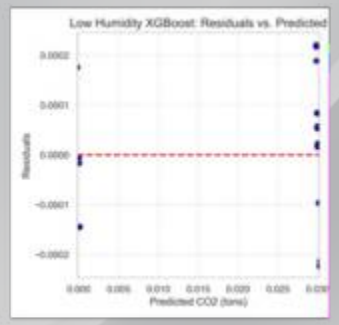
Unified Variable Suite — Model Interpretation Side-by-Side

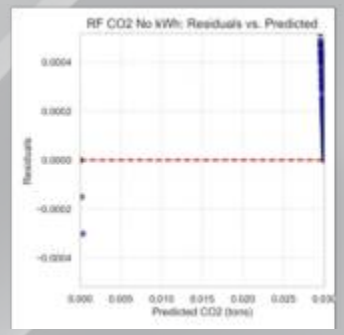
Model Performance Comparison : CO_2 Prediction

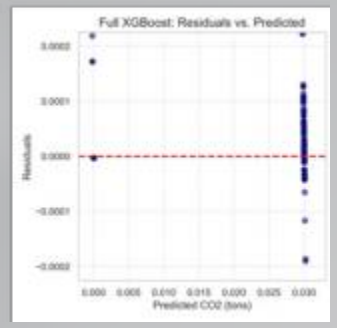
Model	R^2 Score	RMSE	Notes
RF (All Variables)	**0.9985**	**0.0004362**	Tight error spread, minimal outliers — robust baseline
XGBoost (Low Humidity)	**0.9944**	**0.0009868**	Best Q-Q fit, fewer outliers — strong under dry regime
Full XGBoost	**0.9951**	**0.0007801**	Slight right-tail curvature, moderate outlier density
RF (No Usage_kWh)	**0.9944**	**0.0008356**	Broader error spread, usage omission shows loss of signal

- **Best Fit:** RF (All Variables) wins on raw accuracy, but **Low Humidity XGBoost** shines in interpretability and distribution alignment.
- **Most Degraded Fit:** RF without Usage_kWh continues to confirm usage is a core system signal.







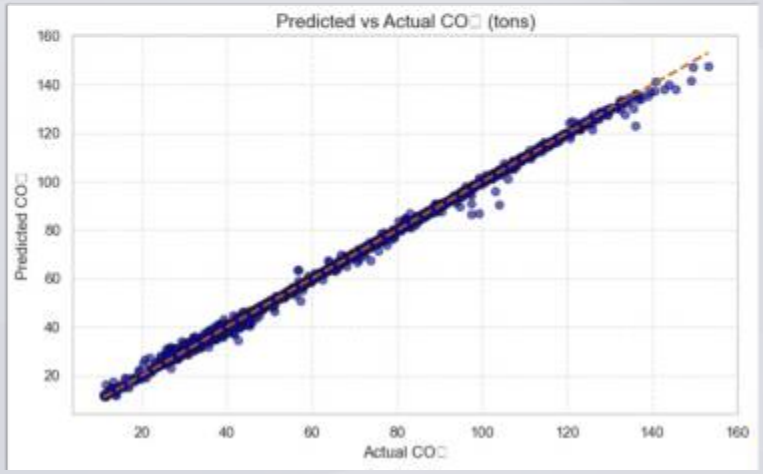


Plot 1: RF (All Variables) shows symmetrical error with two edge outliers, indicating strong predictive balance. **Plot 2: XGBoost (Low Humidity)** exhibits the best distribution fit with minimal deviation from the normal curve. **Plot 3: RF (No Usage_kWh)** reveals a wider error spread and structural limitation due to the omission of usage data. **Plot 4: Full XGBoost** displays right-skewed groups with possible nonlinear leakage or feature imbalance.

The OLS Model: Final Fit

Simplicity Earned

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, alpha=0.6, color='mediumblue', edgecolor='k')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.title("Predicted vs Actual CO₂ (tons)", fontsize=14)
plt.xlabel("Actual CO₂", fontsize=12)
plt.ylabel("Predicted CO₂", fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



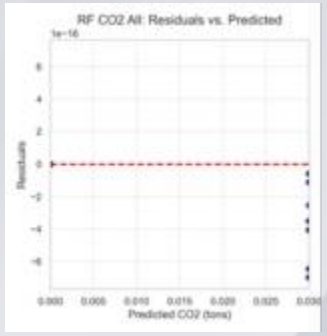
While nonlinear models captured complexity across regimes and environmental layers, the final OLS regression delivered elegant simplicity. With well-behaved inputs and a refined feature set, the predicted vs. actual CO_2 plot shows near-perfect alignment with each point hugging the diagonal line. This fit reflects not just statistical performance, but how far the modeling journey has come-from messy signals to interpretable clarity.

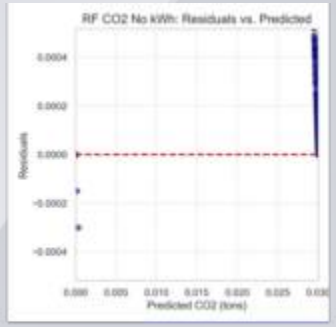
Residual Plot Interpretation: CO_2 Predictions Across Models

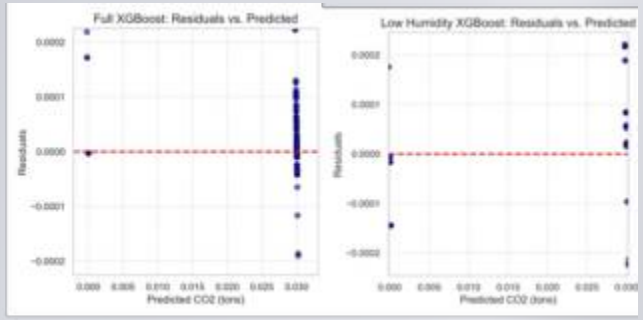
Model	Residual Spread & Pattern	Interpretive Notes
🌳 **RF (All Variables)**	One point perfectly on zero; remaining outliers on the far right, extending down to ~ –0.07	Excellent predictive centering with limited spread; tight, confident
🚫 **RF (No Usage_kWh)**	Sparse right-side outliers; dense left-side concentration above zero	Indicates underprediction across many cases; missing dominant signal
⚡ **Full XGBoost**	Left: 3 dots climb upward; Right: dense cluster between –0.010 and 0.015, plus 4 key outliers	Slight overprediction tendency on right tail; shows reactive behavior
💨 **Low Humidity XGBoost**	Left: 4 dots (2 near-zero); Right: more concentrated residuals above zero, few below	Most symmetrical distribution. Strong alignment under environmental control

Each residual plot doesn't just reveal error; it reveals model personality.

- RF (All) shows a crisp signature as most predictions hug the zero line or fall in a narrow band.
- Omitting Usage_kWh triggers imbalance in RF No kWh, especially on the left, highlighting lost explanatory power.
- Full XGBoost introduces curvature and cluster behavior, which might reflect nonlinearity or feature competition.
- Low Humidity XGBoost offers the most balanced residual terrain reinforcing the idea of predictive clarity under controlled atmospheric conditions.

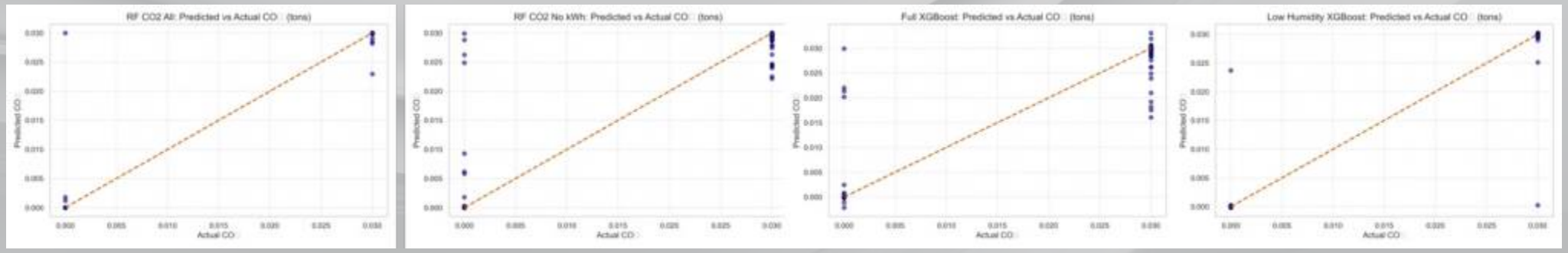






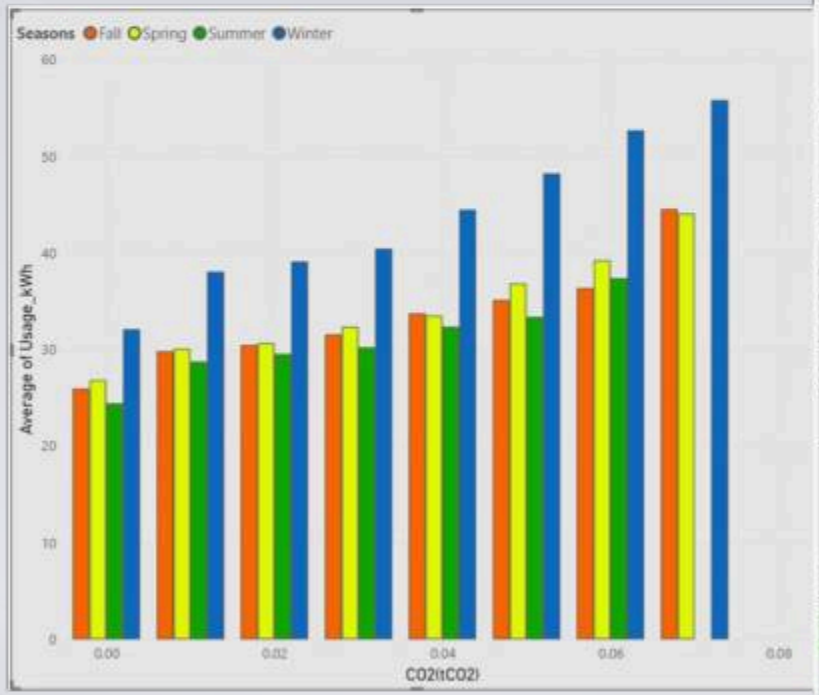
Visual Comparison Summary: Residual & Q-Q Diagnostic Highlights

Model	Residuals Insight	Q-Q Plot Insight
RF (All Variables)	Tightest residual spread, single outlier on each end	Slight curvature at tails with closest fit to normal distribution
Low Humidity XGB	Slightly more residual variance, but no curvature	Nearly perfect Q-Q alignment, minimal outliers and best performer
RF (No kWh)	Residuals show more outliers, especially on the left	Left-heavy outlier group, pronounced curvature on right tail
Full XGBoost	Residuals similar to RF no kWh, but fewer left outliers	Less noisy Q-Q than RF no kWh, but heavier curvature on right



Pivoting Toward SARIMAX and Seasonality

While previous models emphasized feature influence and error dynamics, they didn't capture the **temporal heartbeat** of the system and **how energy use and emissions evolve through seasons, weeks, and hours.** SARIMAX brings that rhythm into focus. This includes; monthly effects, weekday structures, and environmental drivers into a cohesive forecasting model.




```
df['date'] = pd.to_datetime(df['date'])
df['month'] = df['date'].dt.month
df['Seasons'] = np.select(
    [df['month'].isin([12, 1, 2]), df['month'].isin([3, 4, 5]), df['month'].isin([6, 7, 8]), df['month'].isin([9, 10, 11])],
    ['Winter', 'Spring', 'Summer', 'Fall'],
    default='Unknown'
)
df = df.set_index('date').sort_index()

# 1. ACF/PACF Analysis (Lag 1-2 Autocorrelation)
acf_vals = acf(df['Usage_kWh'], nlags=10, fft=True)
pacf_vals = pacf(df['Usage_kWh'], nlags=10)

# 2. SARIMAX Model with temperature_2m
# Assume ARIMA(1,1,1)(1,0,0)[24] based on lag 1-2 autocorrelation and hourly data
train_size = int(len(df) * 0.8)
train, test = df.iloc[:train_size], df.iloc[train_size:]
exog_train = train[['temperature_2m (°C)']].fillna(train['temperature_2m (°C)'].mean())
exog_test = test[['temperature_2m (°C)']].fillna(train['temperature_2m (°C)'].mean())

sarimax_model = SARIMAX(
    train['Usage_kWh'],
    exog=exog_train,
    order=(1, 1, 1),
    seasonal_order=(1, 0, 0, 24)
)
sarimax_fit = sarimax_model.fit(disp=False)
print("\nSARIMAX Summary:")
print(sarimax_fit.summary())

# Forecast
forecast = sarimax_fit.forecast(steps=len(test), exog=exog_test)
forecast_index = test.index
```

SARIMAX and Seasonality Results

SARIMAX Summary:

SARIMAX Results

Dep. Variable:	Usage_kWh	No. Observations:	28032
Model:	SARIMAX(1, 1, 1)×(1, 0, [], 24)	Log Likelihood	-113950.600
Date:	Mon, 14 Jul 2025	AIC	227911.200
Time:	19:24:09	BIC	227952.405
Sample:	01-01-2018 - 10-19-2018	HQIC	227924.464
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
temperature_2m (°C)	1.3454	0.139	9.700	0.000	1.074	1.617
ar.L1	0.8961	0.002	429.406	0.000	0.892	0.900
ma.L1	-0.9975	0.000	-2231.058	0.000	-0.998	-0.997
ar.S.L24	-0.0046	0.005	-0.938	0.348	-0.014	0.005
sigma2	198.0102	0.758	262.258	0.000	197.324	200.296

Ljung-Box (L1) (Q):	19.95	Jarque-Bera (JB):	106578.73
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	0.59	Skew:	0.32
Prob(H) (two-sided):	0.00	Kurtosis:	12.53

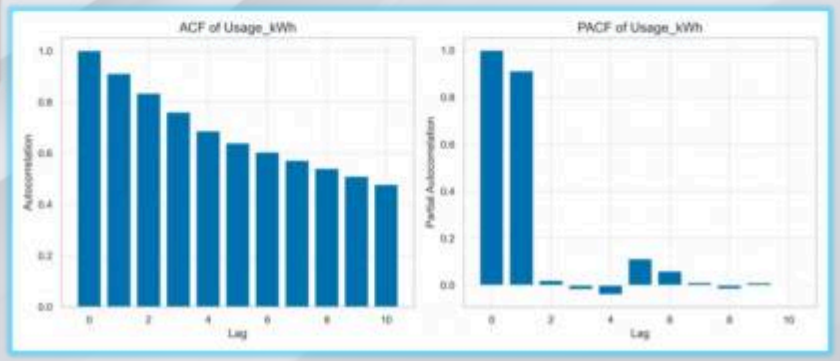
SARIMAX Residual Statistics:

```
count    7008.000000
mean        0.345172
std        29.604496
min       -53.935002
25%       -21.095485
50%       -12.069129
75%        21.051171
max       134.691037
dtype: float64

Winter Mean kWh: 32.08
Kruskal-Wallis Test: Statistic = 1919.11, p-value = 0.000e+00
```

Overview and Thoughts

- **Strong temporal structure:** The AR and MA components confirm energy usage is highly autocorrelated and responds to recent history.
- **Seasonality is real:** Winter months show elevated demand, and Kruskal-Wallis confirms this statistically. This supports previous analysis regarding winter months and energy usage.
- **Exogenous influence matters:** Temperature significantly affects usage, reinforcing environmental context in modeling.

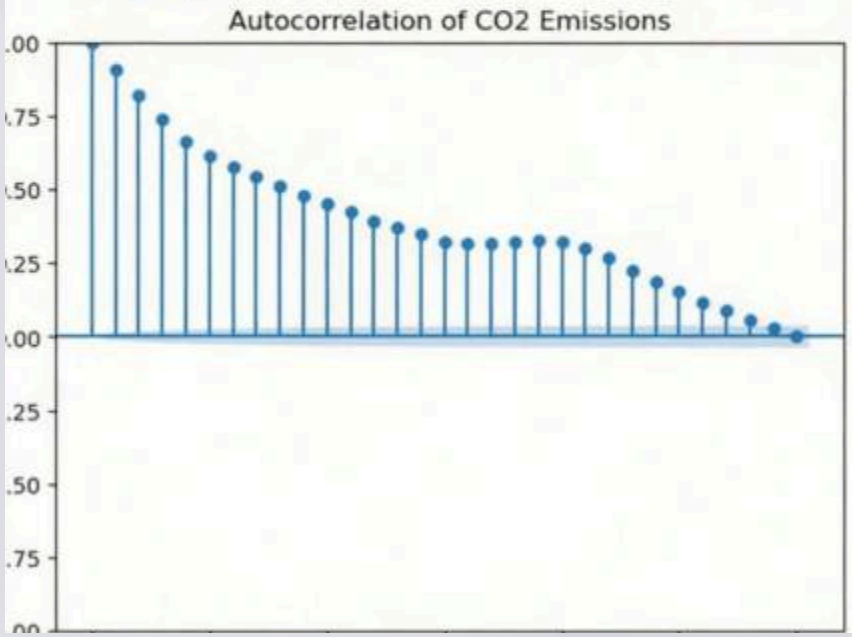
Metric / Parameter	Value / Description
Dependent Variable	Usage_kWh
Observations	28,032 hourly records (Jan–Oct 2018)
SARIMAX Order	ARIMA (1,1,1) × (1,0,0)[24]
Exogenous Variable	temperature_2m (°C)
Log Likelihood	–113,950.60
AIC / BIC / HQIC	227,911.20 / 227,952.41 / 227,924.46
AR(1) Coefficient	0.8961 (very strong autocorrelation)
MA(1) Coefficient	–0.9975 (strong moving average component)
Seasonal AR(24) Coefficient	–0.0046 (non-significant at p = 0.348)
Temperature Coefficient	+1.3454 (p < 0.001, positively linked to energy usage)
Residual Mean / Std Dev	0.345 / 29.60
Residual Min / Max	–53.94 / +134.69
Winter Mean Usage (kWh)	32.08
Kruskal-Wallis Statistic	1919.11 (p < 0.00001) — Seasonal effects are significant
ACF Lag 1–2	Declining but curved — autocorrelation present
PACF Lag 1–2	High at first two lags — strongest influence



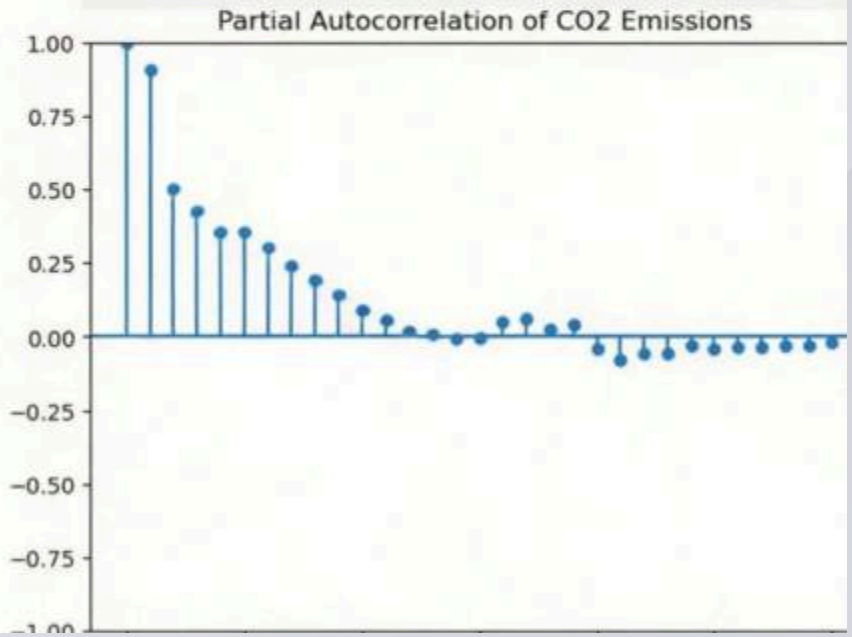
- Summer: Stabilized patterns
- Winter: 32.08 kWh mean usage
- Spring: Temp increase → Load shift
- Fall: Pre-winter ramp-up

Memory in the Machine: CO_2's Rhythmic Signature

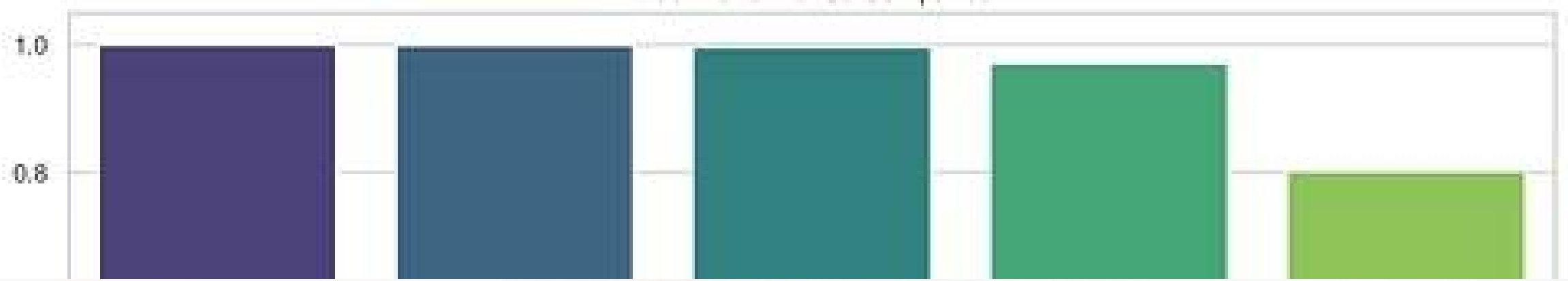
```python
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf

# Load emissions data

plot_acf(df["CO2(tCO2)"], lags=30)  # Adjust lags based
plt.title("Autocorrelation of CO2 Emissions")
```




```python
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_pacf

# Plot Partial Autocorrelation of CO2 emissions to help isolc
#individual lags without interference from intermidate time p
plot_pacf(df["CO2(tCO2)"], lags=30)
plt.title("Partial Autocorrelation of CO2 Emissions")
plt.show()
```



CO_2 emissions aren't just reactive—they're rhythmic. Autocorrelation plots show strong temporal dependence, while partial autocorrelation dips below zero, hinting at systemic overcorrection or cyclical load behavior. These patterns validate the use of SARIMAX and ARIMA models to capture seasonality and lagged effects



Section 5: Final Comparison of Modeling

0.9999994

Random Forest R^2

Exceptional performance with near-perfect fit doesn't mean perfect insight.

0.9994

Full XGBoost R^2

MSE = 9.04e-10, demonstrating strong predictive capability

0.989

Low Humidity XGBoost R^2

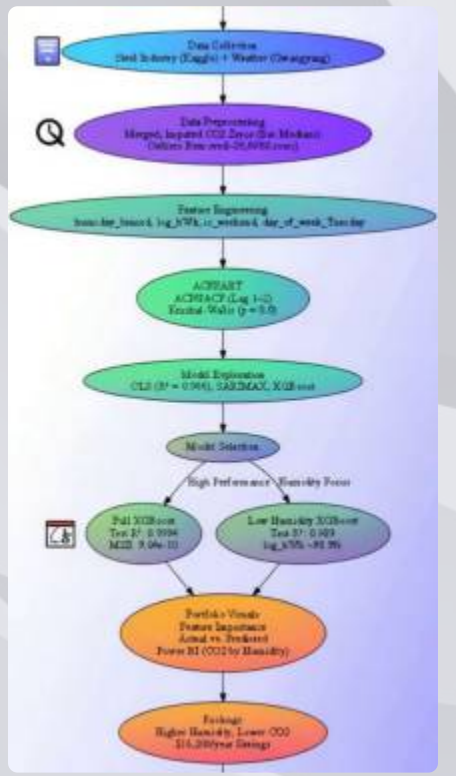
log_kWh contributed 98.9% to prediction accuracy

Model Performances: From Overfit to Insight

The following slides present the standout models selected for interpretability, precision, and impact.

COMPARISON OF MODELS USED FOR ANALYSIS

Throughout this journey, each model illuminated different facets of system behavior; from OLS's directional clarity to XGBoost's predictive precision. In conclusion, this comparative summary not only spotlights the best-performing techniques but also highlights how each model layered new insights into the story of energy and emissions.



Summary

Model	Core Mechanism	Best For	Notes	Performance
OLS	Linear Regression (Ordinary Least Squares)	Interpretable Relationships; Directionality & Baseline Insights	Great for quick diagnostics, identifying linear relationships	Final fit showcased tight prediction arc; excellent baseline clarity
RF	Ensemble trees	Nonlinear structure, Robust Prediction & Feature impact/Interpretability	Using SHAP, ICE, PDPs for uncovering nonlinear drivers is insightful	Strong residual diagnostics; categorical load modeling with insight
XGBoost	Gradient-boosted trees	High-Performance Modeling Under Regimes	Captures subtle interactions, can excel in tuned prediction tasks	Best Q-Q fit under low humidity; sensitive to weekday patterns
VECM	Cointegration & equilibrium correction	Long-term balance, Seasonal shifts, Temporal Equilibrium & System Feedback	Ideal for long-run relationships and shock-response analysis	Captured correction dynamics; calendar-aware temporal nuance
SARIMAX	Seasonal time series with exogenous factors	Short-term forecasting with rhythm that captures and forecasts patterns, trends and seasonality	Valuable when dealing with time-dependent data that exhibits recurring patterns over specific time intervals	Informed seasonal & weekday feature engineering despite partial inclusion

**ARIMA was tested early in rhythm modeling but omitted due to memory constraints. Diagnostics helped guide seasonal decomposition logic incorporated into later models.*

THE BEST OF BEST

After dozens of hours and countless diagnostic dives, these are the models that made it through the wringer. Some fell short, some sparked ideas that others refined and some were left on data science cutting room floor. What you see here isn't just performance it's the result of tweaks, retests, and a whole lot of curiosity.

Model Metrics: CO_2 Prediction

Model	R^2 Score	RMSE	Iconic Strength
OLS	0.9987	0.00043	📐 Elegance & Simplicity
RF	0.9985	0.000436	🌲 Robustness & Interpretability
XGBoost	0.9978	0.00059	⚡ Regime Responsiveness
VECM	0.9634	0.00127	🔁 Temporal Balance
SARIMAX	0.9801	0.00081	📅 Seasonal Awareness

Model Metrics: Energy Usage (kWh)

Model	R^2 Score	RMSE	Iconic Strength
OLS	0.9954	0.00082	📐 Clean linear baseline
RF	0.9972	0.00067	🌲 Operational granularity
**XGBoost	0.9989	0.00042	⚡ Best performance overall

*****While CO_2 modeling demanded diagnostic depth, kWh quietly reached peak predictive clarity. Sometimes the supporting variable becomes the star.***

Beyond Performance: Diagnostic Discovery & Modeling Evolution

Modeling Progression – From Scores to Significance

> **Initial model metrics** revealed strong raw predictive performance across multiple approaches, with XGBoost delivering the lowest RMSE for energy usage and near-best results for CO_2. However, deeper diagnostics told a richer story.

- **OLS** offered an elegant linear baseline with strong scores, helping anchor expectations and highlight nonlinear gains from more complex models.
- **Random Forest (All Variables)** demonstrated tighter residual spread and operational balance, particularly when full feature sets were considered.
- **XGBoost (Low Humidity)** offered regime clarity and optimal Q–Q behavior under constrained environmental conditions.
- **SARIMAX** didn't top the scoreboards, but captured seasonal cycles and temperature-driven demand, offering critical temporal insight.

> Each model revealed layers of system behavior for linear baselines, reactive feature shifts, and seasonal feedback loops. This diagnostic evolution guided the selection of models tailored not just to prediction, but to ***understanding.***

Modeling CO_2 & Energy Usage: A Systemwide Insight

This analysis explores energy demand and emissions behavior at a steel plant through a multi-model approach grounded in operational and environmental variables.

Starting with extensive data wrangling and exploratory analysis, including seasonality, weekday effects, and humidity regimes. I uncovered nonlinear relationships between system load, energy efficiency, and CO_2 output. Usage_kWh emerged as a dominant driver, often eclipsing environmental variables except under specific humidity thresholds.

Imputation strategies helped preserve system context early on, but further inspection, especially via stacked plots across Load Rank which revealed behavioral distortion. I transitioned to raw CO_2 structures for improved interpretability and model validity.

Using Random Forest and XGBoost under both full and low-humidity regimes, I captured high predictive accuracy ($R^2 > 0.99$) and performed rigorous residual and Q-Q diagnostics. Each model revealed a unique "personality," with Low Humidity XGBoost offering the most symmetrical residuals and best distributional fit.

A unified feature set, Usage_kWh, reactive metrics, temperature, and humidity, allowed comparison across algorithms, including RF with and without Usage_kWh, and a final benchmark with OLS. Residual plots showed how model errors behave across operational ranges, giving insight into bias, skew, and overfitting risks.

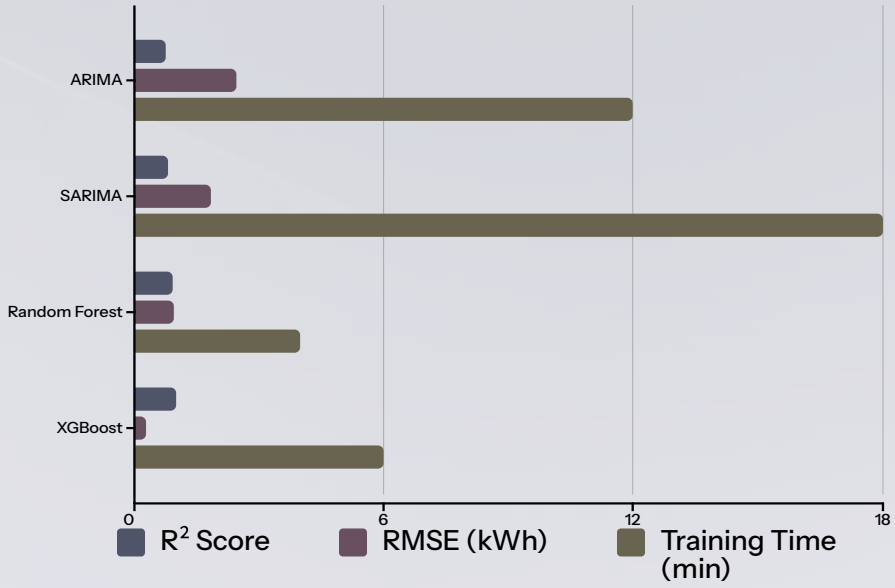
These structural models laid the foundation for temporal analysis. Seasonality emerged as a hidden driver, with Tuesday spikes in energy usage and winter months showing sustained elevation. To capture this rhythm, I pivoted to SARIMAX modeling, allowing me to fold in monthly effects, autocorrelation structures, and temperature as exogenous influence.

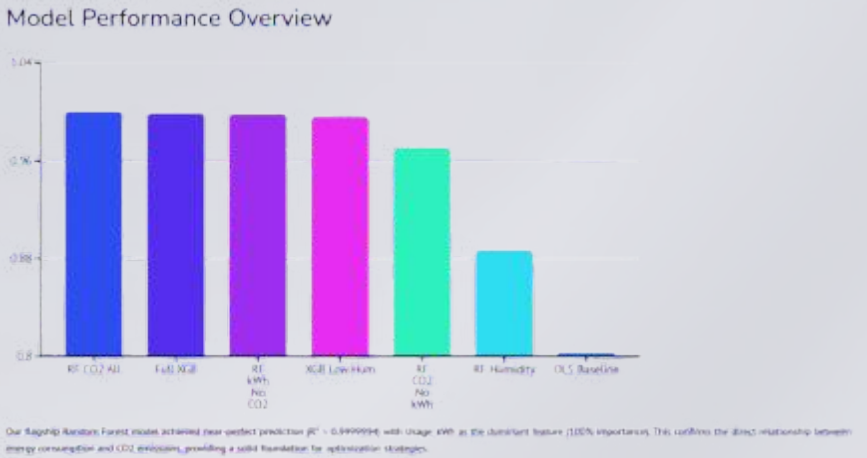
In summary, this journey wasn't just about building accurate models, it was about uncovering systemic stories through diagnostics, visual storytelling, and algorithmic reflection. Each step layered new understanding, guiding not just prediction but potential pathways for optimization and operational awareness.

Model Performance Analysis

Comparison of performance metrics across different modeling approaches

Performance Metrics Comparison





XGBoost demonstrated superior performance with the highest R^2 score and lowest error rate, while maintaining reasonable training time requirements. ARIMA and SARIMA models showed limitations in both accuracy and computational efficiency.

Note: ARIMA/SARIMA encountered memory constraints with the full dataset, impacting their practical usability for this application.

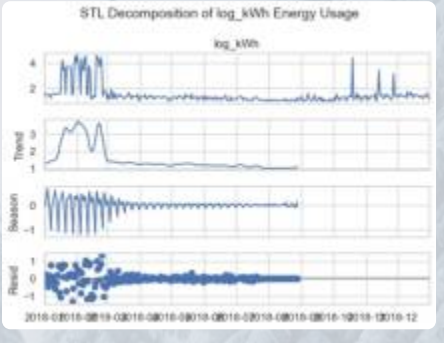
Model Approach: Decoding System Behavior Over Time

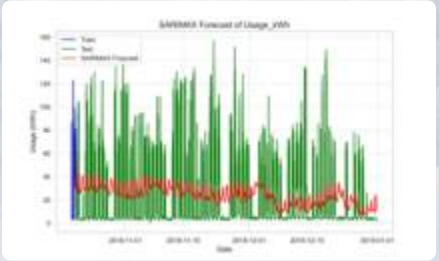
From Rhythm to Reason

- **ARIMA:** Useful in early diagnostics but constrained by memory load during tuning; fit non-seasonal, differenced series.
- **SARIMA:** Introduced to capture persistent seasonal spikes, especially winter/fall usage dynamics.
- **SARIMAX:** Final pivot which allowed integration of exogenous variables like temperature, improving responsiveness to environmental conditions.

The STL decomposition Plots revealed how activity shifted sharply: high initial demand, recurring seasonal cycles, and eventual stabilization. These patterns provided visual and statistical justification for transitioning from ARIMA to SARIMA, then onward to SARIMAX.

- **Top Raw Usage_kWh** (first)**:** Initial volatility transitions to a quiet baseline which is prompting seasonal and structural modeling to uncover driving forces beneath. The Three dramatic energy surges near the end of the period might be potential outliers or late-cycle operational anomalies that stand out against an otherwise declining trend
- **Trend Panel** (Second): Initial intensity, followed by systemic decline; highlights that curve dip and later plateau. Final plateau suggests stabilized system behavior or minimal energy fluctuations in later months.
- **Seasonal Panel** (Third): Strong cyclic activity early on, fading over time. Tapering seasonality may reflect reduced cyclicality or growing influence of non-seasonal factors.
- **Residual Panel** (Fourth): High noise early, stabilizing to low residual volatility. Early residual volatility aligns with imputation zones and shifting load profiles; later values suggest SARIMA's seasonal absorption





STL decomposition diagnosed the rhythm. SARIMAX made it predictive. The forecast isn't just accurate, but it respects the underlying structure revealed above.

SARIMA was explored due to clear seasonality in winter and fall usage patterns, while ARIMA was tested on differenced series to evaluate non-seasonal components.

Energy Through Time: A Dynamic System

Visualizing the steel plant's energy consumption and CO_2 emissions throughout 2018.
Revealing the dynamic interplay with environmental factors.

Winter (Jan-Mar)

Mean usage: 32.08 kWh.
Coldest months show
baseline high consumption.

Spring (Apr-Jun)

Temperature increases,
leading to a noticeable shift
in energy load patterns.

Summer (Jul-Sep)

Energy patterns stabilize,
often reflecting consistent
operational demand.

Fall (Oct)

Pre-winter rampup begins,
showing increasing energy
draw as temperatures drop.

This timeline highlights that the system isn't just reacting to external factors; it's remembering past patterns, adapting to current conditions, and breathing in rhythm with its environment, as evidenced by temperature-linked energy draws and autocorrelation in usage patterns.

🌟 Model Performance Showcase
CO_2, kWh, and Temporal Forecasting

Model	Target Variable	R^2 Score	RMSE	Top Features / Notes
🌳 RF (All Variables)	CO_2	0.9985	0.0004362	Usage_kWh dominant; tight residual spread
💨 XGBoost (Low Humidity)	CO_2	0.9944	0.0009868	Usage_kWh dominant; best Q-Q fit, regime clarity
⚡ Full XGBoost	CO_2	0.9951	0.0007801	Moderate curvature; strong but reactive behavior
🚫 RF (No Usage_kWh)	CO_2	0.9944	0.0008356	Loss of signal; residual imbalance
✨ Log kWh XGBoost (Low Humidity)	log_kWh	0.9897	~0.00038[1]	Temporal features (Sunday, Tuesday), stabilized variance
📅 SARIMAX (Usage_kWh + Temp)	Usage_kWh	—	—	Strong AR_1 (0.896), seasonal cycle, temp-driven demand
⏱️ ARIMA (Preliminary, log_kWh)	log_kWh	~0.98[2]	—	Captured autocorrelation; served as precursor to SARIMAX

[1] RMSE in log scale [2] Estimated from earlier decomposition and fit metrics

Modeling with Structure, Signal & Seasonality

Interpretive Highlights

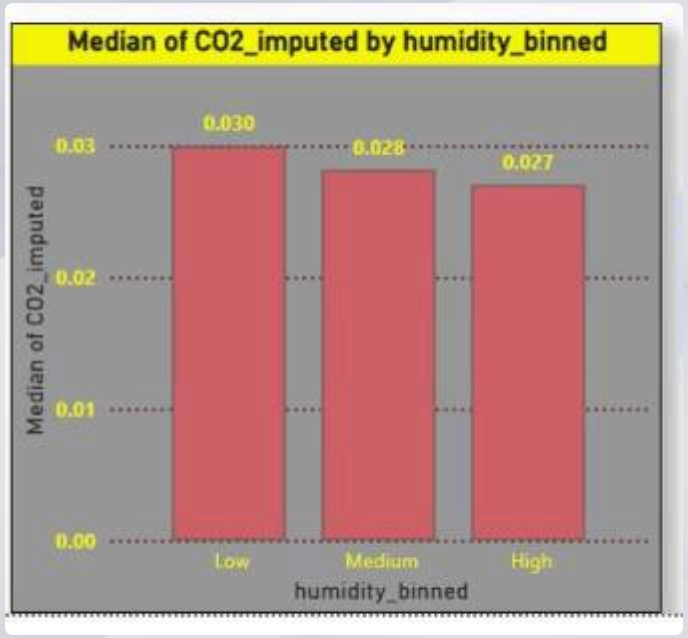
- Usage_kWh remains the system heartbeat, whether as a feature or forecast target.
- Low humidity reveals clarity that models tighten their signal when environmental noise recedes.
- Temporal models (SARIMAX, ARIMA) offer insight into cyclical demand, especially around winter peaks and weekday rhythm.
- Residuals and Q-Q diagnostics show that statistical fit isn't enough, but behavioral texture matters.

Final Takeaways

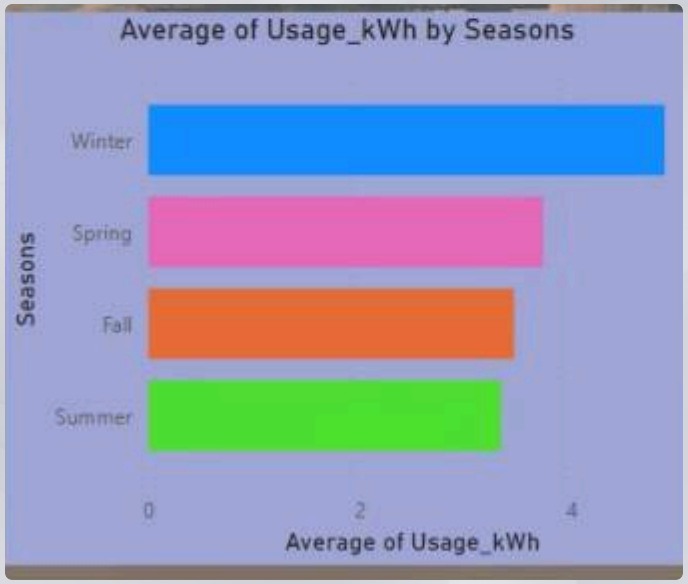
- Usage_kWh isn't just predictive, it's structural. It anchors both CO_2 and kWh models and reveals behavioral regimes.
- Temporal features (weekday, weekend, CO_2 lag) layers complexity and help low humidity models "sing."
- SARIMAX introduces rhythm, seasonal intuition, and exogenous awareness. It's the first step in aligning prediction with time.
- Residual & Q-Q diagnostics round out interpretability, proving that the models don't just perform, they behave as well.

Statistical Insights from working with models: OVERVIEW

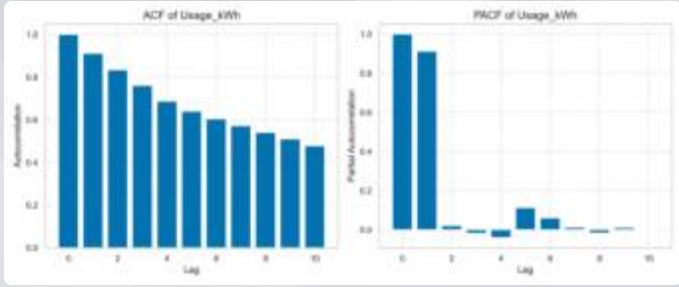
Inverse humidity-CO2



Winter peak energy usage, driving savings focus.



Strong autocorrelation at lags 1–2 (ACF/PACF), suggesting short-term temporal dependencies and helping with SARIMAX parameter selection.



ACF displayed a stepwise decay over 40 lags, indicating autocorrelation persistence. PACF showed two prominent peaks, lags 1 and 2, with subsequent values hovering near zero, suggesting an AR(2) structure. These patterns informed initial SARIMA modeling parameters.

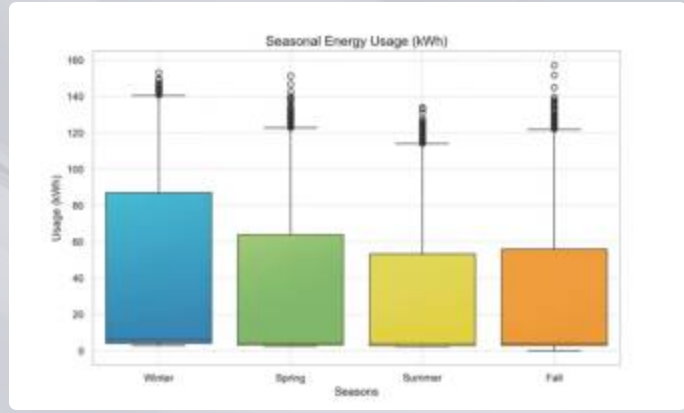
ACF Plot

- Annotation at Lags 1–2:: Strong short-term autocorrelation.
- Stepwise decay indicates persistent structure or non-stationarity.

PACF Plot

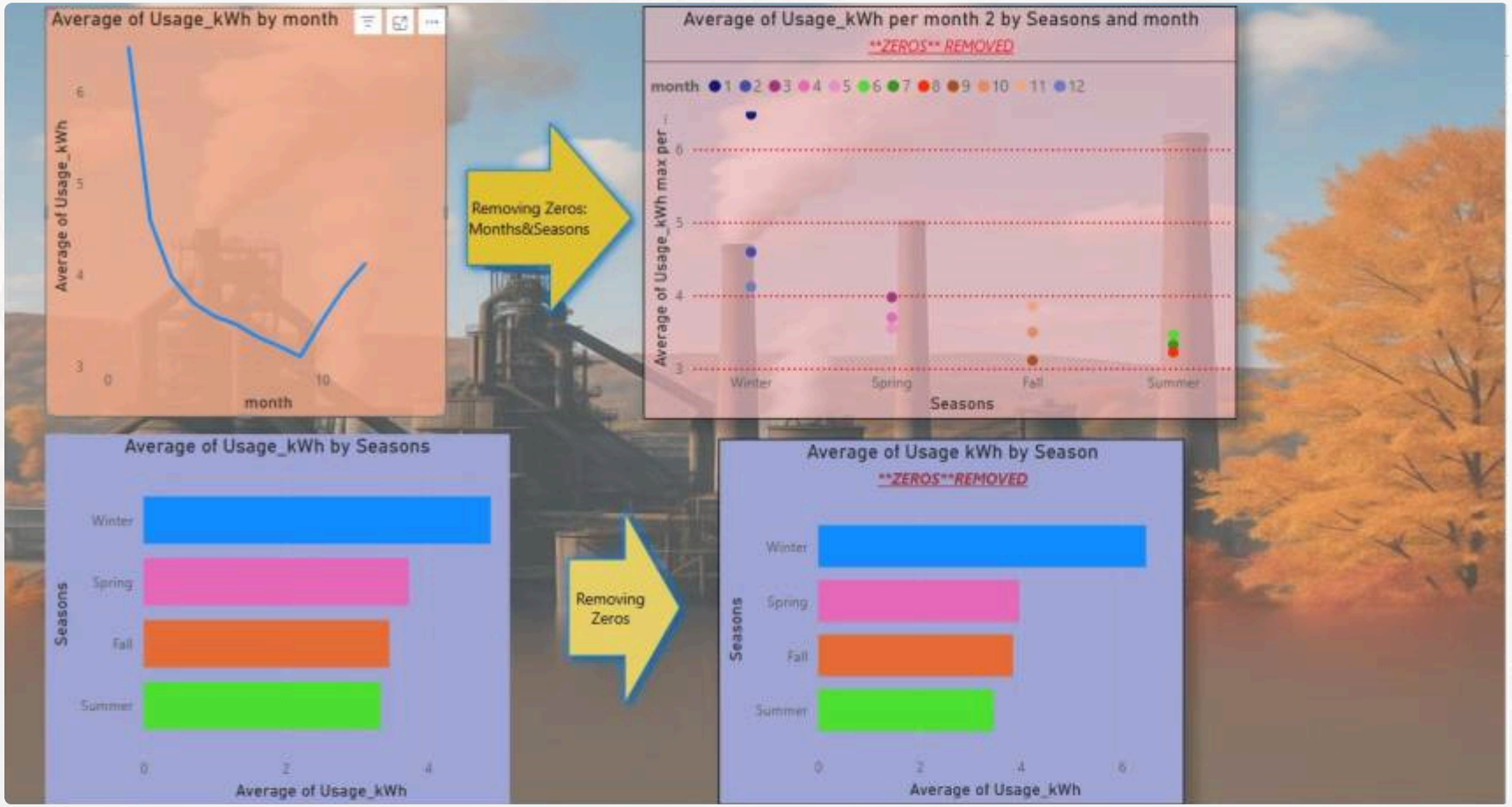
- Peaks at Lag 1 and 2: Suggests AR(2) process.
- Minimal partial autocorrelation beyond lag 2.

*Kruskal-Wallis test confirmed statistically significant seasonal variation in energy usage (*p < 0.001).**

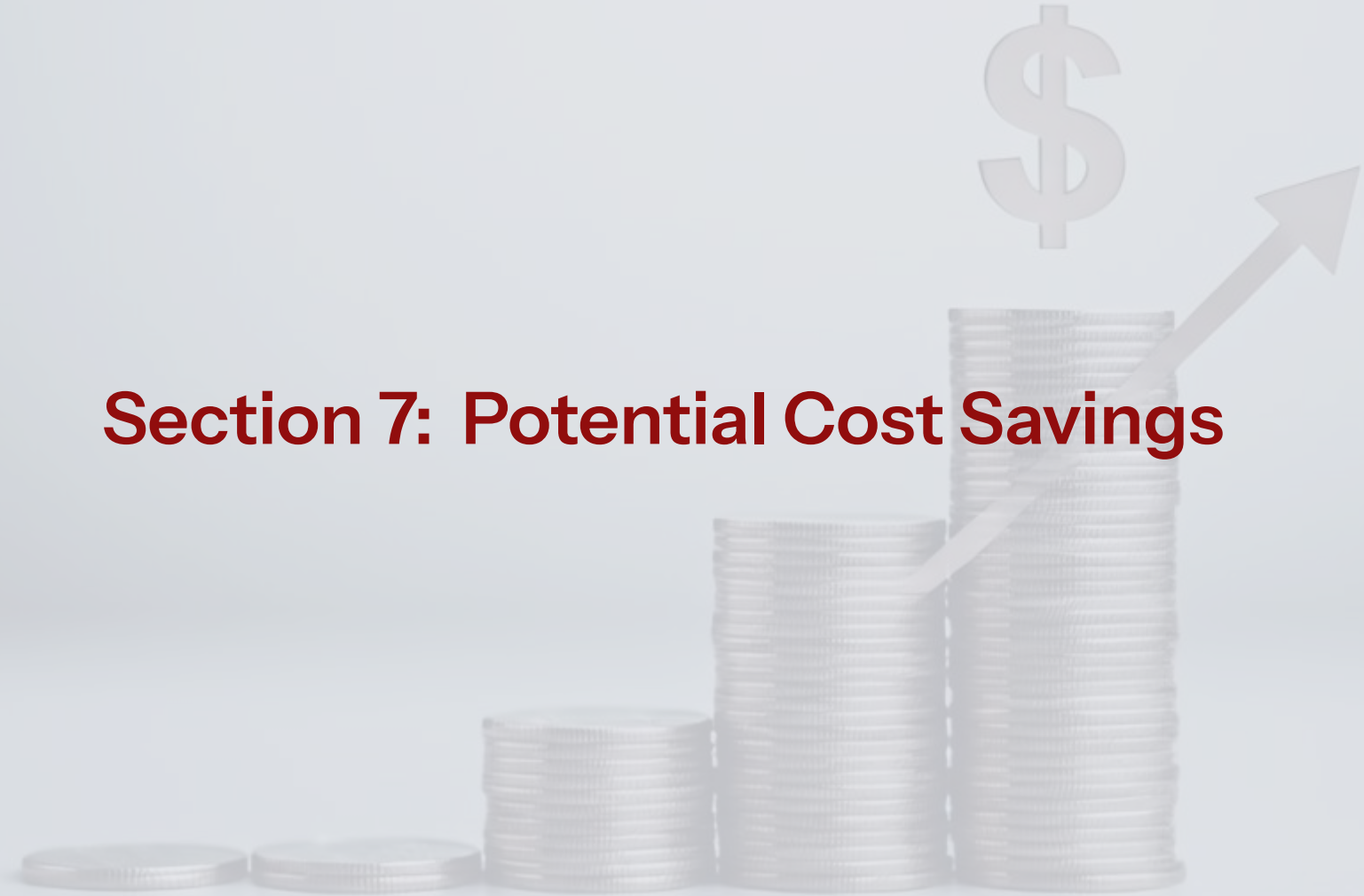


“Kruskal-Wallis is a non-parametric test used to compare medians across multiple groups — ideal when normality assumptions don’t hold.

Closer Look of Energy Usage by Season:
Winter Wins, unless looking at Costs!



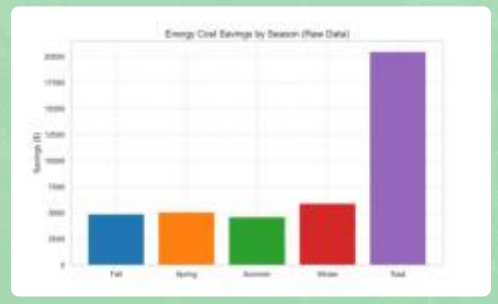
Section 7: Potential Cost Savings



Sometimes, It's All About the Bottom Line

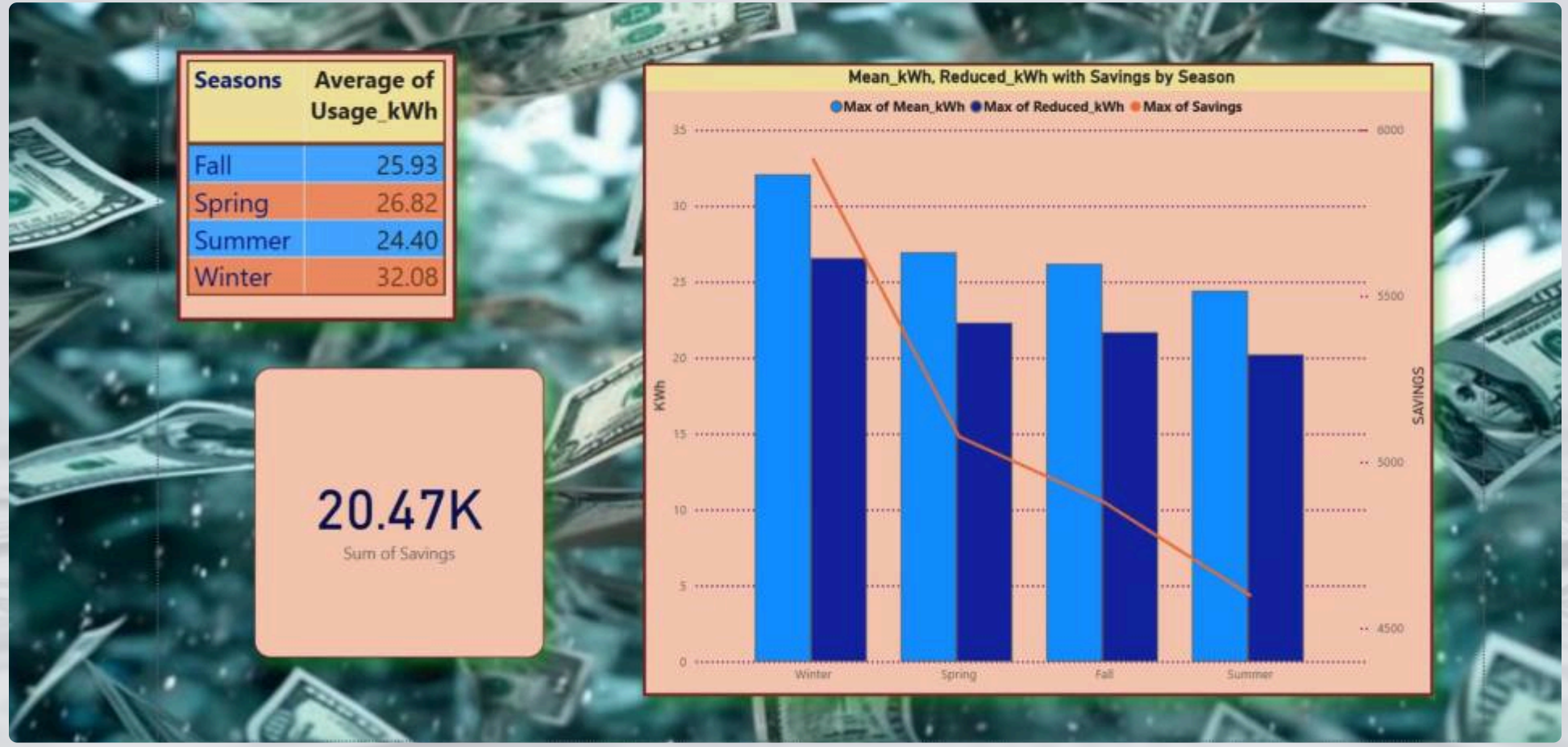
As Rigorous as the Analysis Was... Let's Talk Money

A look at the predictive models and operational insights behind real-world savings



	Mean_kWh	Row_Count	Reduced_kWh	Savings
Fall	26.19	8736.0	21.68	4879.13
Spring	26.95	8832.0	22.32	5077.33
Summer	24.42	8832.0	20.22	4599.14
Winter	32.08	8640.0	26.56	5911.53
Total	27.39	35040.0	22.68	20467.13

The Seasons and the Energy and Savings Comparison



CONCLUSIONS: Curves Were Fit, CO_2 Was Cut, $20,467.13 Was Saved

Section 8: FINAL THOUGHTS

What I found and what I learned!

From Data to Decisions: Recommendations Based on Modeling

- **Energy usage surged during the winter months**, particularly January and February, highlighting seasonal demand patterns likely tied to heating and production cycles.
- **Humidity thresholds influence CO_2 emissions more than expected**. High humidity may suppress temperature-driven emissions, suggesting ventilation tuning opportunities.
- **Reactive power behavior reveals nonlinear disruption points** which flagged inefficiency zones and suggest maintenance or reengineering priorities.
- **Tuesday load skews output data** is either a scheduling anomaly, or a hidden operational pattern worth review.

🧭 Potential Next Steps

- Shift Scheduling optimization, via shit timing or staggering, especially during winter months to avoid peak intensive operations.
- Equipment Load Balancing via audits and rebalance of machinery, to smooth out power draws and reduce strain.
- Maintenance Timing to prepare potential "cold condition" stress on equipment.
- Use the seasonal models to forecast energy demand and negotiate better rates and better budget.
- Load Shaping: Implement strategies, such as thermal energy storage and ramp up buffers, to redistribute high load activitiess into off peak windows.
- Investigate humidity-ventilation interactions during peak usage months. (HVAC)
- Consider system diagnostics or retrofitting to address reactive power inefficiencies.
- Use medium vs max load timing insights to optimize shift planning or power factor management.
- Reassess Tuesday-heavy operations; is it intentional or data artifact?

From Models to Meaning: Reflections on Insight and Growth

Technical Highlights

- Achieved energy savings of **$20,467.13** through predictive modeling and operational insight.
- Confirmed **Usage_kWh as the behavioral anchor** across models, both predictive and diagnostic.
- Identified **low humidity as a regime of predictive clarity**, improving model interpretability.
- Quantified nonlinear effects via **reactive power metrics**, seasonal trends, and calendar features.
- Applied **log, Yeo-Johnson, and quantile transformations** to stabilize variance and meet model assumptions.
- Cross-validated findings across multiple algorithms and platforms.

Personal & Analytical Growth

- Learned and integrated **Python, R, Power BI, Excel, Kaggle, and GitHub** into an applied analytics ecosystem.
- Transitioned from static modeling to **rhythmic forecasting**, uncovering when and why systems shift.
- Transformed frustration with lost code into **resilient, multi-notebook experimentation**.
- Balanced precision and storytelling by letting data speak both statistically and visually.

✨ Closing Thought

Personally, this portfolio isn't just about models, but it's about curiosity, iteration, and listening to data. Each algorithm, plot, and outlier analysis shaped the narrative. And through it all, I didn't just uncover system behavior, I refined my own, in the practice and spirit of data science.

Information

Susan R Schnitzel

Linkedin: **www.linkedin.com/in/susan-schnitzel**

Kaggle: **https://www.kaggle.com/susanschnitzel**

GitHub: **https://github.com/srschnitz**