SDSU | San Diego State University

BDA 696 Advanced Special Topics in Big Data Fall 2023

Final Project Report

Multivariate Time Series Forecasting of Household Power Consumption Data

---

Authors:

Ethan Nyi Nyi

Vedant Ghavate

Table of Contents

Abstract

In the wake of higher emphasis on environmental consciousness, household power consumption habits will have the most impact on overall energy consumption. The supply-demand chain of energy is dependent on consumption habits in households, through the day, month, and seasons of the year. Electricity use is changing with time as people become more conscious about their carbon footprint, and with more efficient household technology, the consumption rate is changing. On the other end, the government's switch to renewable energy sources makes the supply-demand gap even tighter.

This makes it necessary to analyze the trend of consumption from both a single factor and multifactor perspective that includes temperature, customer price index, electricity prices, and the popularity of energy-saving initiatives. We will build an LSTM (Long Short-Term Memory) Recurrent Neural Network model for the univariate forecasting of power consumption.

We also seek to perform a multivariate analysis with a Temporal Fusion transformer to understand the impact of external factors mentioned above on energy consumption.

Introduction

Power consumption habits may depend on a wide range of factors like electricity prices, social trends through the green initiative, atmospheric conditions like temperature, humidity, precipitation, and overall retail inflation. We plan to study the impact of these factors on power consumption and build a model to predict power consumption based on these factors.

We use MSTL with LOESS for EDA as it decomposes the time series into different seasonal and trend elements, the effectiveness of this decomposition can be identified by the residuals, if they lie within the significant range, the trends have been successfully decomposed by the MSTL.

Since most of the data is in time series, we needed a suitable multivariate model that could handle multiple time series data and capture correlations between them. A random walk model is considered as a baseline for our model, while LSTM and Google's TFT (Temporal Fusion Transformer) performance will be compared and analyzed against this baseline model's performance.

Along with multivariate time series analysis, it is also important to address the granularity of the data that we have obtained, the final model should be able to handle granularity and capture trend and seasonality through hours, days, months, and years. Since we are using multiple data sets with different levels of granularities, like the consumer consumption index which changes every month, we will need the final model to be able to handle multivariate data along with multi-step granularity. These two layers of complexity will provide us with training data that can capture trend and seasonality through multiple time series, exogenous, and seasonal features.

We will compare and contrast the baseline model i.e. random walk with TFT through their accuracy. One ability the TFT model possesses is that it can handle multiple seasonal trends thus addressing the issue of multiple granularity.

We aim to predict and understand the trend in the power consumption habits of households and the significance of factors in the overall trend.

Methodology

Preprocessing

The features that we will use as targets for our EDA and our predictive models include the global active power consumption for an individual household in France as well as the energy consumed via sub-metering 1, sub-metering 2, and sub-metering 3 extracted from a dataset in the UCI Machine Learning Repository, the search trend for the term “ampoules basse consommation” translating to “Energy saving bulb” in French, the search trend for the term “Jean-Louis Borloo” and the search trend for the term “Grenelle de l'environnement” from Google Trends data, monthly consumer confidence survey scores from the INSEE, the price index for consumer electricity and fuel prices from Federal Reserve Economic Data (FRED), the hourly air temperature levels in France and the daily precipitation depth extracted from the Integrated Surface Dataset hosted by the National Oceanic and Atmospheric Administration (NOAA). Our features are derived from multiple data sources, which are merged for this project.

Feature Engineering For Time Series Data

We address outliers and different scaling factors for different time series variables in our data using 2 levels of normalization. We first will transform our values to a uniform scale based on the RobustScaler function via the scikit-learn package. This will be beneficial during exploratory data analysis when we decompose our time series as the RobustScaler function can suppress the influence of outliers during normalization. We still observed outliers in Air Temperature and Wind Speed upon applying the RobustScaler function to their values, we filtered outliers for these features based on a value threshold. We replace values that cross this threshold with NaN before we interpolate for these empty values using spline interpolation.

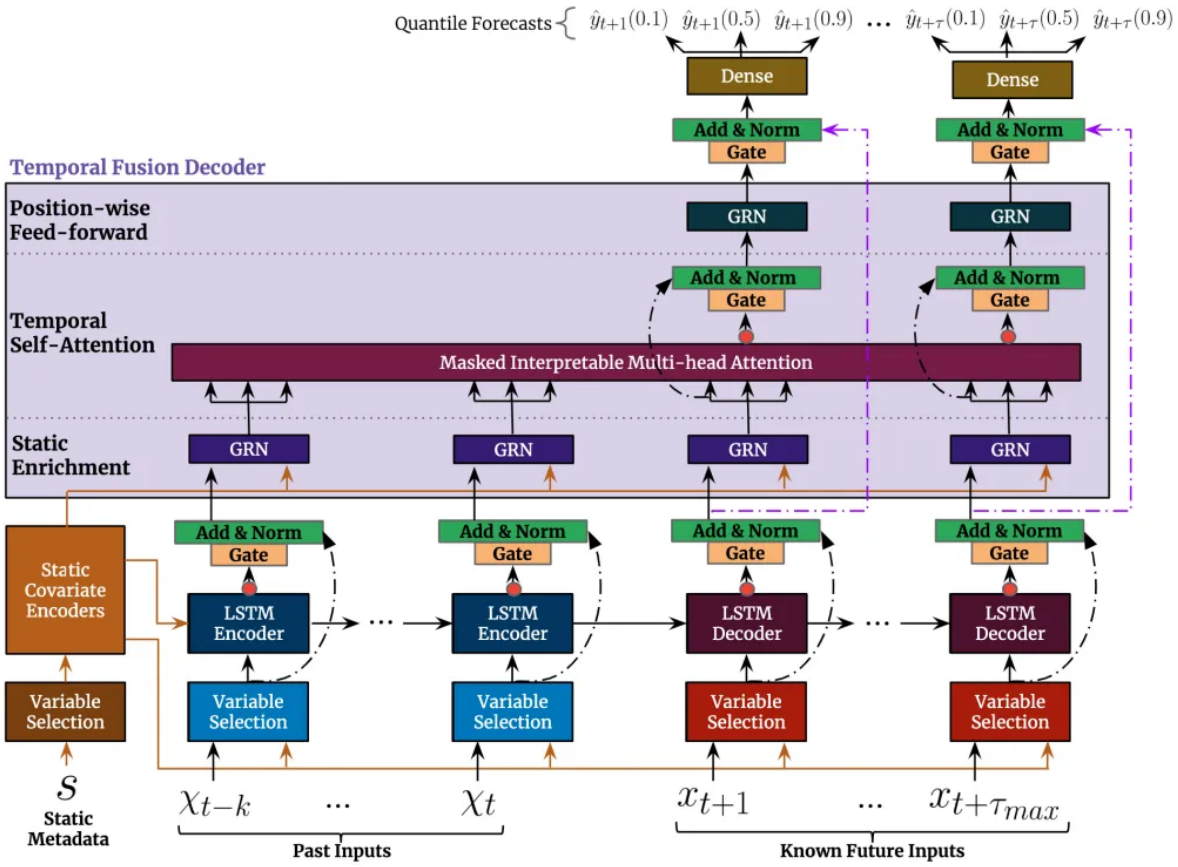
We will use spline interpolation for addressing outliers as well as pre-existing empty values at specific periods in our features. The statsmodel package also offers the MSTL function which decomposes our time series based on multiple levels of seasonality which we will use for our exploratory data analysis to observe seasonal and trend components for each feature.

Temporal Fusion Transformer

The Temporal Fusion Transformer (TFT), is a robust model designed for multi-horizon and multivariate time series forecasting. TFT introduces a predictive approach utilizing past target values, time-dependent exogenous input features, and static covariates. Unlike conventional models providing single-value predictions, TFT generates prediction intervals using quantiles, offering a more comprehensive forecast range.

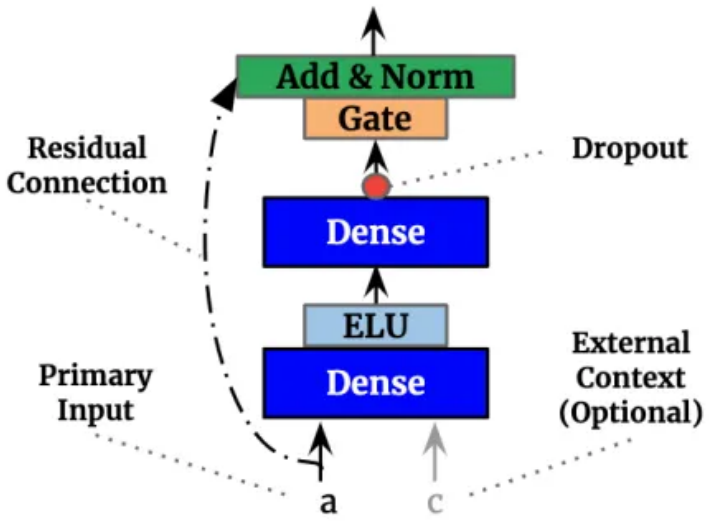
The Temporal Fusion Transformer (TFT) serves as a powerful model for forecasting various time series scenarios. Its predictive capacity leverages historical target values, time-related unknown features, known time-dependent features, and static contextual information to anticipate future trends. This methodology marks a departure from singular value predictions by incorporating prediction intervals through quantiles.

Unlike ARIMA forecasting models, there is no need to make the time series stationary by using log difference transformation, for TFT, the time series can be left as is.



TFT Model Architecture: TFT's architecture comprises distinctive components:

Gated Residual Networks (GRN): Integrated at different levels within TFT, these networks facilitate flexibility by establishing skip/residual connections. Such connections enable the model to bypass unnecessary nonlinear layers, thereby enhancing its adaptability across diverse scenarios and significantly reducing parameter requirements.



Gated Residual Network (GRN)

Static Covariate Encoders: These modules extract context vectors from static metadata, strategically embedding them into the TFT network. This embedding occurs at multiple points, enabling the model to integrate static information into temporal representation learning.

Variable Selection: TFT employs separate variable selection blocks for different input types, determining the significance of each feature. This selective approach enables subsequent layers to process transformed inputs, enhancing the model's ability to learn from continuous and categorical features effectively.

Sequence-to-Sequence Layer: Replacing positional encoding found in Transformers, this layer specializes in capturing local temporal patterns through recurrent connections. Context vectors initialize the LSTM unit, enriching temporal representation with static information.

Interpretable Multi-head Attention: TFT refines the multi-attention mechanism to ensure interpretability. By sharing weights across attention heads, the model facilitates easy identification of relevant values, aiding in understanding past time steps' influence on forecasts and detecting significant changes in temporal patterns.

Quantile Regression: TFT employs quantile prediction, estimating the distribution of target values, allowing for quantification of uncertainty at each time step.

TFT's interpretability features stand out in two primary aspects:

- The ability to trace relevant past time steps for each forecast is traditionally achieved through seasonality and autocorrelation analysis.
- Identification of substantial changes in temporal patterns by comparing average attention patterns per forecast horizon with attention weights at each point.

Google's Temporal Fusion Transformer (TFT) presents a promising methodology for multi-horizon and multivariate time series forecasting. Its interpretability features, variable selection capabilities, and refined attention mechanisms offer valuable insights into understanding temporal patterns and uncertainties in forecasts.

We will use the TemporalFusionTransformer model from the PyTorch Forecasting library and PyTorch Lightning

This process has three steps:

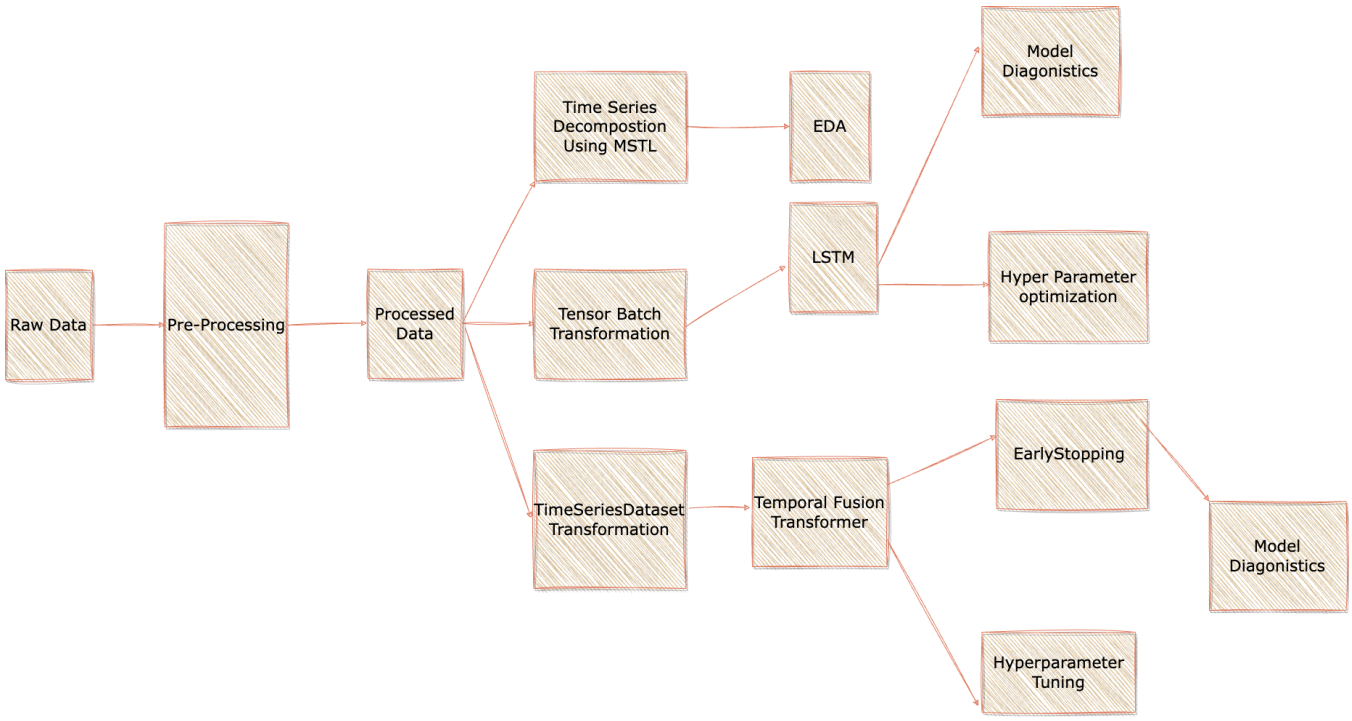
- Wrap our data frame into a TimeSeriesDataset instance.
- Pass our TimeSeriesDataset instance to TemporalFusionTransformer.
- The TimeSeriesDataset is very useful because it helps us specify whether features are time-varying or static. The TemporalFusionTransformer accepts only this format for processing.

Apart from all other parameters, the most relevant for our project will be ‘max_encoder_length’ which defines the lookback period, and max_prediction_length which specifies how many data points will be predicted.

We do not have to write our own Dataloader as the TimeSeriesDataset format will handle the features, and allow for easy normalization of the sequences as they will differ in magnitude, i.e. CCI is released monthly. The GroupNormalizer is used for the task. We also can specify a lookback window for predictions, this is set according to the granularity of our base data, that is hourly and one week prior.

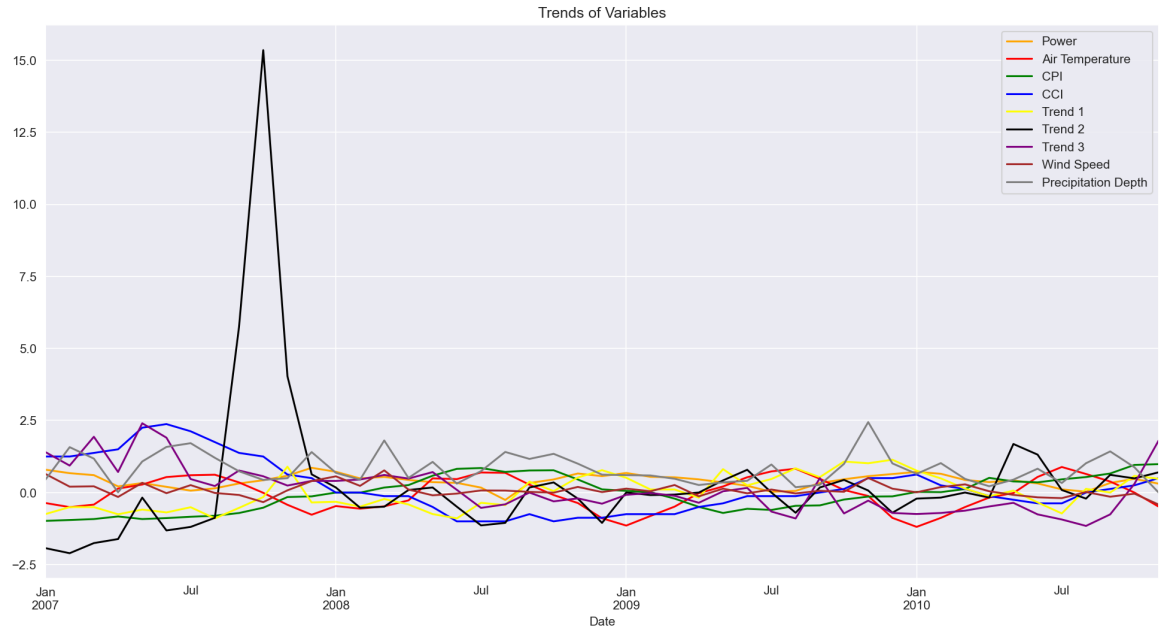
We use the EarlyStopping callback to monitor the validation loss. We use Tensorboard to log our training and validation metrics. Our model uses Quantile Loss - a special type of loss that helps us output the prediction intervals.

Pipeline



Exploratory Data Analysis

Overlay Charts



The constructed overlay charts display the normalized trends of the time series variables - global active power, sub-metering 1, 2, and 3, air temperature, CCI score, CPI index, the 3 search terms, wind speed, and precipitation depth.

In the overlay chart that tracks the time series from January 2007 to November 2010, we take note of a significant peak in the second Google search term corresponding to “Jean-Louis Borloo”. Jean-Louis Borloo references an individual who was the Minister of Ecology, Energy, Sustainable Development, and the Sea for France from 2007 to 2010. It is possible to attribute this spike to the popularity of the “Grenelle de l'environnement" which took place during the summer and fall of 2007 that Jean-Louis Borloo had participated in.

The remaining normalized features have relatively similar levels of peaks and dips. We will examine this relationship further in our next section.

Time Series Decomposition using MSTL with LOESS

For our exploratory analysis, we will decompose the time series of our variables into their level, trend, seasonality, and residual components. An additive time series model can be represented in the following equation as shown:

$$y(t) = \text{Level} + \text{Trend} + \text{Seasonality} + \text{Noise}$$

An additive time series model is when the components of the time series are added together and is commonly used when the seasonality component is relatively constant concerning trends. A multiplicative time series model can be represented in the following equation as shown :

$$y(t) = \text{Level} * \text{Trend} * \text{Seasonality} * \text{Noise}$$

A multiplicative time series model multiplies the components of the time series together and is used to represent time series variables with seasonal fluctuations that have varying amplitudes over time concerning trends, and it differs from the additive model in the relationship between the components of the time series. We will utilize the MSTL package from the stats model, which allows us to measure multiple levels of seasonality within a time series. All our variables exhibit fluctuations that can be modeled by an additive time series model. MSTL utilizes the LOESS (locally estimated scatterplot smoothing) which is a non-parametric regression technique that extracts 3 components, the trend, the seasonality, and residuals from a time series feature.

Appendix B of our Appendix section will contain the output graphs of the MSTL function applied to all the features displaying the trend and seasonality component. We also show magnified views of seasonality components based on a filtered period, and the Python code used to build our visuals and analysis.

The global active power consumption of the specific household aggregated to the hourly frequency, can potentially exhibit 3 levels of seasonality (Visual A). One on the daily level where seasonality patterns exist within a day (Visual B), the weekly level within days of the week (Visual C), and the yearly level where seasonal patterns exist within months of the year (Visual D). The trend, adjusted for all levels of seasonality, is shown to exhibit a downturn in late 2008 to early 2009 before rebounding to a lower average level in mid-2009 until 2010. We will discover as we move forward that dips in level during late 2008 are also seen in weather data. When we focus on the daily seasonality component from global active power, we see that power consumption peaks during midday hours around 12 PM, and drops down to its lowest point around 8 PM. January 1st, 2007 starts on a Monday. The weekly seasonality shows that most power is consumed during weekend hours, which could be explained by the individuals in the household having no work during the weekends which causes more power consumption due to spending longer hours at home. Power consumption is also shown to have a yearly seasonality pattern, where it shows a steadily decreasing trend from January to August before an upshoot from August to December. This could be explained by a greater need for heating and electricity during the Fall and Winter months, August through December in France are shown to have relatively lower levels of air temperatures as we will examine for our next feature.

Air Temperature data recorded in France on an hourly level, is also seen to show strong seasonality within daily and yearly levels (Visuals E and F). Air Temperature is shown to spike during midday hours within a day. On the daily level of seasonality for Air Temperature with hourly frequency, the air temperature is at its highest at noon, before it drops to its lowest between 10 PM and 11 PM. It also peaks during the first quarter of the year before a consistent decline throughout the year. The seasonally adjusted trend (Visual G) shows a consistent

increase in the temperature level over the period from 2007 to late 2010, with minor dips during the last quarter of 2008 and the last quarter of 2009.

For time series that are of monthly granularity such as the CCI score and CPI index, they can exhibit a single level of annual seasonality shown (Visual H and I). For CCI, the yearly seasonal behaves in a pattern that repeats from 2007 to 2009, with consumer confidence scores regarding economic outlook peaking during the first to second quarter of the year and dropping off from around the mid-year to end of year. The seasonal component breaks from this pattern from 2009 onwards until late 2010, as it experiences a downward trend within the first few months of the year before experiencing a general upturn until the end of the year.

The CPI score which tracks the price index of consumer electricity and fuel prices exhibits a constant seasonality pattern that repeats yearly from 2007 to 2010. The yearly seasonality component indicates that August shows the lowest CPI index in the year which means that the consumer price levels of electricity and fuels are at their lowest during this time before peaking in the final quarter of the year around November. The price index was the highest during mid-2008, before dropping off in mid-2009 and rebounding in 2010. There are possible economic implications that can be attached to the behavior of CPI and CCI scores, regarding the economic recession that spanned from late 2008 to 2009.

All 3 search terms have a weekly frequency and therefore can exhibit 2 levels of seasonality which are monthly and annually. The first search term (Visual J) which tracks “Energy saving bulb” experiences an upturn and peaks around mid-2009 before gradually decreasing which shows that there was the greatest interest for terms related to energy-efficient appliances such as light bulbs during that period. The second search term (Visual K), which is the name of the Minister of Ecology, Energy, Sustainable Development and the Sea in France, experienced a surge in popularity from early 2007 to January 2008 before dropping down to a lower average level in mid-2008 before experiencing a rebound around April 2010. The third search term is the keyword “Grenelle de l'environnement” (Visual L). The popularity of the search term shows a consistent decline from early 2007 until late 2010. The initial high popularity of “Grenelle de l'environnement” is attributed to the event taking place in the

summer and fall of 2007. The search term therefore gradually lost popularity possibly due to less media coverage upon completion of the event. The 3 search terms share a similar monthly level of seasonality with a cyclical pattern throughout one month (Visuals M, N, and O). The first search term denoting “Energy saving bulb” tends to dip during the first 2 weeks of the month until it peaks towards the end of each month as shown in the weekly seasonality plot. The other 2 search trends for “Jean-Louis Borloo” and “Grenelle de l'environnement” also display a similar pattern within a monthly cycle based on dips early within the month followed by rebounds from the second half onwards. The yearly seasonalities can also be found (Visuals P, Q, and R). The first search term shows a general decline from the start of the year until mid-August where we see large increases towards the end of the year. The second search term also shows a strong spike towards the end of the year around November, scaling off before January which means that interest in terms related to Jean-Louis Borloo is highest around November before steeply dropping. The third search term shows spikes during the mid-year before decreasing towards the end of the year.

We can draw insights from the household’s electricity consumption patterns when total energy consumption is dissected into portions that represent everyday functions within the household. Sub-metering 1 (Visual S) corresponds to the active energy consumed in appliances located in the kitchen of the household mainly for cooking, heating, and baking purposes. Sub-metering 2 (visual T) corresponds to the energy consumed in the laundry room, and Sub-metering 3 (visual U) corresponds to the combined energy consumption within an electric water heater and air conditioner. The active energy consumed on the daily level (Visuals V through X) for sub-metering 1 drops dips at 10 AM, before rising at 11 AM. Other notable peaks in the day include 4 PM and midnight. Sub-metering 2 peaks at 1 pm and 5 pm, and has the steepest decline during midnight on a daily level. Sub-metering 3 has notable dips at 9 AM, and peaks again at 5 PM and 7 PM. On the weekly level (Visuals Y, Z, and A1), the 3 features share a common trend as we see shared peaks around Sunday with relatively higher levels of energy consumption during the period as compared to the rest of the week. The 3 submetering trends also have yearly patterns. (Visuals A2, A3 and A4). Sub-metering 1 and 2 show cyclical patterns throughout the year with continuous spikes and dips in level across the period. The yearly pattern of sub-metering 3 more closely

resembles that of the household's global active power as it peaks towards the end of the year. The seasonally adjusted trends for Sub-metering 1 show that it has decreased consistently throughout the period from 2007 to late 2010. Sub-metering 2 also exhibits a similar downturn but the trend stagnates around early 2009. Sub-metering 3 on the other hand shows a positive trend, as we see a sharp upturn until it stagnates in late 2009 to early 2010.

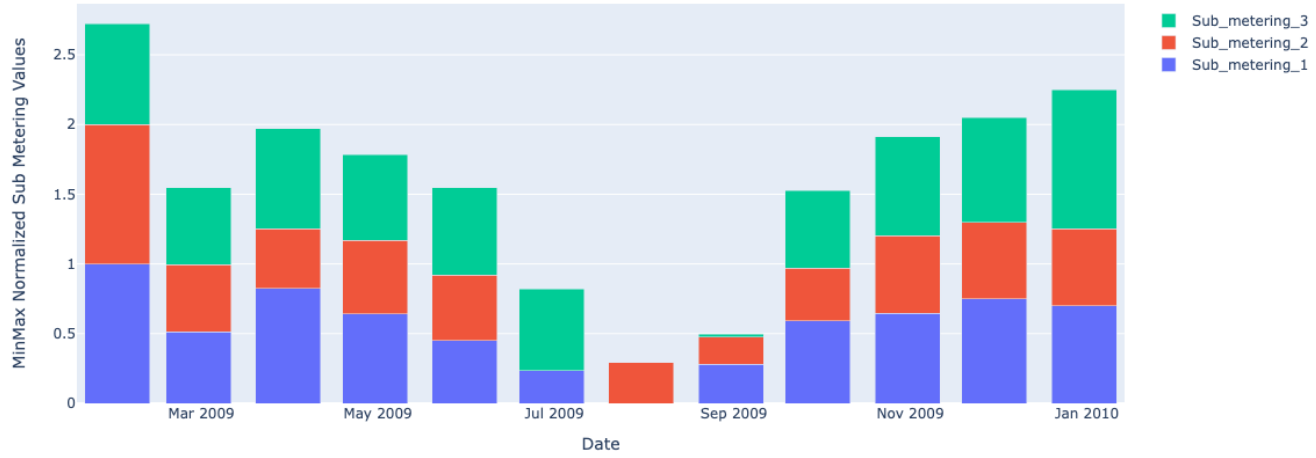
The wind speed is defined as the rate of horizontal travel of air in meters per second. The wind speed is shown to exhibit the highest levels on average during 5 PM within a day, whilst it drops to its lowest points during odd hours of 12 AM to 5 AM (Visual A5). The wind speed shows an upturn with cyclical patterns until September where it starts to decline until the end of the year, before rebounding into another peak (Visual A6). This tells us that the relevant region in France has relatively windier months from September to December. The wind speed over the period from 2007 to 2010 has shown to have upturns and then points of the plateau before dropping off, one in late 2008 and one in early 2010 (Visual A7).

Precipitation depth measures the total amount of rainfall within 24 hours distributed evenly over a watershed area. The precipitation depth started to show a steep decline in its seasonally adjusted trend around the middle of 2008 (Visual A8), which gives the inference that rainfall during this period was relatively lower than the other periods before seeing a rebound from the middle of 2008 onwards. The precipitation depth is shown to exhibit cyclical levels of seasonality throughout the year (Visual 8). There are many peaks across the year, with the most notable peak being towards the end of the year in early December before dropping off rapidly. Similar to wind speed, the seasonally adjusted trend of precipitation depth also varies across the period and does not display stationarity. The recorded precipitation depth had seen a strong downward trend from late 2008 onwards until mid-2009 before increasing to a higher level until the end of the period in 2010 November. From this trend, we see that the period from late 2008 to the second quarter of 2009 displays abnormal behavior. The dip in precipitation depth coincides with the plateau in wind speed and a dip in air temperature during the last quarter period of 2008. This potentially suggests that the period in question shows relatively lower levels of rainfall, wind, and temperature as compared to other periods when adjusted for seasonality.

Extrapolating the previous insight onto our aforementioned analysis on the global active power consumption and sub-meterings shows that amid this dip in weather metrics, the power consumption also declined during the same period of late 2008. We can assume that this decline is correlated with the decline in energy consumption in sub-meterings 1 and 2, which correspond to electricity use in the kitchen and laundry room respectively during the same period. There are potential socioeconomic connections we can draw to explain this behavior, shown by the CCI score which is an indicator that reflects the general economic outlook of households in France. The CCI score also saw a large decline in the last quarter of 2008 which we can interpret as a more negative economic outlook.

It is important to examine the residuals in each feature's time series decomposition, to gauge for the independence and randomness of the residuals. Independent residuals are residuals that show no serial autocorrelation, and this means that the MSTL model was able to exhaustively extract all trends and seasonal components from the time series. Residuals that show serial autocorrelations, give us the inference that there are still underlying trends additional to the pre-existing ones in our time series that we can still discover through more sophisticated techniques, which will be beyond the scope of this paper. The Durbin-Watson statistic is a statistical test that can check for the presence of serial correlation in the residuals of a regression model. The null hypothesis is defined to be that first-order autocorrelation does not exist within residuals whereas the alternative hypothesis is that first-order autocorrelation exists. Values for the Durbin-Watson statistic range from 0 to 4. Values that are below or greater than 2 are likely to have some level of autocorrelation, with values lower than 1 or greater than 3 showing significant likelihood of autocorrelation. In our data frame of test results for each feature (Visual A9) we see that when we apply the test via the stats model package to our features' MSTL residuals, there are several features that exhibit a high likelihood of autocorrelation. The features are global active power consumption, sub-metering 3, CCI score, CPI index, the second search term "Jean-Louis Borloo", air temperature, and precipitation depth. We conclude, therefore, that given that we have extracted and analyzed the trend components and multi-seasonality patterns for arbitrarily defined seasonality levels, there is the possibility of uncovering additional trends and seasonality beyond what we have explored in the paper.

Stacked Bar Graph



The three submetering levels represent different types of appliances that consume electricity:

- Sub Metering 1 - Appliances in the Kitchen, such as a dishwasher, an oven, and a microwave.
- Sub Metering 2 - Washing Machine, Tumble Dryer
- Sub Metering 3 - Electric water heater and an Air Conditioner

The above-stacked bar graph shows the monthly sums of the three sub-metering levels. Since the watt-per-hour consumption of every metering cannot be scaled on the same graph due to a large difference in the sums, we have used min-max normalization for visualization purposes.

From the graph, it is clear that the submetering levels have a yearly trend and change according to the weather seasons. The months with moderate cold and heat show decreased submetering for level 3, i.e. air conditioning and heater as the moderate weather does not require additional air conditioning or heater use. January and February show the highest consumption of electricity for all the submetering levels, this may be due to the holiday break when most people have returned to their normal schedule and the weather is at its coldest.

During July and August, The submitting levels 1 and 2 have also shrunk, this may be due to the vacation season and people not being in their households to consume electricity.

Since the datasets take into account a particular household, the negligible consumption in August for all three sub-metering levels indicates that the household was unoccupied during those months. Overall, the three sub-metering levels reflect power consumption through habits through the seasons and this can be explained by the weather at that time.

Dataset description

Power Consumption: Individual Household Consumption Dataset (UCI) - This dataset contains data from one household and includes the following fields: Date, Time, Global Active Power, Global Reactive Power, Global Intensity Voltage, Sub Metering 1, 2 & 3.

Integrated Surface Dataset for Daily Weather Data (NOAA): This dataset contains synoptic surface information from numerous weather stations. We use the hourly air temperature in degrees Celsius recorded by a weather station in Abbeville, France. (a 2-hour drive from Sceaux, where the household is located), daily precipitation depth in meters, and hourly wind speed in meters per second from January 2007 to November 2010.

Monthly CPI: Electricity, Gas and Other Fuels for France (FRED) - reflects the monthly electricity prices from 2007 to 2010

CCI: The monthly consumer confidence index (CCI) by INSEE is an indicator based on a consumer survey conducted on households in France. The survey aims to explore general attitudes regarding economic outlook through the lens of expenditure and saving habits. The dataset contains monthly CCI scores from January 2007 to November 2010.

Google Trends: We have selected three terms related to policy changes and popular green energy efforts that may have affected the power consumption habits as shown below

File Name	Term	Meaning
GoogleTrends1.csv	ampoules basse consommation	Energy saving bulb
GoogleTrends2.csv	Jean-Louis Borloo	Minister of Ecology, Energy, Sustainable Development and the Sea
GoogleTrends3.csv	Grenelle de l'environnement	The 'Grenelle de l'environnement' was an open multi-party debate that took place in France in the summer and fall of 2007 to define key points of public policy on environmental and sustainable development over the following five-year period.

Data cleaning

Like every dataset, out of the six we have chosen, three of them, power consumption datasets, CCI datasets, and weather datasets have some missing values. Since the data is in a time series, we have the option to fill in missing values by using spline interpolation. Spline interpolation involves creating a smooth curve that passes through given points by using piecewise-defined polynomials. Spline interpolation offers smoother results than other methods due to its continuity and differentiability properties.

The spline interpolated values do not influence the trend of the time series and they are based on the adjacent values and the overall trend, thus this provides for a smooth estimate of the missing value. The datetime column of every dataset needs to be formatted in the most readable, consistent date time column through all the six datasets, pandas have a 'to_datetime' function for converting any string value to datetime for any format.

```
df['DATE'] = pd.to_datetime(df['DATE'],format="%Y-%m-%d")
df2010['DATE'] = pd.to_datetime(df2010['DATE'],format='%Y-%m-%dT%H:%M:%S')
df['timestamp'] = pd.to_datetime(df['timestamp'],format='%Y-%m-%dT%H:%M:%S')
```

Model Building and Training

Data Preparation for TFT

Power consumption data such as global power usage, and sub-metering 1,2, and 3 are resampled with seasonality every 6 hours. Other hourly data such as wind speed, precipitation depth, and air temperature are also aggregated based on averages to match the level of the seasonality of power consumption. Values that are of a higher level of seasonality such as CCI, CPI, and Google search trends are converted to 6-hour intervals through forward-filling the monthly values to be constant within datetime periods during the month.

The TFT model requires that we define time indexes within our dataset. We specify the time-varying known categorical variables to be the day, week, month, hour. These are columns we create based on the Date column to specify categorical inputs known into the future. We define the time-varying known reals to be time index and year as they are numerical indexes. Time-varying unknown reals are defined to be time series features that are not known into the future, these variables are our time series features in question. Power consumption globally and for the 3 sub-meterings, CPI Index, CCI Score, the 3 Google search trends, and weather data. Since some of our time series features have a weekly level of seasonality, we set one week as the minimum lookback period and a year as the maximum. The model will utilize past information from within the aforementioned period, to predict global power consumption 6 months into the future as a maximum. These parameter values, along with others, are set in the TimeSeriesDataset class for the conversion of our dataset. This can also be found in the appendix.

Hyperparameter Tuning

The hyperparameters to be tuned in the Temporal Fusion Transformer are the hidden size units, continuous hidden size units, dropout rate, and learning rate. The hidden size units denote the dimensionality of the hidden states within feedforward neural networks in the GRN unit. The continuous hidden size shares a

similar interpretation but is for static inputs. The dropout rate is the proportion at which the neurons in a neural network layer are made to have their output values set to zero. The dropout rate influences the impact of overfitting through this mechanism as it allows for generalization. In the context of the Temporal Fusion Transformer, the dropout rate refers to the dropout layers within the GRN unit. The learning rate is defined as the rate at which the model's weights are updated after each iteration. It can be considered to be the step size during the loss optimization process of the Temporal Fusion Transformer.

We utilize the PyTorch tuning class to access the optimize_hyperparameter function, as well as the optuna package as PyTorch utilizes optuna to iterate through studies, which each contain a collection of hyperparameter combinations that are to be set during loss optimization to gauge which combination provides for optimal results. We optimize against the Quantile Loss function.

The Quantile Loss function is defined as the following: Machine learning models work by minimizing (or maximizing) an objective function. An objective function translates the problem we are trying to solve into a mathematical formula to be minimized by the model. As the name suggests, the quantile regression loss function is applied to predict quantiles. A quantile is the value below which a fraction of observations in a group falls. For example, a prediction for quantile 0.9 should over-predict 90% of the time. ("Blog: Quantile loss function for machine learning")

Results

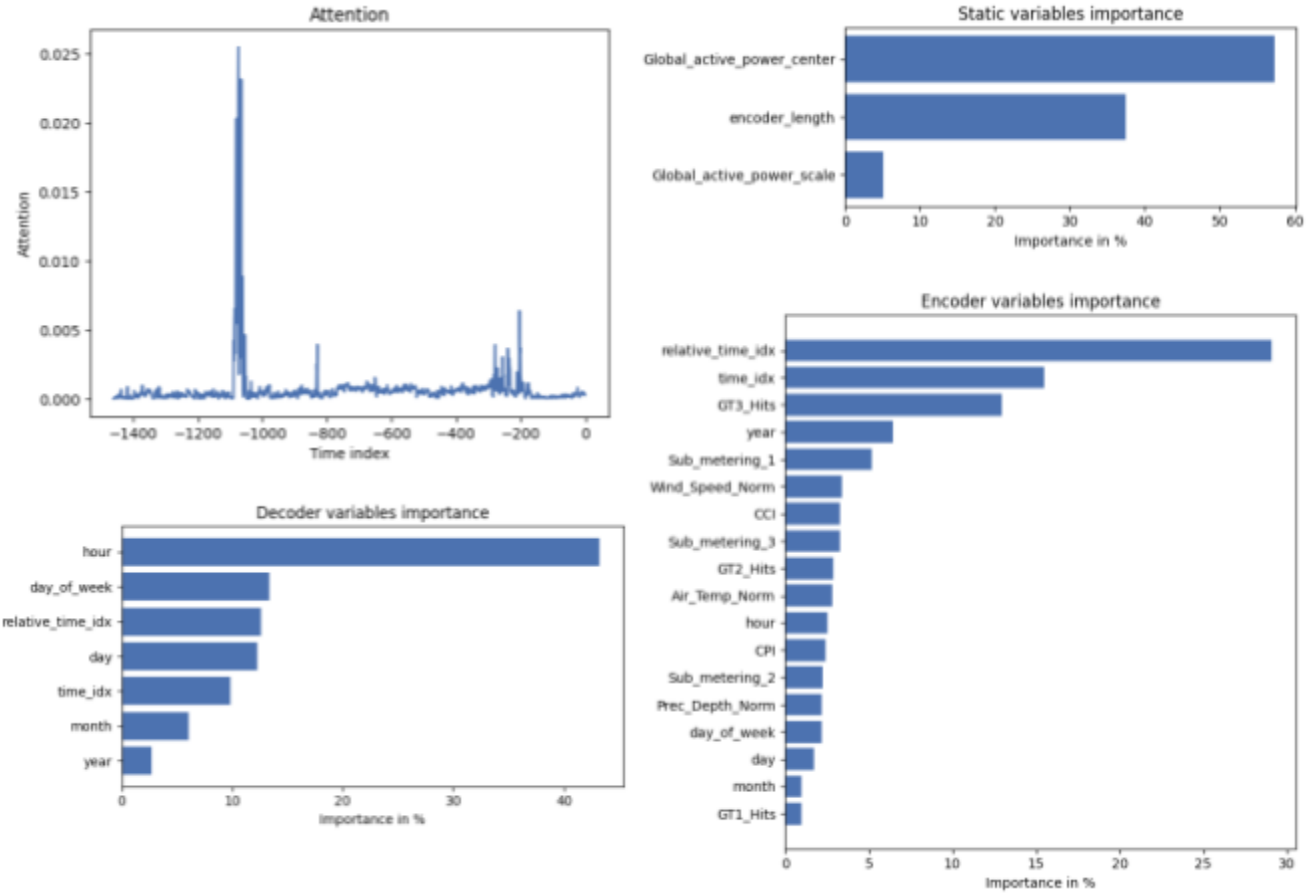
Loss functions

Loss Function - Points	TFT	Baseline
MAPE (Mean absolute percentage)	0.996	1.760
SMAPE (Symmetric mean absolute percentage)	0.675	0.729
Poisson Loss	3.187	3.587
Quantile Loss	0.953	1.001

Table : Comparison of TFT and Baseline on various loss functions

Various Loss Functions show the TFT performs much better than the Naive Bayes-based baseline model.

Variable importance



Observed vs Predicted

The TFT model has multiple prediction formats - raw, data frame, and prediction. Using the ‘raw’ we plot the quantile results of the predictions to the observed values. The raw prediction mode also provides the attention score of the TFT for the relative time index.

Using 'raw' predict mode

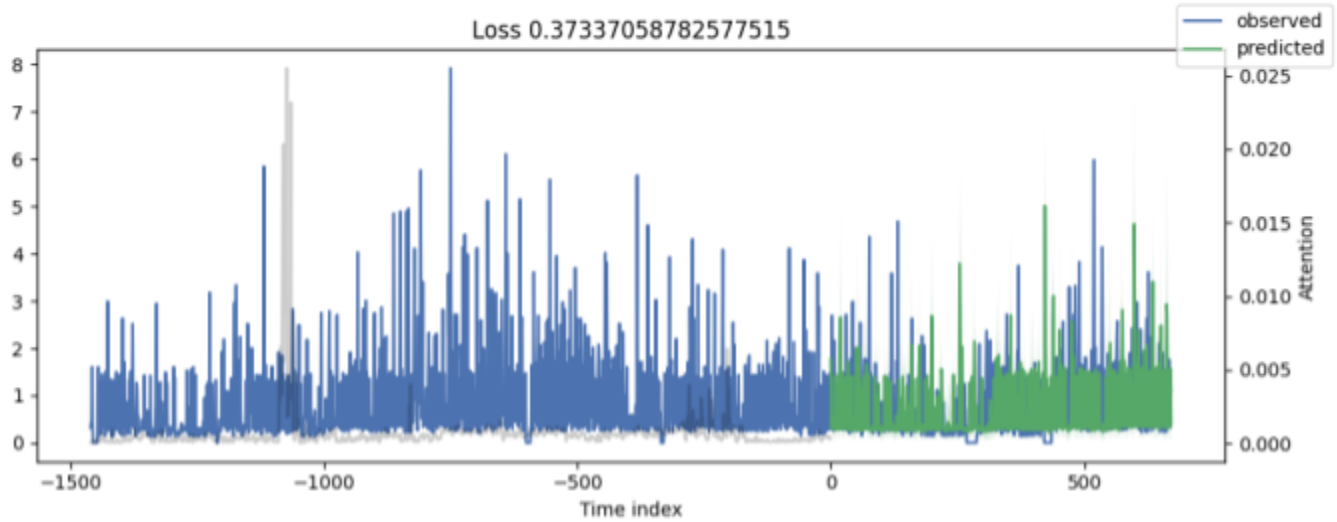


Figure 3.1 Observed vs Predicted Using four years for Hourly Predictions for a year (Grey line indicates attention score during training)

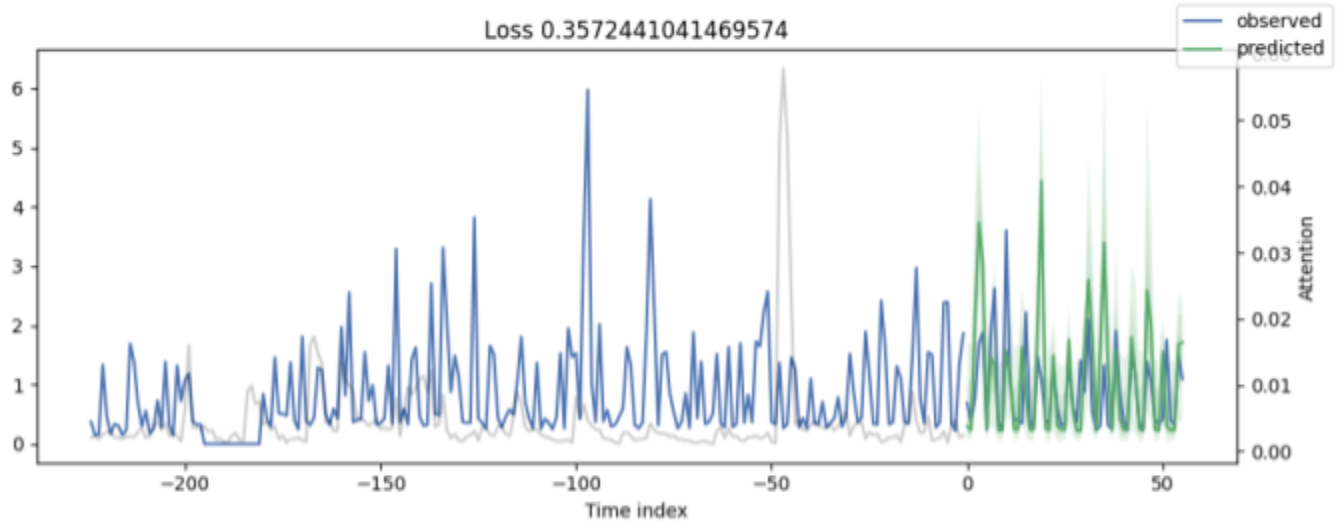


Figure 3.1 Observed vs Predicted Using one month for Hourly Predictions for a week (Grey line indicates attention score during training)

Using 'prediction' mode

Sub-Metering Levels:

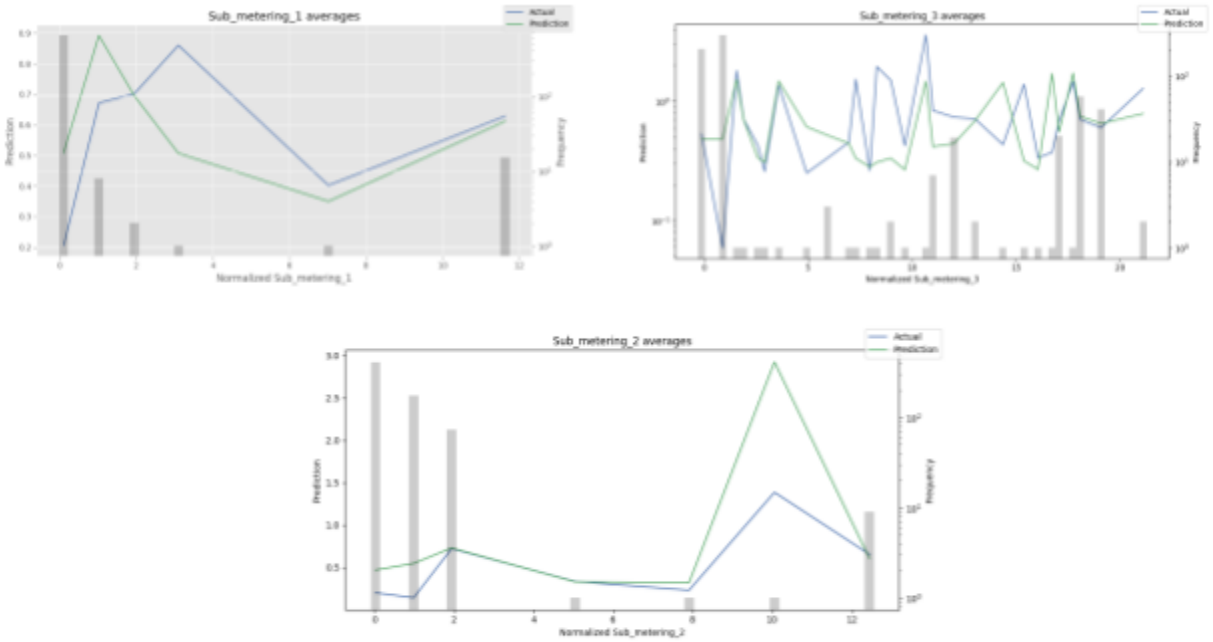


Figure: Observed vs Predicted for Sub Metering 1(top left), 3(top right), 2(bottom)

The submetering levels show similar moments in the line graph above, The magnitude of the y-axis is drastically different, however, the slope, rises and dips are well-tracked by the model.

Weather:

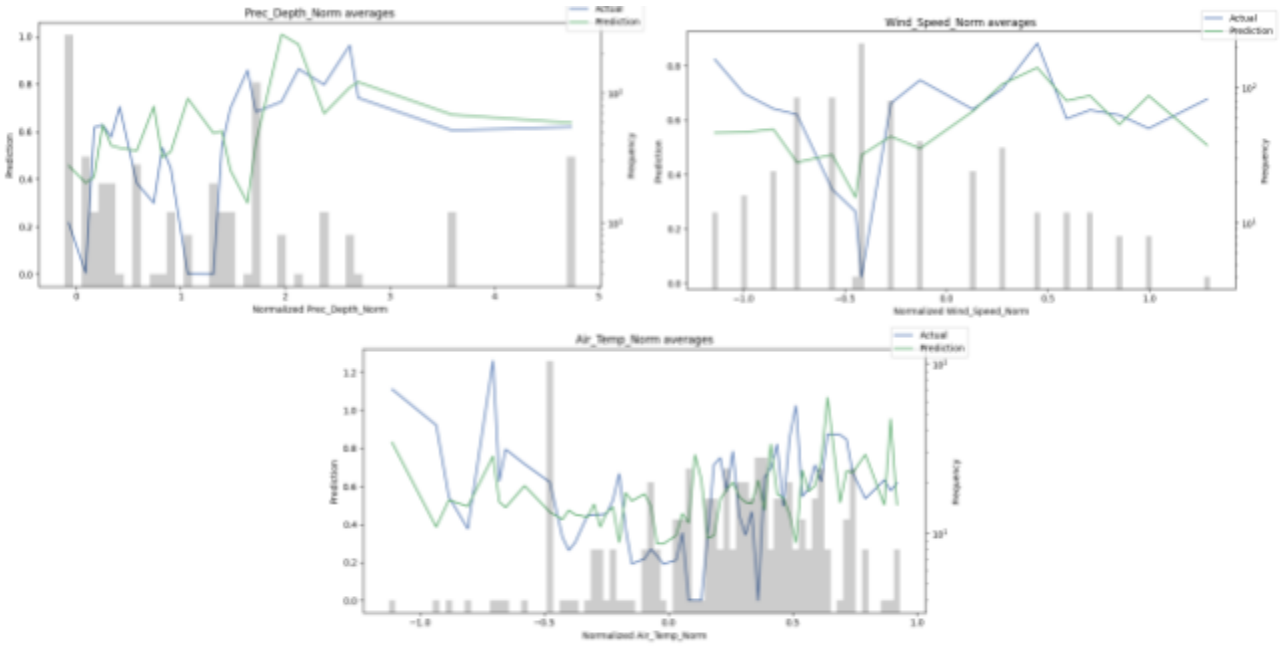


Figure: Observed vs Predicted for Precipitation(top left), Wind Speed(top right), Air Temperature(bottom)

The model fails to predict accurately the air temperature, wind speed, and precipitation rate, there may be various reasons for this including location issues of the household and the weather station. Overall, the model predicts the correct magnitude of the y-axis. The grey bar indicates the attention score by the TFT model, which is consistent as the transformer is unable to correlate the weather trend with global active power.

Google Trends:

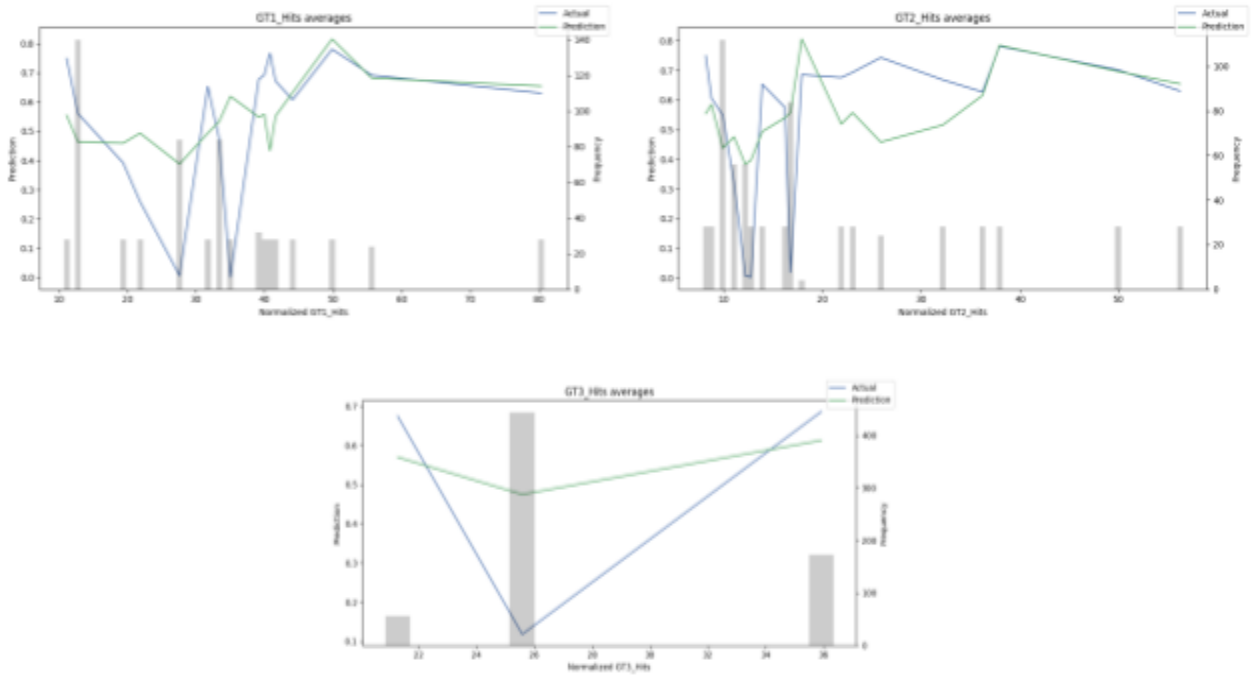


Figure: Observed vs Predicted for Google Trends 'energy saving bulb'(top left), the energy minister(top right), green energy movement (bottom)

The model can assign a high attention score during the peaks and dips of the Google trends, the model does catch up on understanding the magnitude of the trend. Thus, the model can effectively amalgamate the weekly Google Trends to predict hourly Global Active Power.

CPI and CCI:

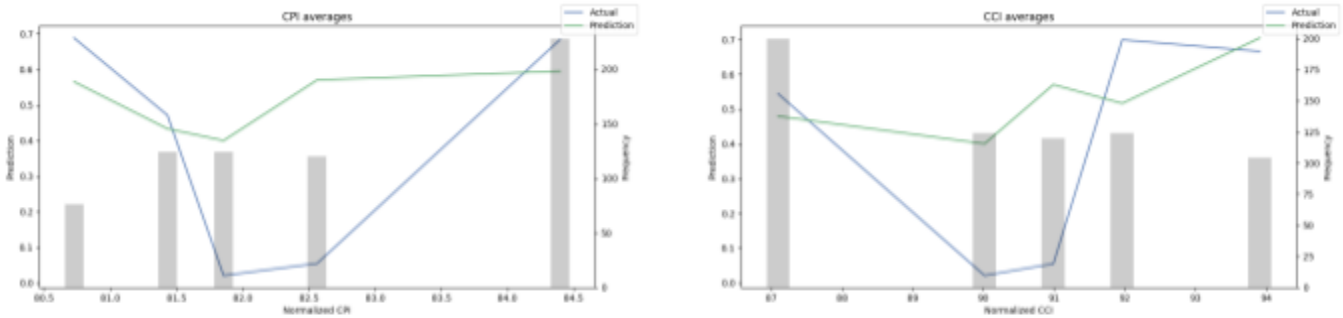


Figure: Observed vs Predicted for CPI(left), CCI(right)

Conclusion

The most significant variable in detecting the global active power for the next hour is the hour (categorical) and relative time index. Since we had considered four points of time in a day, it was found that the global active power for that particular hour is dependent on which point of time it is during the day and the relative time index i.e. the index number of the reading.

The weekly features like Google Trends have significant importance and the Temporal Fusion Transformer can utilize these to make predictions, the TFT is also able to understand the impact of these features over different granularities. Google Trend 3 has the most importance among the features. The monthly features i.e. CPI and CCI have a relatively low impact on the global active power prediction.

The three weather features i.e. precipitation rate, wind speed, and air temperature have low to no correlation and importance to the prediction of global active power. The weather features have the same hourly granularity as the global active power, however, the model does not utilize these features for prediction. A lag is also observed between the predictions and actuals for the above-mentioned three features, this may be due to relevant weather features existing at other points in time other than the ones considered. The location of the weather station might have significantly different weather than the household considered for electricity consumption.

Thus, the TFT can comprehend and predict temporal features across granularities and can utilize static, categorical, and numeric inputs. The TFT also allows for easy interpretation of the model using variable importance graphs and feature-wise prediction graphs. The TFT allows us to illustrate the attention of the model while training, this tells us more about the influential points and how the model learns and improves.

Limitations

The Temporal Fusion Transformer proved computationally expensive as the optimizing algorithm used 200 compute units on Google Collab Pro to output the hyperparameters after 5 trials.

The lag seen in weather predictions may be due to the weather station collecting data from a location away from the selected household. The lag indicates that weather follows the household after some time from the weather station. There might be a mismatch between the location of the household and the location of the weather station where both might have different weather.

Various loss functions like Multivariate distribution loss cannot be compared against the baseline, as temporal models that predict quantiles are not currently available apart from TFT. Also, the functions available for loss calculations are not robust and lack attributes to exclude and include certain conditions.

Since the TFT can handle multiple targets, there needs to be prediction and output functions designed to handle multiple normalized targets. The output functions for graphs also cannot robustly handle multiple normalized targets. There are also limitations of the Dataloader with normalizing the targets.

Future Work

The predictions of a large set of households can help determine the general power consumption habits and predict the energy needs of households during peak time, vacation, and holiday seasons.

In countries with acute electricity shortages, a power-sharing plan between industries and households can be developed based on TFT predictions. The power-sharing plan may involve a limited power supply on hours that have less usage or a day-wise sharing plan to ration the power supply.

Inconsistent power supply may cause massive economic losses and such events need to be dealt with most robustly. In France, the nuclear power plant for supplying electricity failed and power had to be bought at exorbitant prices from neighbors during the Ukro-Russo War. This import of power was unexpected and the volume to be imported was misjudged. The volume of power to be imported was revised several times and this led to additional costs. Such situations can be averted by the actual load known on different levels of granularity.

Green Energy Policy can be framed by recommending power consumption habits to households based on a shared power community plan, where the generation capacity is in sync with the capacity of electricity consumption. Since the generation of green energy is limited, a community-based power-sharing plan will help suffice energy needs through green energy entirely.

The Temporal Fusion has a variety of use cases for all kinds of datasets, like agriculture where some static and categorical variables like soil quality, soil nutrients, market prices, and weather indicators can be used to predict the ideal harvest time for crops. In manufacturing, the TFT can be used to predict real-time issues with machinery during the actual working of the machinery. Another domain where the use of TFT can prove to be useful in predicting pollution levels.

About Authors

Vedant Ghavate

Vedant has completed his Bachelor of Engineering Degree from Saviritbai Phule Pune University. He has experience working as a freelancer for the digital migration of a business, as an intern for a startup, and as a Business intelligence for a Multinational Company. Currently, he is pursuing his Masters in Big Data Analytics Degree.

Ethan Nyi Nyi

Ethan has completed his Bachelor of Business Administration with Concentrations in Information Systems and Finance from Boston University. He has experience working within the asset management sector as a data analyst, where he specialized in providing solutions for mortgage originations and credit risk modeling. Currently, he is pursuing his Master's in Big Data Analytics at San Diego State University.

References

[1] "Blog: Quantile loss function for machine learning." *Evergreen Innovations*, https://www.evergreeninnovations.co/blog-quantile-loss-function-for-machine-learning/. Accessed 19 December 2023.

[2] https://towardsdatascience.com/temporal-fusion-transformer-time-series-forecasting-with-deep-learning-complete-tutorial-d32c1e51cd91

[3] https://medium.com/@nikoskafritsas

[4] https://medium.com/@mouna.labiadh/forecasting-book-sales-with-temporal-fusion-transformer-dd482a7a257c

[5] https://www.kaggle.com/code/tomwarrens/temporal-fusion-transformer-in-pytorch

[6] https://towardsdatascience.com/temporal-fusion-transformer-googles-model-for-interpretable-time-series-forecasting-5aa17beb621

[7] https://gist.github.com/ChadFulton/82744b500a5dcb0283624c80fd10c92b

[8] https://pytorch-forecasting.readthedocs.io/en/stable/tutorials/stallion.html

[9] https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption

[10] https://www.advancinganalytics.co.uk/blog/2021/06/22/10-incredibly-useful-time-series-forecasting-algorithms

[11] https://medium.com/dataness-ai/understanding-temporal-fusion-transformer-9a7a4fcde74b

[12] https://facebook.github.io/prophet/

[13] https://weather.com/weather/today/l/Sceaux+Hauts+de+Seine+France?canonicalCityId=b6cc512ee717e71314095f4f1914180576f96983cc60638f6ccb3c55ad62b651

[14] https://www.kaggle.com/code/tomwarrens/temporal-fusion-transformer-in-pytorch

[15] https://machinelearningmastery.com/how-to-develop-lstm-models-for-multi-step-time-series-forecasting-of-household-power-consumption/

[16] https://en.wikipedia.org/wiki/Spline_interpolation

[17] https://towardsdatascience.com/multi-seasonal-time-series-decomposition-using-mstl-in-python-136630e67530

[18] https://arxiv.org/pdf/2107.13462.pdf

[19] https://towardsdatascience.com/multi-seasonal-time-series-decomposition-using-mstl-in-python-136630e67530

[20] https://towardsdatascience.com/temporal-fusion-transformer-time-series-forecasting-with-deep-learning-complete-tutorial-d32c1e51cd91

[21] https://kunalbharadkar.medium.com/advanced-techniques-for-complex-time-series-analysis-15cac7ba945

[22] https://www.youtube.com/watch?v=RZQJk0hf3sI&t=203s

[23] https://www.sciencedirect.com/science/article/pii/S0169207021000637

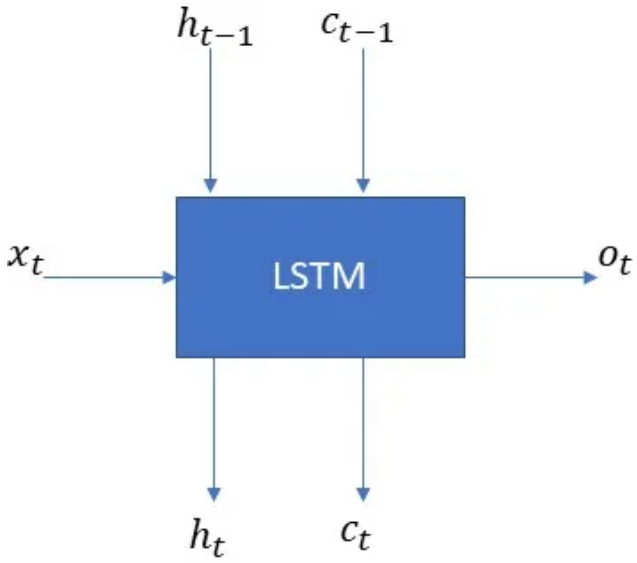
[24] https://arxiv.org/pdf/1912.09363.pdf

[25] https://neptune.ai/blog/arima-vs-prophet-vs-lstm

[26] https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e

Appendix A

- The power consumption dataset has 25k nan/null values which are interpolated, these null/0.0 values may be due to power failure on the domestic household level such as MCB tripping and overload on the fuse.
- Since the dataset includes data for only one household, we have to take into account the specific power consumption habits of the residents of that particular household (spending weekends away from home, annual vacation).
- Min Max normalization is used for the stacked bar graph
- Standard scalar and robust scalar are used for the overlay chart
- Date start and end are from 1/1/2007 to 11/26/2010 since power consumption data is available only for those values.
- Spline interpolation is used to fill in missing values



σ_g : $sigmoid$
σ_c : tanh
. : element wise multiplication

$$f_t = \sigma_g\,(W_f \times x_t + U_f \times h_{t-1} + b_f)$$
$$i_t = \sigma_g\,(W_i \times x_t + U_i \times h_{t-1} + b_i)$$
$$o_t = \sigma_g\,(W_o \times x_t + U_o \times h_{t-1} + b_o)$$
$$c'_t = \sigma_c\,(W_c \times x_t + U_c \times h_{t-1} + b_c)$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t$$
$$h_t = o_t \cdot \sigma_c(c_t)$$

f_t *is the forget gate*
i_t *is the input gate*
o_t *is the output gate*
c_t *is the cell state*
h_t *is the hidden state*

Shown above: The figure shows the input and outputs of an LSTM cell for a single timestep.

Appendix B

Data Preprocessing Notebook

Exploratory Data Analysis (Overlay Chart) Notebook

Exploratory Data Analysis (MSTL) Notebook

Training and Testing TFT Model Notebook

GitHub Link:

https://github.com/vedantghavate259/TFT

In [45]:
```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import register_matplotlib_converters
from statsmodels.tsa.seasonal import MSTL
from statsmodels.tsa.seasonal import DecomposeResult
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.seasonal import STL
from sklearn.preprocessing import StandardScaler, RobustScaler
import seaborn as sns
import numpy as np
import matplotlib.dates as mdate

register_matplotlib_converters()
sns.set_style("darkgrid")


# Replace 'your_file.csv' with the path to your CSV file.
file_path = '/Users/ethan/Downloads/UCI_Power_Consumption_Dataset.txt'

# Read the CSV file into a pandas DataFrame.
df = pd.read_csv(file_path, sep=';')
```

```
/var/folders/pj/894szprx04q1zqpqhy4bgvqr0000gn/T/ipykernel_46678/1232310424.py:21: DtypeWarning: Columns (2,3,4,5,6,7) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv(file_path, sep=';')
```

In [46]:
```
file_path = '/Users/ethan/Downloads/UCI_Power_Consumption_Dataset.txt'
df = pd.read_csv(file_path, sep=';')
```

```
/var/folders/pj/894szprx04q1zqpqhy4bgvqr0000gn/T/ipykernel_46678/2128151062.py:2: DtypeWarning: Columns (2,3,4,5,6,7) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv(file_path, sep=';')
```

In [47]:
```
df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y')
```

In [48]:
```
# Remove the first 3 rows.
df = df[df['Date'] >= '2007-01-01']

# Save the modified DataFrame back to a new CSV file.
df.to_csv('/Users/ethan/Downloads/UCI_Cleaned_Power_Consumption.csv', index=False)
```

In [49]:
```
ISD = pd.read_csv('/Users/ethan/Downloads/ISD_Data.csv')
CPI = pd.read_csv('/Users/ethan/Downloads/France_CPI_N.csv')
CCI = pd.read_csv('/Users/ethan/Downloads/CCI_monthly_values (4).csv', sep=';')
CCI = CCI.loc[3:]
trend1 = pd.read_csv('/Users/ethan/Downloads/YMW_Clean_GoogleTrends1.csv')
trend2 = pd.read_csv('/Users/ethan/Downloads/YMW_Clean_GoogleTrends2.csv')
trend3 = pd.read_csv('/Users/ethan/Downloads/YMW_Clean_GoogleTrends3.csv')
```

In [50]:
```
trend1['Date'] = pd.to_datetime(trend1['Date'], format='%Y-%m-%d')
trend2['Date'] = pd.to_datetime(trend2['Date'], format='%Y-%m-%d')
trend3['Date'] = pd.to_datetime(trend3['Date'], format='%Y-%m-%d')
trend1.set_index('Date')
trend2.set_index('Date')
trend3.set_index('Date')

trend1_agg = trend1.groupby(trend1['Date'].dt.to_period('M'))['Hits'].mean()
trend1_agg = pd.DataFrame(trend1_agg)
trend2_agg = trend2.groupby(trend2['Date'].dt.to_period('M'))['Hits'].mean()
trend2_agg = pd.DataFrame(trend2_agg)
trend3_agg = trend3.groupby(trend3['Date'].dt.to_period('M'))['Hits'].mean()
trend3_agg = pd.DataFrame(trend3_agg)
```

In [51]:
```
print(trend1_agg)
```

```
              Hits
Date
2007-01   5.250000
2007-02  13.750000
2007-03  13.750000
2007-04   5.000000
2007-05  10.750000
2007-06   7.250000
2007-07  13.600000
2007-08   0.000000
2007-09  12.800000
2007-10  26.000000
2007-11  63.000000
2007-12  19.400000
2008-01  20.250000
2008-02  14.250000
2008-03  23.600000
2008-04  17.000000
2008-05   5.250000
2008-06   0.000000
2008-07  19.000000
2008-08  16.200000
2008-09  44.750000
2008-10  33.250000
2008-11  49.800000
2008-12  58.750000
2009-01  49.500000
2009-02  35.250000
2009-03  35.200000
2009-04  25.250000
2009-05  60.000000
2009-06  40.250000
2009-07  48.000000
2009-08  61.000000
2009-09  50.250000
2009-10  69.250000
2009-11  67.200000
2009-12  71.500000
2010-01  58.200000
2010-02  48.750000
2010-03  36.250000
2010-04  27.000000
2010-05  33.800000
2010-06  19.500000
2010-07   6.000000
2010-08  36.000000
2010-09  31.500000
2010-10  51.800000
2010-11  48.333333
```

In [52]:
```
ISD[['Air_Temp', 'Air_Temp_QualityCheck']] = ISD['TMP'].str.split(",", expand = True)
ISD[['Prec_Period', 'Prec_Depth', 'Prec_Cond', 'Prec_Qual']] = ISD['AA3'].str.split(",", expand = True)
ISD[['Wind_Angle', 'Wind_DirQuality', 'Wind_Type', 'Wind_Speed', 'Wind_SpeedQual']] = ISD['WND'].str.split(",", expand = True)
```

In [53]:
```
ISD['Air_Temp'] = ISD['Air_Temp'].str.replace('+', '').astype(int)
```

In [54]:
```
ISD['Air_Temp'] = (ISD['Air_Temp']/10)
ISD['Air_Temp'].round(1)
ISD['Wind_Speed'] = ISD['Wind_Speed'].astype(int)
ISD['Wind_Speed'] = (ISD['Wind_Speed']/10)
```

In [55]:
```
df['Date'] = df['Date'].astype('str')
ISD['New_Date']= pd.to_datetime(ISD['DATE'])
ISD.set_index('New_Date')
ISD.sort_values(by='New_Date', ascending = True)
ISD['Prec_Depth'] = ISD['Prec_Depth'].str.replace('9999', '0000')
ISD['Prec_Depth']= ISD['Prec_Depth'].ffill()
df['New_Date'] = pd.to_datetime(df['Date'])
df['New_Datetime'] = pd.to_datetime(df['Date'] + ' ' + df['Time'])
CCI['New_Date'] = pd.to_datetime(CCI['Label'], format='%Y-%m')
CPI['New_Date'] = pd.to_datetime(CPI['DATE'], format='%d/%m/%Y')
```

In [56]:
```
ISD.head()
```

Out[56]:

	STATION	DATE	SOURCE	LATITUDE	LONGITUDE	ELEVATION	NAME	REPORT_TYPE	CALL_SIGN	QUALITY_CONTROL	...	Prec_Period	Prec_Depth	Prec_Cond	Prec_Qual	Wind_Angle	Wind_DirQuality	Wind_Type	Wind_Speed	Wind_SpeedQual	New_Date
0	7005099999	2010-12-31T23:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	LFOI	V020	...	NaN	NaN	NaN	NaN	120	1	N	1.5	1	2010-12-31 23:00:00
1	7005099999	2010-12-31T22:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	LFOI	V020	...	NaN	NaN	NaN	NaN	150	1	N	1.0	1	2010-12-31 22:00:00
2	7005099999	2010-12-31T21:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	LFOI	V020	...	NaN	NaN	NaN	NaN	150	1	N	2.1	1	2010-12-31 21:00:00
3	7005099999	2010-12-31T20:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	LFOI	V020	...	NaN	NaN	NaN	NaN	130	1	N	1.5	1	2010-12-31 20:00:00
4	7005099999	2010-12-31T19:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	LFOI	V020	...	NaN	NaN	NaN	NaN	140	1	N	2.6	1	2010-12-31 19:00:00

5 rows × 54 columns

In [57]:
```
def mean_normalized(df, column):
    df[column] = pd.to_numeric(df[column], errors='coerce')
    SC = RobustScaler()
    df_column_scaled = SC.fit_transform(df[[column]])
    return df_column_scaled
```

In [58]:
```
ISD = ISD[((ISD['New_Date'] >= '2007-01-01') & (ISD['New_Date'] <= '2010-11-01'))]
df = df[((df['New_Date'] >= '2007-01-01') & (df['New_Date'] <= '2010-11-01'))]
CCI = CCI[((CCI['New_Date'] >= '2007-01-01') & (CCI['New_Date'] <= '2010-11-01'))]
CPI = CPI[((CPI['New_Date'] >= '2007-01-01') & (CPI['New_Date'] <= '2010-11-01'))]
CCI.set_index('New_Date')
```

Out[58]:

New_Date	Label	Monthly consumer confidence survey	Codes
2007-01-01	2007-01	100	A
2007-02-01	2007-02	100	A
2007-03-01	2007-03	101	A
2007-04-01	2007-04	102	A
2007-05-01	2007-05	108	A
2007-06-01	2007-06	109	A
2007-07-01	2007-07	107	A
2007-08-01	2007-08	104	A
2007-09-01	2007-09	101	A
2007-10-01	2007-10	100	A
2007-11-01	2007-11	95	A
2007-12-01	2007-12	94	A
2008-01-01	2008-01	90	A
2008-02-01	2008-02	90	A
2008-03-01	2008-03	89	A
2008-04-01	2008-04	89	A
2008-05-01	2008-05	86	A
2008-06-01	2008-06	82	A
2008-07-01	2008-07	82	A
2008-08-01	2008-08	82	A
2008-09-01	2008-09	84	A
2008-10-01	2008-10	82	A
2008-11-01	2008-11	83	A
2008-12-01	2008-12	83	A
2009-01-01	2009-01	84	A
2009-02-01	2009-02	84	A
2009-03-01	2009-03	84	A
2009-04-01	2009-04	86	A
2009-05-01	2009-05	87	A
2009-06-01	2009-06	89	A
2009-07-01	2009-07	89	A
2009-08-01	2009-08	89	A
2009-09-01	2009-09	90	A
2009-10-01	2009-10	91	A
2009-11-01	2009-11	94	A
2009-12-01	2009-12	94	A
2010-01-01	2010-01	95	A
2010-02-01	2010-02	92	A
2010-03-01	2010-03	91	A
2010-04-01	2010-04	89	A
2010-05-01	2010-05	88	A
2010-06-01	2010-06	87	A
2010-07-01	2010-07	87	A
2010-08-01	2010-08	90	A
2010-09-01	2010-09	91	A
2010-10-01	2010-10	92	A
2010-11-01	2010-11	94	A

In [59]:
```
df['Global_active_power'] = pd.to_numeric(df['Global_active_power'], errors='coerce')
```

In [60]:
```python
trend1_agg['Hits_Norm'] = mean_normalized(trend1_agg,'Hits')
trend2_agg['Hits_Norm'] = mean_normalized(trend2_agg,'Hits')
trend3_agg['Hits_Norm'] = mean_normalized(trend3_agg,'Hits')
ISD['Air_Temp_Norm'] = mean_normalized(ISD,'Air_Temp')
df['Global_active_power_Norm'] = mean_normalized(df, 'Global_active_power')
df['Sub_metering_1'] = mean_normalized(df, 'Sub_metering_1')
df['Sub_metering_2'] = mean_normalized(df, 'Sub_metering_2')
df['Sub_metering_3'] = mean_normalized(df, 'Sub_metering_3')
CCI['Monthly consumer confidence survey_Norm'] = mean_normalized(CCI, "Monthly consumer confidence survey")
CPI['CPI_Score_Norm'] = mean_normalized(CPI, 'CPI_Score')
ISD['Wind_Speed_Norm'] = mean_normalized(ISD, 'Wind_Speed')
ISD['Prec_Depth_Norm'] = mean_normalized(ISD, 'Prec_Depth')
```

```
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

In [61]:
```python
ISD['Air_Temp_Norm'].describe()
# Identify outliers based on a threshold (e.g., 100)
outlier_threshold = 100
filter = ISD['Air_Temp_Norm'] > outlier_threshold

# Remove outliers
ISD.loc[filter, 'Air_Temp_Norm'] = np.nan

ISD['Prec_Depth_Norm'].describe()
outlier_threshold = 200
filter = ISD['Prec_Depth_Norm'] > outlier_threshold
ISD.loc[filter, 'Prec_Depth_Norm'] = np.nan

ISD['Wind_Speed_Norm'].describe()
outlier_threshold = 200
filter = ISD['Wind_Speed_Norm'] > outlier_threshold
ISD.loc[filter, 'Wind_Speed_Norm'] = np.nan
```

In [62]:
```
df['Global_active_power_Norm'] = df['Global_active_power_Norm'].interpolate(option='spline')
ISD['Air_Temp_Norm'] = ISD['Air_Temp_Norm'].interpolate(option='spline')
ISD['Prec_Depth_Norm'] = ISD['Prec_Depth_Norm'].interpolate(option='spline')
ISD['Wind_Speed_Norm'] = ISD['Wind_Speed_Norm'].interpolate(option='spline')
ISD_agg = ISD.groupby(ISD['New_Date'].dt.to_period('M'))['Air_Temp_Norm'].mean()
df_agg = df.groupby(df['New_Date'].dt.to_period('M'))['Global_active_power_Norm'].mean()
df['Sub_metering_1'] = df['Sub_metering_1'].interpolate(option = 'spline')
df['Sub_metering_2'] = df['Sub_metering_2'].interpolate(option = 'spline')
df['Sub_metering_3'] = df['Sub_metering_3'].interpolate(option = 'spline')
df_agg_sub1 = df.groupby(df['New_Date'].dt.to_period('M'))['Sub_metering_1'].mean()
df_agg_sub2 = df.groupby(df['New_Date'].dt.to_period('M'))['Sub_metering_2'].mean()
df_agg_sub3 = df.groupby(df['New_Date'].dt.to_period('M'))['Sub_metering_3'].mean()
Wind_Speed_Agg = ISD.groupby(ISD['New_Date'].dt.to_period('M'))['Wind_Speed_Norm'].mean()
Prec_Depth_Agg = ISD.groupby(ISD['New_Date'].dt.to_period('M'))['Prec_Depth_Norm'].mean()
```

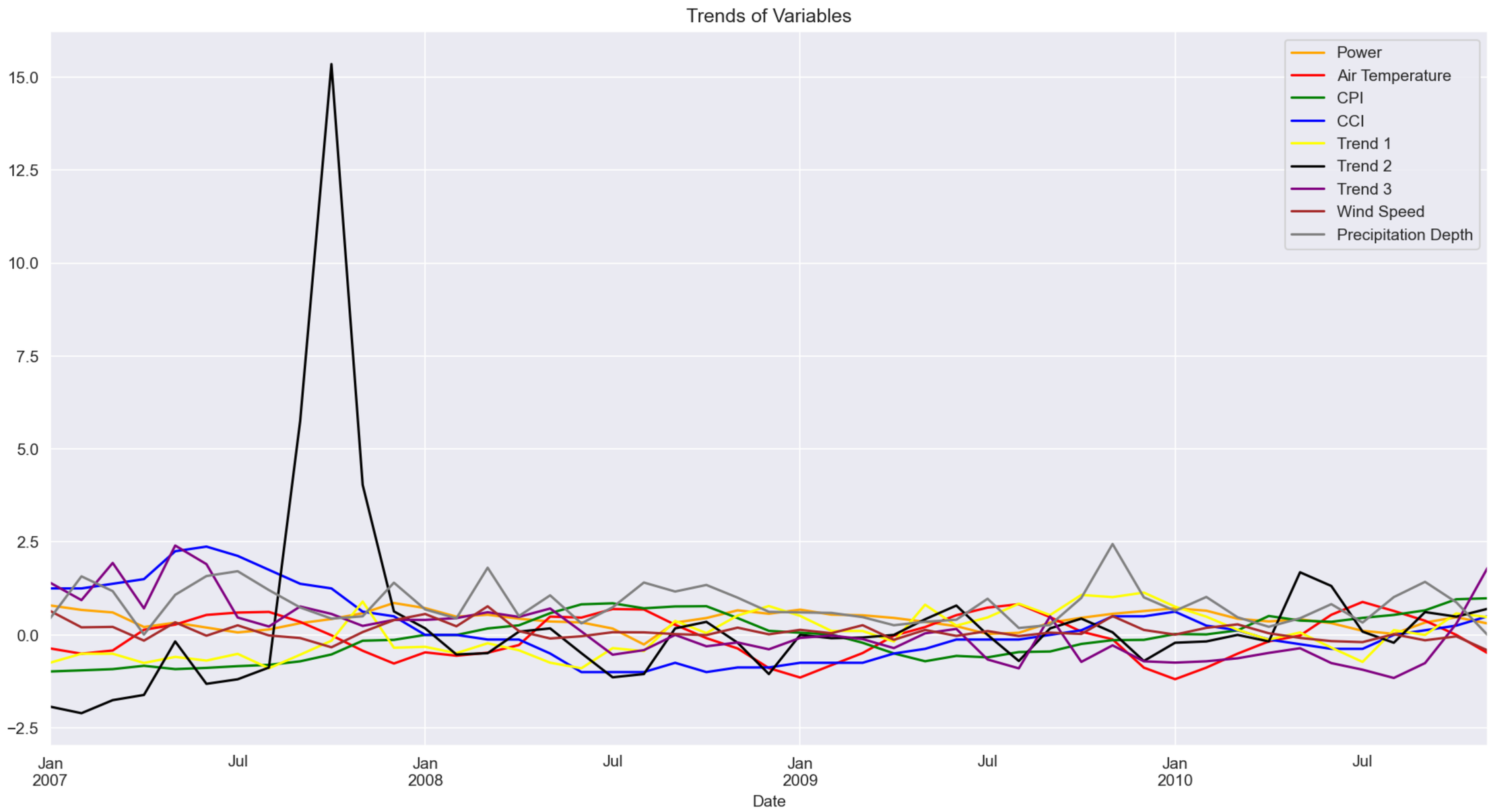
In [63]:
```
ISD_agg = pd.DataFrame(ISD_agg)
df_agg = pd.DataFrame(df_agg)
df_agg_sub1 = pd.DataFrame(df_agg_sub1)
df_agg_sub2 = pd.DataFrame(df_agg_sub2)
df_agg_sub3 = pd.DataFrame(df_agg_sub3)
prec_depth = pd.DataFrame()
Wind_Speed_Agg = pd.DataFrame(Wind_Speed_Agg)
Prec_Depth_Agg = pd.DataFrame(Prec_Depth_Agg)
```

In [64]:
```
CCI = CCI.set_index('New_Date')
CPI = CPI.set_index('New_Date')
Wind_Speed_Agg.index
Prec_Depth_Agg.index
df_agg.index
ISD_agg.index
CCI.index
CPI.index
print(trend1)
print(trend2)
```

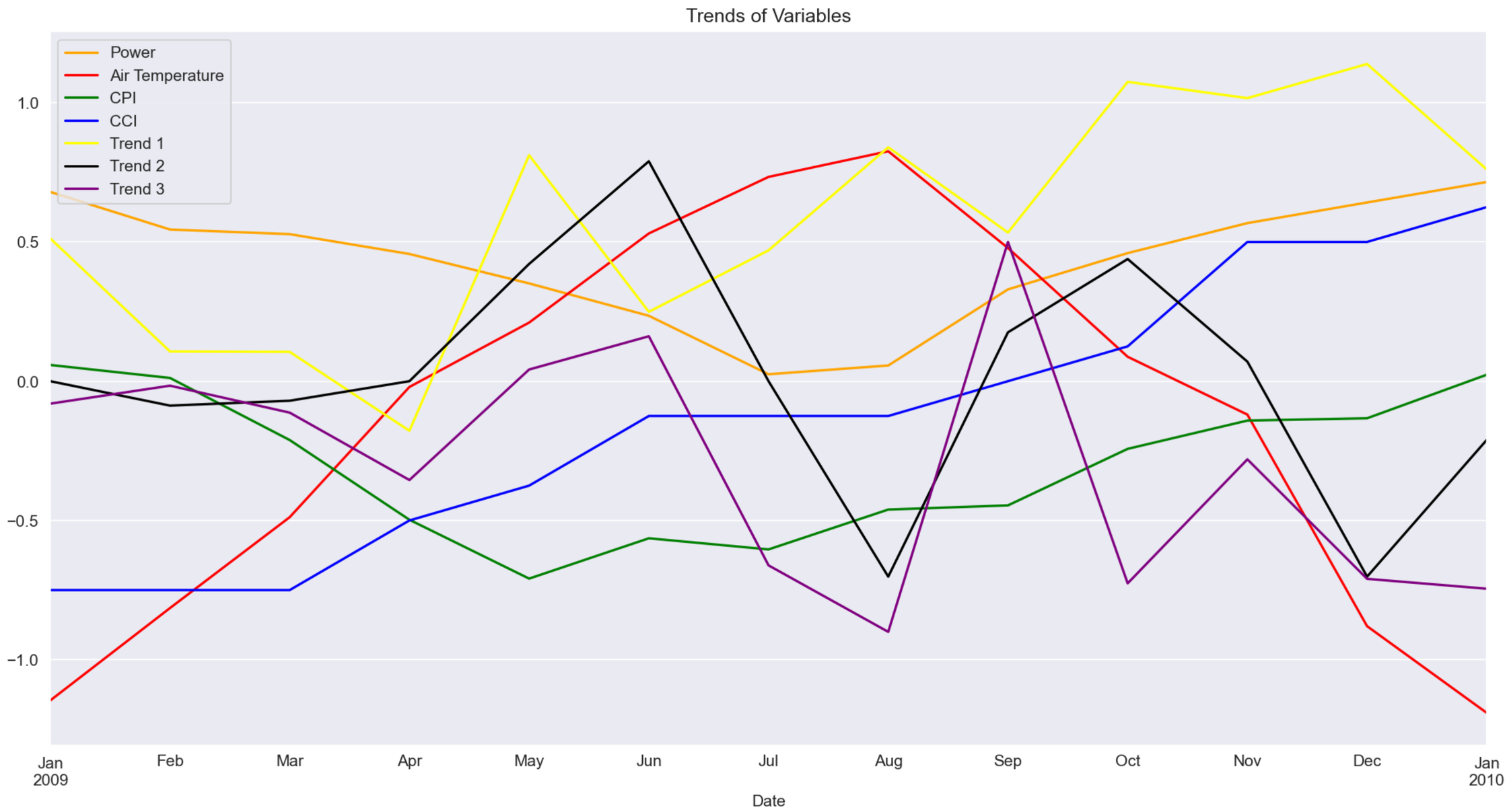
```
          Date  Hits
0   2007-01-07    21
1   2007-01-14     0
2   2007-01-21     0
3   2007-01-28     0
4   2007-02-04    15
..         ...   ...
198 2010-10-24    63
199 2010-10-31    60
200 2010-11-07    57
201 2010-11-14    52
202 2010-11-21    36

[203 rows x 2 columns]
          Date  Hits
0   2007-01-07     3
1   2007-01-14     1
2   2007-01-21     2
3   2007-01-28     4
4   2007-02-04     1
..         ...   ...
198 2010-10-24     8
199 2010-10-31    10
200 2010-11-07     9
201 2010-11-14    10
202 2010-11-21    11

[203 rows x 2 columns]
```

In [65]:
```
joined_table = df_agg.join(other = [ISD_agg, CCI, CPI])
```

In [66]:
```
plt.figure(figsize=(16, 8), dpi=150)
df_agg['Global_active_power_Norm'].plot(label='Power', color='orange')
ISD_agg['Air_Temp_Norm'].plot(label='Air Temperature', color = 'red')
CPI['CPI_Score_Norm'].plot(label='CPI', color= 'green')
CCI['Monthly consumer confidence survey_Norm'].plot(label='CCI', color = 'blue')
trend1_agg['Hits_Norm'].plot(label = 'Trend 1', color = 'yellow')
trend2_agg['Hits_Norm'].plot(label = 'Trend 2', color = 'black')
trend3_agg['Hits_Norm'].plot(label = 'Trend 3', color = 'purple')
Wind_Speed_Agg['Wind_Speed_Norm'].plot(label = 'Wind Speed', color = 'brown')
Prec_Depth_Agg['Prec_Depth_Norm'].plot(label = 'Precipitation Depth', color = 'gray')
plt.title('Trends of Variables')

# adding Label to the x-axis
plt.xlabel('Date')

# adding legend to the curve
plt.legend()
plt.show()
```



In [67]:
```
plt.figure(figsize=(16, 8), dpi=150)
df_agg.loc['2009-01-01':'2010-01-01', 'Global_active_power_Norm'].plot(label='Power', color='orange')
ISD_agg.loc['2009-01-01':'2010-01-01', 'Air_Temp_Norm'].plot(label='Air Temperature', color='red')
CPI.loc['2009-01-01':'2010-01-01', 'CPI_Score_Norm'].plot(label='CPI', color= 'green')
CCI.loc['2009-01-01':'2010-01-01', 'Monthly consumer confidence survey_Norm'].plot(label='CCI', color = 'blue')
trend1_agg.loc['2009-01-01':'2010-01-01', 'Hits_Norm'].plot(label = 'Trend 1', color = 'yellow')
trend2_agg.loc['2009-01-01':'2010-01-01', 'Hits_Norm'].plot(label = 'Trend 2', color = 'black')
trend3_agg.loc['2009-01-01':'2010-01-01', 'Hits_Norm'].plot(label = 'Trend 3', color = 'purple')
plt.title('Trends of Variables')

# adding Label to the x-axis
plt.xlabel('Date')

# adding legend to the curve
plt.legend()
plt.show()
```



In [68]:
```
df_null = df.isna().sum()
print(df_null)
df.set_index('New_Datetime')
```

```
Date                          0
Time                          0
Global_active_power       25975
Global_reactive_power         0
Voltage                       0
Global_intensity              0
Sub_metering_1                0
Sub_metering_2                0
Sub_metering_3                0
New_Date                      0
New_Datetime                  0
Global_active_power_Norm      0
dtype: int64
```

Out[68]:

New_Datetime	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	New_Date	Global_active_power_Norm
2007-01-01 00:00:00	2007-01-01	00:00:00	2.580	0.136	241.970	10.600	0.0	0.0	-0.058824	2007-01-01	1.646281
2007-01-01 00:01:00	2007-01-01	00:01:00	2.552	0.100	241.750	10.400	0.0	0.0	-0.058824	2007-01-01	1.623140
2007-01-01 00:02:00	2007-01-01	00:02:00	2.550	0.100	241.640	10.400	0.0	0.0	-0.058824	2007-01-01	1.621488
2007-01-01 00:03:00	2007-01-01	00:03:00	2.550	0.100	241.710	10.400	0.0	0.0	-0.058824	2007-01-01	1.621488
2007-01-01 00:04:00	2007-01-01	00:04:00	2.554	0.100	241.980	10.400	0.0	0.0	-0.058824	2007-01-01	1.624793
...	...	...	...	...	...	...	...	...	...	...	...
2010-11-01 23:55:00	2010-11-01	23:55:00	1.780	0.076	249.52	7.2	0.0	0.0	0.000000	2010-11-01	0.985124
2010-11-01 23:56:00	2010-11-01	23:56:00	1.782	0.074	249.49	7.2	0.0	0.0	0.000000	2010-11-01	0.986777
2010-11-01 23:57:00	2010-11-01	23:57:00	1.530	0.07	249.45	6.2	0.0	0.0	-0.058824	2010-11-01	0.778512
2010-11-01 23:58:00	2010-11-01	23:58:00	1.280	0.068	249.91	5.2	0.0	0.0	0.000000	2010-11-01	0.571901
2010-11-01 23:59:00	2010-11-01	23:59:00	1.270	0.07	249.27	5.2	0.0	0.0	0.000000	2010-11-01	0.563636

2017440 rows × 11 columns

In [69]:
```
df1 = df.resample('H', on='New_Datetime').Global_active_power_Norm.mean()
df1 = df1.reset_index()
df_null = df1.isna().sum()
print(df_null)
df1.set_index('New_Datetime')
```

```
New_Datetime                0
Global_active_power_Norm    0
dtype: int64
```

Out[69]:

New_Datetime	Global_active_power_Norm
2007-01-01 00:00:00	1.622011
2007-01-01 01:00:00	1.599504
2007-01-01 02:00:00	1.648209
2007-01-01 03:00:00	1.614601
2007-01-01 04:00:00	1.560110
...	...
2010-11-01 19:00:00	-0.211625
2010-11-01 20:00:00	-0.093388
2010-11-01 21:00:00	0.396474
2010-11-01 22:00:00	0.662534
2010-11-01 23:00:00	0.731928

33624 rows × 1 columns

In [70]:
```
df_sub1 = df.resample('H', on = 'New_Datetime').Sub_metering_1.mean()
df_sub1 = df_sub1.reset_index()
df_null = df_sub1.isna().sum()
print(df_null)
df_sub1.set_index('New_Datetime')
```

```
New_Datetime      0
Sub_metering_1    0
dtype: int64
```

Out[70]:

New_Datetime	Sub_metering_1
2007-01-01 00:00:00	0.0
2007-01-01 01:00:00	0.0
2007-01-01 02:00:00	0.0
2007-01-01 03:00:00	0.0
2007-01-01 04:00:00	0.0
...	...
2010-11-01 19:00:00	0.0
2010-11-01 20:00:00	0.0
2010-11-01 21:00:00	0.0
2010-11-01 22:00:00	0.0
2010-11-01 23:00:00	0.0

33624 rows × 1 columns

In [71]:
```
df_sub2 = df.resample('H', on = 'New_Datetime').Sub_metering_2.mean()
df_sub2 = df_sub2.reset_index()
df_null = df_sub2.isna().sum()
print(df_null)
df_sub2.set_index('New_Datetime')
```

```
New_Datetime      0
Sub_metering_2    0
dtype: int64
```

Out[71]:

New_Datetime	Sub_metering_2
2007-01-01 00:00:00	0.583333
2007-01-01 01:00:00	0.000000
2007-01-01 02:00:00	0.333333
2007-01-01 03:00:00	0.266667
2007-01-01 04:00:00	0.000000
...	...
2010-11-01 19:00:00	0.000000
2010-11-01 20:00:00	0.350000
2010-11-01 21:00:00	1.200000
2010-11-01 22:00:00	0.466667
2010-11-01 23:00:00	0.000000

33624 rows × 1 columns

In [72]:
```
df_sub3 = df.resample('H', on = 'New_Datetime').Sub_metering_3.mean()
df_sub3 = df_sub3.reset_index()
df_null = df_sub3.isna().sum()
print(df_null)
df_sub3.set_index('New_Datetime')
```

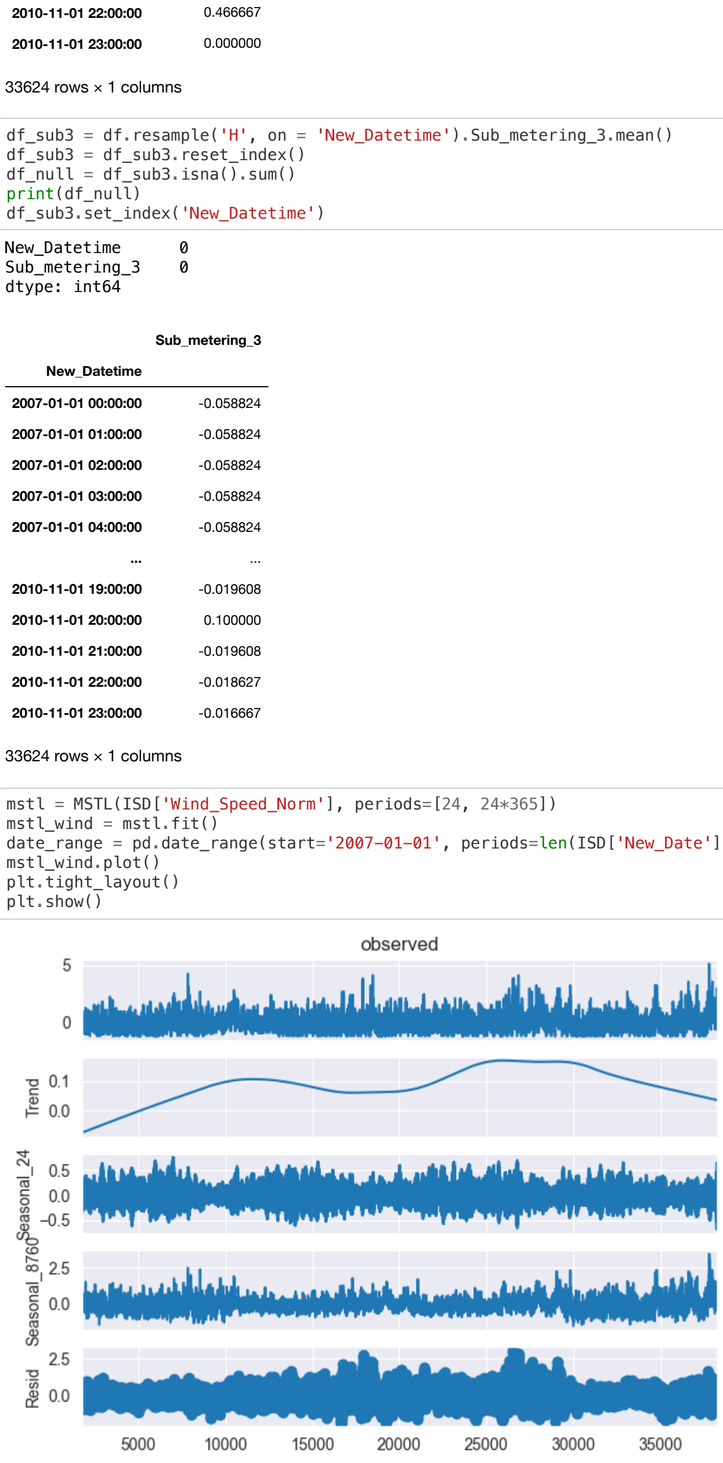
```
New_Datetime      0
Sub_metering_3    0
dtype: int64
```

Out[72]:

New_Datetime	Sub_metering_3
2007-01-01 00:00:00	-0.058824
2007-01-01 01:00:00	-0.058824
2007-01-01 02:00:00	-0.058824
2007-01-01 03:00:00	-0.058824
2007-01-01 04:00:00	-0.058824
...	...
2010-11-01 19:00:00	-0.019608
2010-11-01 20:00:00	0.100000
2010-11-01 21:00:00	-0.019608
2010-11-01 22:00:00	-0.018627
2010-11-01 23:00:00	-0.016667

33624 rows × 1 columns

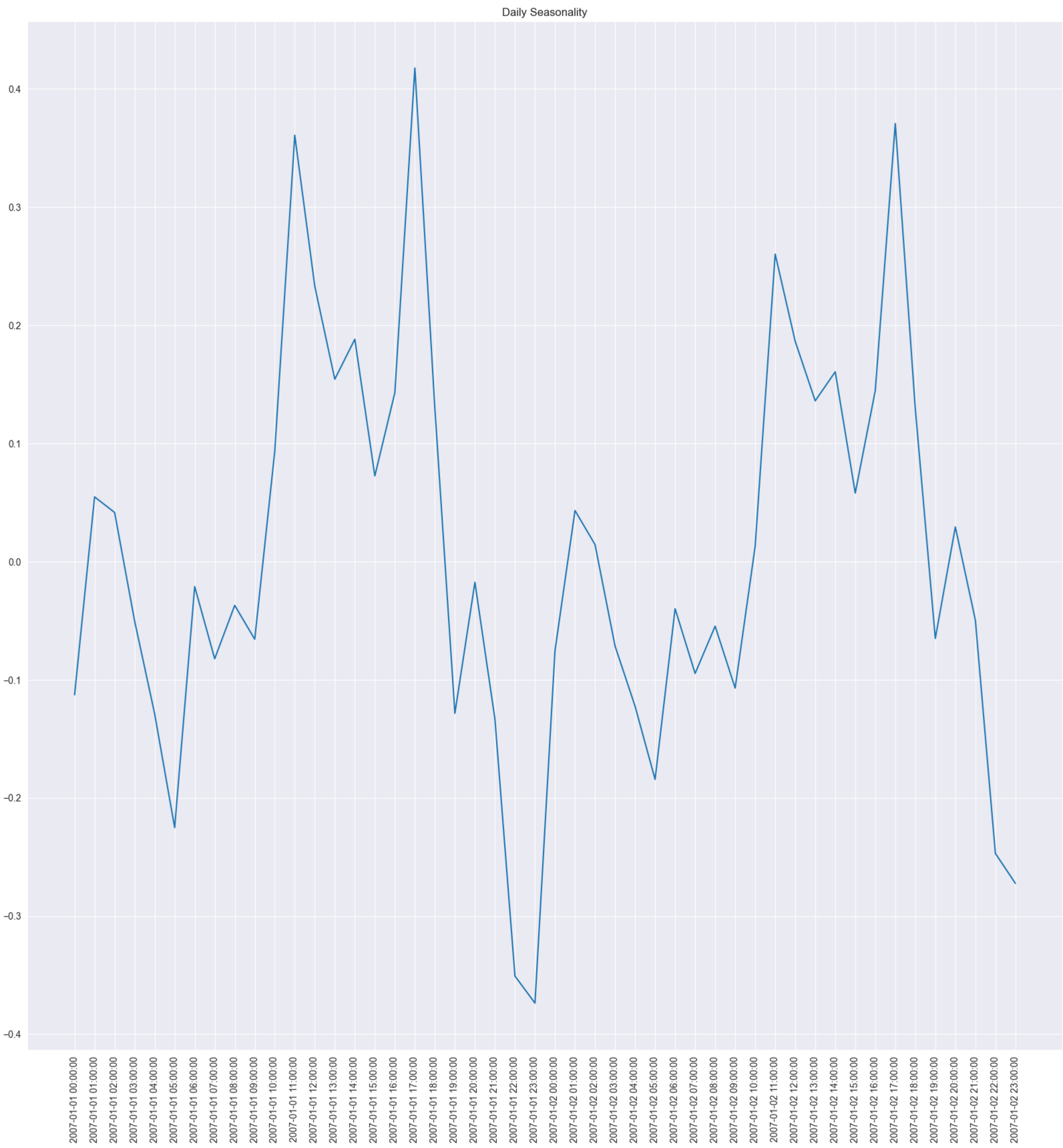
In [73]:
```
mstl = MSTL(ISD['Wind_Speed_Norm'], periods=[24, 24*365])
mstl_wind = mstl.fit()
date_range = pd.date_range(start='2007-01-01', periods=len(ISD['New_Date']), freq='H')
mstl_wind.plot()
plt.tight_layout()
plt.show()
```



In [74]:
```
mstl_wind.observed
```

Out[74]:
```
1758    -0.416667
1759     0.000000
1760    -0.416667
1761    -0.416667
1762    -0.138889
           ...
38234    2.444444
38235    2.138889
38236    2.138889
38237    2.138889
38238    1.583333
Name: observed, Length: 36481, dtype: float64
```

In [75]:
```
wind_24 = mstl_wind._seasonal.seasonal_24[:48]
fig, ax = plt.subplots(figsize=(20, 20))
wind_24.plot(title='Daily Seasonality')
date_range_48h = pd.date_range(start='2007-01-01', periods=48, freq='H')
plt.xticks(ticks=range(1758, 1806), labels=date_range_48h, rotation = 90)
plt.show()
```



In [76]:
```
wind_24.head()
```

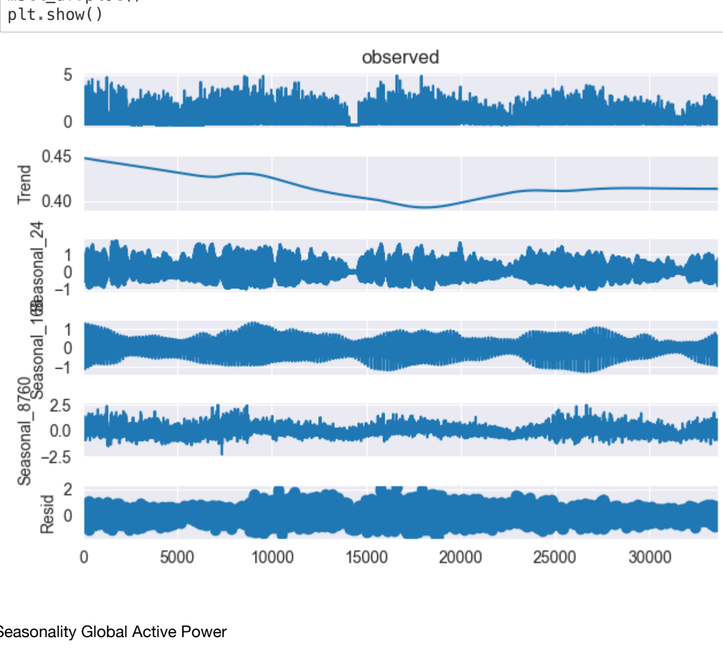
Out[76]:
```
1758   -0.112824
1759    0.054659
1760    0.041462
1761   -0.050430
1762   -0.128919
Name: seasonal_24, dtype: float64
```

In [77]:
```
mstl = MSTL(ISD['Air_Temp_Norm'], periods=[24, 24*365])
mstl_ISD = mstl.fit()
```

Appendix B

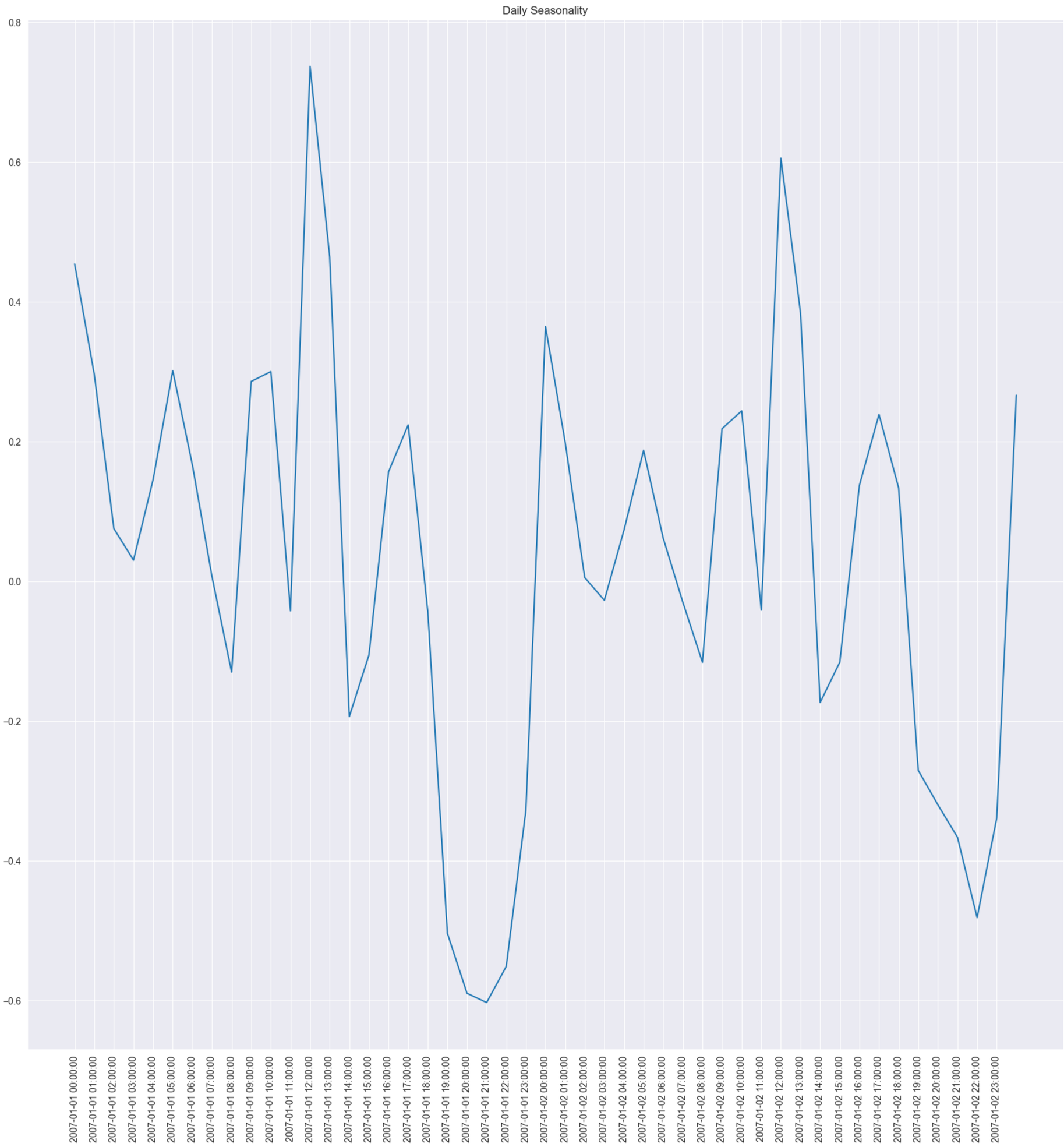
Visual A : Full MSTL output for Global Active Power

In [78]:
```
mstl = MSTL(df1['Global_active_power_Norm'], periods=[24, 24*7, 24*365])
mstl_df = mstl.fit()
mstl_df.plot()
plt.show()
```



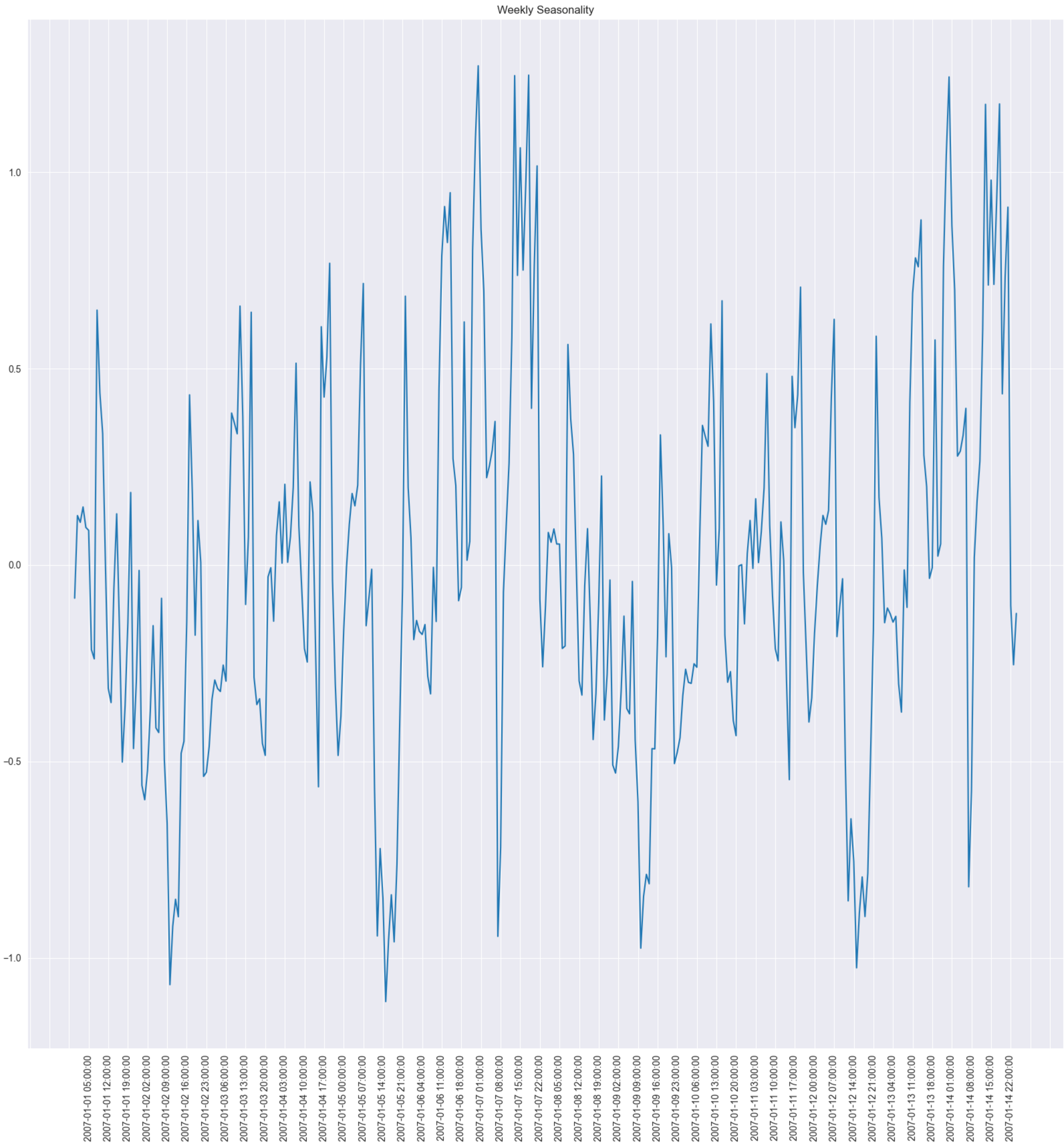
Visual B : Daily Seasonality Global Active Power

In [79]:
```
df_24 = mstl_df._seasonal.seasonal_24[:49]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_48h = pd.date_range(start='2007-01-01', periods=48, freq='H')
df_24.plot(title='Daily Seasonality')
plt.xticks(rotation=90, ticks=list(range(0,48)), labels=date_range_48h)
plt.show()
```



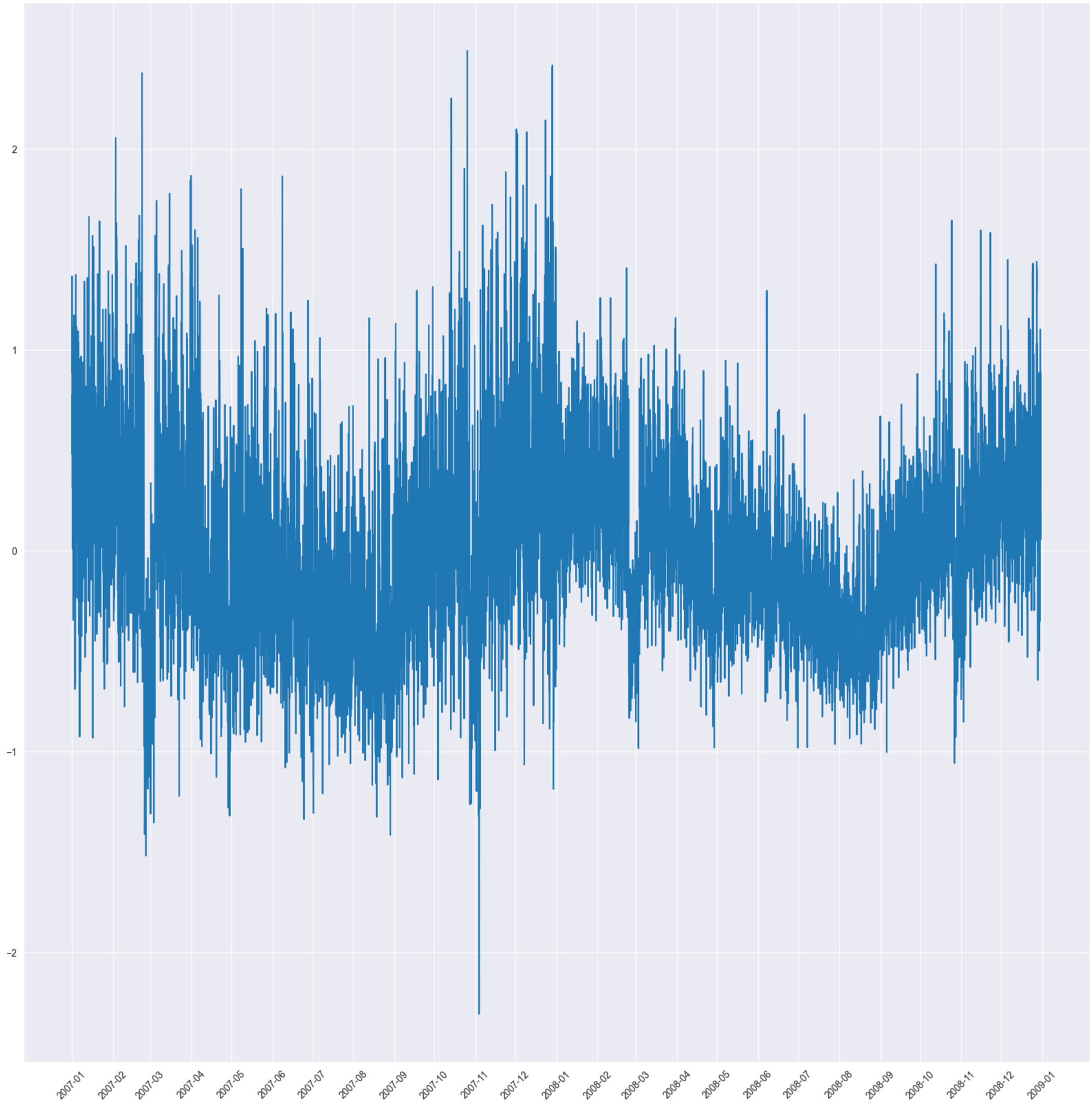
Visual C : Weekly Seasonality Global Active Power

In [80]:
```
df_168 = mstl_df._seasonal.seasonal_168[:168*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_168h = pd.date_range(start='2007-01-01', periods=168*2+1, freq='H')
df_168.plot(title='Weekly Seasonality')

plt.xticks(rotation=90, ticks=list(range(0,168*2+1)), labels=date_range_168h)
plt.gca().xaxis.set_major_locator(mdate.WeekdayLocator())
plt.show()
```



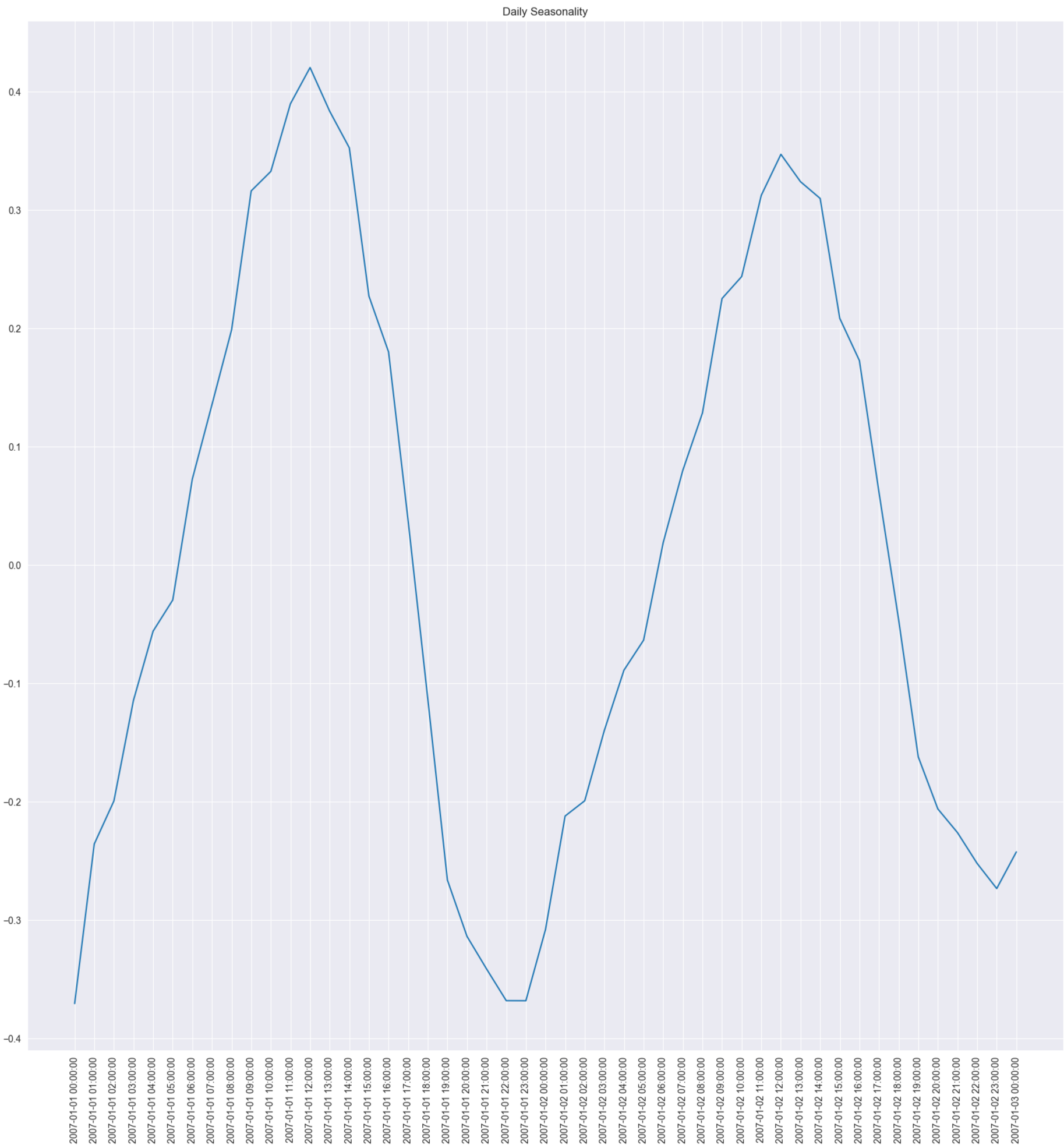
Visual D: Yearly Seasonality Global Active Power

In [81]:
```
df_8760 = mstl_df._seasonal.seasonal_8760[:8760*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_8760h = pd.date_range(start='2007-01-01', periods=8760*2+1, freq='H')
ax.plot(date_range_8760h, df_8760),
monthly_ticks = pd.date_range(start='2007-01-01', end='2009-01-01',freq='MS')  # MS stands for Month Start
plt.xticks(monthly_ticks, [x.strftime('%Y-%m') for x in monthly_ticks], rotation=45)
plt.show()
```



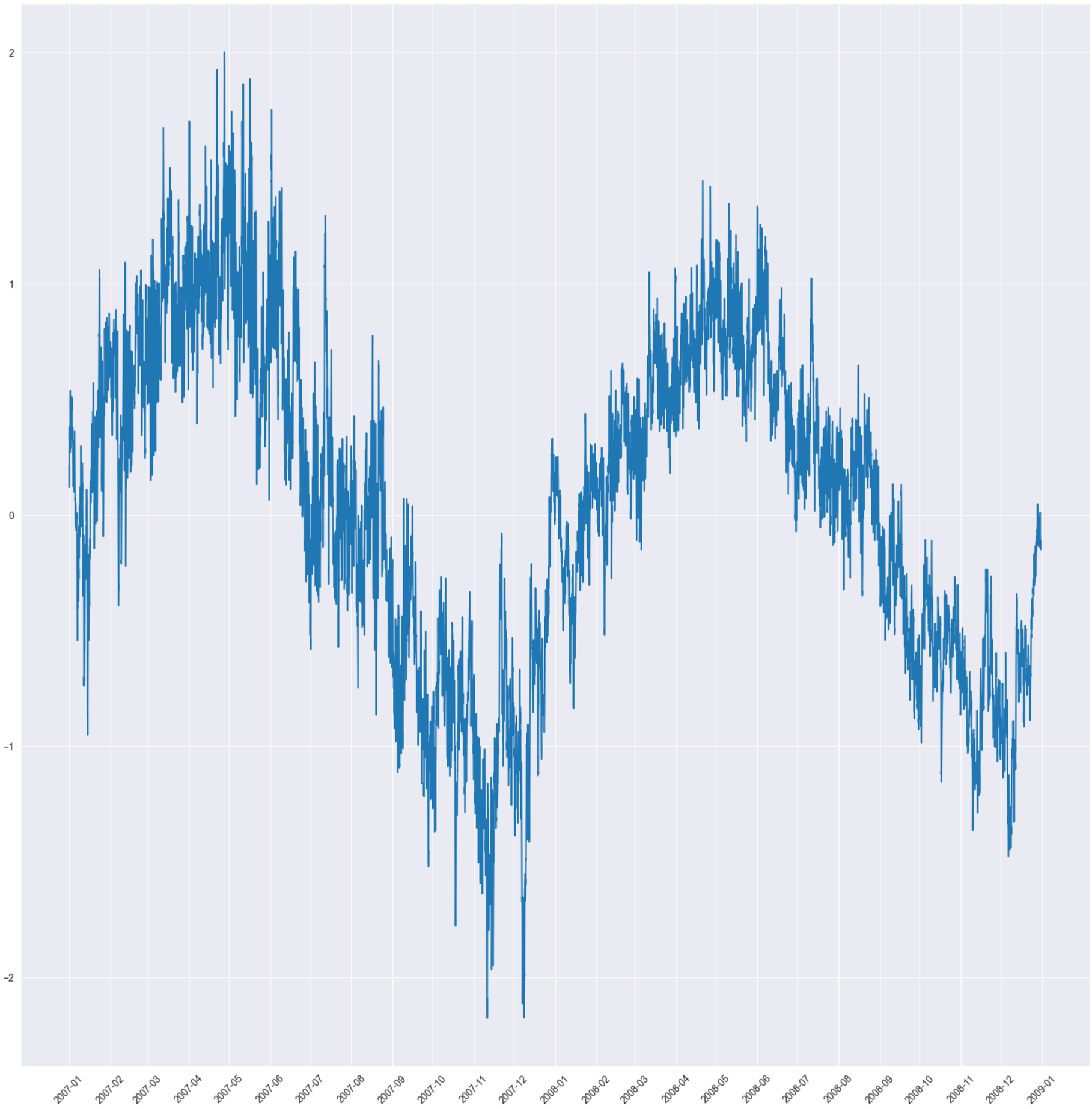
Visual E: Daily Seasonality Air Temperature

In [82]:
```
ISD_24 = mstl_ISD._seasonal.seasonal_24[:49]
fig, ax = plt.subplots(figsize=(20, 20))
ISD_24.plot(title='Daily Seasonality')
date_range_48h = pd.date_range(start='2007-01-01', periods=49, freq='H')
plt.xticks(ticks=range(1758, 1807), labels=date_range_48h, rotation = 90)
plt.show()
```



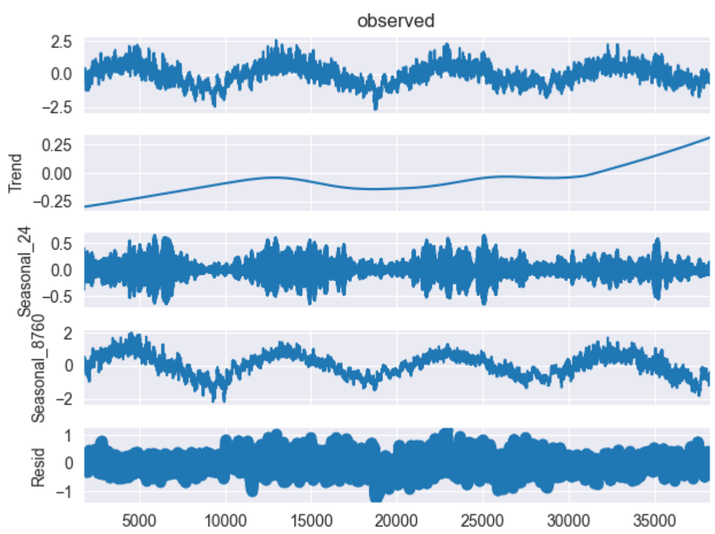
Visual F: Yearly Seasonality Air Temperature

In [83]:
```
df_8760 = mstl_ISD._seasonal.seasonal_8760[:8760*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_8760h = pd.date_range(start='2007-01-01', periods=8760*2+1, freq='H')
ax.plot(date_range_8760h, df_8760),
monthly_ticks = pd.date_range(start='2007-01-01', end='2009-01-01',freq='MS')  # MS stands for Month Start
plt.xticks(monthly_ticks, [x.strftime('%Y-%m') for x in monthly_ticks], rotation=45)
plt.show()
```

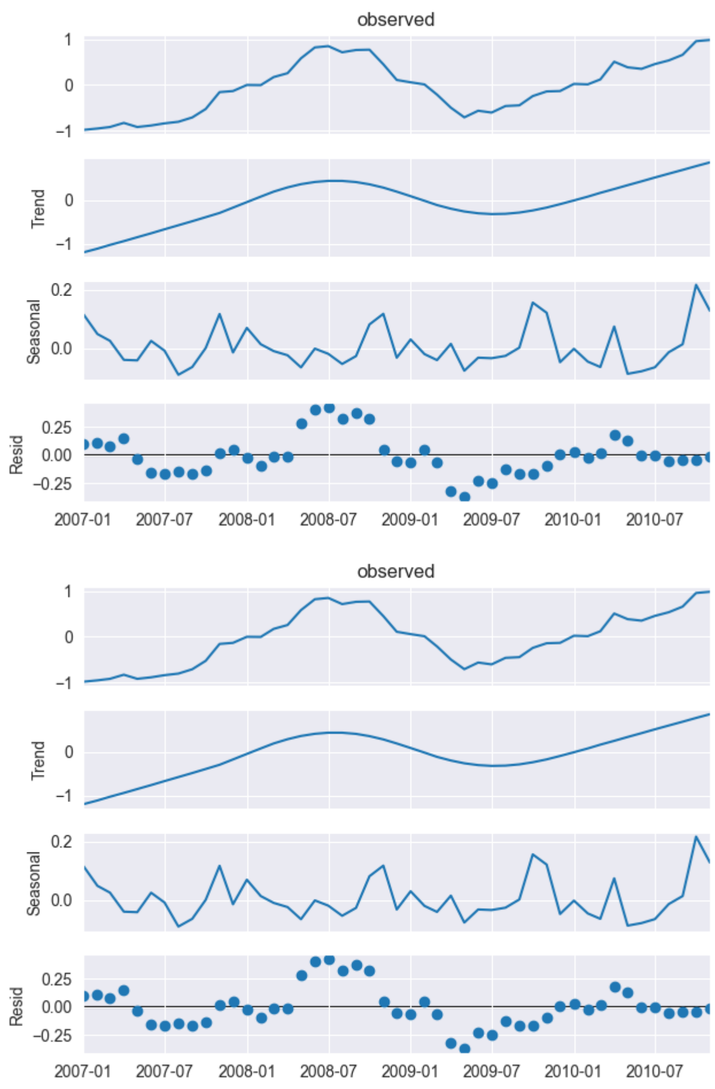


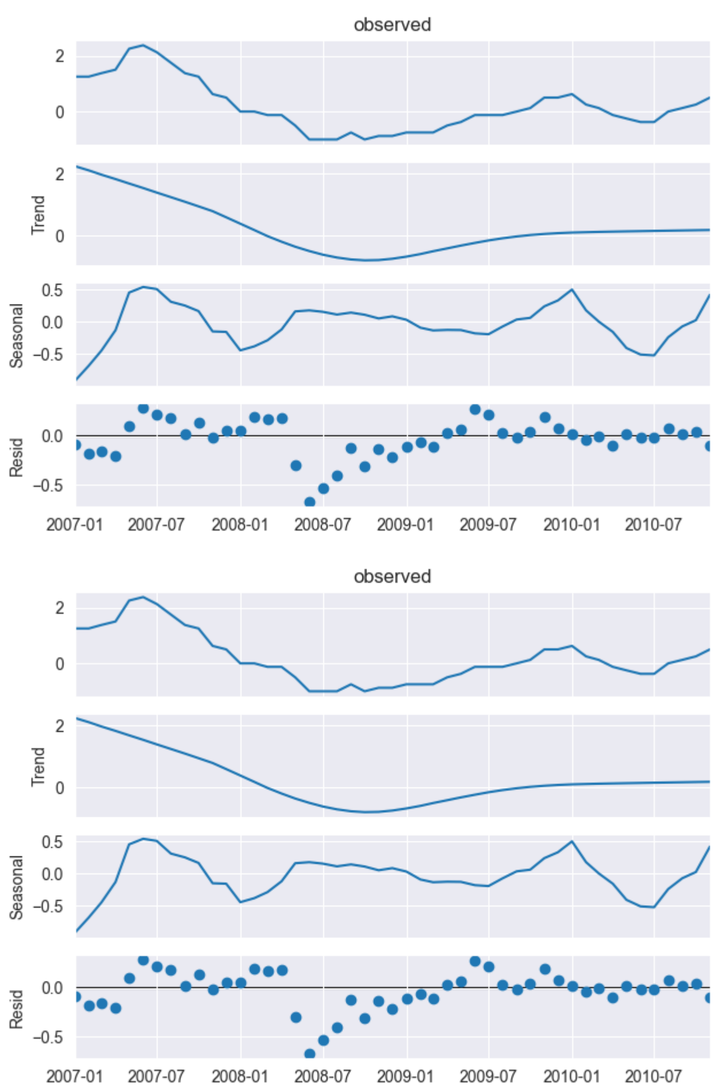
Visual G: Full MSTL Output Air Temperature

In [84]:
```
mstl_ISD.plot()
plt.tight_layout()
```



Visual H: Full MSTL Output of CPI





In [85]:
```
mstl = MSTL(CPI['CPI_Score_Norm'], periods = 12)
mstl_CPI = mstl.fit()
mstl_CPI.plot()
```

Out[85]:

Visual I: Full MSTL Output of CCI

In [86]:
```
mstl = MSTL(CCI['Monthly consumer confidence survey_Norm'], periods = 12)
mstl_CCI = mstl.fit()
mstl_CCI.plot()
```

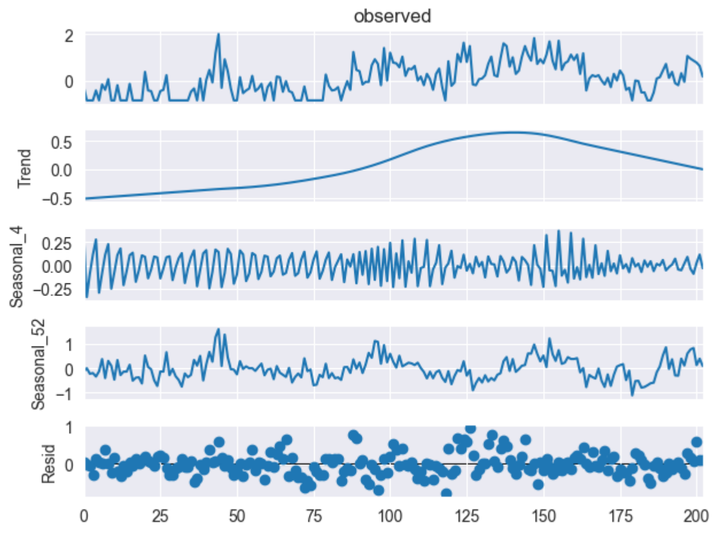
Out[86]:

Visual J: Full MSTL Output of Search Trend “ampoules basse consommation”

In [87]:
```
trend1.Date = trend1.Date.astype('datetime64[ns]')
trend1.set_index('Date')
trend1['Hits_Norm'] = mean_normalized(trend1, 'Hits')
trend2['Hits_Norm'] = mean_normalized(trend2, 'Hits')
trend3['Hits_Norm'] = mean_normalized(trend3, 'Hits')
```

```
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:757: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:595: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/Users/ethan/Documents/ethan/lib/python3.10/site-packages/sklearn/utils/validation.py:604: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

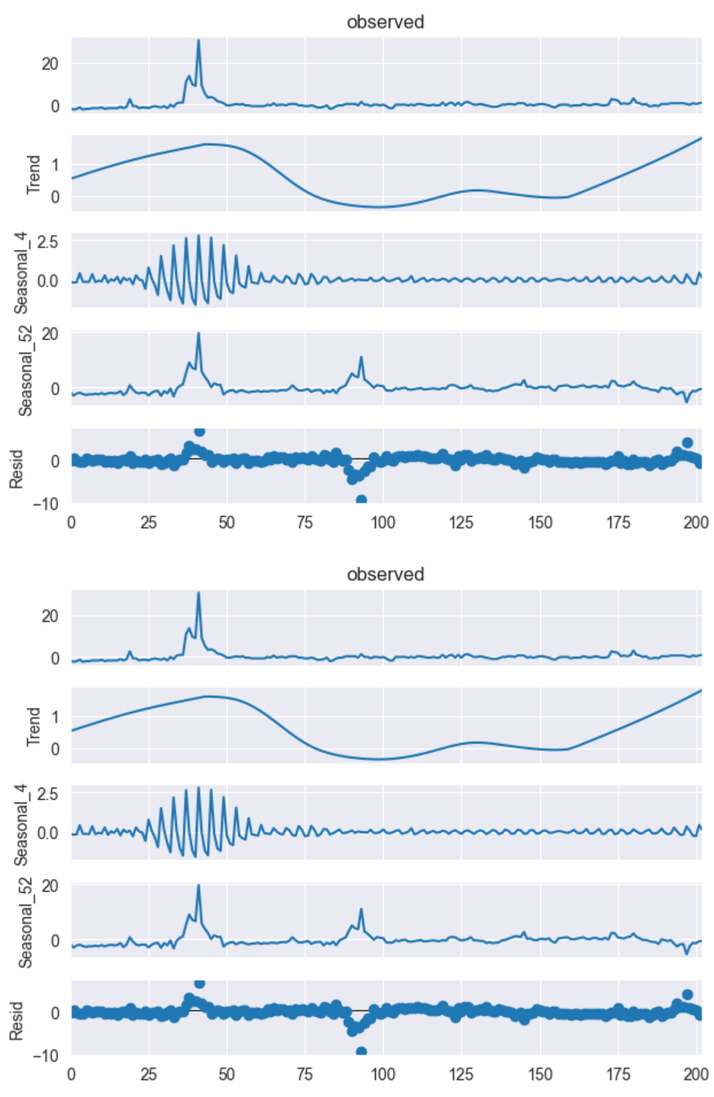
In [88]:
```
mstl = MSTL(trend1['Hits_Norm'], periods = [4,52])
mstl_trend1 = mstl.fit()
mstl_trend1.plot()
plt.tight_layout()
```



Visual K: Full MSTL Output of Search Trend “Jean-Louis Borloo”

In [89]:
```
trend2.Date = trend2.Date.astype('datetime64[ns]')
trend2.set_index('Date')

mstl = MSTL(trend2['Hits_Norm'], periods = [4, 52])
mstl_trend2 = mstl.fit()
mstl_trend2.plot()
```

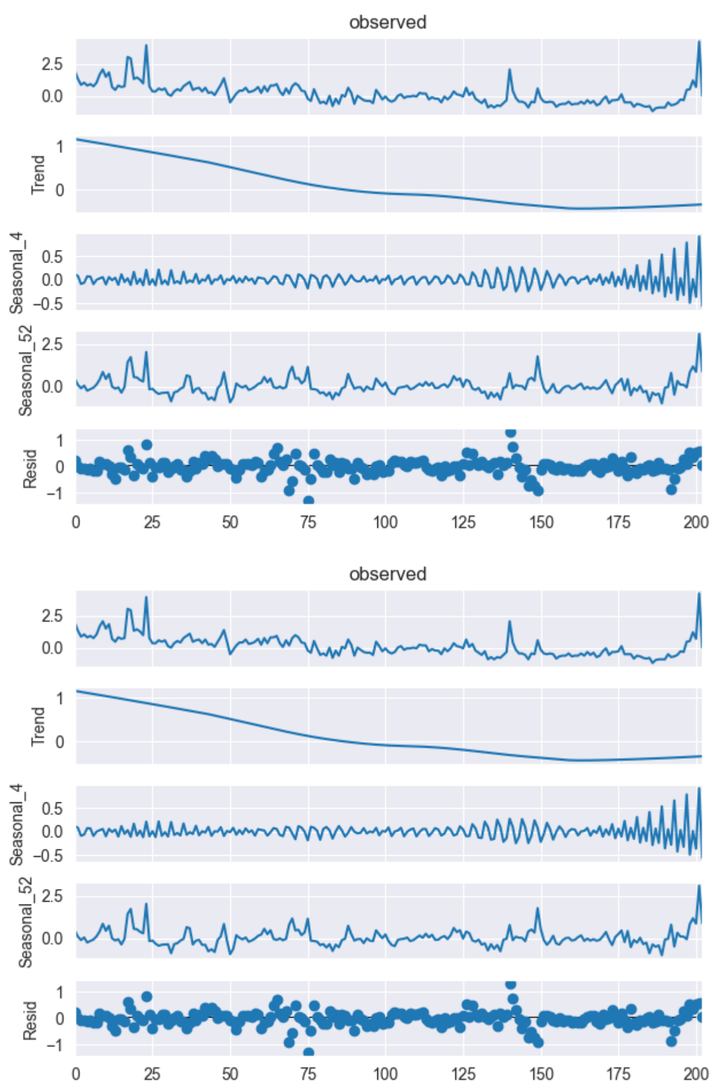
Out[89]:



Visual L: Full MSTL Output of Search Trend “Grenelle de l'environnement”

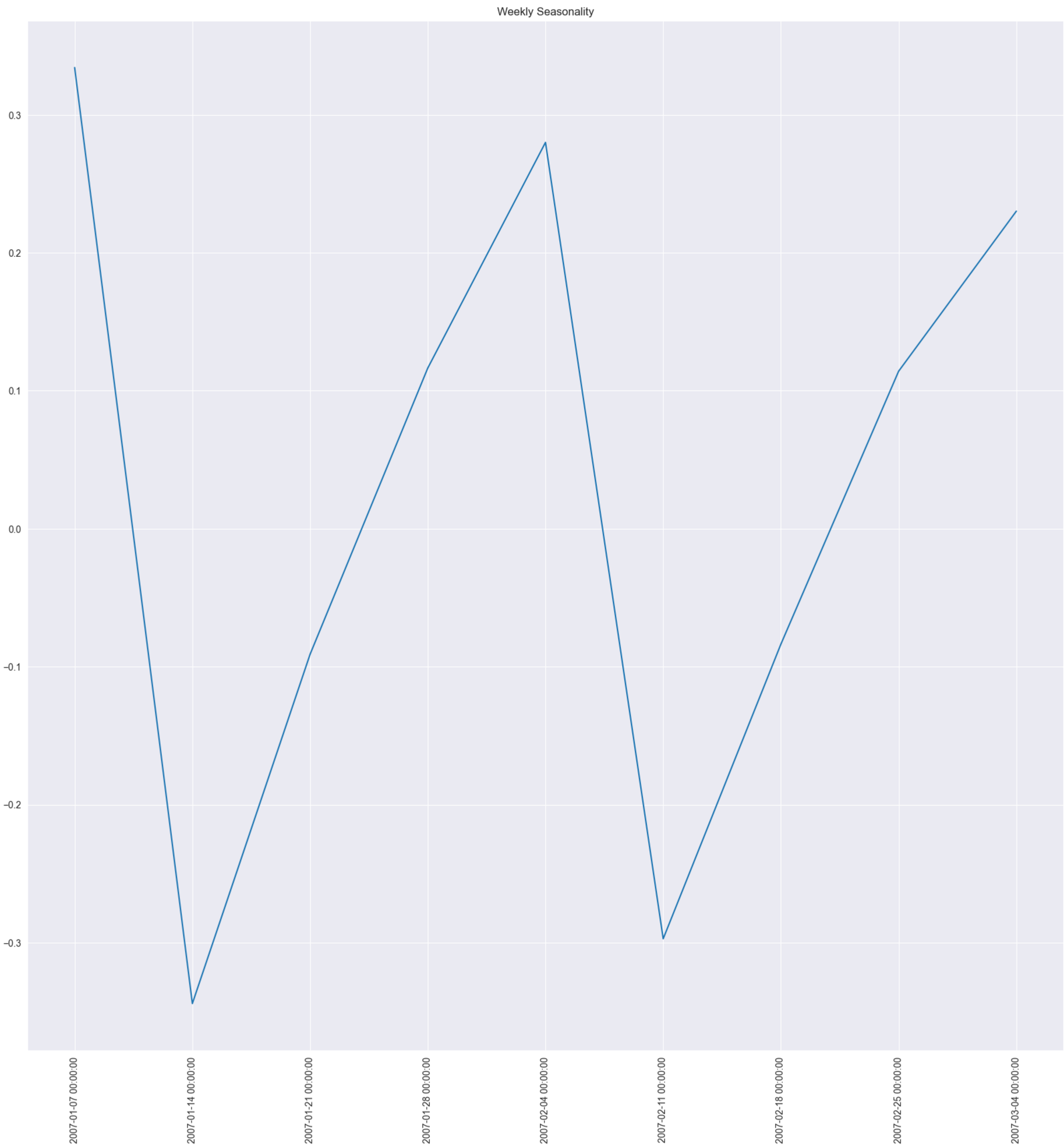
In [90]:
```
trend3.Date = trend3.Date.astype('datetime64[ns]')
trend3.set_index('Date')

mstl = MSTL(trend3['Hits_Norm'], periods = [4, 52], iterate = 5)
mstl_trend3 = mstl.fit()
mstl_trend3.plot()
```

Out[90]:



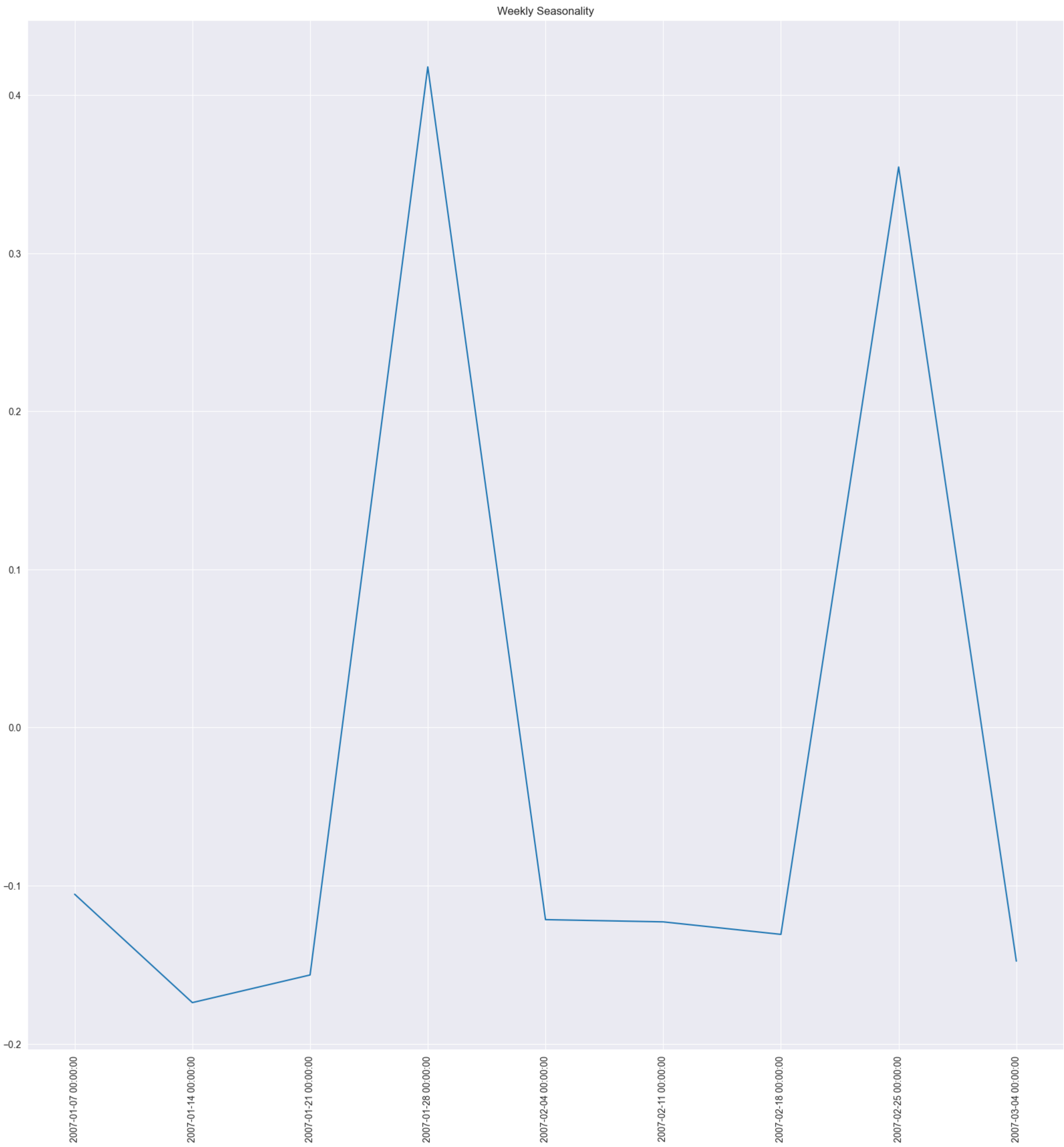
Visual M: Monthly Seasonality Search Trend “ampoules basse consommation”

In [91]:
```
df_trend1_4 = mstl_trend1._seasonal.seasonal_4[:4*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range = pd.date_range(start='2007-01-01', periods=9, freq='W')
df_trend1_4.plot(title='Weekly Seasonality')
plt.xticks(rotation=90, ticks=list(range(0,9)), labels=date_range)
plt.show()
```



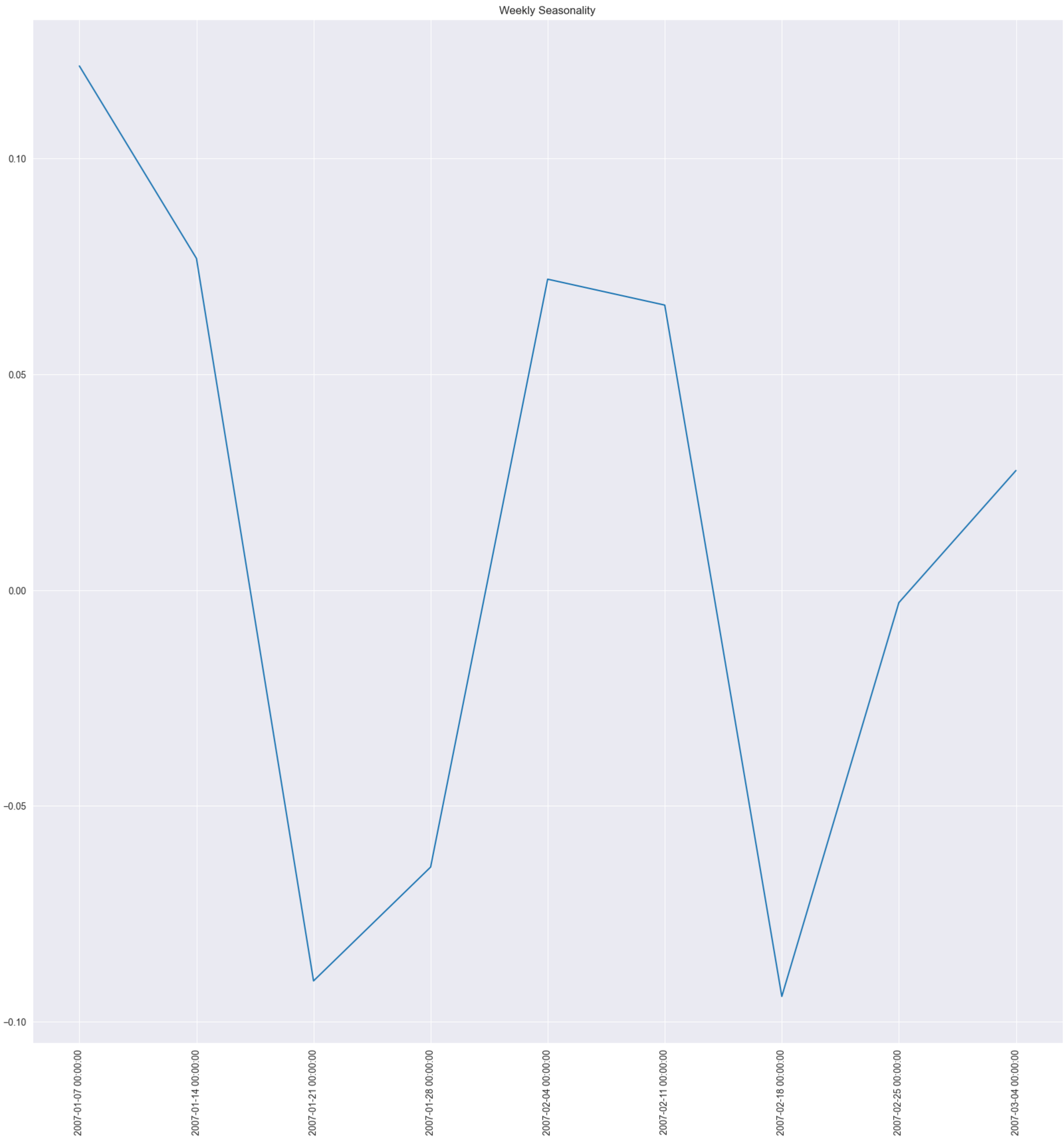
Visual N: Monthly Seasonality Search Trend “Jean-Louis Borloo”

In [92]:
```
df_trend2_4 = mstl_trend2._seasonal.seasonal_4[:4*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range = pd.date_range(start='2007-01-01', periods=9, freq='W')
df_trend2_4.plot(title='Weekly Seasonality')
plt.xticks(rotation=90, ticks=list(range(0,9)), labels=date_range)
plt.show()
```



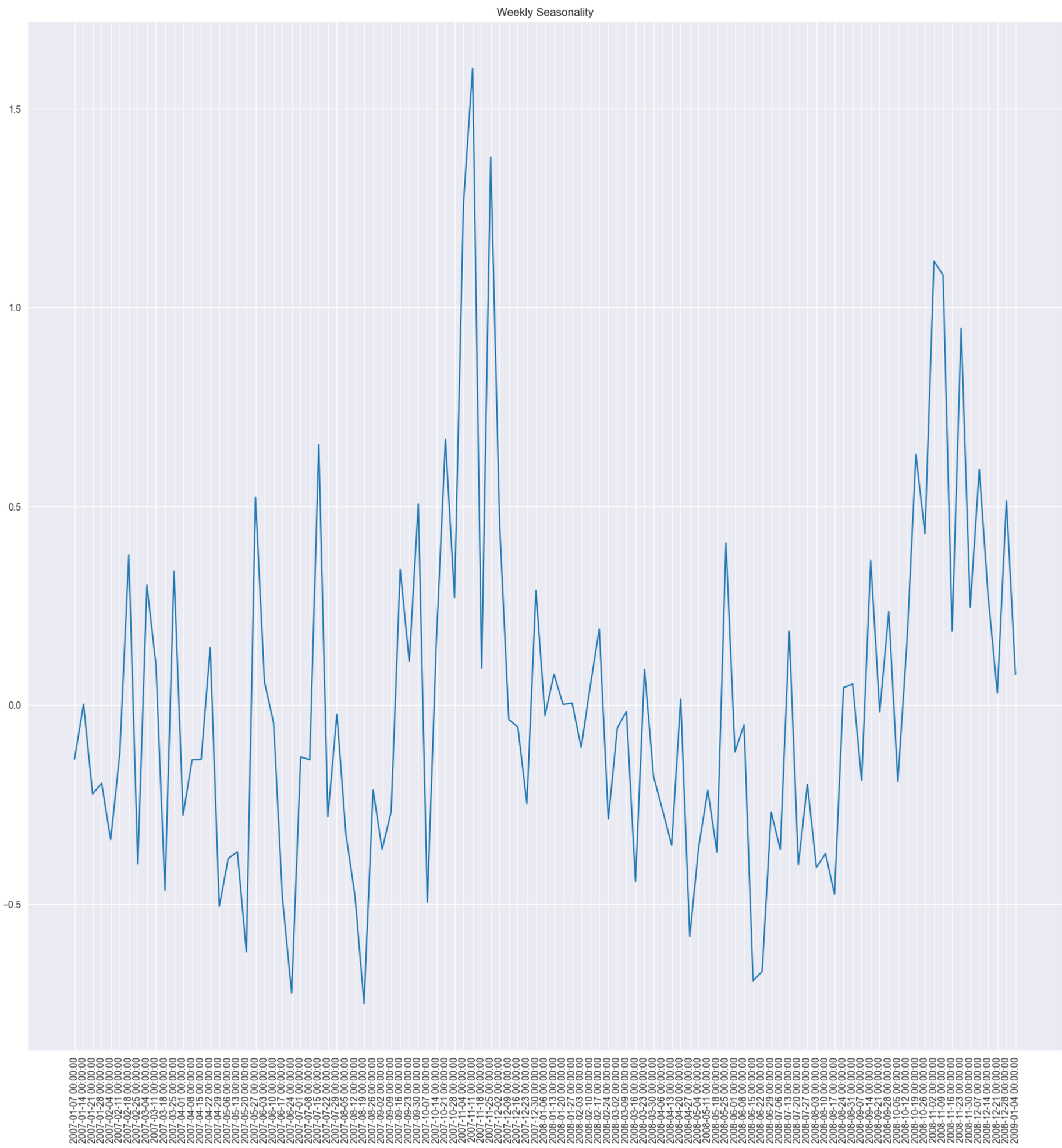
Visual O: Monthly Seasonality Search Trend “Grenelle de l'environnement”

In [93]:
```
df_trend3_4 = mstl_trend3._seasonal.seasonal_4[:4*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range = pd.date_range(start='2007-01-01', periods=9, freq='W')
df_trend3_4.plot(title='Weekly Seasonality')
plt.xticks(rotation=90, ticks=list(range(0,9)), labels=date_range)
plt.show()
```



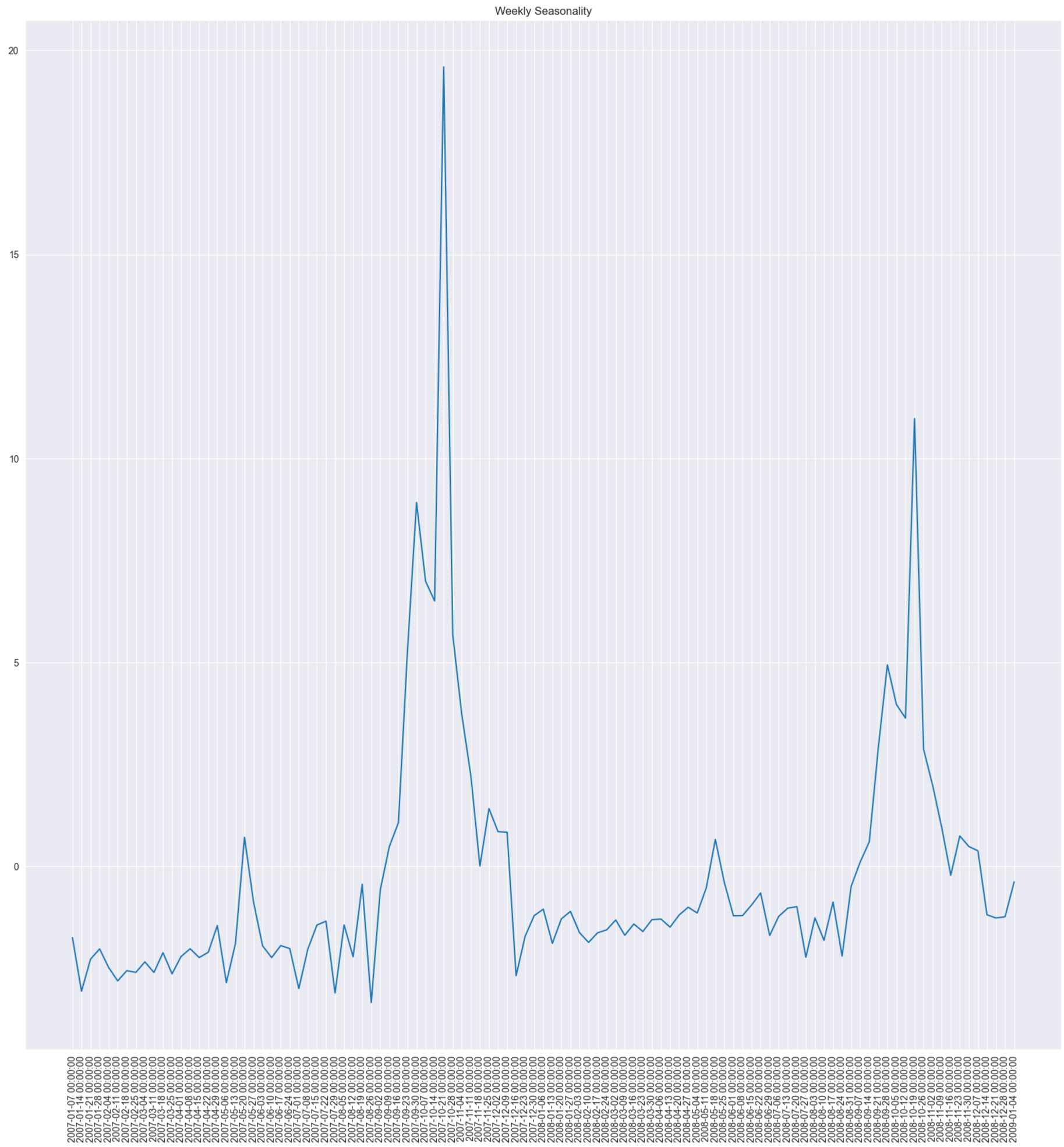
Visual P: Yearly Seasonality Search Trend “ampoules basse consommation”

In [94]:
```
df_trend1_52 = mstl_trend1._seasonal.seasonal_52[:52*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range = pd.date_range(start='2007-01-01', periods=52*2+1, freq='W')
df_trend1_52.plot(title='Weekly Seasonality')
plt.xticks(rotation=90, ticks=list(range(0,52*2+1)), labels=date_range)
plt.show()
```



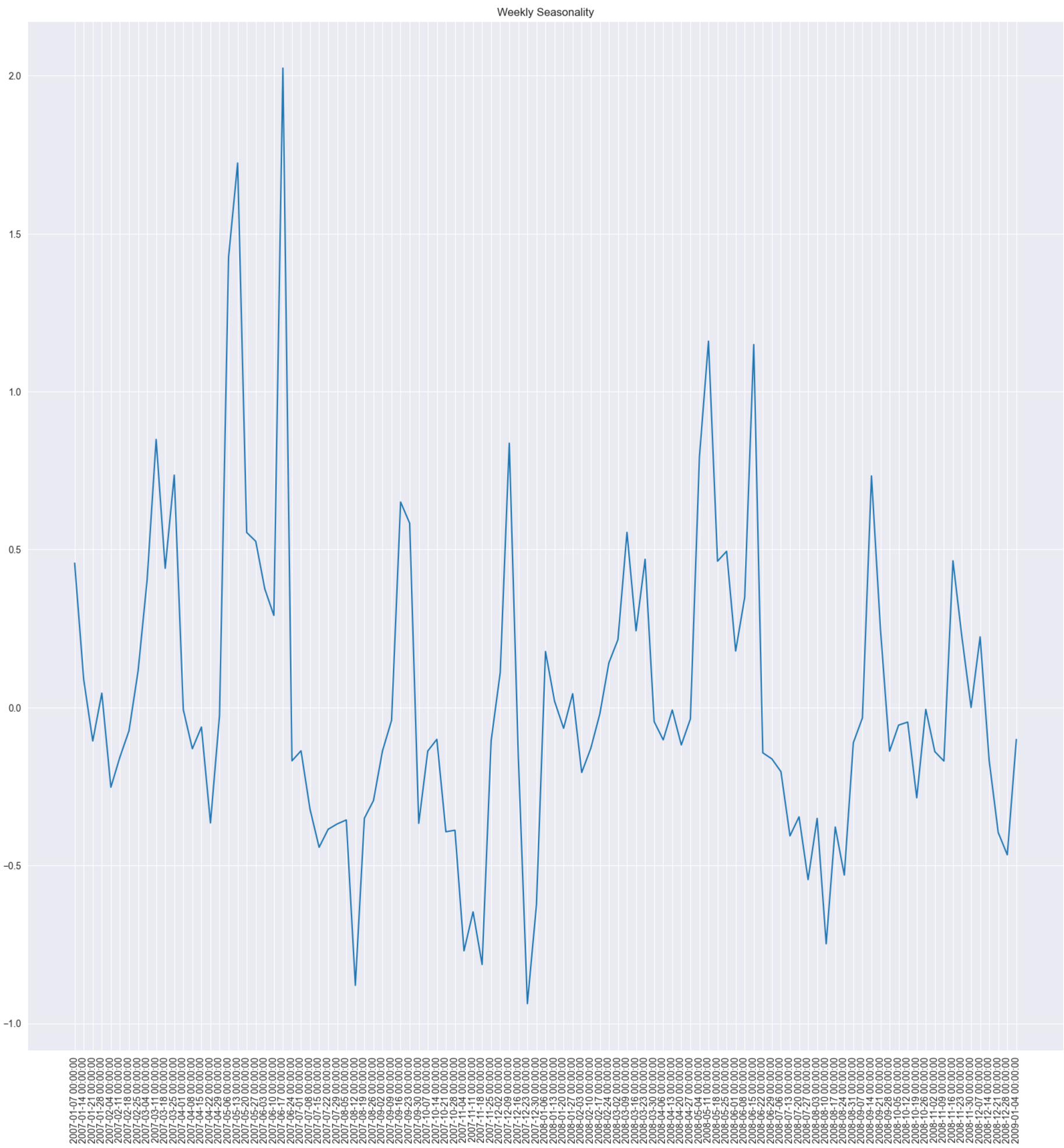
Visual Q: Yearly Seasonality Search Trend “Jean-Louis Borloo”

In [95]:
```
df_trend2_52 = mstl_trend2._seasonal.seasonal_52[:52*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range = pd.date_range(start='2007-01-01', periods=52*2+1, freq='W')
df_trend2_52.plot(title='Weekly Seasonality')
plt.xticks(rotation=90, ticks=list(range(0,52*2+1)), labels=date_range)
plt.show()
```



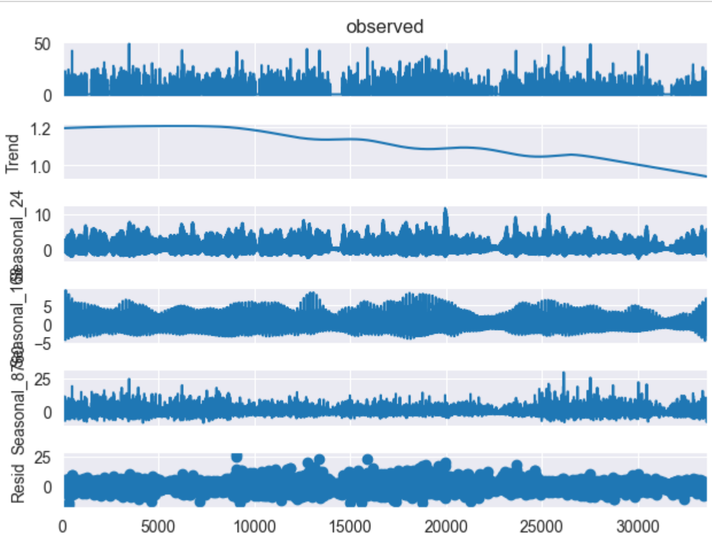
Visual R: Yearly Seasonality Search Trend “Grenelle de l'environnement”

In [96]:
```
df_trend3_52 = mstl_trend3._seasonal.seasonal_52[:52*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range = pd.date_range(start='2007-01-01', periods=52*2+1, freq='W')
df_trend3_52.plot(title='Weekly Seasonality')
plt.xticks(rotation=90, ticks=list(range(0,52*2+1)), labels=date_range)
plt.show()
```



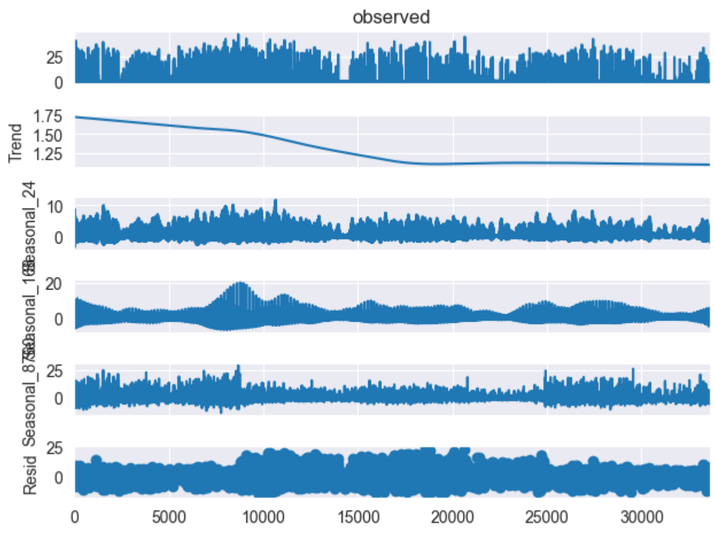
Visual S: Full MSTL Output of Sub-Metering 1

In [97]:
```
mstl = MSTL(df_sub1['Sub_metering_1'], periods = [24, 24*7, 24*365])
mstl_dfsub1 = mstl.fit()
mstl_dfsub1.plot()
plt.show()
```



Visual T: Full MSTL Output of Sub-Metering 2

In [98]:
```
mstl = MSTL(df_sub2['Sub_metering_2'], periods = [24, 24*7, 24*365])
mstl_dfsub2 = mstl.fit()
mstl_dfsub2.plot()
plt.show()
```

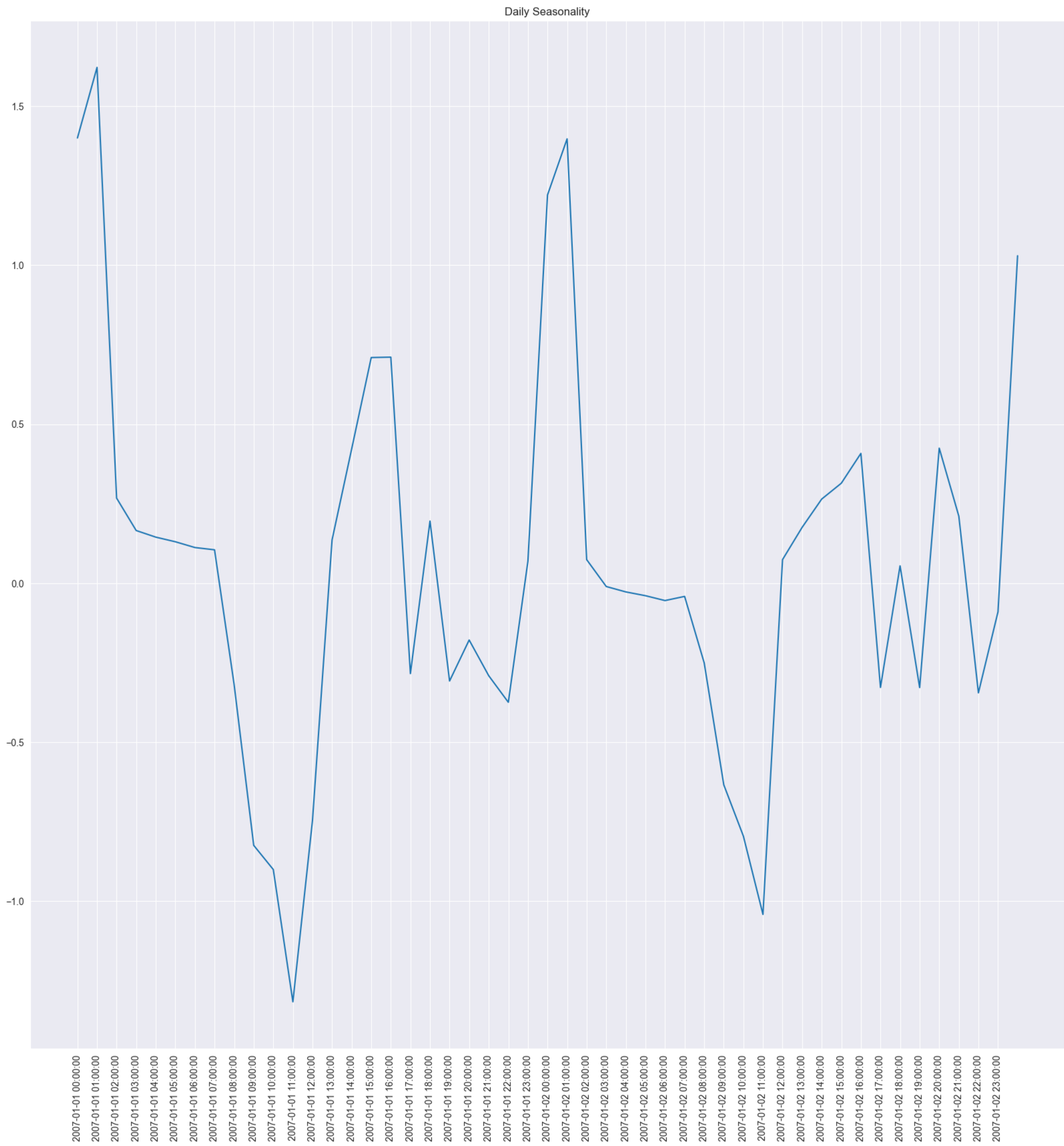


Visual U: Full MSTL Output of Sub-Metering 3

In [99]:
```
mstl = MSTL(df_sub3['Sub_metering_3'], periods = [24, 24*7, 24*365])
mstl_dfsub3 = mstl.fit()
mstl_dfsub3.plot()
plt.show()
```

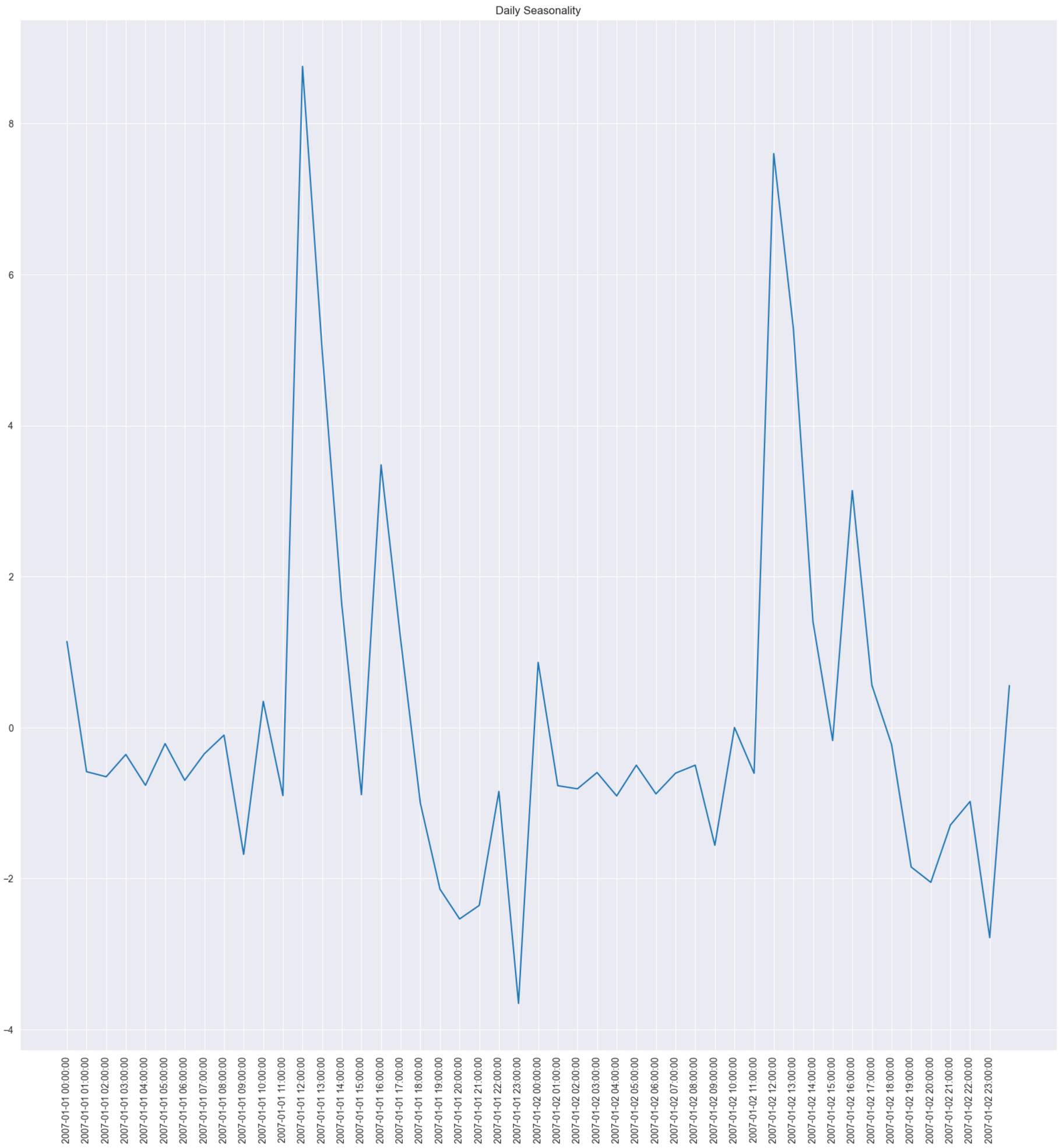

Visual V: Daily Seasonality Sub-Metering 1

In [100]:
```
dfsub1_24 = mstl_dfsub1._seasonal.seasonal_24[:49]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_48h = pd.date_range(start='2007-01-01', periods=48, freq='H')
dfsub1_24.plot(title='Daily Seasonality')
plt.xticks(rotation=90, ticks=list(range(0,48)), labels=date_range_48h)
plt.show()
```



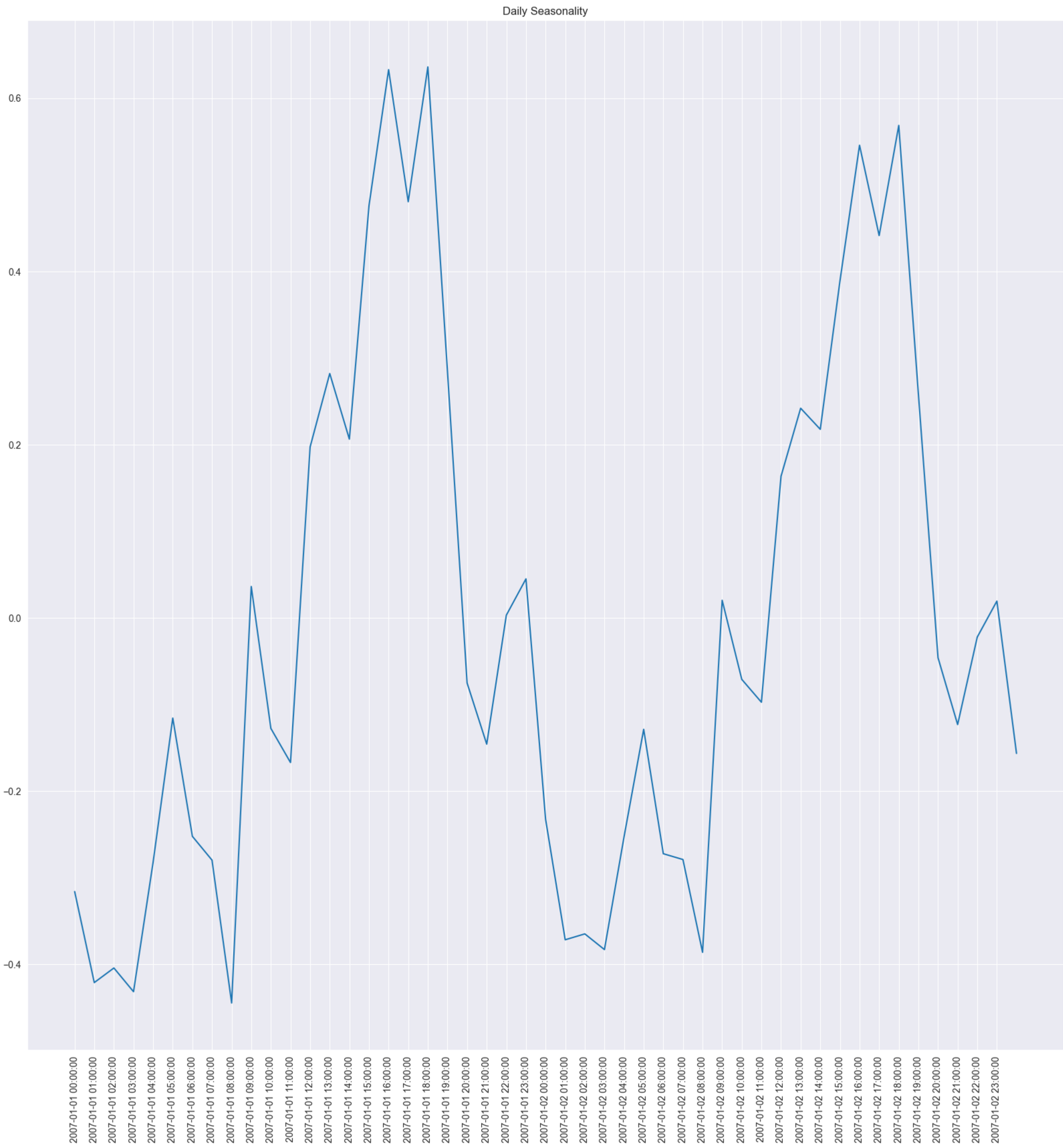
Visual W: Daily Seasonality Sub-Metering 2

In [101]:
```
dfsub2_24 = mstl_dfsub2._seasonal.seasonal_24[:49]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_48h = pd.date_range(start='2007-01-01', periods=48, freq='H')
dfsub2_24.plot(title='Daily Seasonality')
plt.xticks(rotation=90, ticks=list(range(0,48)), labels=date_range_48h)
plt.show()
```



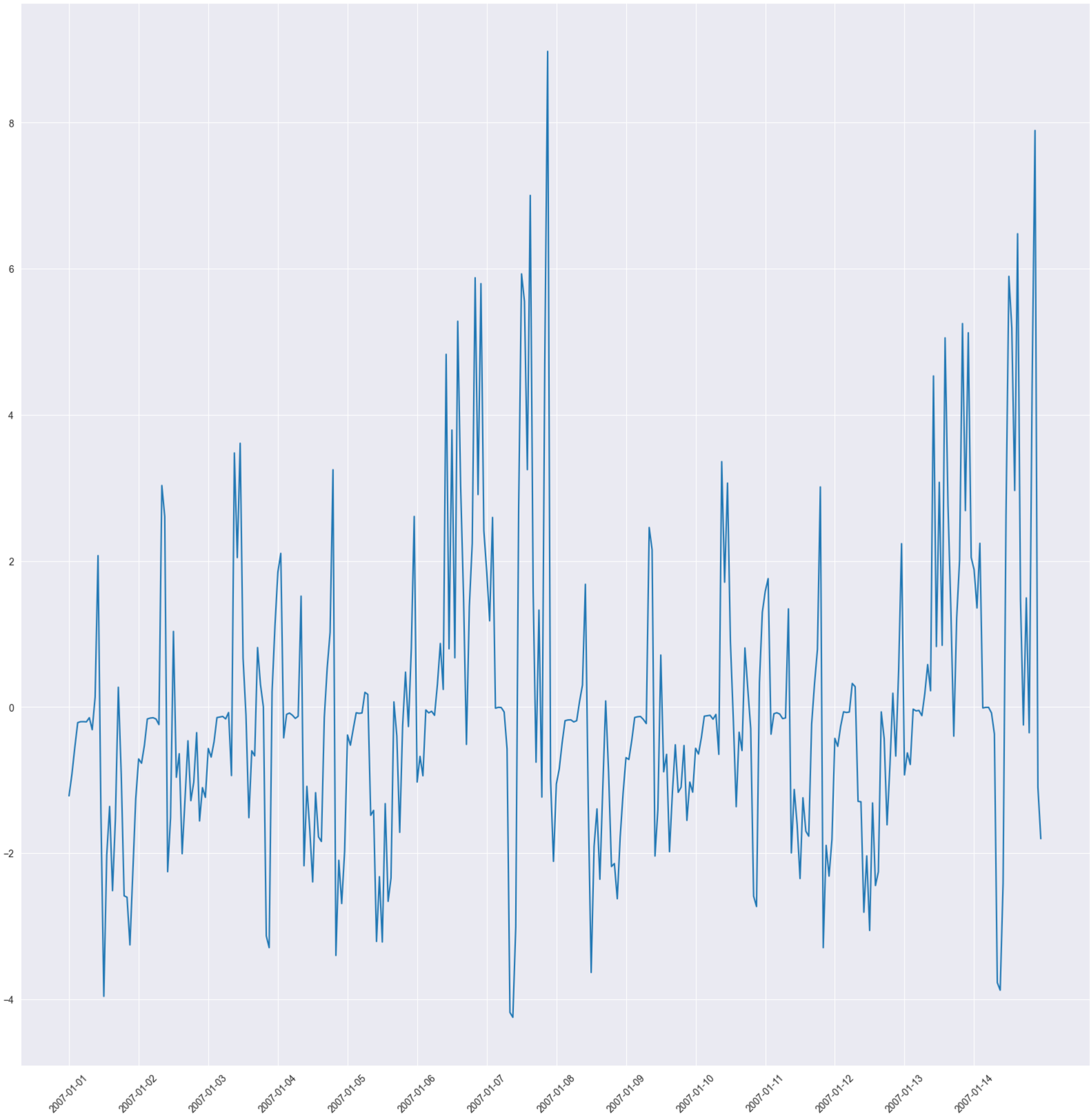
Visual X: Daily Seasonality Sub-Metering 3

In [102]:
```
dfsub3_24 = mstl_dfsub3._seasonal.seasonal_24[:49]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_48h = pd.date_range(start='2007-01-01', periods=48, freq='H')
dfsub3_24.plot(title='Daily Seasonality')
plt.xticks(rotation=90, ticks=list(range(0,48)), labels=date_range_48h)
plt.show()
```



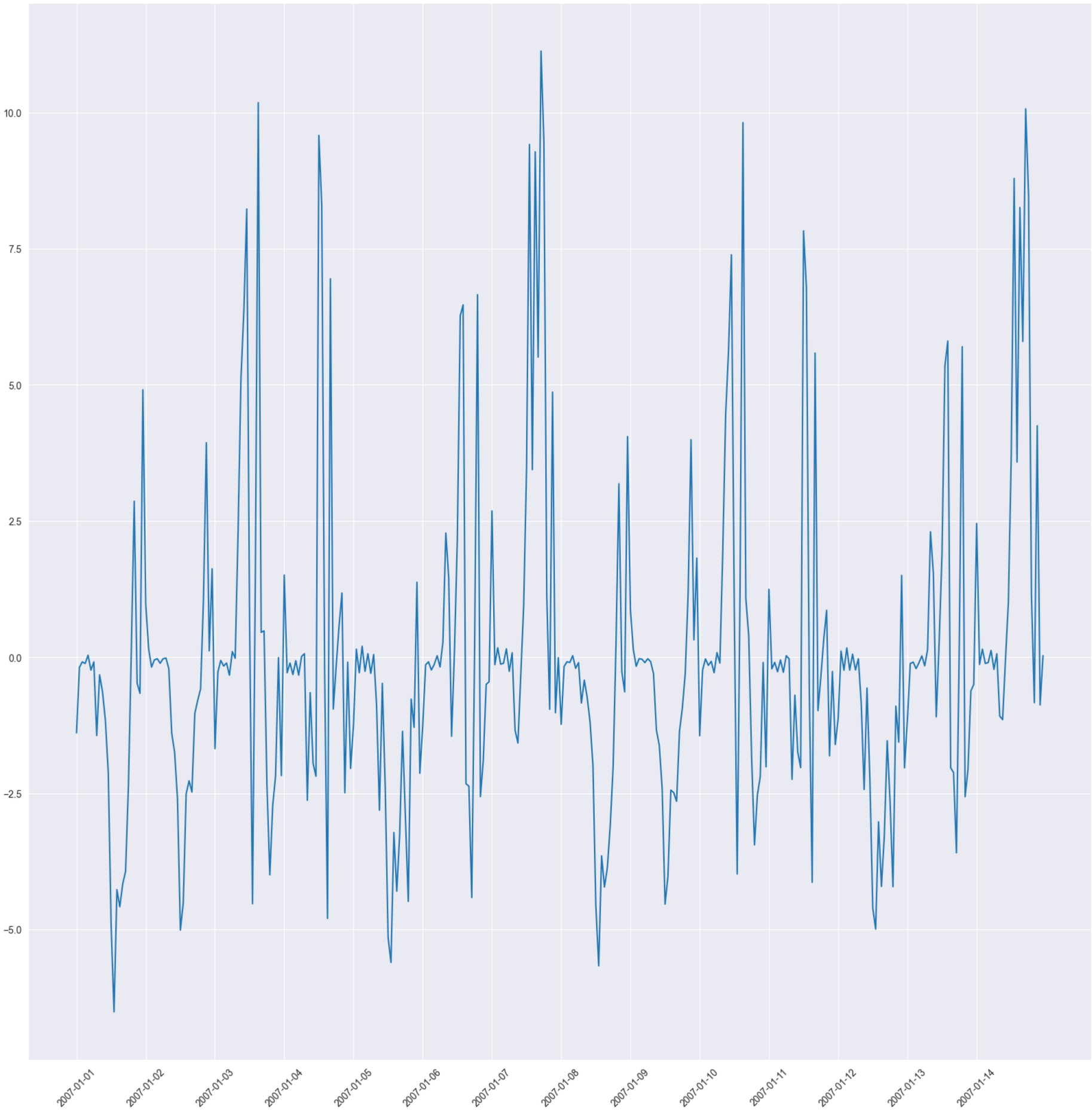
Visual Y: Weekly Seasonality Sub-Metering 1

In [103]:
```
dfsub1_168 = mstl_dfsub1._seasonal.seasonal_168[:168*2]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_168h = pd.date_range(start='2007-01-01', periods=168*2, freq='H')
ax.plot(date_range_168h, dfsub1_168),
daily_ticks = pd.date_range(start='2007-01-01', end='2007-01-14',freq='D')  # MS stands for Month Start
plt.xticks(daily_ticks,rotation=45)
plt.show()
```



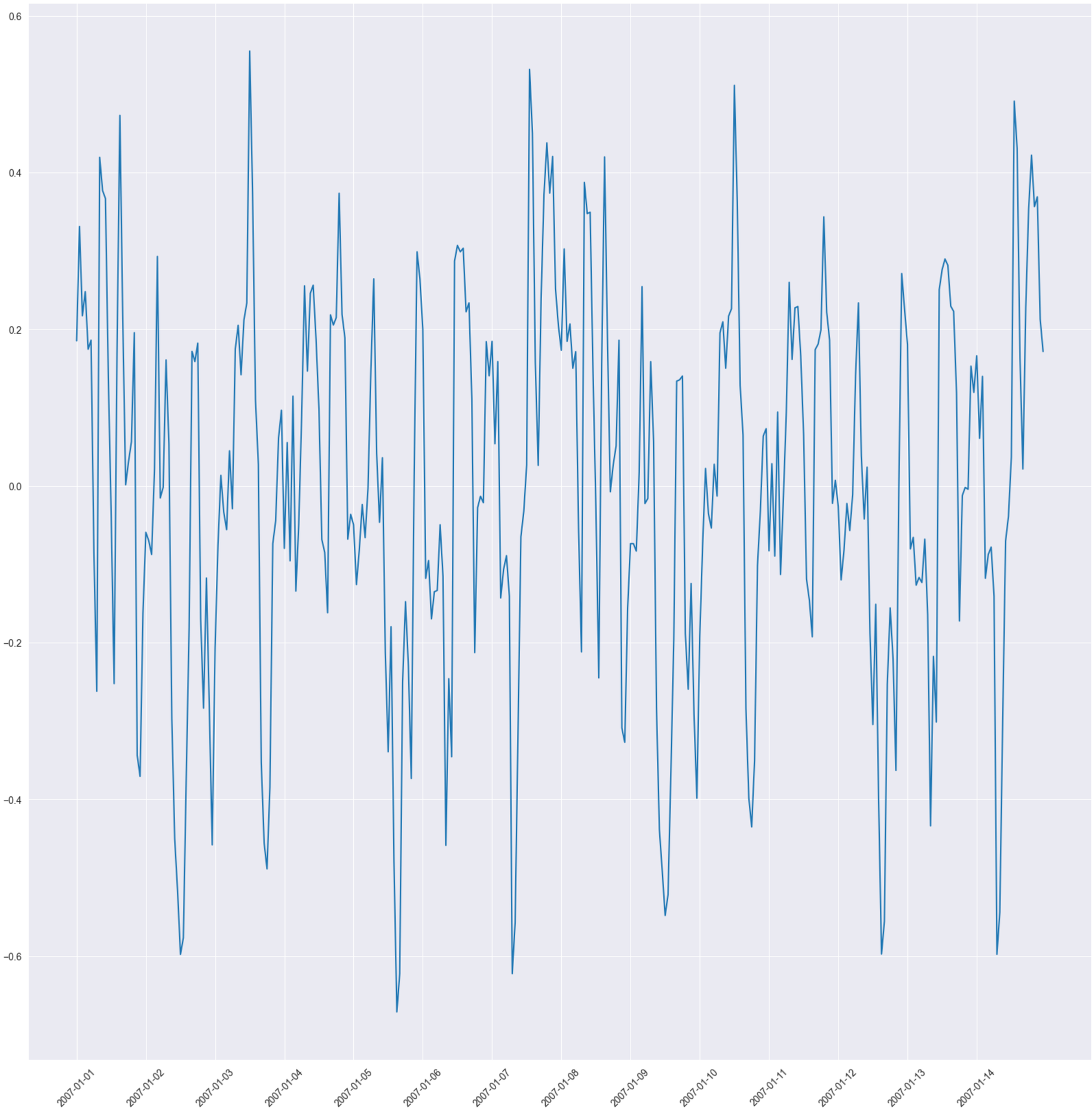
Visual Z: Weekly Seasonality Sub-Metering 2

In [104]:
```
dfsub2_168 = mstl_dfsub2._seasonal.seasonal_168[:168*2]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_168h = pd.date_range(start='2007-01-01', periods=168*2, freq='H')
ax.plot(date_range_168h, dfsub2_168),
daily_ticks = pd.date_range(start='2007-01-01', end='2007-01-14',freq='D')  # MS stands for Month Start
plt.xticks(daily_ticks,rotation=45)
plt.show()
```



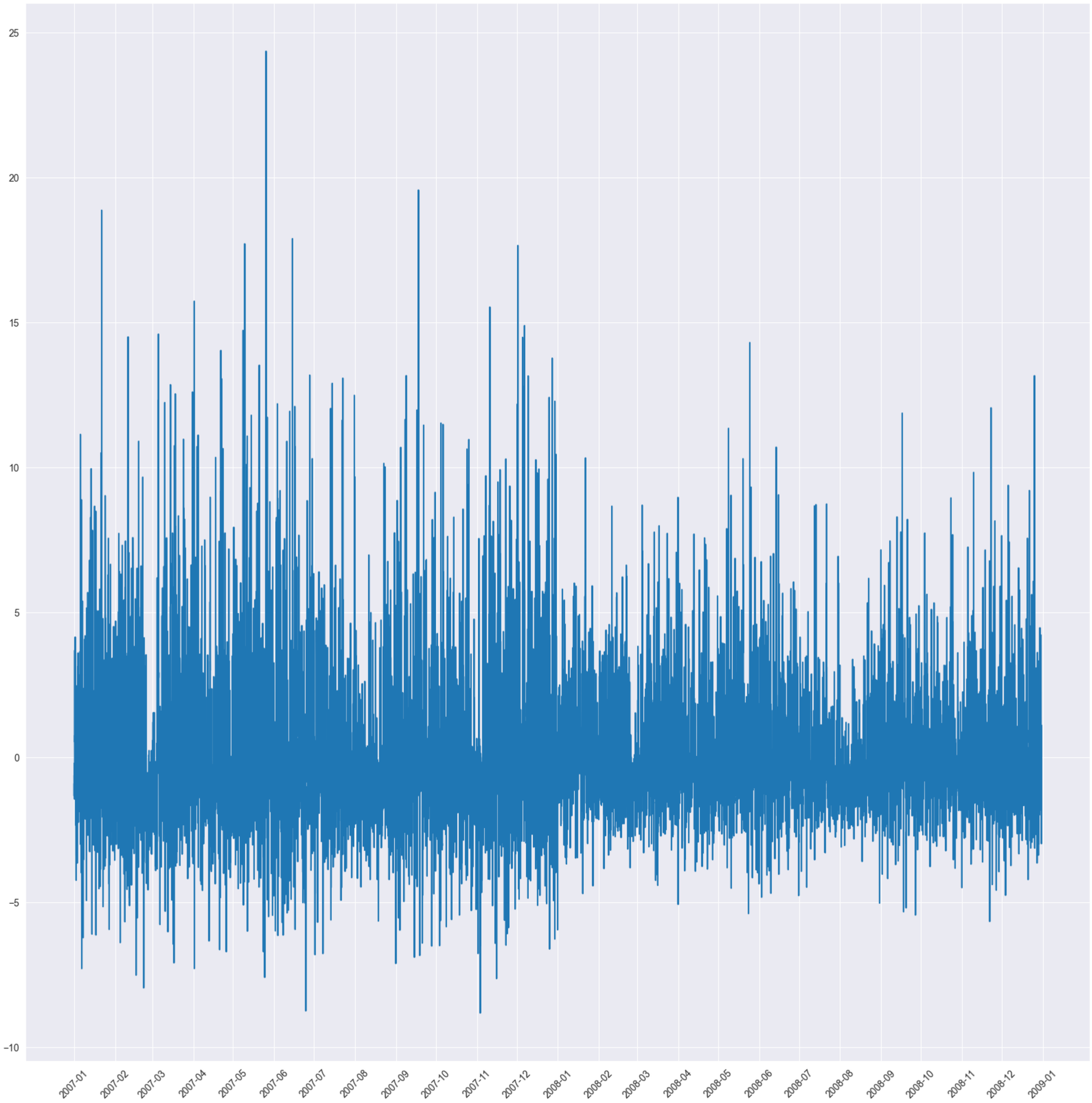
Visual A1: Weekly Seasonality Sub-Metering 3

In [105]:
```
dfsub3_168 = mstl_dfsub3._seasonal.seasonal_168[:168*2]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_168h = pd.date_range(start='2007-01-01', periods=168*2, freq='H')
ax.plot(date_range_168h, dfsub3_168),
daily_ticks = pd.date_range(start='2007-01-01', end='2007-01-14',freq='D')  # MS stands for Month Start
plt.xticks(daily_ticks,rotation=45)
plt.show()
```



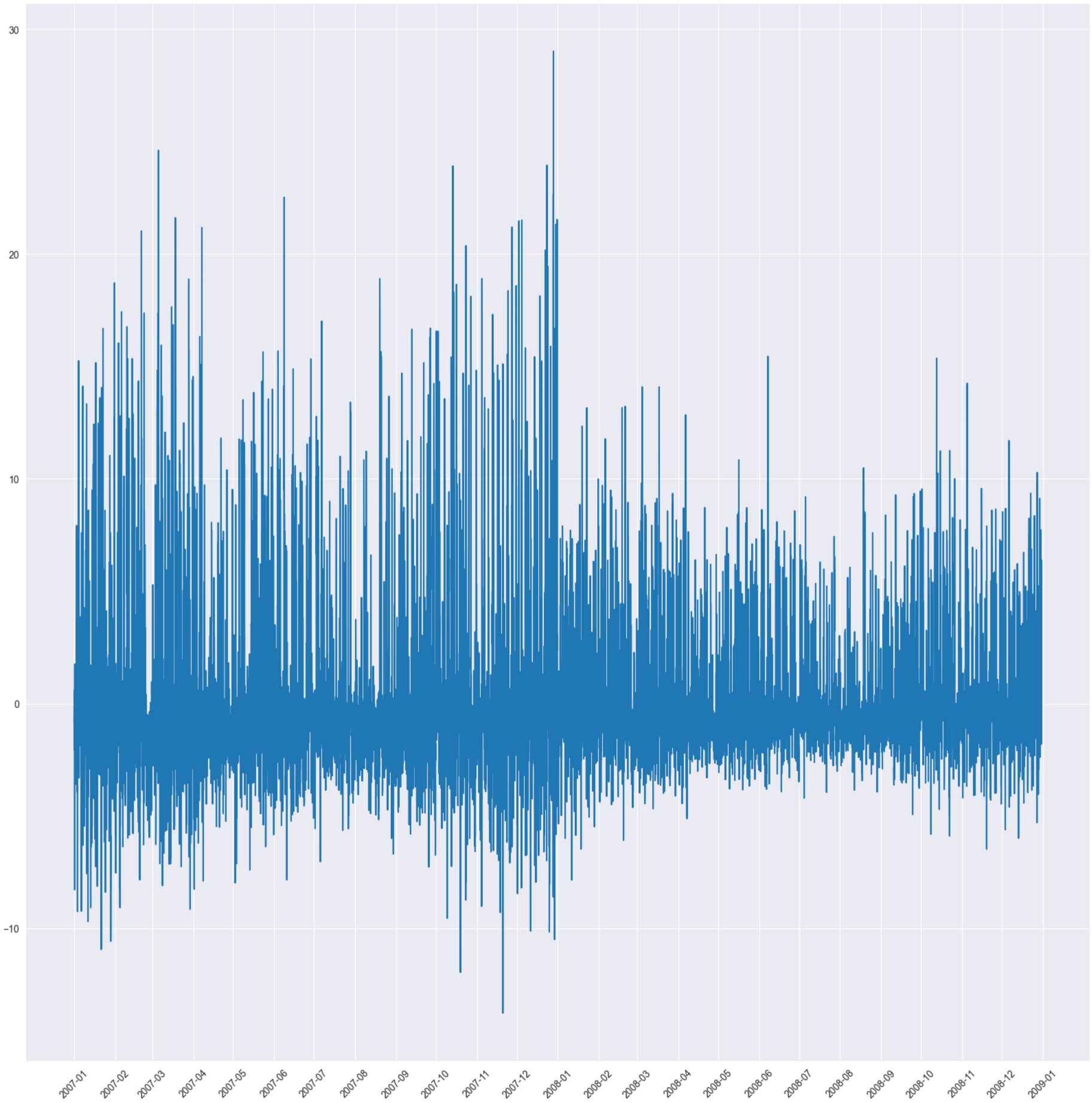
Visual A2: Yearly Seasonality Sub-Metering 1

In [106]:
```
df_8760 = mstl_dfsub1._seasonal.seasonal_8760[:8760*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_8760h = pd.date_range(start='2007-01-01', periods=8760*2+1, freq='H')
ax.plot(date_range_8760h, df_8760),
monthly_ticks = pd.date_range(start='2007-01-01', end='2009-01-01',freq='MS')  # MS stands for Month Start
plt.xticks(monthly_ticks, [x.strftime('%Y-%m') for x in monthly_ticks], rotation=45)
plt.show()
```



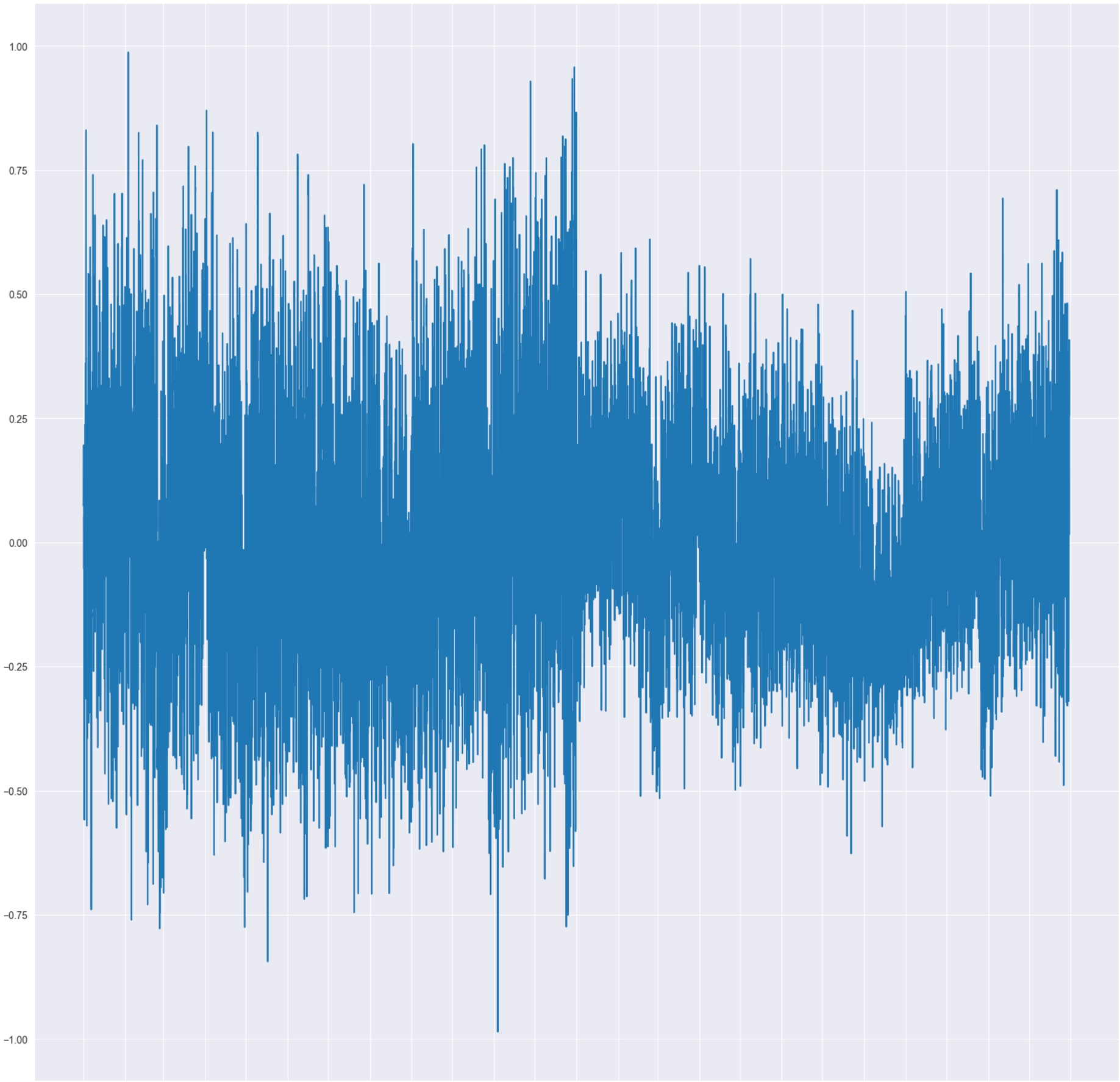
Visual A3: Yearly Seasonality Sub-Metering 2

In [107]:
```
df_8760 = mstl_dfsub2._seasonal.seasonal_8760[:8760*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_8760h = pd.date_range(start='2007-01-01', periods=8760*2+1, freq='H')
ax.plot(date_range_8760h, df_8760),
monthly_ticks = pd.date_range(start='2007-01-01', end='2009-01-01',freq='MS')  # MS stands for Month Start
plt.xticks(monthly_ticks, [x.strftime('%Y-%m') for x in monthly_ticks], rotation=45)
plt.show()
```



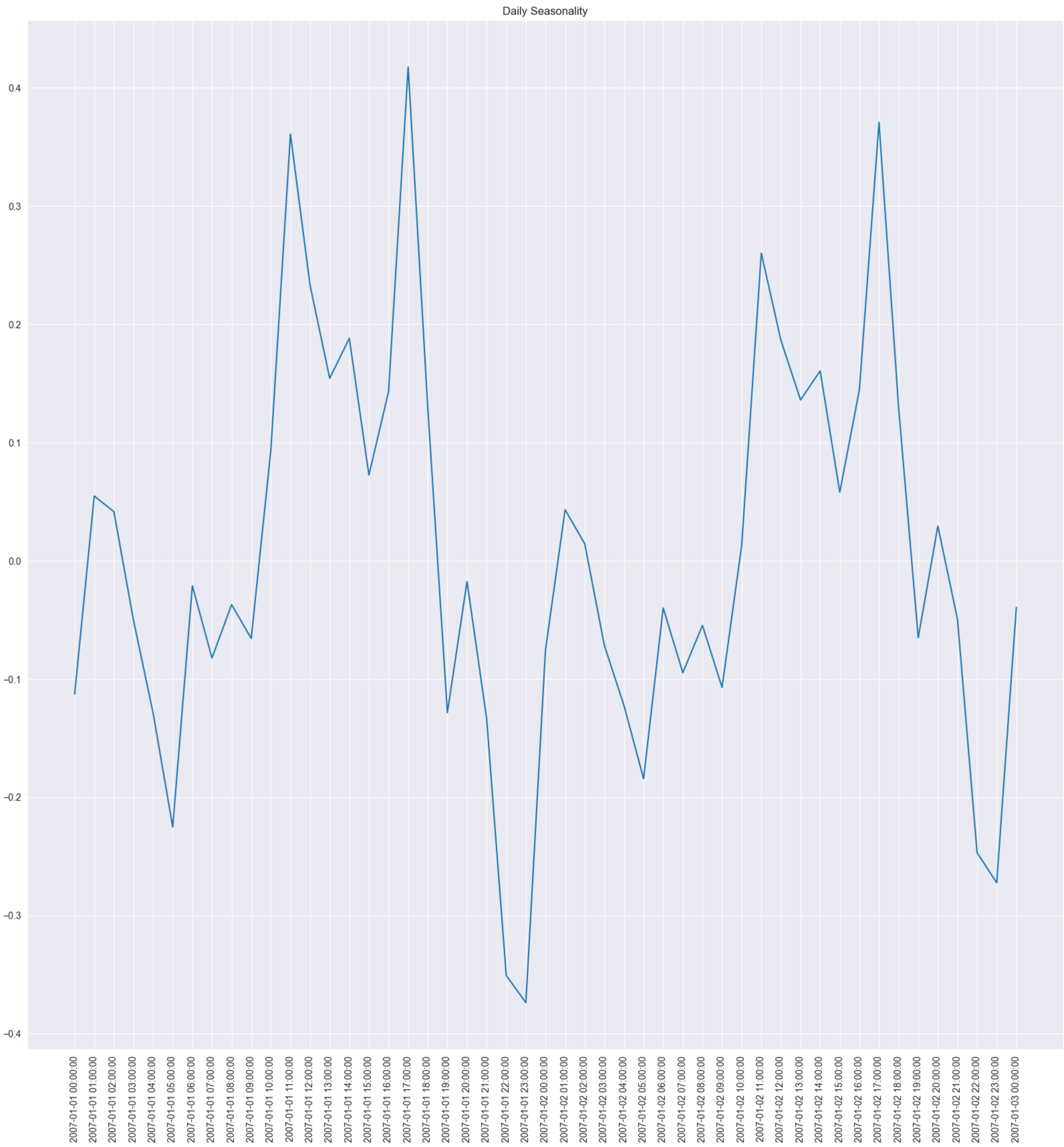
Visual A4: Yearly Seasonality Sub-Metering 3

In [108]:
```
df_8760 = mstl_dfsub3._seasonal.seasonal_8760[:8760*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_8760h = pd.date_range(start='2007-01-01', periods=8760*2+1, freq='H')
ax.plot(date_range_8760h, df_8760),
monthly_ticks = pd.date_range(start='2007-01-01', end='2009-01-01',freq='MS')  # MS stands for Month Start
plt.xticks(monthly_ticks, [x.strftime('%Y-%m') for x in monthly_ticks], rotation=45)
plt.show()
```



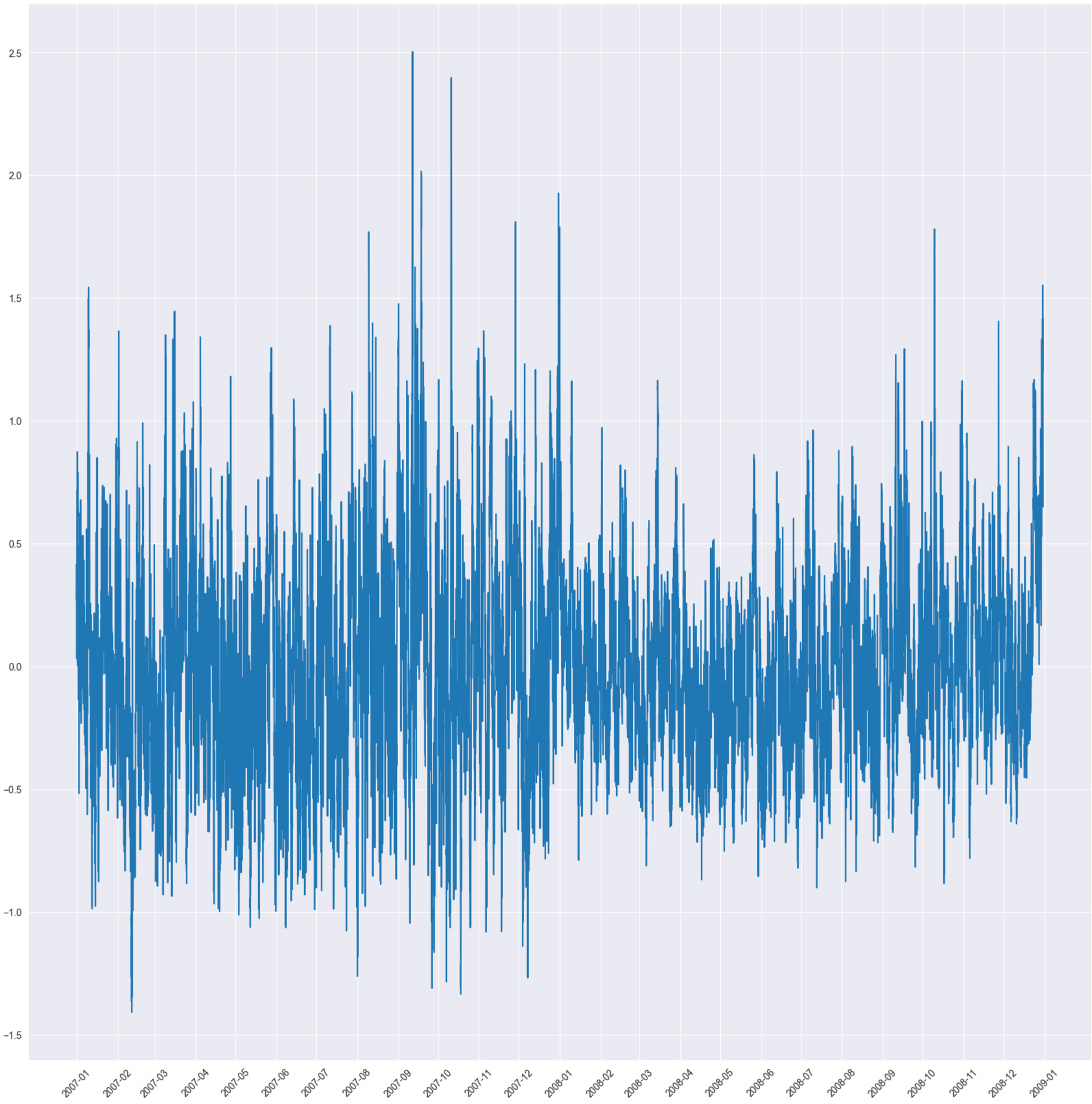
Visual A5: Daily Seasonality Wind Speed

In [109]:
```
wind_24 = mstl_wind._seasonal.seasonal_24[:49]
fig, ax = plt.subplots(figsize=(20, 20))
wind_24.plot(title='Daily Seasonality')
date_range_48h = pd.date_range(start='2007-01-01', periods=49, freq='H')
plt.xticks(ticks=range(1758, 1807), labels=date_range_48h, rotation = 90)
plt.show()
```



Visual A6: Yearly Seasonality Wind Speed

In [110]:
```
df_8760 = mstl_wind._seasonal.seasonal_8760[:8760*2+1]
fig, ax = plt.subplots(figsize=(20, 20))
date_range_8760h = pd.date_range(start='2007-01-01', periods=8760*2+1, freq='H')
ax.plot(date_range_8760h, df_8760),
monthly_ticks = pd.date_range(start='2007-01-01', end='2009-01-01',freq='MS')  # MS stands for Month Start
plt.xticks(monthly_ticks, [x.strftime('%Y-%m') for x in monthly_ticks], rotation=45)
plt.show()
```



Visual A7: Full MSTL Output Wind Speed

In [111]:
```
mstl_wind.plot()
plt.tight_layout()
```

Visual A8: Full MSTL Output Precipitation Depth

In [112]:
```
ISD_prec = ISD.resample('D', on = 'New_Date').Prec_Depth_Norm.mean()
ISD_prec = ISD_prec.reset_index()
prec_null = ISD_prec.isna().sum()
print(prec_null)
ISD_prec.set_index('New_Date')
```

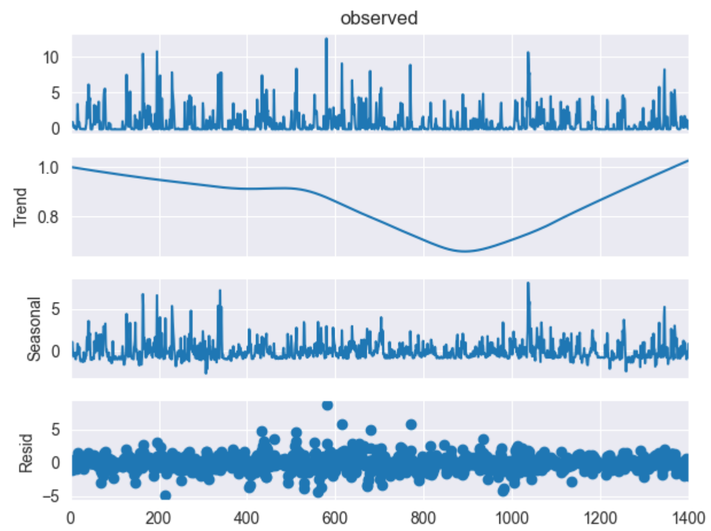
```
New_Date          0
Prec_Depth_Norm    0
dtype: int64
```

Out[112]:

New_Date	Prec_Depth_Norm
2007-01-01	1.465909
2007-01-02	0.386364
2007-01-03	0.488636
2007-01-04	0.727273
2007-01-05	0.923636
...	...
2010-10-28	0.392727
2010-10-29	1.196970
2010-10-30	0.465035
2010-10-31	0.000000
2010-11-01	0.000000

1401 rows × 1 columns

In [113]:
```
mstl = MSTL(ISD_prec['Prec_Depth_Norm'].interpolate(option='spline'), periods=365)
mstl_prec = mstl.fit()
mstl_prec.plot()
plt.show()
```



Visual A9: Durbin Watson results for each feature

In [114]:
```
from statsmodels.stats.stattools import durbin_watson
resid_test = []
dw_df = durbin_watson(mstl_df.resid)
dw_df1 = durbin_watson(mstl_dfsub1.resid)
dw_df2 = durbin_watson(mstl_dfsub2.resid)
dw_df3 = durbin_watson(mstl_dfsub3.resid)
dw_CCI = durbin_watson(mstl_CCI.resid)
dw_CPI= durbin_watson(mstl_CPI.resid)
dw_trend1 = durbin_watson(mstl_trend1.resid)
dw_trend2 = durbin_watson(mstl_trend2.resid)
dw_trend3 = durbin_watson(mstl_trend3.resid)
dw_airtmp = durbin_watson(mstl_ISD.resid)
dw_wind = durbin_watson(mstl_wind.resid)
dw_prec = durbin_watson(mstl_prec.resid)
resid_test.extend([dw_df, dw_df1, dw_df2, dw_df3, dw_CCI, dw_CPI, dw_trend1, dw_trend2, dw_trend3, dw_airtmp, dw_wind, dw_prec])
DW_Stat = pd.DataFrame(resid_test, index=['Global active power', 'Sub-metering 1', 'Sub-metering 2', 'Sub-metering 3', 'Consumer Confidence Score', 'CPI Index', 'Search Term 1', 'Search Term 2', 'Search Term 3', 'Air Temp', 'Wind Speed',
'Precipitation Depth'])
DW_Stat
```

Out[114]:

	0
Global active power	0.775400
Sub-metering 1	1.257313
Sub-metering 2	1.230871
Sub-metering 3	0.997189
Consumer Confidence Score	0.567085
CPI Index	0.344158
Search Term 1	1.276659
Search Term 2	0.763675
Search Term 3	1.220660
Air Temp	0.039417
Wind Speed	0.179383
Precipitation Depth	0.983584

```
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

CPI

```
df = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/raw/France_CPI_Electricity.csv")
df.isnull().values.sum()
df.head()
```

	DATE	CP0450FRM086NEST
0	2006-01-01	70.63
1	2006-02-01	70.88
2	2006-03-01	71.11
3	2006-04-01	73.06
4	2006-05-01	73.73

```
df['DATE'] = pd.to_datetime(df['DATE'],format="%Y-%m-%d")
```

```
df=df.query("DATE>=2007 and DATE<='2010-11'")
```

```
df.columns= ['Date','CPI']
df[['Date','CPI']].to_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YM_Clean_France_CPI_Electricity.csv",index=None)
```

Main File - Power Consumption

```
df =pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/raw/UCI_Power_Consumption_Dataset.txt",sep=';')
df.isnull().values.sum()
```

```
<ipython-input-54-95d0e62cd5e1>:1: DtypeWarning: Columns (2,3,4,5,6,7) have mixed types. Specify dtype option on import or set low_memory=False.
  df =pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/raw/UCI_Power_Consumption_Dataset.txt",sep=';')
25979
```

```
df['Global_active_power']= df['Global_active_power'].interpolate(option='spline')
df['Global_reactive_power']= df['Global_reactive_power'].interpolate(option='spline')
df['Voltage']= df['Voltage'].interpolate(option='spline')
df['Global_intensity']= df['Global_intensity'].interpolate(option='spline')
df['Sub_metering_1']= df['Sub_metering_1'].interpolate(option='spline')
df['Sub_metering_2']= df['Sub_metering_2'].interpolate(option='spline')
df['Sub_metering_3']= df['Sub_metering_3'].interpolate(option='spline')
```

```
df.isnull().values.sum()
```

```
0
```

```
df['Date'] = pd.to_datetime(df['Date'],format="%d/%m/%Y")
df.query('Date>=2007').head()
```

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
21996	2007-01-01	00:00:00	2.580	0.136	241.970	10.600	0.000	0.000	0.0
21997	2007-01-01	00:01:00	2.552	0.100	241.750	10.400	0.000	0.000	0.0
21998	2007-01-01	00:02:00	2.550	0.100	241.640	10.400	0.000	0.000	0.0
21999	2007-01-01	00:03:00	2.550	0.100	241.710	10.400	0.000	0.000	0.0
22000	2007-01-01	00:04:00	2.554	0.100	241.980	10.400	0.000	0.000	0.0

```
df.to_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMH_Clean_UCI_Power_Consumption_Dataset.csv",index=None)
```

Temp_ISD

```
df2007 = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/raw/France_Temp_ISD/ISD_2007.csv")
df2008 = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/raw/France_Temp_ISD/ISD_2008.csv")
df2009 = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/raw/France_Temp_ISD/ISD_2009.csv")
df2010 = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/raw/France_Temp_ISD/ISD_2010.csv")
```

```
df2010.head()
```

	STATION	DATE	SOURCE	LATITUDE	LONGITUDE	ELEVATION	NAME	REPORT_TYPE	CALL_S
0	7005099999	2010-01-01T00:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	
1	7005099999	2010-01-01T01:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	
2	7005099999	2010-01-01T02:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	
3	7005099999	2010-01-01T03:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	
4	7005099999	2010-01-01T04:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	

5 rows × 43 columns

```
df2010['DATE'] = pd.to_datetime(df2010['DATE'],format='%Y-%m-%dT%H:%M:%S')
df2010=df2010.query("DATE<='2010-11-26'")
```

```
frames = [df2007, df2008,df2009,df2010]
df = pd.concat(frames)
df.head()
```

	STATION	DATE	SOURCE	LATITUDE	LONGITUDE	ELEVATION	NAME	REPORT_TYPE	CALL_S
0	7005099999	2007-01-01T00:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	
1	7005099999	2007-01-01T01:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	
2	7005099999	2007-01-01T02:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	
3	7005099999	2007-01-01T03:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	
4	7005099999	2007-01-01T04:00:00	4	50.143492	1.831892	67.05	ABBEVILLE, FR	FM-12	

5 rows × 43 columns

```
df.to_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMH_Clean_TempISD.csv")

```

CCI

```
df_cci = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/raw/CCI_monthly_values.csv",sep=';')
df_cci=df_cci.iloc[3:,:2]
df_cci.columns = ['Date', 'CCI']
df_cci=df_cci.query("Date>='2007' and Date<='2010-11'")
df_cci[['Date', 'CCI']].to_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YM_Clean_CCI.csv",index=False)
df_cci.head()
```

	Date	CCI
15	2007-01	100
16	2007-02	100
17	2007-03	101
18	2007-04	102
19	2007-05	108

Google Trends

```
df1 = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/raw/GoogleTrends1.csv")
df2 = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/raw/GoogleTrends2.csv")
df3 = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/raw/GoogleTrends3.csv")
```

```
df1['Week'] = pd.to_datetime(df1['Week'],format='%m/%d/%y')
df1.columns = ['Date', 'Hits']
df1[['Date', 'Hits']].to_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMW_Clean_GoogleTrends1.csv",index=False)

```

```
df2['Week'] = pd.to_datetime(df2['Week'],format='%m/%d/%y')
df2
df2.columns = ['Date', 'Hits']
df2[['Date', 'Hits']].to_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMW_Clean_GoogleTrends2.csv",index=False)

```

```
df3['Week'] = pd.to_datetime(df3['Week'],format='%m/%d/%y')
df3.columns = ['Date', 'Hits']
df3[['Date', 'Hits']].to_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMW_Clean_GoogleTrends3.csv",index=False)

```

Temp

```
df=pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/raw/dataexport_20231114T035544.csv",skiprows=9)
df.head()
```

	timestamp	Basel Temperature [2 m elevation corrected]	Basel Precipitation Total	Basel Relative Humidity [2 m]	Basel Sunshine Duration
0	20070101T0000	55.544440	0.5	74.879456	0.0
1	20070101T0100	56.174440	0.7	78.361946	0.0
2	20070101T0200	56.912440	0.6	81.754720	0.0
3	20070101T0300	56.606440	0.5	83.333110	0.0
4	20070101T0400	56.012444	0.9	89.276825	0.0

```
df['timestamp'] = pd.to_datetime(df['timestamp'],format='%Y-%m-%dT%H:%M:%S')
df.columns = ['Date', 'Temperature','Precipitation', 'Humidity','Sunshine']
df.head()
#df.to_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMH_Weather.csv",index=False)

```

	Date	Temperature	Precipitation	Humidity	Sunshine
0	2007-01-01 00:00:00	55.544440	0.5	74.879456	0.0
1	2007-01-01 01:00:00	56.174440	0.7	78.361946	0.0
2	2007-01-01 02:00:00	56.912440	0.6	81.754720	0.0
3	2007-01-01 03:00:00	56.606440	0.5	83.333110	0.0
4	2007-01-01 04:00:00	56.012444	0.9	89.276825	0.0

```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.graph_objs as go

##Show plots in Colab
import plotly.io as pio
pio.renderers.default = "colab"
```

```
def m_normalize(df):
  adf= (df-df.min())/(df.max()-df.min())
  return adf
```

Power Consumption Dataset

```
df_main=pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMH_Clean_UCI_Power_Consumption_Dataset.csv")
df_main.head()
```

```
<ipython-input-16-2a949e5a53fe>:1: DtypeWarning:

Columns (2,3,4,5,6,7) have mixed types. Specify dtype option on import or set low_memory=False.
```

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
0	2006-12-16	17:24:00	4.216	0.418	234.840	18.400	0.000	1.000	17.0
1	2006-12-16	17:25:00	5.360	0.436	233.630	23.000	0.000	1.000	16.0
2	2006-12-16	17:26:00	5.374	0.498	233.290	23.000	0.000	2.000	17.0
3	2006-12-16	17:27:00	5.388	0.502	233.740	23.000	0.000	1.000	17.0
4	2006-12-16	17:28:00	3.666	0.528	235.680	15.800	0.000	1.000	17.0

Grouping Data by Date

```
df_main['Date'] = pd.to_datetime(df_main['Date'])
df_main_2009=df_main.query("Date>='2009-01-01' and Date<='2009-12-31'")
df_main_2009=df_main_2009[['Date','Global_active_power' ,'Global_reactive_power'    ,'Voltage'  ,'Global_intensity' ,'Sub_metering_1'   ,'Sub_met
df_main_2009['Global_active_power']   = df_main_2009['Global_active_power'].apply(pd.to_numeric, errors = 'coerce').dropna().astype(float)
df_main_2009['Global_reactive_power'] = df_main_2009['Global_reactive_power'].apply(pd.to_numeric, errors = 'coerce').dropna().astype(float)
df_main_2009['Voltage']               = df_main_2009['Voltage'].apply(pd.to_numeric, errors = 'coerce').dropna().astype(float)
df_main_2009['Global_intensity']      = df_main_2009['Global_intensity'].apply(pd.to_numeric, errors = 'coerce').dropna().astype(float)
df_main_2009['Sub_metering_1']        = df_main_2009['Sub_metering_1'].apply(pd.to_numeric, errors = 'coerce').dropna().astype(float)
df_main_2009['Sub_metering_2']        = df_main_2009['Sub_metering_2'].apply(pd.to_numeric, errors = 'coerce').dropna().astype(float)
df_main_2009['Sub_metering_3']        = df_main_2009['Sub_metering_3'].apply(pd.to_numeric, errors = 'coerce').dropna().astype(float)
df_main_2009_sum=df_main_2009.groupby(pd.Grouper(key='Date', axis=1, freq='D'),as_index=True).sum().reset_index()
df_main_2009_sum.head()
```

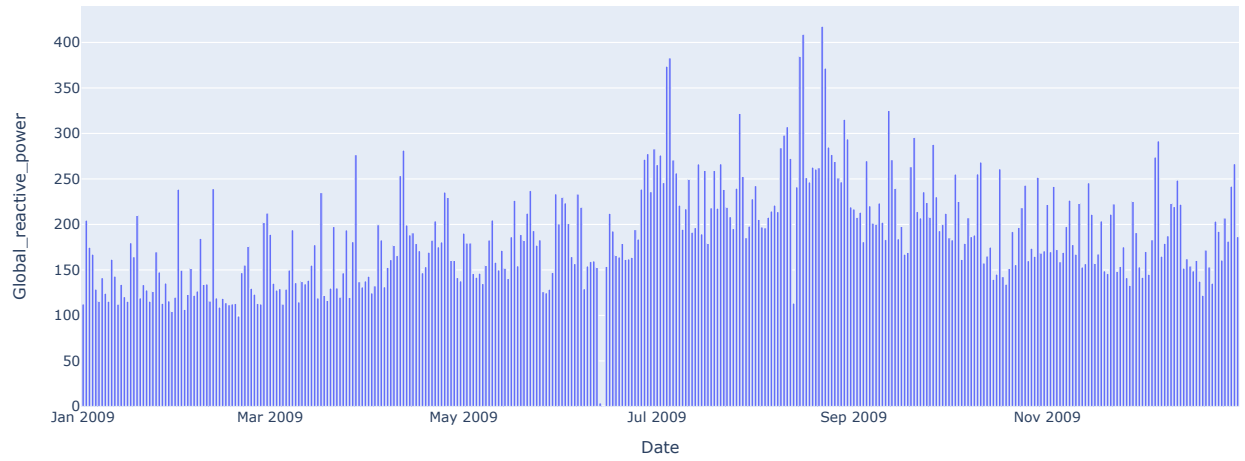
	Date	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
0	2009-01-01	1406.692	112.668	349683.93	5975.8	765.0	3845.0	2803.0
1	2009-01-02	2327.510	204.730	349495.04	9702.4	2417.0	6018.0	11841.0
2	2009-01-03	1913.496	174.912	350901.30	7964.4	6440.0	328.0	10439.0
3	2009-01-04	1939.994	167.338	352544.30	8111.8	1457.0	5251.0	3245.0
4	2009-01-05	1604.000	128.938	352556.32	6676.2	1247.0	322.0	8650.0

```
df_main_2009_sum.query("Date>='2009-06' and Date<'2009-07'").head()
```

	Date	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
151	2009-06-01	1443.260	229.876	347521.14	6208.2	2789.0	461.0	11361.0
152	2009-06-02	1090.638	223.696	347947.63	4849.6	0.0	506.0	7892.0
153	2009-06-03	1356.578	200.926	346184.53	5870.4	940.0	1370.0	11577.0
154	2009-06-04	1035.660	164.662	345614.69	4440.2	1906.0	455.0	6905.0
155	2009-06-05	1230.360	156.950	345403.02	5256.0	0.0	456.0	12515.0

```
fig = go.Figure()
fig.add_trace(go.Bar(x = df_main_2009_sum['Date'],y=df_main_2009_sum['Global_reactive_power'], name = 'Actual'))
```

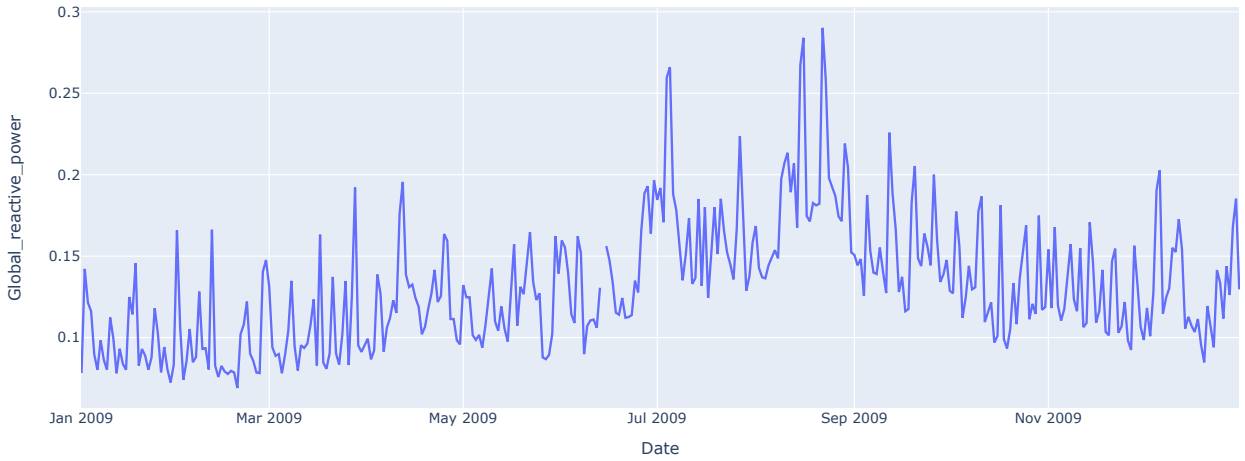
```
fig.update_layout(xaxis_title = 'Date', yaxis_title = 'Global_reactive_power')
fig.show()
```




```
1
```

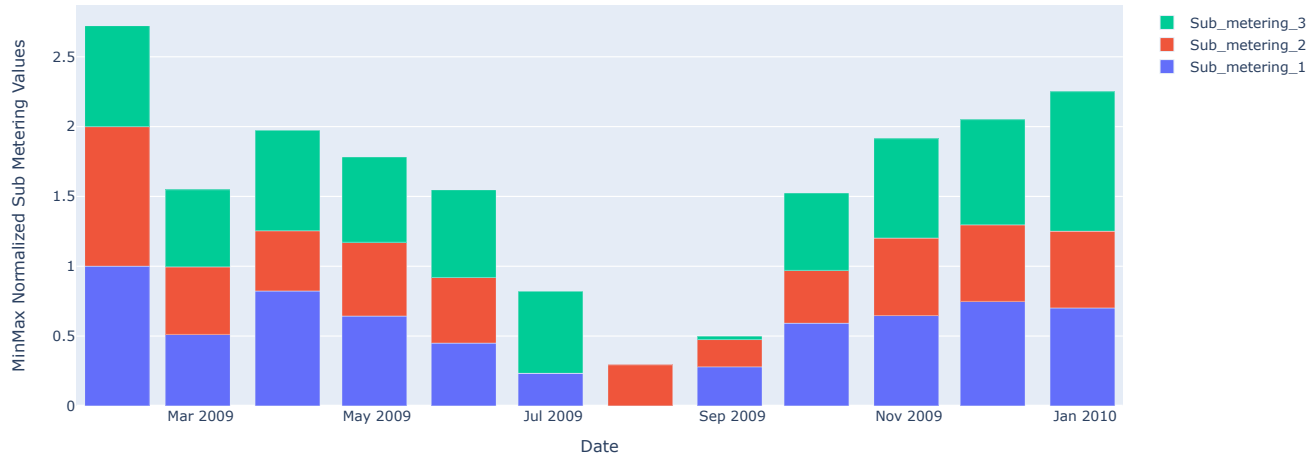
Grouping Data by Month

```
df_main_2009_mean=df_main_2009.groupby(pd.Grouper(key='Date', axis=1, freq='D'),as_index=True).mean().reset_index()
#df_main_2009_mean['Global_reactive_power']=df_main_2009_mean['Global_reactive_power'].apply(lamda x:x)
#df_main_2009_sum=df_main_2009.groupby(['Date'], as_index=False)['Global_reactive_power'].sum()
fig = go.Figure()
#fig.add_trace(go.Scatter(x = df_main_2009_sum['Date'],y=df_main_2009_sum['Global_reactive_power'], mode = 'lines', name = 'Sum'))
fig.add_trace(go.Scatter(x = df_main_2009_mean['Date'],y=df_main_2009_mean['Global_reactive_power'], mode = 'lines', name = 'Mean'))
fig.update_layout(xaxis_title = 'Date', yaxis_title = 'Global_reactive_power')
fig.show()
```




```
df_main_2009_sum=df_main_2009.groupby(pd.Grouper(key='Date', axis=1, freq='M'),as_index=True).sum().reset_index()
#df_main_2009_sum['Date']  = df_main_2009_sum['Date'] .dt.to_period('M')
#df_main_2009_sum=df_main_2009.groupby(['Date'], as_index=False)['Global_reactive_power'].sum()
#df_main_2009_sum['Global_reactive_power'] = m_normalize(df_main_2009_sum['Global_reactive_power'])
df_main_2009_sum['Global_active_power']   = m_normalize(df_main_2009_sum['Global_active_power'])
df_main_2009_sum['Global_reactive_power'] = m_normalize(df_main_2009_sum['Global_reactive_power'])
df_main_2009_sum['Voltage']               = m_normalize(df_main_2009_sum['Voltage'])
df_main_2009_sum['Global_intensity']      = m_normalize(df_main_2009_sum['Global_intensity'])
df_main_2009_sum['Sub_metering_1']        = m_normalize(df_main_2009_sum['Sub_metering_1'])
df_main_2009_sum['Sub_metering_2']        = m_normalize(df_main_2009_sum['Sub_metering_2'])
df_main_2009_sum['Sub_metering_3']        = m_normalize(df_main_2009_sum['Sub_metering_3'])
```

```
fig = go.Figure()
#fig.add_trace(go.Bar(x = df_main_2009_sum['Date'],y=df_main_2009_sum['Global_active_power'],offsetgroup=0,name='Global_active_power'))
#fig.add_trace(go.Bar(x = df_main_2009_sum['Date'],y=df_main_2009_sum['Global_reactive_power'],offsetgroup=0,name='Global_reactive_power'))
#fig.add_trace(go.Bar(x = df_main_2009_sum['Date'],y=df_main_2009_sum['Voltage'],offsetgroup=0,name='Voltage'))
#fig.add_trace(go.Bar(x = df_main_2009_sum['Date'],y=df_main_2009_sum['Global_intensity'],offsetgroup=0,name='Global_intensity'))
fig.add_trace(go.Bar(x = df_main_2009_sum['Date'],y=df_main_2009_sum['Sub_metering_1'],offsetgroup=0,name='Sub_metering_1'))
fig.add_trace(go.Bar(x = df_main_2009_sum['Date'],y=df_main_2009_sum['Sub_metering_2'],offsetgroup=0,name='Sub_metering_2'))
fig.add_trace(go.Bar(x = df_main_2009_sum['Date'],y=df_main_2009_sum['Sub_metering_3'],offsetgroup=0,name='Sub_metering_3'))

fig.update_layout(barmode='stack',xaxis_title = 'Date', yaxis_title = 'MinMax Normalized Sub Metering Values')
fig.show()
```



Google Trends

```
df_gt1=pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMW_Clean_GoogleTrends1.csv")
df_gt1['Date'] = pd.to_datetime(df_gt1['Date'])
df_gt1=df_gt1.query("Date>='2009-01-01' and Date<='2009-12-31'")

df_gt2=pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMW_Clean_GoogleTrends2.csv")
df_gt2['Date'] = pd.to_datetime(df_gt2['Date'])
df_gt2=df_gt2.query("Date>='2009-01-01' and Date<='2009-12-31'")

df_gt3=pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMW_Clean_GoogleTrends3.csv")
df_gt3['Date'] = pd.to_datetime(df_gt3['Date'])
df_gt3=df_gt3.query("Date>='2009-01-01' and Date<='2009-12-31'")

```

CPI

```
df_cpi=pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YM_Clean_France_CPI_Electricity.csv")
df_cpi['Date'] = pd.to_datetime(df_cpi['Date'])
df_cpi=df_cpi.query("Date>='2009-01-01' and Date<='2009-12-31'")
```

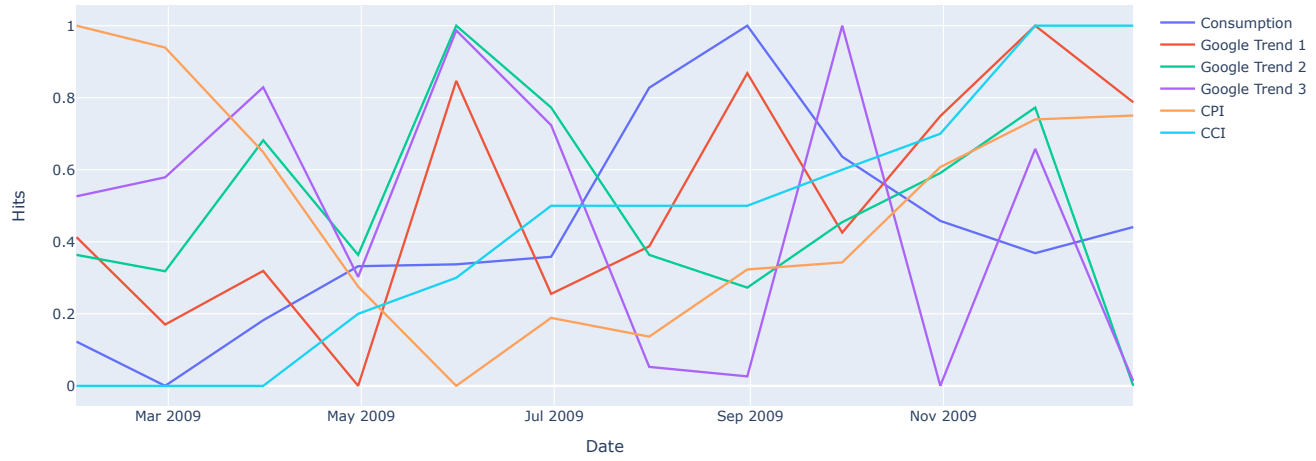
CCI

```
df_cci=pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YM_Clean_CCI.csv")
df_cci['Date'] = pd.to_datetime(df_cci['Date'])
df_cci=df_cci.query("Date>='2009-01-01' and Date<='2009-12-31'")
df_cci.head()
```

	Date	CCI
26	2009-03-01	84

Month Based Overlay Chart for all Datasets

```
df_gt1_sum=df_gt1.groupby(pd.Grouper(key='Date', axis=1, freq='M'),as_index=True).sum().reset_index()
df_gt1_sum['Hits']=m_normalize(df_gt1_sum['Hits'])

df_gt2_sum=df_gt2.groupby(pd.Grouper(key='Date', axis=1, freq='M'),as_index=True).sum().reset_index()
df_gt2_sum['Hits']=m_normalize(df_gt2_sum['Hits'])

df_gt3_sum=df_gt3.groupby(pd.Grouper(key='Date', axis=1, freq='M'),as_index=True).sum().reset_index()
df_gt3_sum['Hits']=m_normalize(df_gt3_sum['Hits'])

df_cpi_sum=df_cpi.groupby(pd.Grouper(key='Date', axis=1, freq='M'),as_index=True).sum().reset_index()
df_cpi_sum['CPI']=m_normalize(df_cpi_sum['CPI'])

df_cci_sum=df_cci.groupby(pd.Grouper(key='Date', axis=1, freq='M'),as_index=True).sum().reset_index()
df_cci_sum['CCI']=m_normalize(df_cci_sum['CCI'])


fig = go.Figure()
fig.add_trace(go.Scatter(x = df_main_2009_sum['Date'],y=df_main_2009_sum['Global_reactive_power'], mode = 'lines', name = 'Consumptior
fig.add_trace(go.Scatter(x = df_gt1_sum['Date'],y=df_gt1_sum['Hits'], mode = 'lines', name = 'Google Trend 1'))
fig.add_trace(go.Scatter(x = df_gt2_sum['Date'],y=df_gt2_sum['Hits'], mode = 'lines', name = 'Google Trend 2'))
fig.add_trace(go.Scatter(x = df_gt3_sum['Date'],y=df_gt3_sum['Hits'], mode = 'lines', name = 'Google Trend 3'))
fig.add_trace(go.Scatter(x = df_cpi_sum['Date'],y=df_cpi_sum['CPI'], mode = 'lines', name = 'CPI'))
fig.add_trace(go.Scatter(x = df_cci_sum['Date'],y=df_cci_sum['CCI'], mode = 'lines', name = 'CCI'))

fig.update_layout(xaxis_title = 'Date', yaxis_title = 'Hits')
fig.show()
```




```
1
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```
import pandas as pd
```

In [ ]:
```
# df    = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMH_Clean_UCI_Power_Consu
# cpi   = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YM_Clean_France_CPI_Elect
# temp  = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMH_Clean_TempISD.csv")
# cci   = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YM_Clean_CCI.csv")
# gt1   = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMW_Clean_GoogleTrends1.c
# gt2   = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMW_Clean_GoogleTrends2.c
# gt3   = pd.read_csv("/content/drive/MyDrive/BDA 696 Project/Datasets/processsed/YMW_Clean_GoogleTrends3.c
```

In [ ]:
```
df    = pd.read_csv("Downloads/processsed/YMH_Clean_UCI_Power_Consumption_Dataset.csv")
cpi   = pd.read_csv("Downloads/processsed/YM_Clean_France_CPI_Electricity.csv")
temp  = pd.read_csv("Downloads/processsed/YMH_Clean_TempISD.csv")
cci   = pd.read_csv("Downloads/processsed/YM_Clean_CCI.csv")
gt1   = pd.read_csv("Downloads/processsed/YMW_Clean_GoogleTrends1.csv")
gt2   = pd.read_csv("Downloads/processsed/YMW_Clean_GoogleTrends2.csv")
gt3   = pd.read_csv("Downloads/processsed/YMW_Clean_GoogleTrends3.csv")
```

```
C:\Users\vghavate3103\AppData\Local\Temp\ipykernel_15040\2729922107.py:1: DtypeWarning: Columns (2,3,4,5,6,
7) have mixed types. Specify dtype option on import or set low_memory=False.
  df    = pd.read_csv("Downloads/processsed/YMH_Clean_UCI_Power_Consumption_Dataset.csv")
C:\Users\vghavate3103\AppData\Local\Temp\ipykernel_15040\2729922107.py:3: DtypeWarning: Columns (20) have m
ixed types. Specify dtype option on import or set low_memory=False.
  temp  = pd.read_csv("Downloads/processsed/YMH_Clean_TempISD.csv")
```

In [ ]:
```
df['Date'] = pd.to_datetime(df['Date'])
df=df.query('Date.dt.minute==0')# or Date.dt.minute==30')
df=df.replace('?',0.0)
df
```

Out[ ]:

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	S
0	2007-01-01 00:00:00	00:00:00	2.580	0.136	241.970	10.600	0.000	
60	2007-01-01 01:00:00	01:00:00	2.466	0.000	241.090	10.200	0.000	
120	2007-01-01 02:00:00	02:00:00	2.504	0.088	239.260	10.400	0.000	
180	2007-01-01 03:00:00	03:00:00	2.648	0.254	241.630	11.000	0.000	
240	2007-01-01 04:00:00	04:00:00	2.400	0.000	241.960	9.800	0.000	
...	...	...	...	...	...	...	...	
2053020	2010-11-26 17:00:00	17:00:00	0.898	0.0	237.22	3.8	0.0	
2053080	2010-11-26 18:00:00	18:00:00	1.096	0.142	239.14	4.6	0.0	
2053140	2010-11-26 19:00:00	19:00:00	1.81	0.0	235.59	7.6	0.0	
2053200	2010-11-26 20:00:00	20:00:00	1.456	0.0	238.18	6.2	0.0	
2053260	2010-11-26 21:00:00	21:00:00	0.938	0.0	239.82	3.8	0.0	

34222 rows × 9 columns

In [ ]:
```
#temp['Date'] = pd.to_datetime(cci['Date'])
#df=pd.merge_asof(df,temp, on='Date')

#df.dtypes

# cpi['Date'] = pd.to_datetime(cpi['Date'])
# df=df.merge(cpi, on='Date', how='left')

# cci['Date'] = pd.to_datetime(cci['Date'])
# df=df.merge(cci, on='Date', how='left')

gt1['Date'] = pd.to_datetime(gt1['Date'])
df=df.merge(gt1, on='Date', how='left')

gt2['Date'] = pd.to_datetime(gt2['Date'])
df=df.merge(gt2, on='Date', how='left')

gt3['Date'] = pd.to_datetime(gt3['Date'])
df=df.merge(gt3, on='Date', how='left')


df['GT1_Hits']=df['GT1_Hits'].fillna(0)
df['GT2_Hits']=df['GT2_Hits'].fillna(0)
df['GT3_Hits']=df['GT3_Hits'].fillna(0)
```

In [ ]:
```
df.head()
```

Out[ ]:

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_met
0	2007-01-01 00:00:00	00:00:00	2.580	0.136	241.970	10.600	0.000	
1	2007-01-01 01:00:00	01:00:00	2.466	0.000	241.090	10.200	0.000	
2	2007-01-01 02:00:00	02:00:00	2.504	0.088	239.260	10.400	0.000	
3	2007-01-01 03:00:00	03:00:00	2.648	0.254	241.630	11.000	0.000	
4	2007-01-01 04:00:00	04:00:00	2.400	0.000	241.960	9.800	0.000	

In [ ]:

```
!pip install lightning
!pip install pytorch_forecasting
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: lightning in c:\users\vghavate3103\appdata\roaming\python\python311\site-packages (2.1.2)
Requirement already satisfied: PyYAML<8.0,>=5.4 in c:\programdata\anaconda3\lib\site-packages (from lightning) (6.0)
Requirement already satisfied: fsspec[http]<2025.0,>2021.06.0 in c:\programdata\anaconda3\lib\site-packages (from lightning) (2023.4.0)
Requirement already satisfied: lightning-utilities<2.0,>=0.8.0 in c:\users\vghavate3103\appdata\roaming\python\python311\site-packages (from lightning) (0.10.0)
Requirement already satisfied: numpy<3.0,>=1.17.2 in c:\programdata\anaconda3\lib\site-packages (from lightning) (1.24.3)
Requirement already satisfied: packaging<25.0,>=20.0 in c:\programdata\anaconda3\lib\site-packages (from lightning) (23.1)
Requirement already satisfied: torch<4.0,>=1.12.0 in c:\users\vghavate3103\appdata\roaming\python\python311\site-packages (from lightning) (2.1.1)
Requirement already satisfied: torchmetrics<3.0,>=0.7.0 in c:\users\vghavate3103\appdata\roaming\python\python311\site-packages (from lightning) (1.2.1)
Requirement already satisfied: tqdm<6.0,>=4.57.0 in c:\programdata\anaconda3\lib\site-packages (from lightning) (4.65.0)
Requirement already satisfied: typing-extensions<6.0,>=4.0.0 in c:\programdata\anaconda3\lib\site-packages (from lightning) (4.7.1)
Requirement already satisfied: pytorch-lightning in c:\users\vghavate3103\appdata\roaming\python\python311\site-packages (from lightning) (2.1.2)
Requirement already satisfied: requests in c:\programdata\anaconda3\lib\site-packages (from fsspec[http]<2025.0,>2021.06.0->lightning) (2.31.0)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in c:\programdata\anaconda3\lib\site-packages (from fsspec[http]<2025.0,>2021.06.0->lightning) (3.8.5)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-packages (from lightning-utilities<2.0,>=0.8.0->lightning) (68.0.0)
Requirement already satisfied: filelock in c:\programdata\anaconda3\lib\site-packages (from torch<4.0,>=1.12.0->lightning) (3.9.0)
Requirement already satisfied: sympy in c:\programdata\anaconda3\lib\site-packages (from torch<4.0,>=1.12.0->lightning) (1.11.1)
Requirement already satisfied: networkx in c:\programdata\anaconda3\lib\site-packages (from torch<4.0,>=1.12.0->lightning) (3.1)
Requirement already satisfied: jinja2 in c:\programdata\anaconda3\lib\site-packages (from torch<4.0,>=1.12.0->lightning) (3.1.2)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-packages (from tqdm<6.0,>=4.57.0->lightning) (0.4.6)
Requirement already satisfied: attrs>=17.3.0 in c:\programdata\anaconda3\lib\site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (22.1.0)
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in c:\programdata\anaconda3\lib\site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (2.0.4)
Requirement already satisfied: multidict<7.0,>=4.5 in c:\programdata\anaconda3\lib\site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (6.0.2)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in c:\programdata\anaconda3\lib\site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (4.0.2)
Requirement already satisfied: yarl<2.0,>=1.0 in c:\programdata\anaconda3\lib\site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (1.8.1)
Requirement already satisfied: frozenlist>=1.1.1 in c:\programdata\anaconda3\lib\site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (1.3.3)
Requirement already satisfied: aiosignal>=1.1.2 in c:\programdata\anaconda3\lib\site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (1.2.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\programdata\anaconda3\lib\site-packages (from jinja2->torch<4.0,>=1.12.0->lightning) (2.1.1)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests->fsspec[http]<2025.0,>2021.06.0->lightning) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests->fsspec[http]<2025.0,>2021.06.0->lightning) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from requests->fsspec[http]<2025.0,>2021.06.0->lightning) (2023.7.22)
Requirement already satisfied: mpmath>=0.19 in c:\programdata\anaconda3\lib\site-packages (from sympy->torch<4.0,>=1.12.0->lightning) (1.3.0)
Defaulting to user installation because normal site-packages is not writeable
Collecting pytorch_forecasting
  Using cached pytorch_forecasting-0.10.1-py3-none-any.whl (127 kB)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages (from pytorch_forecasting) (3.7.2)
Collecting optuna<3.0.0,>=2.3.0 (from pytorch_forecasting)
  Using cached optuna-2.10.1-py3-none-any.whl (308 kB)
Collecting pandas<2.0.0,>=1.3.0 (from pytorch_forecasting)
  Using cached pandas-1.5.3-cp311-cp311-win_amd64.whl (10.3 MB)
Collecting pytorch-lightning<2.0.0,>=1.2.4 (from pytorch_forecasting)
  Using cached pytorch_lightning-1.9.5-py3-none-any.whl (829 kB)
Collecting scikit-learn<1.1,>=0.24 (from pytorch_forecasting)
```

```
Using cached scikit-learn-1.0.2.tar.gz (6.7 MB)
Installing build dependencies: started
Installing build dependencies: finished with status 'done'
Getting requirements to build wheel: started
Getting requirements to build wheel: finished with status 'done'
Preparing metadata (pyproject.toml): started
Preparing metadata (pyproject.toml): finished with status 'error'
```

```
  error: subprocess-exited-with-error

  Preparing metadata (pyproject.toml) did not run successfully.
  exit code: 1

  [66 lines of output]
  Partial import of sklearn during the build process.
  setup.py:128: DeprecationWarning:

    `numpy.distutils` is deprecated since NumPy 1.23.0, as a result
    of the deprecation of `distutils` itself. It will be removed for
    Python >= 3.12. For older Python versions it will remain present.
    It is recommended to use `setuptools < 60.0` for those Python versions.
    For more details, see:
      https://numpy.org/devdocs/reference/distutils_status_migration.html


    from numpy.distutils.command.build_ext import build_ext  # noqa
  INFO: No module named 'numpy.distutils._msvccompiler' in numpy.distutils; trying from distutils
  Traceback (most recent call last):
    File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_vendor\pyproject_hooks\_in_process\_in_process.p
y", line 353, in <module>
      main()
    File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_vendor\pyproject_hooks\_in_process\_in_process.p
y", line 335, in main
      json_out['return_val'] = hook(**hook_input['kwargs'])
                               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_vendor\pyproject_hooks\_in_process\_in_process.p
y", line 149, in prepare_metadata_for_build_wheel
      return hook(metadata_directory, config_settings)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-build-env-5kpyzdik\overlay\Lib\site-packages\setupto
ols\build_meta.py", line 174, in prepare_metadata_for_build_wheel
      self.run_setup()
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-build-env-5kpyzdik\overlay\Lib\site-packages\setupto
ols\build_meta.py", line 268, in run_setup
      self).run_setup(setup_script=setup_script)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-build-env-5kpyzdik\overlay\Lib\site-packages\setupto
ols\build_meta.py", line 158, in run_setup
      exec(compile(code, __file__, 'exec'), locals())
    File "setup.py", line 319, in <module>
      setup_package()
    File "setup.py", line 315, in setup_package
      setup(**metadata)
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-build-env-5kpyzdik\overlay\Lib\site-packages\numpy\d
istutils\core.py", line 135, in setup
      config = configuration()
               ^^^^^^^^^^^^^^^
    File "setup.py", line 201, in configuration
      config.add_subpackage("sklearn")
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-build-env-5kpyzdik\overlay\Lib\site-packages\numpy\d
istutils\misc_util.py", line 1050, in add_subpackage
      config_list = self.get_subpackage(subpackage_name, subpackage_path,
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-build-env-5kpyzdik\overlay\Lib\site-packages\numpy\d
istutils\misc_util.py", line 1016, in get_subpackage
      config = self._get_configuration_from_setup_py(
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-build-env-5kpyzdik\overlay\Lib\site-packages\numpy\d
istutils\misc_util.py", line 958, in _get_configuration_from_setup_py
      config = setup_module.configuration(*args)
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-install-vz5byaae\scikit-learn_8388def4b00a47b99534f7
99919104e2\sklearn\setup.py", line 85, in configuration
      cythonize_extensions(top_path, config)
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-install-vz5byaae\scikit-learn_8388def4b00a47b99534f7
99919104e2\sklearn\_build_utils\__init__.py", line 47, in cythonize_extensions
      basic_check_build()
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-install-vz5byaae\scikit-learn_8388def4b00a47b99534f7
99919104e2\sklearn\_build_utils\pre_build_helpers.py", line 114, in basic_check_build
      compile_test_program(code)
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-install-vz5byaae\scikit-learn_8388def4b00a47b99534f7
99919104e2\sklearn\_build_utils\pre_build_helpers.py", line 70, in compile_test_program
      ccompiler.compile(
```

```
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-build-env-5kpyzdik\overlay\Lib\site-packages\setuptools\_distutils\_msvccompiler.py", line 327, in compile
      self.initialize()
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-build-env-5kpyzdik\overlay\Lib\site-packages\setuptools\_distutils\_msvccompiler.py", line 224, in initialize
      vc_env = _get_vc_env(plat_spec)
               ^^^^^^^^^^^^^^^^^^^^^^
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-build-env-5kpyzdik\overlay\Lib\site-packages\setuptools\msvc.py", line 316, in msvc14_get_vc_env
      return _msvc14_get_vc_env(plat_spec)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "C:\Users\vghavate3103\AppData\Local\Temp\pip-build-env-5kpyzdik\overlay\Lib\site-packages\setuptools\msvc.py", line 270, in _msvc14_get_vc_env
      raise distutils.errors.DistutilsPlatformError(
  distutils.errors.DistutilsPlatformError: Microsoft Visual C++ 14.0 or greater is required. Get it with "Microsoft C++ Build Tools": https://visualstudio.microsoft.com/visual-cpp-build-tools/
  [end of output]

  note: This error originates from a subprocess, and is likely not a problem with pip.
error: metadata-generation-failed

Encountered error while generating package metadata.

See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
```

In [ ]:
```
import lightning.pytorch as pl
from lightning.pytorch.callbacks import EarlyStopping, LearningRateMonitor
from lightning.pytorch.loggers import TensorBoardLogger
import numpy as np
import pandas as pd
import torch

from pytorch_forecasting import Baseline, TemporalFusionTransformer, TimeSeriesDataSet
from pytorch_forecasting.data import GroupNormalizer
from pytorch_forecasting.metrics import MAE, SMAPE, PoissonLoss, QuantileLoss
from pytorch_forecasting.models.temporal_fusion_transformer.tuning import optimize_hyperparameters
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[13], line 8
      5 import pandas as pd
      6 import torch
----> 8 from pytorch_forecasting import Baseline, TemporalFusionTransformer, TimeSeriesDataSet
      9 from pytorch_forecasting.data import GroupNormalizer
     10 from pytorch_forecasting.metrics import MAE, SMAPE, PoissonLoss, QuantileLoss

ModuleNotFoundError: No module named 'pytorch_forecasting'
```

In [ ]:
```
earliest_time= df['Date'].min()
```

In [ ]:
```
df["day_of_week"] = df['Date'].dt.dayofweek.astype(str).astype("int")  # categories have be strings
df["week_of_year"] = df['Date'].dt.isocalendar().week.astype(str).astype("int")  # categories have be stri
df["month"] = df['Date'].dt.month.astype(str).astype("int")
df['hour'] = df['Date'].dt.hour.astype(str).astype("int")
df['day'] = df['Date'].dt.day.astype(str).astype("int")
#df['minute'] = df['Date'].dt.minute.astype(str).astype("int")
df['time_idx']=df.index
df['hours_from_start'] = (df['Date'] - earliest_time).dt.seconds / 60 / 60 + (df['Date'] - earliest_time).d
df['hours_from_start'] = df['hours_from_start'].astype('int')
df['days_from_start'] = (df['Date'] - earliest_time).dt.days
df['group']=0
df['Global_active_power']=df['Global_active_power'].astype("float64")
time_df = df[[
    'Global_active_power'
,'GT1_Hits'
,'GT2_Hits'
,'GT3_Hits'
,'day_of_week'
,'week_of_year'
,'month'
,'hour'
,'day'
```

```
,'time_idx'
,'hours_from_start'
,'days_from_start'
,'group'
]]
df.dtypes
```

In [ ]:
```
time_df.isna().values.sum()
```

In [ ]:
```
#Hyperparameters
#batch size=64
#number heads=4, hidden sizes=160, lr=0.001, gr_clip=0.1

max_prediction_length = 24
max_encoder_length = 7*24
training_cutoff = time_df["hours_from_start"].max() - max_prediction_length

training = TimeSeriesDataSet(
    time_df[lambda x: x.hours_from_start <= training_cutoff],
    time_idx="hours_from_start",
    target="Global_active_power",
    group_ids=['group'],
    min_encoder_length=max_encoder_length // 2,
    max_encoder_length=max_encoder_length,
    min_prediction_length=1,
    max_prediction_length=max_prediction_length,
    #static_categoricals=["consumer_id"],
    time_varying_known_reals=["hours_from_start","day","day_of_week", "month", 'hour'],
    time_varying_unknown_reals=['Global_active_power'],
    target_normalizer=GroupNormalizer(
          groups=["group"], transformation="softplus"
      ),  # we normalize by group
    add_relative_time_idx=True,
    add_target_scales=True,
    add_encoder_length=True,
    allow_missing_timesteps=True
)


validation = TimeSeriesDataSet.from_dataset(training, time_df, predict=True, stop_randomization=True)

# create dataloaders for  our model
batch_size = 64
# if you have a strong GPU, feel free to increase the number of workers
train_dataloader = training.to_dataloader(train=True, batch_size=batch_size, num_workers=5)
val_dataloader = validation.to_dataloader(train=False, batch_size=batch_size * 10, num_workers=5)
```

In [ ]:
```
early_stop_callback = EarlyStopping(monitor="val_loss", min_delta=1e-4, patience=5, verbose=True, mode="mi
lr_logger = LearningRateMonitor()
logger = TensorBoardLogger("lightning_logs")

trainer = pl.Trainer(
    max_epochs=45,
    accelerator='auto',
    devices=1,
    enable_model_summary=True,
    gradient_clip_val=0.1,
    callbacks=[lr_logger, early_stop_callback],
    logger=logger)

tft = TemporalFusionTransformer.from_dataset(
    training,
    learning_rate=0.01,
    hidden_size=160,
    attention_head_size=4,
    dropout=1,
    hidden_continuous_size=160,
    output_size=7,  # there are 7 quantiles by default: [0.02, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98]
    loss=QuantileLoss(),
    log_interval=10,
    reduce_on_plateau_patience=4)

trainer.fit(
    tft,
```

```
    train_dataloaders=train_dataloader,
    val_dataloaders=val_dataloader)
```

In [ ]:
```
best_model_path = trainer.checkpoint_callback.best_model_path
print(best_model_path)
best_tft = TemporalFusionTransformer.load_from_checkpoint(best_model_path)
```

In [ ]:
```
#Take a look at what the raw_predictions variable contains

raw_predictions = best_tft.predict(val_dataloader, mode="raw", return_x=True)
print(raw_predictions._fields)
#('output', 'x', 'index', 'decoder_lengths', 'y')

print('\n')
print(raw_predictions.output._fields)
# ('prediction',
# 'encoder_attention',
# 'decoder_attention',
# 'static_variables',
# 'encoder_variables',
# 'decoder_variables',
# 'decoder_lengths',
# 'encoder_lengths')

print('\n')
print(raw_predictions.output.prediction.shape)
#torch.Size([5, 24, 7])

# We get predictions of 5 time-series for 24 days.
# For each day we get 7 predictions - these are the 7 quantiles:
#[0.02, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98]
# We are mostly interested in the 4th quantile which represents, let's say, the 'median loss'
# fyi, although docs use the term quantiles, the most accurate term are percentiles

# We get predictions of 5 time-series for 24 days.
# For each day we get 7 predictions - these are the 7 quantiles:
#[0.02, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98]
# We are mostly interested in the 4th quantile which represents, let's say, the 'median loss'
# fyi, although docs use the term quantiles, the most accurate term are percentiles
```

In [ ]:
```
import matplotlib.pyplot as plt
for idx in range(5):  # plot all 5 consumers
    fig, ax = plt.subplots(figsize=(10, 4))
    best_tft.plot_prediction(raw_predictions.x, raw_predictions.output, idx=idx, add_loss_to_title=Quantile
```

In [ ]:
```
raw_predictions= best_tft.predict(val_dataloader, mode="raw", return_x=True)
interpretation = best_tft.interpret_output(raw_predictions.output, reduction="sum")
best_tft.plot_interpretation(interpretation)
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive",
force_remount=True).
```

In [ ]:
```
!pip install lightning
!pip install pytorch_forecasting
```

```
Requirement already satisfied: lightning in /usr/local/lib/python3.10/dist-packages (2.1.2)
Requirement already satisfied: PyYAML<8.0,>=5.4 in /usr/local/lib/python3.10/dist-packages (from lightning)
(6.0.1)
Requirement already satisfied: fsspec[http]<2025.0,>2021.06.0 in /usr/local/lib/python3.10/dist-packages (f
rom lightning) (2023.6.0)
Requirement already satisfied: lightning-utilities<2.0,>=0.8.0 in /usr/local/lib/python3.10/dist-packages
(from lightning) (0.10.0)
Requirement already satisfied: numpy<3.0,>=1.17.2 in /usr/local/lib/python3.10/dist-packages (from lightnin
g) (1.23.5)
Requirement already satisfied: packaging<25.0,>=20.0 in /usr/local/lib/python3.10/dist-packages (from light
ning) (23.2)
Requirement already satisfied: torch<4.0,>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from lightnin
g) (2.1.0+cu121)
Requirement already satisfied: torchmetrics<3.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from li
ghtning) (1.2.1)
Requirement already satisfied: tqdm<6.0,>=4.57.0 in /usr/local/lib/python3.10/dist-packages (from lightnin
g) (4.66.1)
Requirement already satisfied: typing-extensions<6.0,>=4.0.0 in /usr/local/lib/python3.10/dist-packages (fr
om lightning) (4.9.0)
Requirement already satisfied: pytorch-lightning in /usr/local/lib/python3.10/dist-packages (from lightnin
g) (2.1.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from fsspec[http]<2025.
0,>2021.06.0->lightning) (2.31.0)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.10/dist-packages (from
fsspec[http]<2025.0,>2021.06.0->lightning) (3.9.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from lightning-utilit
ies<2.0,>=0.8.0->lightning) (67.7.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch<4.0,>=1.12.0
->lightning) (3.13.1)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch<4.0,>=1.12.0->l
ightning) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch<4.0,>=1.12.0
->lightning) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch<4.0,>=1.12.0->
lightning) (3.1.2)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-packages (from torch<4.0,>=
1.12.0->lightning) (2.1.0)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.
0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (23.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohtt
p!=4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (6.0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.
0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (1.9.4)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=
4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=
4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (1.3.1)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aio
http!=4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning) (4.0.3)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->tor
ch<4.0,>=1.12.0->lightning) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from re
quests->fsspec[http]<2025.0,>2021.06.0->lightning) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->fssp
ec[http]<2025.0,>2021.06.0->lightning) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests
->fsspec[http]<2025.0,>2021.06.0->lightning) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests
->fsspec[http]<2025.0,>2021.06.0->lightning) (2023.11.17)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch<
4.0,>=1.12.0->lightning) (1.3.0)
Requirement already satisfied: pytorch_forecasting in /usr/local/lib/python3.10/dist-packages (1.0.0)
Requirement already satisfied: fastapi>=0.80 in /usr/local/lib/python3.10/dist-packages (from pytorch_forec
asting) (0.105.0)
Requirement already satisfied: lightning<3.0.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from pyt
orch_forecasting) (2.1.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from pytorch_forecast
ing) (3.7.1)
Requirement already satisfied: optuna<4.0.0,>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from pytorc
h_forecasting) (3.5.0)
Requirement already satisfied: pandas<=3.0.0,>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from pytor
ch_forecasting) (1.5.3)
Requirement already satisfied: pytorch-optimizer<3.0.0,>=2.5.1 in /usr/local/lib/python3.10/dist-packages
(from pytorch_forecasting) (2.12.0)
Requirement already satisfied: scikit-learn<2.0,>=1.2 in /usr/local/lib/python3.10/dist-packages (from pyto
```

```
rch_forecasting) (1.2.2)
Requirement already satisfied: scipy<2.0,>=1.8 in /usr/local/lib/python3.10/dist-packages (from pytorch_for
ecasting) (1.11.4)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dist-packages (from pytorch_forecas
ting) (0.14.0)
Requirement already satisfied: torch<3.0.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from pytorch
_forecasting) (2.1.0+cu121)
Requirement already satisfied: anyio<4.0.0,>=3.7.1 in /usr/local/lib/python3.10/dist-packages (from fastapi
>=0.80->pytorch_forecasting) (3.7.1)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,!=2.1.0,<3.0.0,>=1.7.4 in /usr/local/l
ib/python3.10/dist-packages (from fastapi>=0.80->pytorch_forecasting) (1.10.13)
Requirement already satisfied: starlette<0.28.0,>=0.27.0 in /usr/local/lib/python3.10/dist-packages (from f
astapi>=0.80->pytorch_forecasting) (0.27.0)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from fa
stapi>=0.80->pytorch_forecasting) (4.9.0)
Requirement already satisfied: PyYAML<8.0,>=5.4 in /usr/local/lib/python3.10/dist-packages (from lightning<
3.0.0,>=2.0.0->pytorch_forecasting) (6.0.1)
Requirement already satisfied: fsspec[http]<2025.0,>2021.06.0 in /usr/local/lib/python3.10/dist-packages (f
rom lightning<3.0.0,>=2.0.0->pytorch_forecasting) (2023.6.0)
Requirement already satisfied: lightning-utilities<2.0,>=0.8.0 in /usr/local/lib/python3.10/dist-packages
(from lightning<3.0.0,>=2.0.0->pytorch_forecasting) (0.10.0)
Requirement already satisfied: numpy<3.0,>=1.17.2 in /usr/local/lib/python3.10/dist-packages (from lightnin
g<3.0.0,>=2.0.0->pytorch_forecasting) (1.23.5)
Requirement already satisfied: packaging<25.0,>=20.0 in /usr/local/lib/python3.10/dist-packages (from light
ning<3.0.0,>=2.0.0->pytorch_forecasting) (23.2)
Requirement already satisfied: torchmetrics<3.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from li
ghtning<3.0.0,>=2.0.0->pytorch_forecasting) (1.2.1)
Requirement already satisfied: tqdm<6.0,>=4.57.0 in /usr/local/lib/python3.10/dist-packages (from lightning
<3.0.0,>=2.0.0->pytorch_forecasting) (4.66.1)
Requirement already satisfied: pytorch-lightning in /usr/local/lib/python3.10/dist-packages (from lightning
<3.0.0,>=2.0.0->pytorch_forecasting) (2.1.2)
Requirement already satisfied: alembic>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from optuna<4.0.
0,>=3.1.0->pytorch_forecasting) (1.13.0)
Requirement already satisfied: colorlog in /usr/local/lib/python3.10/dist-packages (from optuna<4.0.0,>=3.
1.0->pytorch_forecasting) (6.8.0)
Requirement already satisfied: sqlalchemy>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from optuna<4.
0.0,>=3.1.0->pytorch_forecasting) (2.0.23)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pand
as<=3.0.0,>=1.3.0->pytorch_forecasting) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas<=3.0.0,
>=1.3.0->pytorch_forecasting) (2023.3.post1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn<
2.0,>=1.2->pytorch_forecasting) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit
-learn<2.0,>=1.2->pytorch_forecasting) (3.2.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch<3.0.0,>=2.0.
0->pytorch_forecasting) (3.13.1)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch<3.0.0,>=2.0.0->
pytorch_forecasting) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch<3.0.0,>=2.0.
0->pytorch_forecasting) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch<3.0.0,>=2.0.0-
>pytorch_forecasting) (3.1.2)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-packages (from torch<3.0.0,>
=2.0.0->pytorch_forecasting) (2.1.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib
->pytorch_forecasting) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->py
torch_forecasting) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotli
b->pytorch_forecasting) (4.46.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotli
b->pytorch_forecasting) (1.4.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->p
ytorch_forecasting) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib
->pytorch_forecasting) (3.1.1)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-packages (from statsmodels->p
ytorch_forecasting) (0.5.4)
Requirement already satisfied: Mako in /usr/local/lib/python3.10/dist-packages (from alembic>=1.5.0->optuna
<4.0.0,>=3.1.0->pytorch_forecasting) (1.3.0)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<4.0.0,>=3.
7.1->fastapi>=0.80->pytorch_forecasting) (3.6)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-packages (from anyio<4.0.0,>=
3.7.1->fastapi>=0.80->pytorch_forecasting) (1.3.0)
```

```
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<4.0.0,
>=3.7.1->fastapi>=0.80->pytorch_forecasting) (1.2.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from fsspec[http]<2025.
0,>2021.06.0->lightning<3.0.0,>=2.0.0->pytorch_forecasting) (2.31.0)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.10/dist-packages (from
fsspec[http]<2025.0,>2021.06.0->lightning<3.0.0,>=2.0.0->pytorch_forecasting) (3.9.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from lightning-utilit
ies<2.0,>=0.8.0->lightning<3.0.0,>=2.0.0->pytorch_forecasting) (67.7.2)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.2->statsmode
ls->pytorch_forecasting) (1.16.0)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-packages (from sqlalchemy
>=1.3.0->optuna<4.0.0,>=3.1.0->pytorch_forecasting) (3.0.2)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->tor
ch<3.0.0,>=2.0.0->pytorch_forecasting) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch<
3.0.0,>=2.0.0->pytorch_forecasting) (1.3.0)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.
0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning<3.0.0,>=2.0.0->pytorch_forecasting) (23.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohtt
p!=4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning<3.0.0,>=2.0.0->pytorch_forecasting) (6.0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.
0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning<3.0.0,>=2.0.0->pytorch_forecasting) (1.9.4)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=
4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning<3.0.0,>=2.0.0->pytorch_forecasting) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=
4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning<3.0.0,>=2.0.0->pytorch_forecasting) (1.3.1)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aio
http!=4.0.0a0,!=4.0.0a1->fsspec[http]<2025.0,>2021.06.0->lightning<3.0.0,>=2.0.0->pytorch_forecasting) (4.
0.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from re
quests->fsspec[http]<2025.0,>2021.06.0->lightning<3.0.0,>=2.0.0->pytorch_forecasting) (3.3.2)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests
->fsspec[http]<2025.0,>2021.06.0->lightning<3.0.0,>=2.0.0->pytorch_forecasting) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests
->fsspec[http]<2025.0,>2021.06.0->lightning<3.0.0,>=2.0.0->pytorch_forecasting) (2023.11.17)
```

In [ ]:
```python
import lightning.pytorch as pl
from lightning.pytorch.callbacks import EarlyStopping, LearningRateMonitor
from lightning.pytorch.loggers import TensorBoardLogger
import numpy as np
import pandas as pd
import torch

import optuna
from optuna.integration import PyTorchLightningPruningCallback


from pytorch_forecasting import Baseline, TemporalFusionTransformer, TimeSeriesDataSet
from pytorch_forecasting.data import GroupNormalizer, TorchNormalizer, MultiNormalizer,EncoderNormalizer
from pytorch_forecasting.metrics import MAE, SMAPE, PoissonLoss, QuantileLoss, MAPE, MultivariateNormalDist

import pandas as pd
import numpy as np


#from pytorch_forecasting.models.temporal_fusion_transformer.tuning import optimize_hyperparameters
```

Creating Multi Step Dataset

In [ ]:
```python
df    = pd.read_csv("/content/drive/MyDrive/BDA696Project/Datasets/processsed/YMH_Clean_UCI_Power_Consumpti
cpi   = pd.read_csv("/content/drive/MyDrive/BDA696Project/Datasets/processsed/YM_Clean_France_CPI_Electric
temp  = pd.read_csv("/content/drive/MyDrive/BDA696Project/Datasets/processsed/YMH_Clean_TempISD.csv")
cci   = pd.read_csv("/content/drive/MyDrive/BDA696Project/Datasets/processsed/YM_Clean_CCI.csv")
gt1   = pd.read_csv("/content/drive/MyDrive/BDA696Project/Datasets/processsed/YMW_Clean_GoogleTrends1.csv")
gt2   = pd.read_csv("/content/drive/MyDrive/BDA696Project/Datasets/processsed/YMW_Clean_GoogleTrends2.csv")
gt3   = pd.read_csv("/content/drive/MyDrive/BDA696Project/Datasets/processsed/YMW_Clean_GoogleTrends3.csv")
```

```
<ipython-input-33-968a71a3a344>:1: DtypeWarning: Columns (2,3,4,5,6,7) have mixed types. Specify dtype opti
on on import or set low_memory=False.
  df    = pd.read_csv("/content/drive/MyDrive/BDA696Project/Datasets/processsed/YMH_Clean_UCI_Power_Consump
tion_Dataset.csv")
<ipython-input-33-968a71a3a344>:3: DtypeWarning: Columns (20) have mixed types. Specify dtype option on imp
ort or set low_memory=False.
  temp  = pd.read_csv("/content/drive/MyDrive/BDA696Project/Datasets/processsed/YMH_Clean_TempISD.csv")
```

In [ ]:
```
ISD = pd.read_csv('/content/drive/MyDrive/BDA696Project/TFT_Model_Datasets/ISD_cleaned.csv')
ISD['Date'] = pd.to_datetime(ISD['New_Date'])
ISD=ISD[['Date','Wind_Speed_Norm','Prec_Depth_Norm','Air_Temp_Norm']]
# ISD = ISD.drop_duplicates(subset='New_Date', keep='first')
# ISD.sort_values(by='New_Date', ascending = False)
# Wind_Speed_Agg = ISD.resample('D', on='New_Date').Wind_Speed_Norm.mean()
# Prec_Depth_Agg = ISD.resample('D', on='New_Date').Prec_Depth_Norm.mean()
# Air_Temp_Agg = ISD.resample('D', on='New_Date').Air_Temp_Norm.mean()
```

```
<ipython-input-34-a0624b3daacb>:1: DtypeWarning: Columns (27) have mixed types. Specify dtype option on imp
ort or set low_memory=False.
  ISD = pd.read_csv('/content/drive/MyDrive/BDA696Project/TFT_Model_Datasets/ISD_cleaned.csv')
```

In [ ]:
```
# df    = pd.read_csv("Downloads/processsed/YMH_Clean_UCI_Power_Consumption_Dataset.csv")
# cpi   = pd.read_csv("Downloads/processsed/YM_Clean_France_CPI_Electricity.csv")
# temp  = pd.read_csv("Downloads/processsed/YMH_Clean_TempISD.csv")
# cci   = pd.read_csv("Downloads/processsed/YM_Clean_CCI.csv")
# gt1   = pd.read_csv("Downloads/processsed/YMW_Clean_GoogleTrends1.csv")
# gt2   = pd.read_csv("Downloads/processsed/YMW_Clean_GoogleTrends2.csv")
# gt3   = pd.read_csv("Downloads/processsed/YMW_Clean_GoogleTrends3.csv")
```

In [ ]:
```
df['Date'] = pd.to_datetime(df['Date'])
df=df.query('(Date.dt.hour==0 or Date.dt.hour==6 or Date.dt.hour==12 or Date.dt.hour==18) and Date.dt.minut
df=df.replace('?',0.0)
df.head()
```

Out[ ]:

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_
0	2007-01-01 00:00:00	00:00:00	2.580	0.136	241.970	10.600	0.000	
360	2007-01-01 06:00:00	06:00:00	2.460	0.064	241.130	10.200	0.000	
720	2007-01-01 12:00:00	12:00:00	2.478	0.000	235.300	10.400	0.000	
1080	2007-01-01 18:00:00	18:00:00	1.416	0.000	239.130	5.800	0.000	
1440	2007-01-02 00:00:00	00:00:00	0.442	0.122	241.060	1.800	0.000	

Creating and using function to foward fill weekly and monthly values

In [ ]:
```
def fillblanks(df_temp):
  temp = pd.DataFrame(pd.date_range(start=df_temp['Date'].min(), end= df_temp['Date'].max()),columns=['Date
  temp['Date'] = pd.to_datetime(temp['Date'])
  temp = temp.merge(df_temp, on='Date', how='left')
  temp = temp.replace(0, np.nan).ffill()
  return temp
```

In [ ]:
```
#temp['Date'] = pd.to_datetime(cci['Date'])
#df=pd.merge_asof(df,temp, on='Date')

#df.dtypes

cpi['Date'] = pd.to_datetime(cpi['Date'])
df=df.merge(fillblanks(cpi), on='Date', how='left')
df['CPI'] = df['CPI'].replace(0, np.nan).ffill()
```

```
cci['Date'] = pd.to_datetime(cci['Date'])
df=df.merge(fillblanks(cci), on='Date', how='left')
df['CCI'] = df['CCI'].replace(0, np.nan).ffill()

ISD['Date'] = pd.to_datetime(ISD['Date'])
df=df.merge(fillblanks(ISD), on='Date', how='left')
df['Wind_Speed_Norm'] = df['Wind_Speed_Norm'].replace(0, np.nan).ffill()
df['Prec_Depth_Norm'] = df['Prec_Depth_Norm'].replace(0, np.nan).ffill()
df['Air_Temp_Norm'] = df['Air_Temp_Norm'].replace(0, np.nan).ffill()


gt1['Date'] = pd.to_datetime(gt1['Date'])
df=df.merge(fillblanks(gt1), on='Date', how='left')
df['GT1_Hits'] = df['GT1_Hits'].replace(0, np.nan).ffill()


gt2['Date'] = pd.to_datetime(gt2['Date'])
df=df.merge(fillblanks(gt2), on='Date', how='left')
df['GT2_Hits'] = df['GT2_Hits'].replace(0, np.nan).ffill()


gt3['Date'] = pd.to_datetime(gt3['Date'])
df=df.merge(fillblanks(gt3), on='Date', how='left')
df['GT3_Hits'] = df['GT3_Hits'].replace(0, np.nan).ffill()
```

In [ ]:
```
df=df.fillna(value=0)
```

In [ ]:
```
earliest_time= df['Date'].min()
```

In [ ]:
```
df["day_of_week"] = df['Date'].dt.dayofweek.astype(str) # categories have be strings
df["week_of_year"] = df['Date'].dt.isocalendar().week.astype(str)  # categories have be strings
df["month"] = df['Date'].dt.month.astype(str)
df['hour'] = df['Date'].dt.hour.astype(str)
df['day'] = df['Date'].dt.day.astype(str)
df['year'] = df['Date'].dt.year-2006
df['year']=df['year'].astype('int')
#df['minute'] = df['Date'].dt.minute.astype(str).astype("int")
df['time_idx']=df.index
df['hours_from_start'] = (df['Date'] - earliest_time).dt.seconds / 60 / 60 + (df['Date'] - earliest_time).d
df['hours_from_start'] = df['hours_from_start'].astype('int')
df['days_from_start'] = (df['Date'] - earliest_time).dt.days
df['group']=0
df['Global_active_power']=df['Global_active_power'].astype("float64")
df['Voltage']=df['Voltage'].astype("float64")
df['Global_intensity']=df['Global_intensity'].astype("float64")
df['Sub_metering_1']=df['Sub_metering_1'].astype("float64")
df['Sub_metering_2']=df['Sub_metering_2'].astype("float64")
df['Wind_Speed_Norm'] = df['Wind_Speed_Norm'].astype("float64")
df['Prec_Depth_Norm'] = df['Prec_Depth_Norm'].astype("float64")
df['Air_Temp_Norm'] = df['Air_Temp_Norm'].astype("float64")


time_df = df[[
    'Global_active_power'
    ,'Sub_metering_1'
    ,'Sub_metering_2'
    ,'Sub_metering_3'
,'CPI'
,'CCI'
,'GT1_Hits'
,'GT2_Hits'
,'GT3_Hits'
,'Wind_Speed_Norm'
,'Prec_Depth_Norm'
,'Air_Temp_Norm'
,'day_of_week'
,'week_of_year'
,'year'
,'month'
,'hour'
,'day'
,'time_idx'
,'days_from_start'
```

```
,'group'
,'Date'
]]
df.dtypes
```

Out[ ]:
```
Date                    datetime64[ns]
Time                            object
Global_active_power            float64
Global_reactive_power           object
Voltage                        float64
Global_intensity               float64
Sub_metering_1                 float64
Sub_metering_2                 float64
Sub_metering_3                 float64
CPI                            float64
CCI                            float64
Wind_Speed_Norm                float64
Prec_Depth_Norm                float64
Air_Temp_Norm                  float64
GT1_Hits                       float64
GT2_Hits                       float64
GT3_Hits                       float64
day_of_week                     object
week_of_year                    object
month                           object
hour                            object
day                             object
year                             int64
time_idx                         int64
hours_from_start                 int64
days_from_start                  int64
group                            int64
dtype: object
```

In [ ]:
```
time_df.isna().values.sum()
```

Out[ ]:
```
0
```

In [ ]:
```
time_df.head()
```

Out[ ]:

	Global_active_power	Sub_metering_1	Sub_metering_2	Sub_metering_3	CPI	CCI	GT1_Hits	GT2_Hits	GT3_Hits
0	2.580	0.0	0.0	0.0	72.71	100.0	45.0	41.0	0.0
1	2.460	0.0	0.0	0.0	72.71	100.0	45.0	41.0	0.0
2	2.478	0.0	0.0	0.0	72.71	100.0	45.0	41.0	0.0
3	1.416	0.0	0.0	17.0	72.71	100.0	45.0	41.0	0.0
4	0.442	0.0	0.0	0.0	72.71	100.0	45.0	41.0	0.0

5 rows × 22 columns

Setting up Time Series DataLoaders

In [ ]:
```
#Hyperparameters
#batch size=64
#number heads=4, hidden sizes=160, lr=0.001, gr_clip=0.1

max_prediction_length = 4*7*4*6 #sixmonths
max_encoder_length = 4*365 #one year
training_cutoff = time_df["time_idx"].max() - max_prediction_length
```

```
training = TimeSeriesDataSet(
    time_df[lambda x: x.time_idx <= training_cutoff],
    time_idx="time_idx",
    target="Global_active_power",
    group_ids=['group'],
    min_encoder_length=4*7, #one week
    max_encoder_length=max_encoder_length,
    min_prediction_length=4, #one day
    max_prediction_length=max_prediction_length,
    #static_categoricals=["consumer_id"],
    time_varying_known_categoricals=["day","day_of_week", "month", 'hour' ], #year to be added
    time_varying_known_reals=["time_idx",'year'],
    time_varying_unknown_reals=['Sub_metering_1', 'Sub_metering_2', 'Sub_metering_3','CPI','CCI','GT1_Hits
    target_normalizer=GroupNormalizer(
          groups=["group"], transformation="softplus"
      ),  # we normalize by group
    add_relative_time_idx=True,
    add_target_scales=True,
    add_encoder_length=True,
    #add_nan=True,
    allow_missing_timesteps=True
)


validation = TimeSeriesDataSet.from_dataset(training, time_df, predict=True, stop_randomization=True)

# create dataloaders for  our model
batch_size = 64
# if you have a strong GPU, feel free to increase the number of workers
train_dataloader = training.to_dataloader(train=True, batch_size=batch_size, num_workers=10)
val_dataloader = validation.to_dataloader(train=False, batch_size=batch_size * 10, num_workers=10)
```

```
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader wi
ll create 10 worker processes in total. Our suggested max number of worker in current system is 8, which is
smaller than what this DataLoader is going to create. Please be aware that excessive worker creation might
get DataLoader running slow or even freeze, lower the worker number to avoid potential slowness/freeze if n
ecessary.
  warnings.warn(_create_warning_msg(
```

[I 2023-12-10 07:14:28,911] Trial 3 finished with value: 0.31155362725257874 and parameters: {'gradient_clip_val': 0.8223697202530604, 'hidden_size': 22, 'dropout': 0.2972505585187706, 'hidden_continuous_size': 9, 'attention_head_size': 1, 'learning_rate': 0.014035308702029799}. Best is trial 2 with value: 0.29049456119537354.

Saving the Model

In [ ]:
```
#!unzip /content/drive/MyDrive/BDA696Project/Model/model.zip
#torch.save(best_tft.state_dict(), path)
```

In [ ]:
```
best_model_path='/content/drive/MyDrive/BDA696Project/Model/model_saved/lightning_logs/lightning_logs/vers
best_tft = TemporalFusionTransformer.load_from_checkpoint(best_model_path)
```

```
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/utilities/parsing.py:198: Attribute 'loss' is an
instance of `nn.Module` and is already saved during checkpointing. It is recommended to ignore them using `
self.save_hyperparameters(ignore=['loss'])`.
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/utilities/parsing.py:198: Attribute 'logging_metr
ics' is an instance of `nn.Module` and is already saved during checkpointing. It is recommended to ignore t
hem using `self.save_hyperparameters(ignore=['logging_metrics'])`.
```

In [ ]:
```
#Take a look at what the raw_predictions variable contains

raw_predictions = best_tft.predict(val_dataloader, mode="raw", return_x=True)
print(raw_predictions._fields)
#('output', 'x', 'index', 'decoder_lengths', 'y')

print('\n')
#print(raw_predictions.output._fields)
# ('prediction',
# 'encoder_attention',
# 'decoder_attention',
# 'static_variables',
# 'encoder_variables',
```

```
# 'decoder_variables',
# 'decoder_lengths',
# 'encoder_lengths')

print('\n')
#print(raw_predictions.output.prediction.shape)
#torch.Size([5, 24, 7])

# We get predictions of 5 time-series for 24 days.
# For each day we get 7 predictions - these are the 7 quantiles:
#[0.02, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98]
# We are mostly interested in the 4th quantile which represents, let's say, the 'median loss'
# fyi, although docs use the term quantiles, the most accurate term are percentiles

# We get predictions of 5 time-series for 24 days.
# For each day we get 7 predictions - these are the 7 quantiles:
#[0.02, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98]
# We are mostly interested in the 4th quantile which represents, let's say, the 'median loss'
# fyi, although docs use the term quantiles, the most accurate term are percentiles
```

```
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader wi
ll create 10 worker processes in total. Our suggested max number of worker in current system is 8, which is
smaller than what this DataLoader is going to create. Please be aware that excessive worker creation might
get DataLoader running slow or even freeze, lower the worker number to avoid potential slowness/freeze if n
ecessary.
  warnings.warn(_create_warning_msg(
('output', 'x', 'index', 'decoder_lengths', 'y')
```

Loss Comparision: TFT VS Baseline

Using 'prediction' mode for calculating loss function

In [ ]:
```
actuals = torch.cat([y[0] for x, y in iter(val_dataloader)])
predictions = best_tft.predict(val_dataloader, mode="prediction").to('cpu')
baseline_predictions = Baseline().predict(val_dataloader).to('cpu')
```

```
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
```

MAE

In [ ]:
```
mae_loss = MAE()
print(f"TFT: {mae_loss.loss(actuals, predictions).mean(axis = 1).median().item()}")
print(f"Baseline: {mae_loss.loss(actuals, baseline_predictions).mean(axis = 1).median().item()}")
```

```
TFT: 0.549065113067627
Baseline: 0.5105714201927185
```

SMAPE

In [ ]:
```
sm = SMAPE()
print(f"TFT: {sm.loss(actuals, predictions).mean(axis = 1).median().item()}")
print(f"Baseline: {sm.loss(actuals, baseline_predictions).mean(axis = 1).median().item()}")
```

```
TFT: 0.6751478910446167
Baseline: 0.729117214679718
```

MAPE

In [ ]:
```
mape = MAPE()
print(f"TFT: {mape.loss(actuals, predictions).mean(axis = 1).median().item()}")
print(f"Baseline: {mape.loss(actuals, baseline_predictions).mean(axis = 1).median().item()}")
```

```
TFT: 0.9968411326408386
Baseline: 1.7605912685394287
```

Poisson Loss

In [ ]:
```
pl = PoissonLoss()
print(f"TFT: {pl.loss(actuals, predictions).mean(axis = 1).median().item()}")
print(f"Baseline: {pl.loss(actuals, baseline_predictions).mean(axis = 1).median().item()}")
```

```
TFT: 3.1876018047332764
Baseline: 3.5891759395599365
```

Quantile Loss

In [ ]:
```
ql = QuantileLoss()
print(f"TFT: {ql.loss(actuals.view(1, -1),predictions.view(1, -1)).mean().item()}")
print(f"Baseline: {ql.loss(actuals.view(1, -1),baseline_predictions.view(1, -1)).mean().item()}")
```

```
TFT: 0.9532883763313293
Baseline: 1.0016515254974365
```

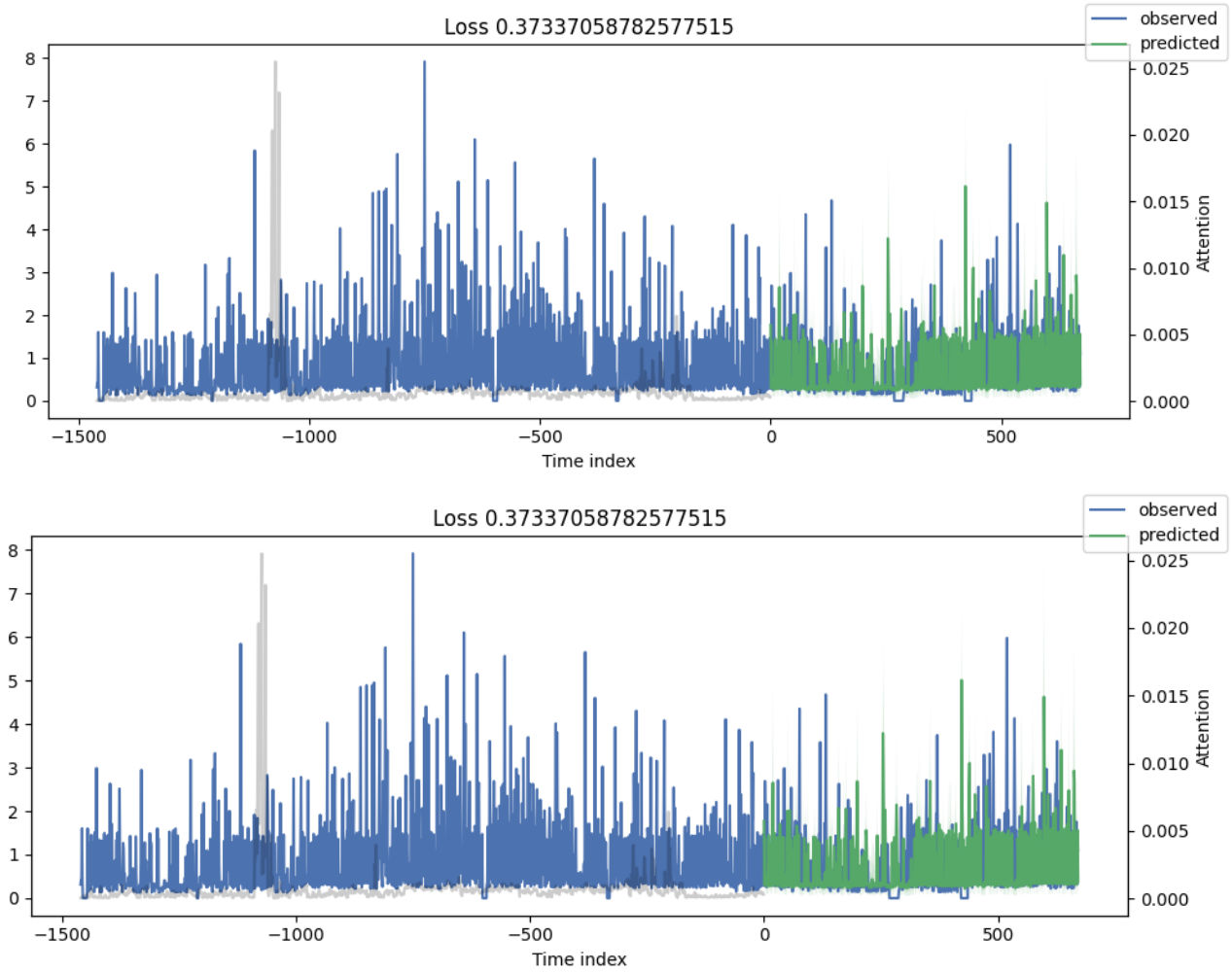
Observed vs Predicted

Using 'raw' mode

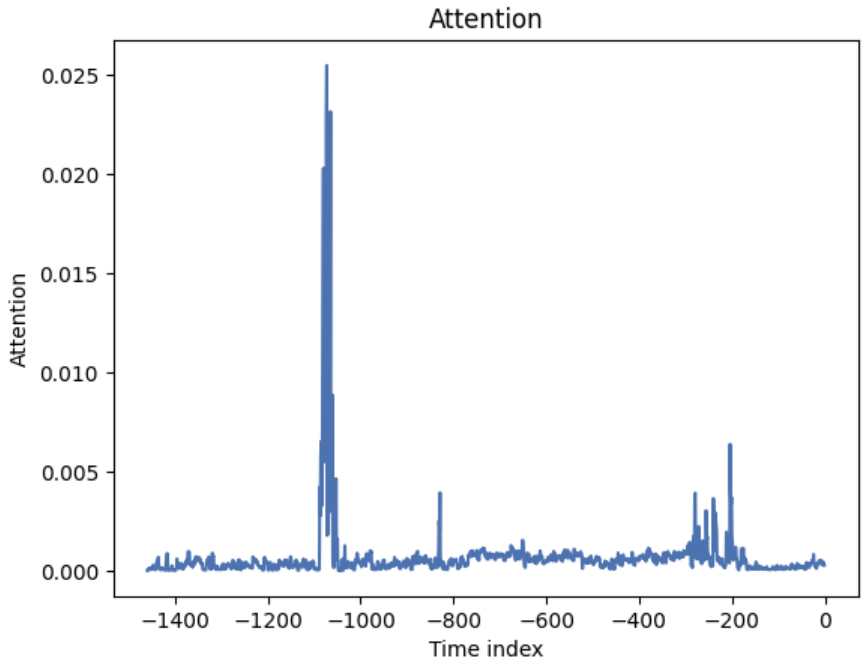
In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('seaborn-deep')
fig, ax = plt.subplots(figsize=(10, 4))
best_tft.plot_prediction(raw_predictions.x, raw_predictions.output, idx=0, add_loss_to_title=True,ax=ax)
```

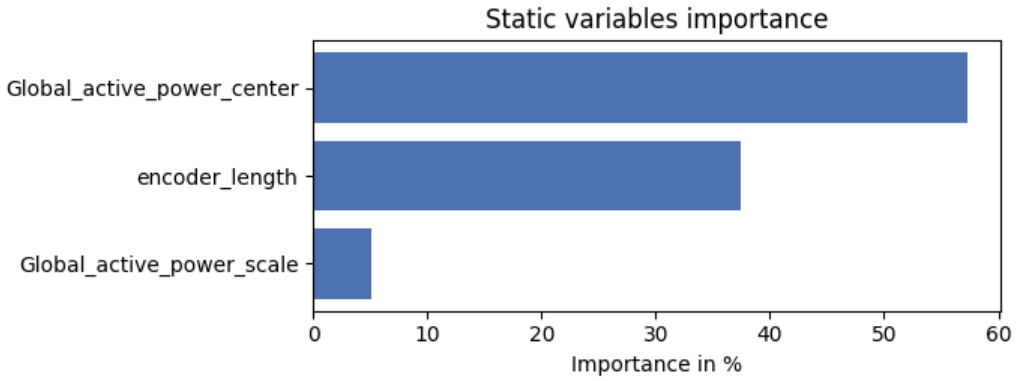
```
<ipython-input-54-86e257ce8775>:4: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib a
re deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn. However, they will
remain available as 'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn API instead.
  plt.style.use('seaborn-deep')
```

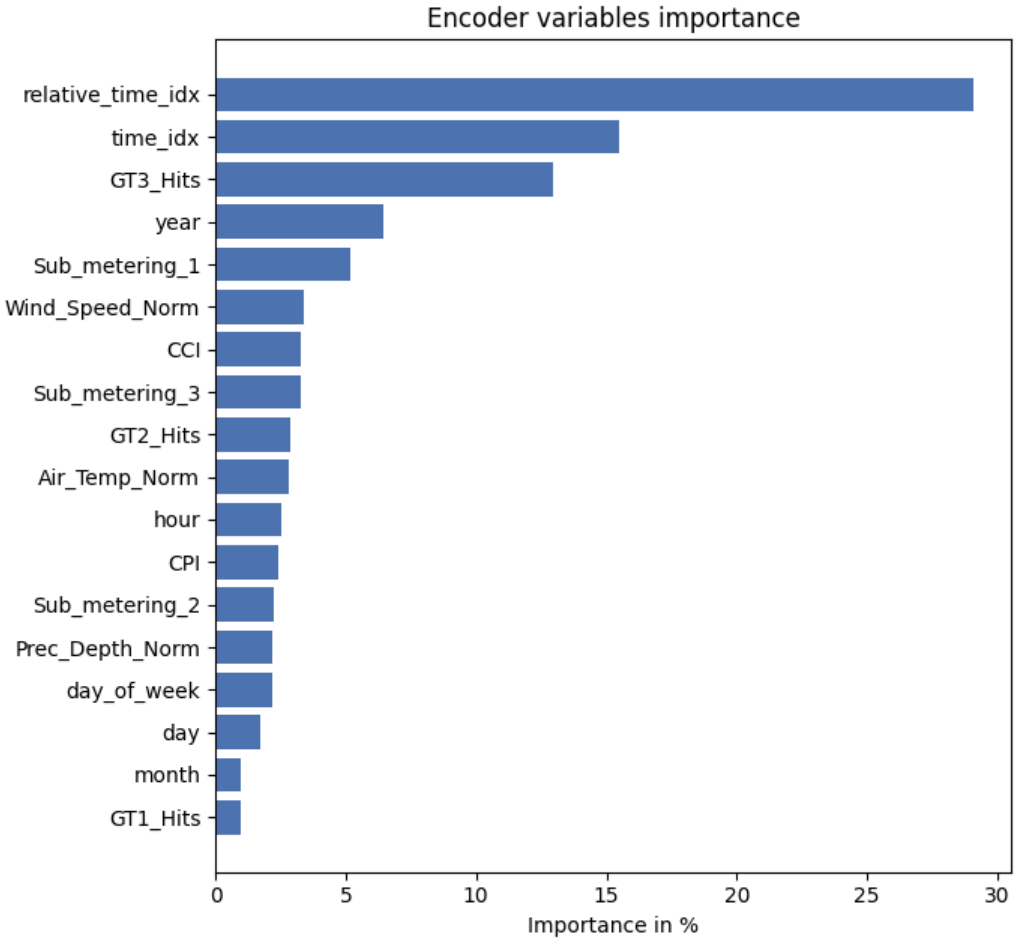
Out[ ]:

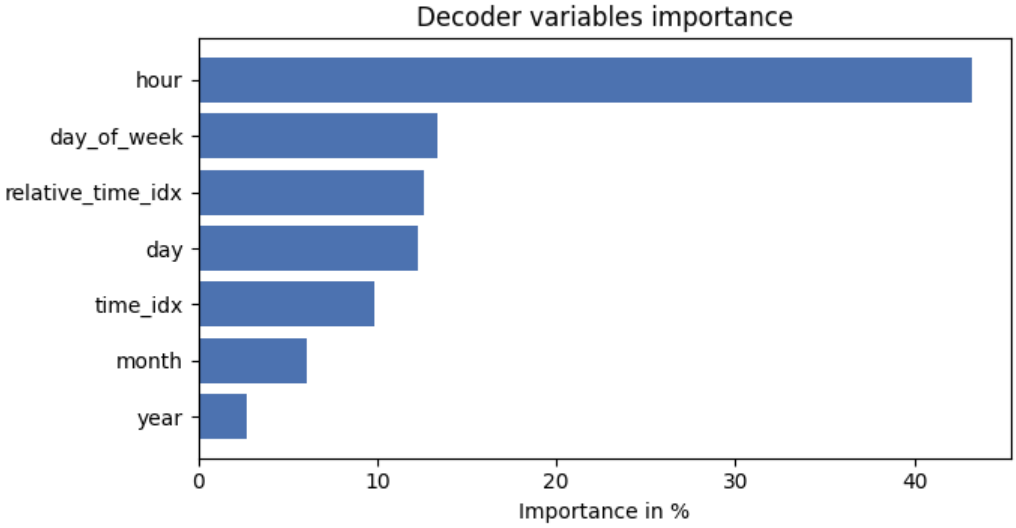


In [ ]:
```
interpretation = best_tft.interpret_output(raw_predictions.output, reduction="sum")
#interpretation
plt.show(best_tft.plot_interpretation(interpretation))
```




Static variables importance
Global_active_power_center
encoder_length
Global_active_power_scale
0
10
20
30
40
50
60
Importance in %


Encoder variables importance
relative_time_idx
time_idx
GT3_Hits
year
Sub_metering_1
Wind_Speed_Norm
CCI
Sub_metering_3
GT2_Hits
Air_Temp_Norm
hour
CPI
Sub_metering_2
Prec_Depth_Norm
day_of_week
day
month
GT1_Hits
0
5
10
15
20
25
30
Importance in %

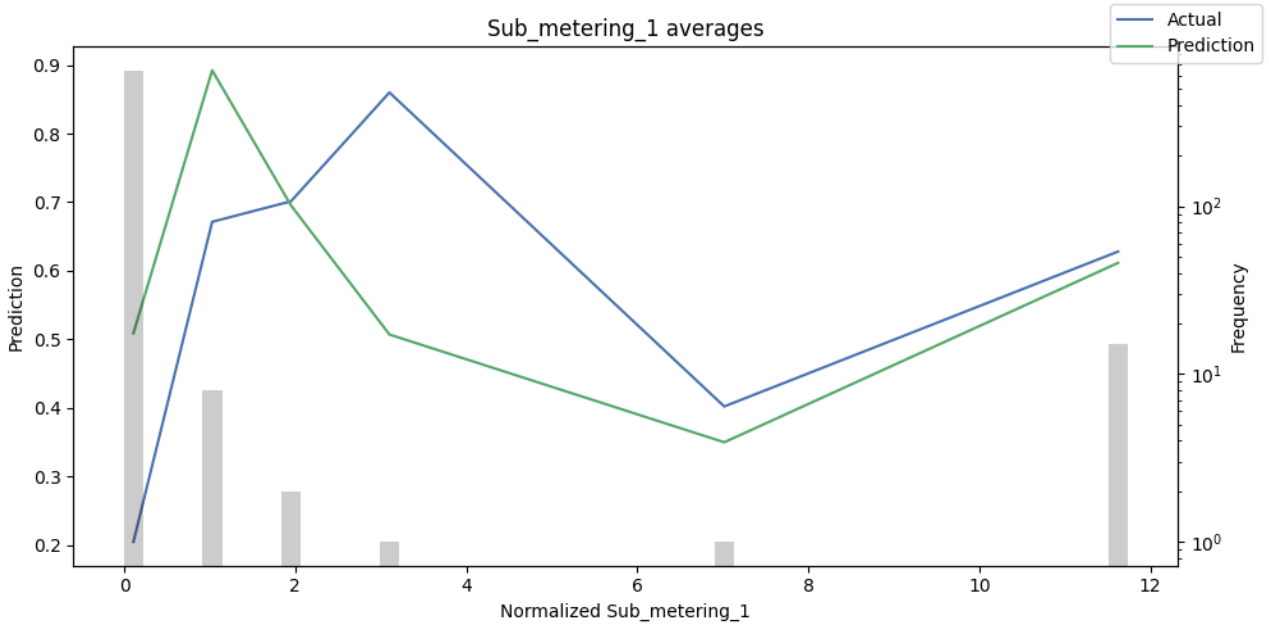

Decoder variables importance
hour
day_of_week
relative_time_idx
day
time_idx
month
year
0
10
20
30
40
Importance in %

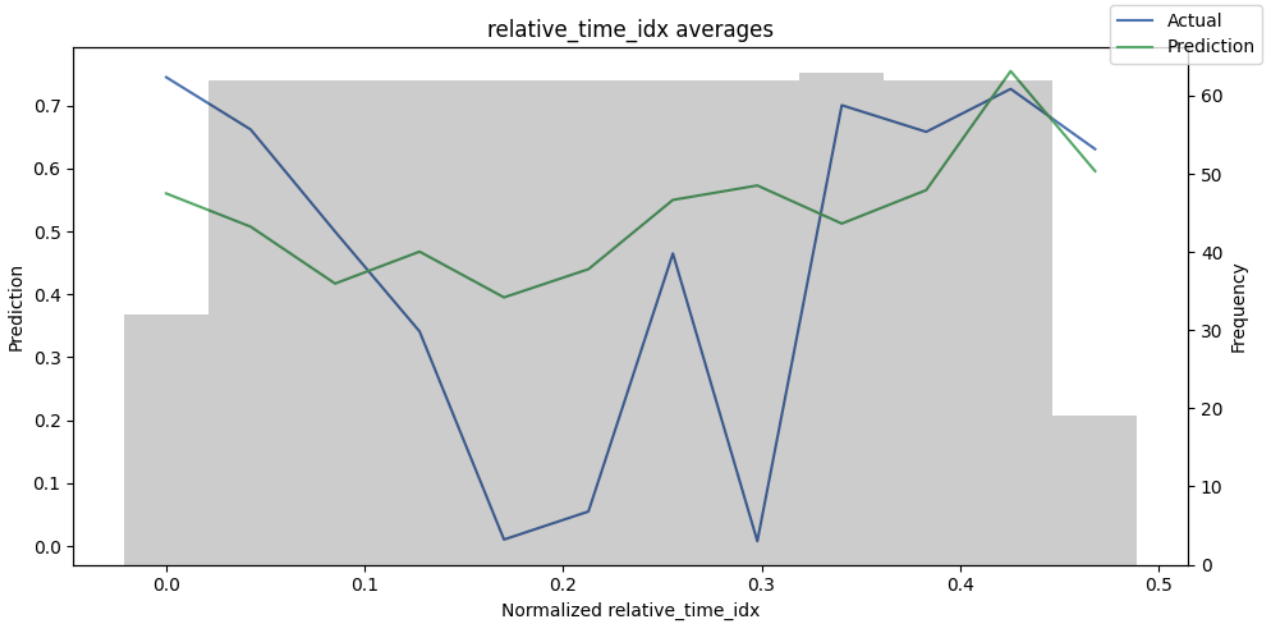
Using 'prediction' mode

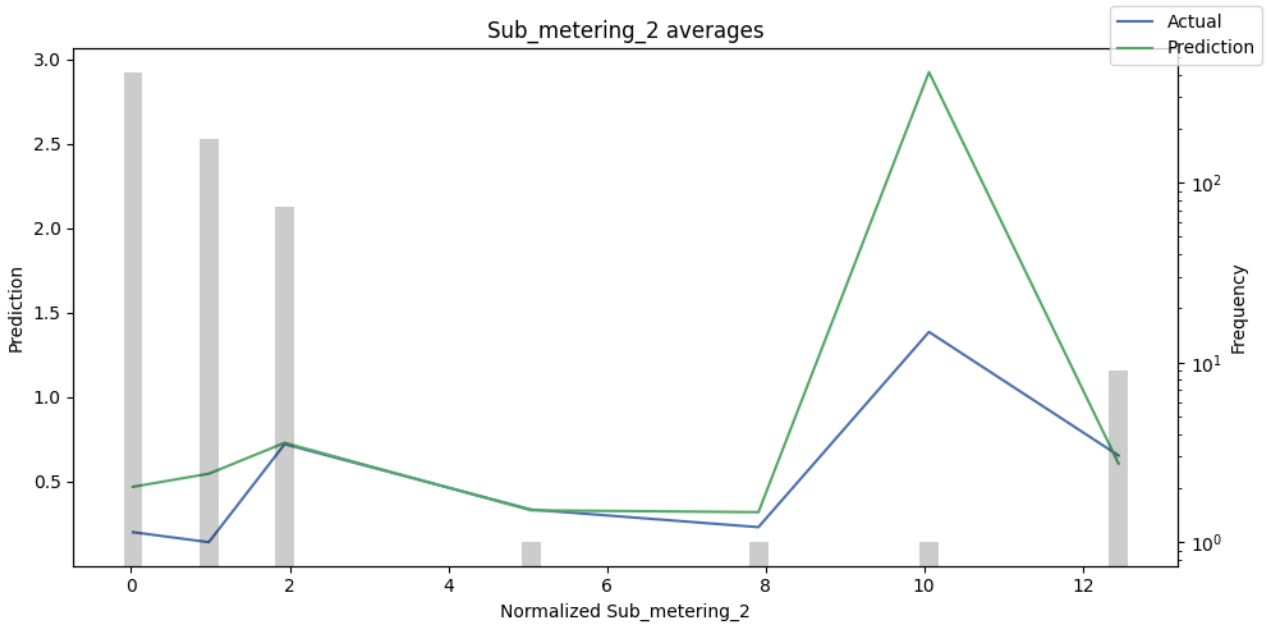
In [ ]:
```
predictions=best_tft.predict(val_dataloader,mode='prediction',return_x=True)
```

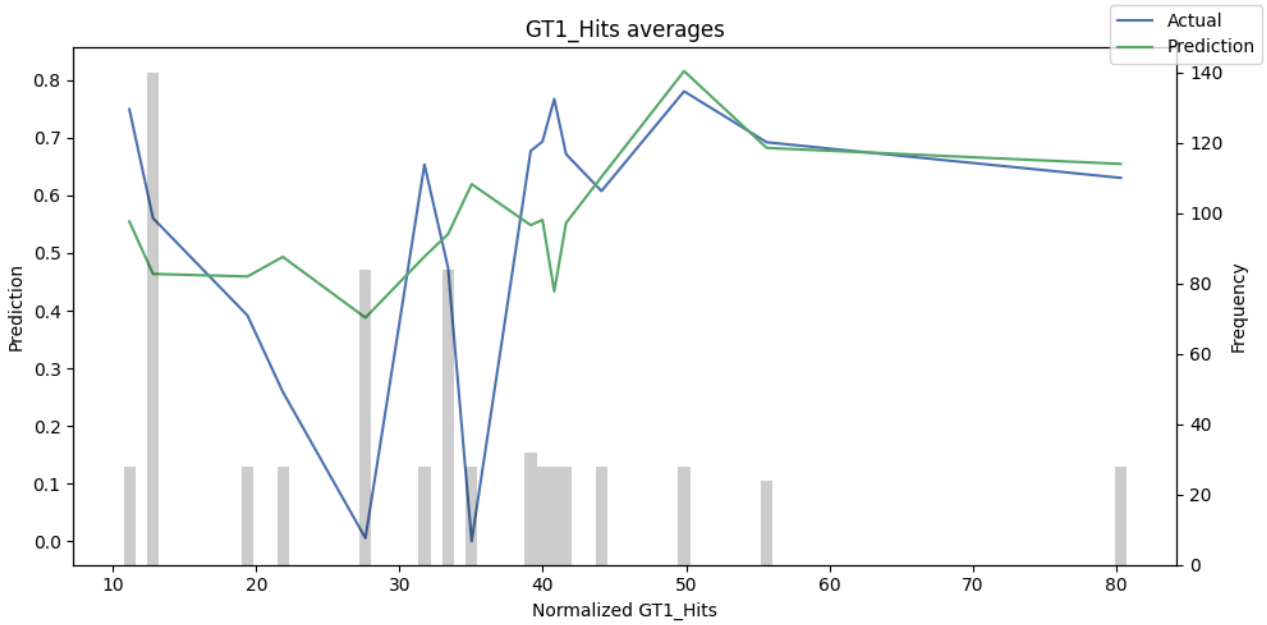
```
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader wi
ll create 10 worker processes in total. Our suggested max number of worker in current system is 8, which is
smaller than what this DataLoader is going to create. Please be aware that excessive worker creation might
get DataLoader running slow or even freeze, lower the worker number to avoid potential slowness/freeze if n
ecessary.
  warnings.warn(_create_warning_msg(
```

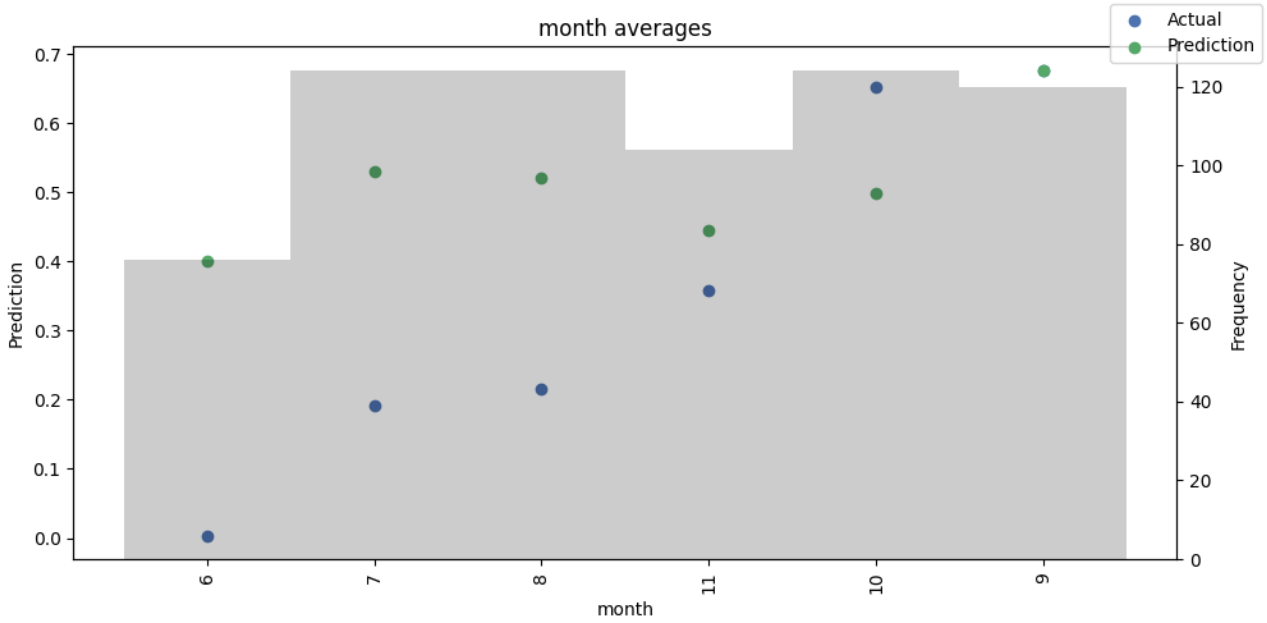
In [ ]:
```
predictions_vs_actuals = best_tft.calculate_prediction_actual_by_variable(predictions.x, predictions.output
all_features = list(set(predictions_vs_actuals['support'].keys())-set(['year', 'encoder_length', 'Global_a
for feature in all_features:
    best_tft.plot_prediction_actual_by_variable(predictions_vs_actuals, name=feature);
```

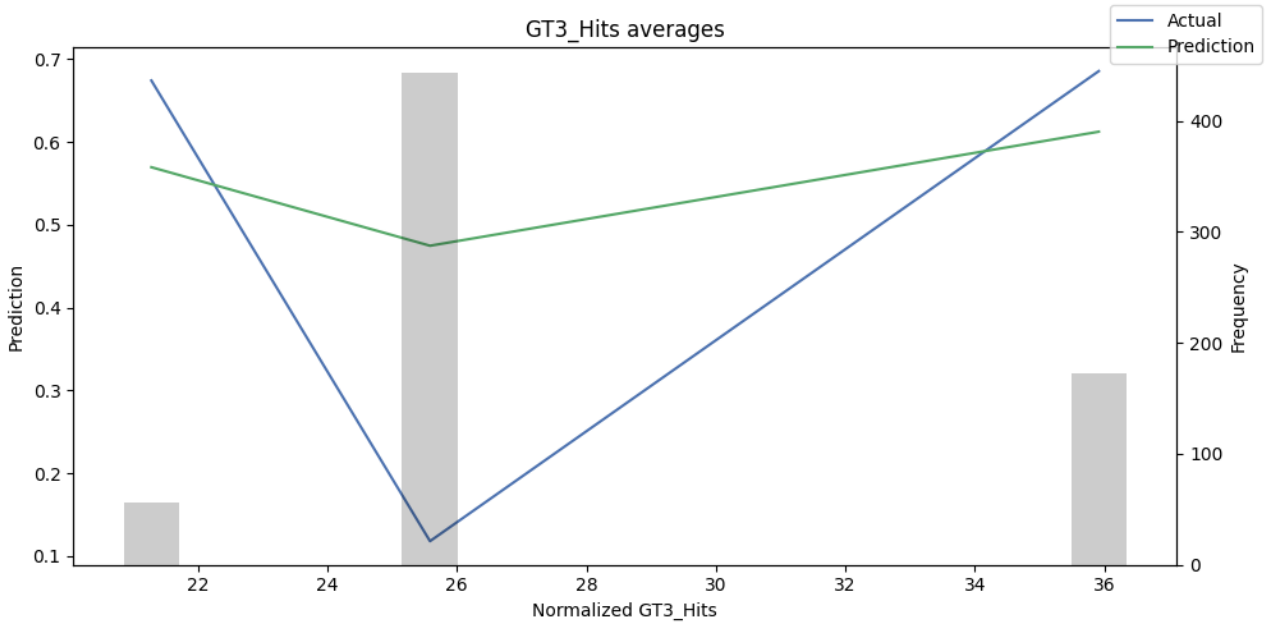


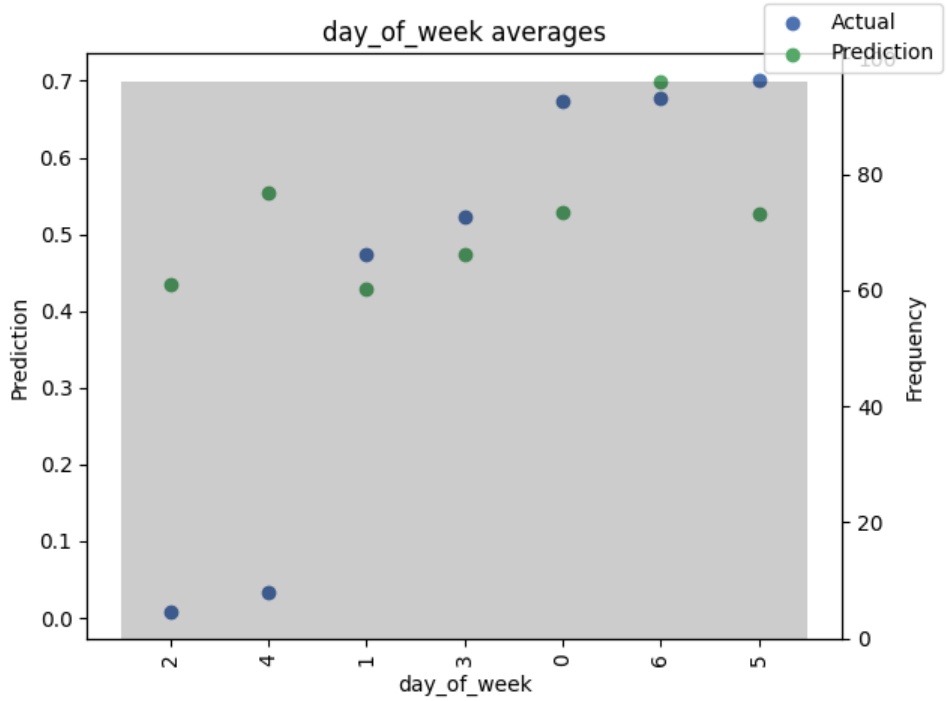

relative_time_idx averages
Actual
Prediction
Prediction
Frequency
Normalized relative_time_idx
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
10
20
30
40
50
60

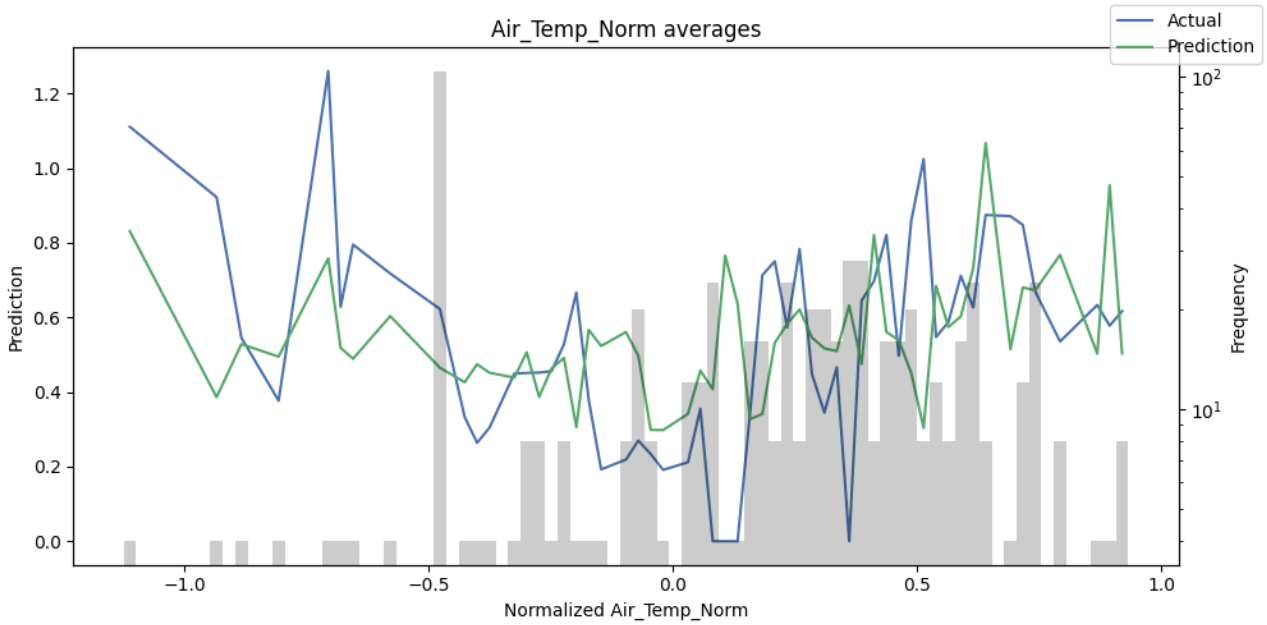

Sub_metering_2 averages
Actual
Prediction
Prediction
Frequency
Normalized Sub_metering_2
0
2
4
6
8
10
12
0.5
1.0
1.5
2.0
2.5
3.0
10^0
10^1
10^2

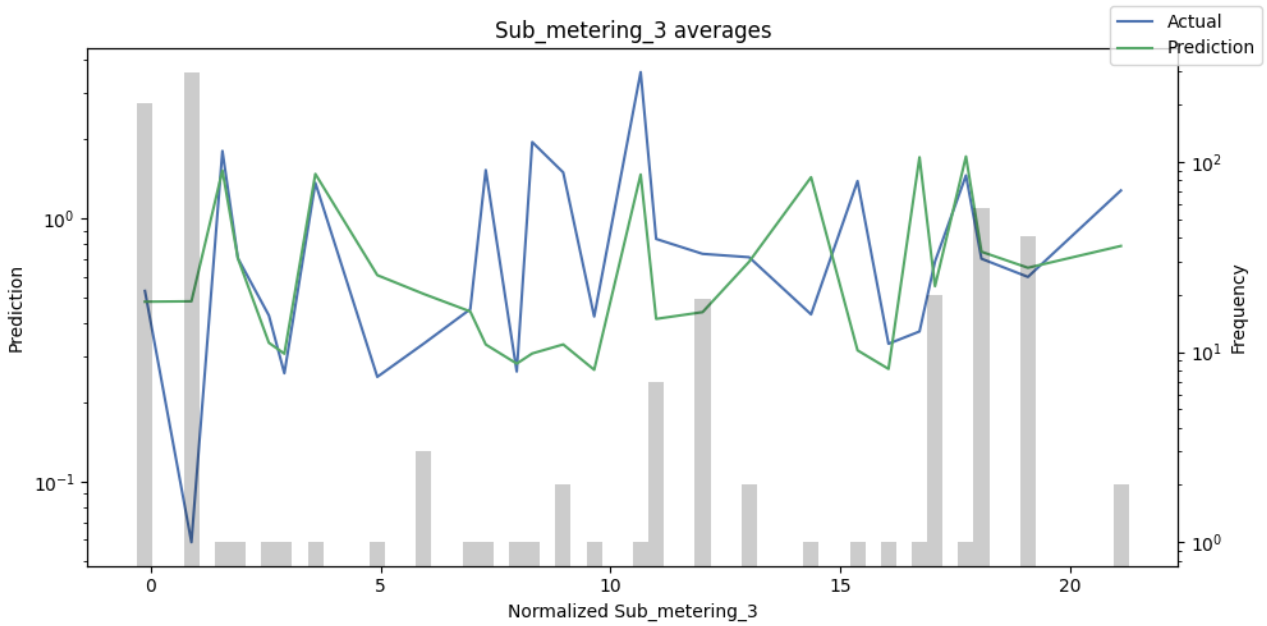

GT1_Hits averages
Actual
Prediction
Prediction
Frequency
Normalized GT1_Hits

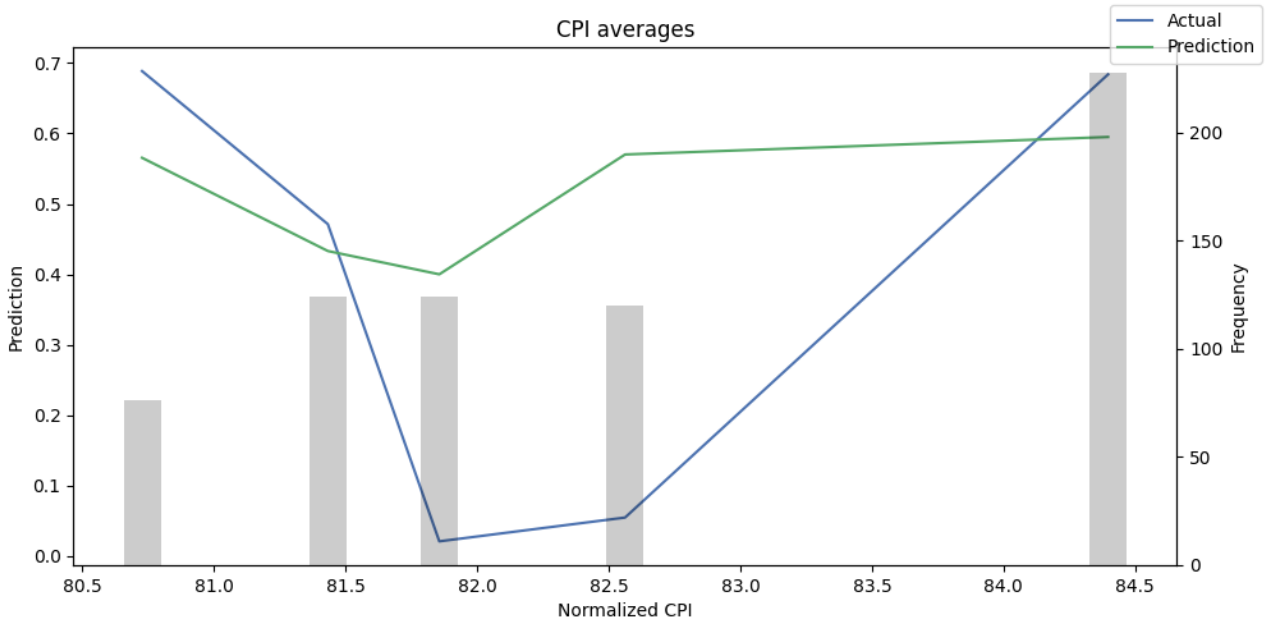

month averages
Actual
Prediction
Prediction
Frequency
month

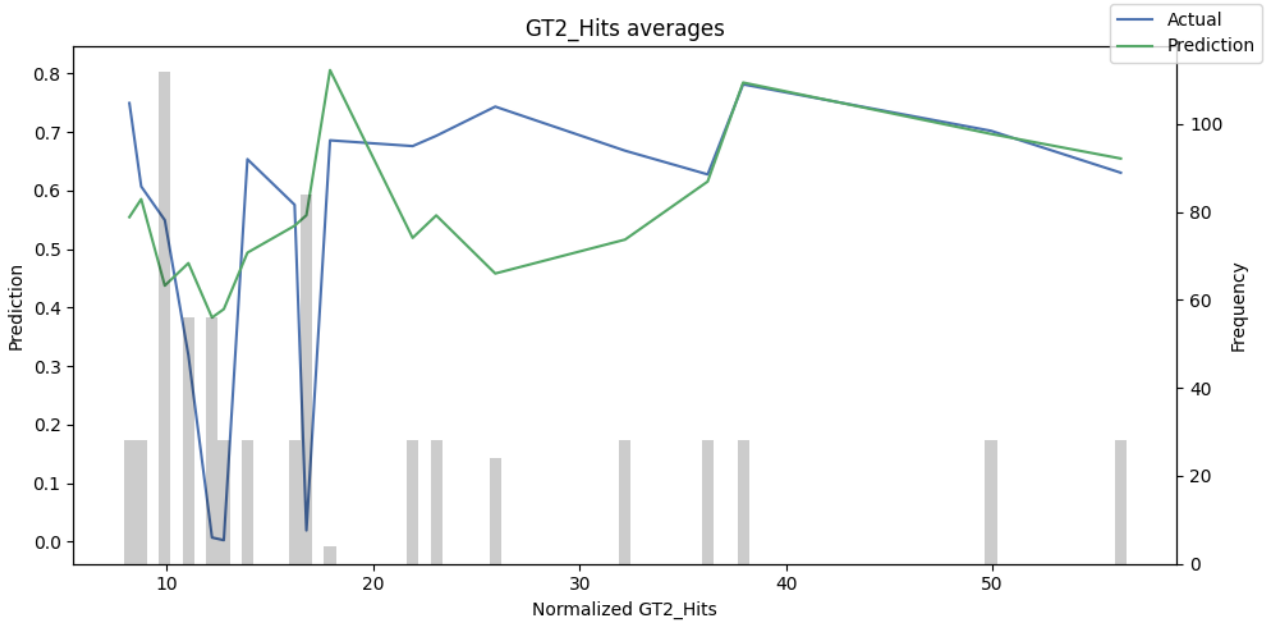

GT3_Hits averages
Actual
Prediction
Prediction
Frequency
Normalized GT3_Hits
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
100
200
300
400
22
24
26
28
30
32
34
36

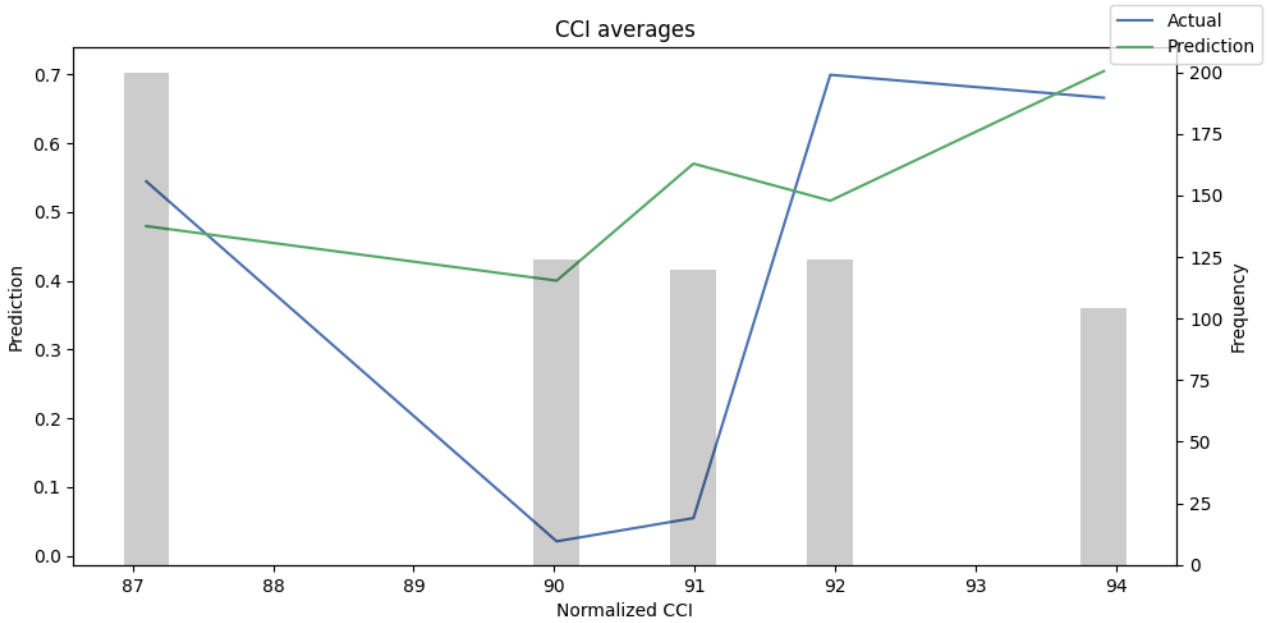

day_of_week averages
Actual
Prediction
Prediction
Frequency
day_of_week
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
2
4
1
3
0
6
5

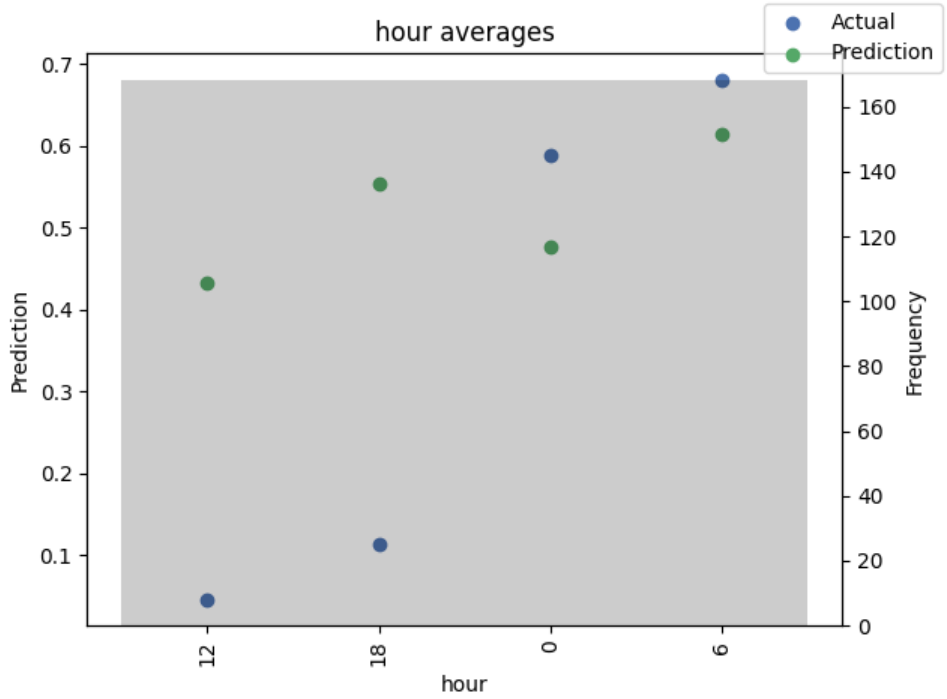

Air_Temp_Norm averages
Actual
Prediction
Prediction
Frequency
Normalized Air_Temp_Norm

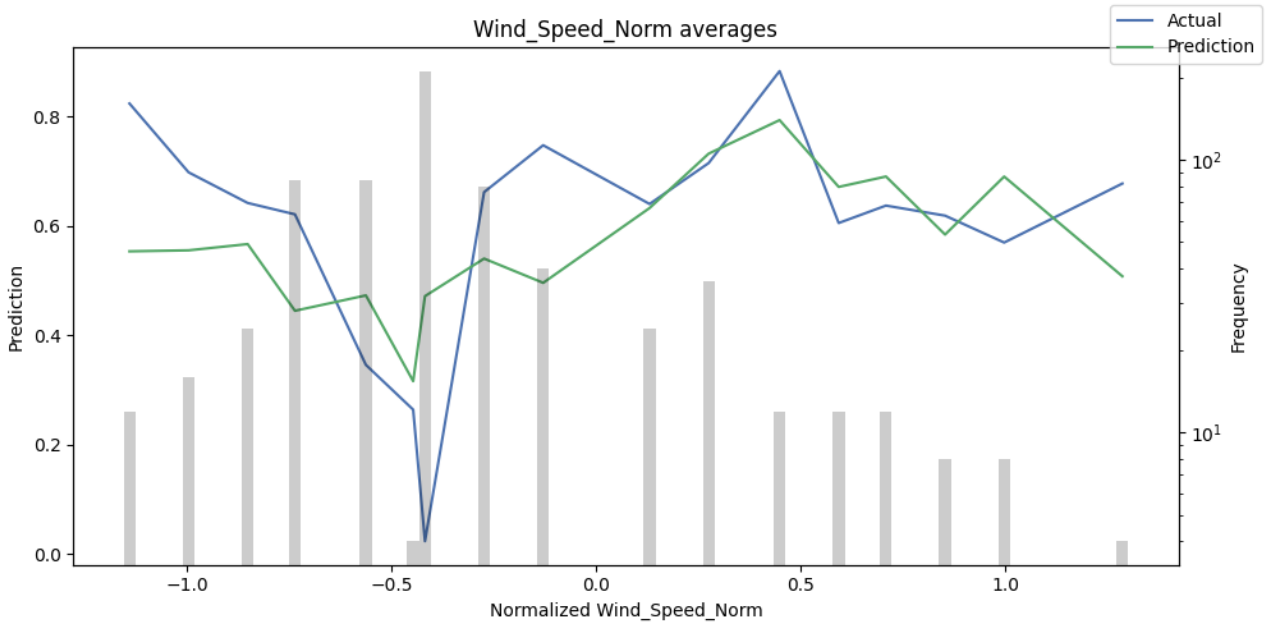

Sub_metering_3 averages
Actual
Prediction
Prediction
Frequency
Normalized Sub_metering_3

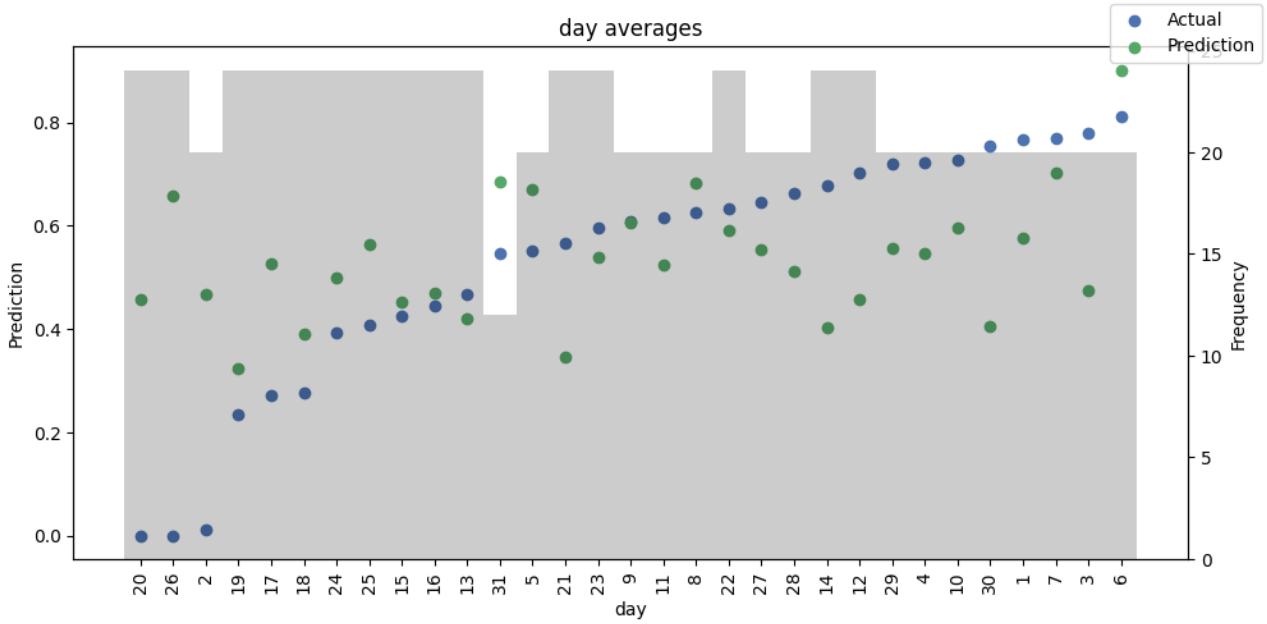

CPI averages
Actual
Prediction
Prediction
Frequency
Normalized CPI
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
50
100
150
200
80.5
81.0
81.5
82.0
82.5
83.0
83.5
84.0
84.5

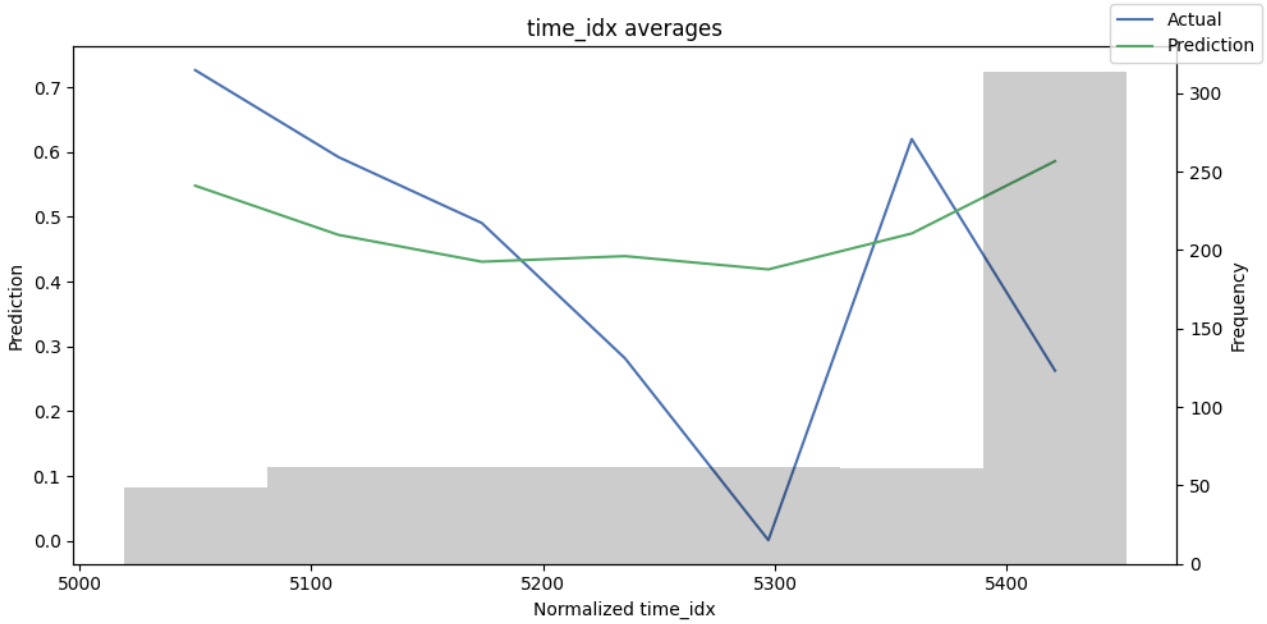

GT2_Hits averages
Actual
Prediction
Prediction
Frequency
Normalized GT2_Hits
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
20
40
60
80
100
10
20
30
40
50

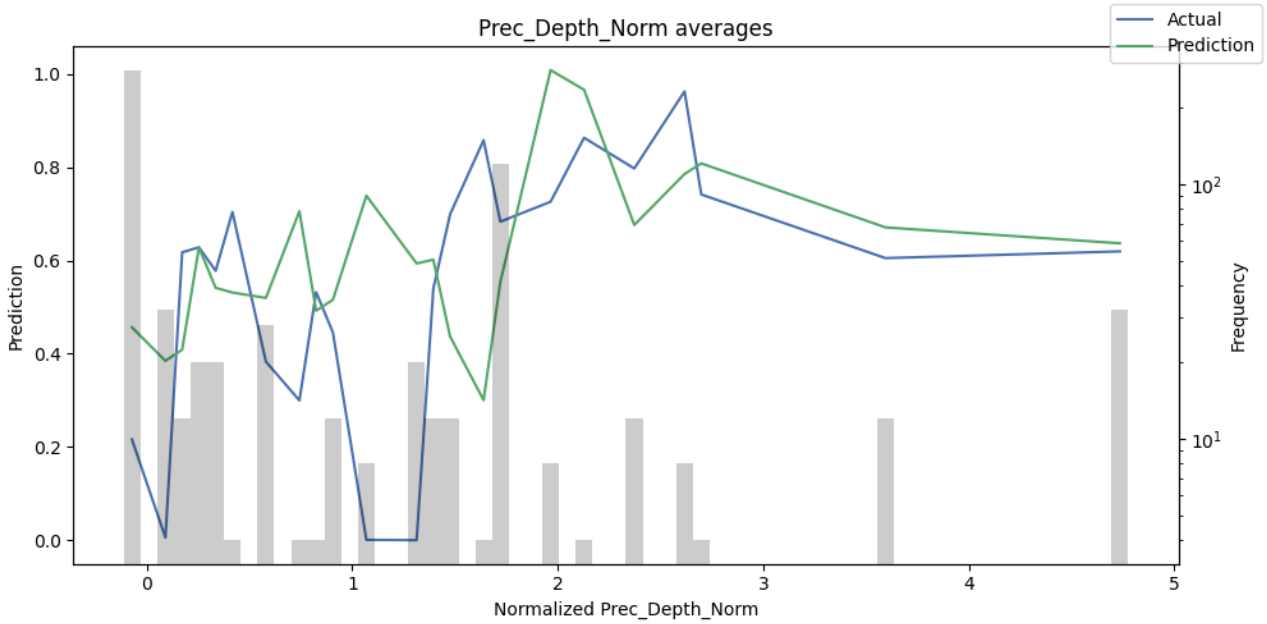

CCI averages
Actual
Prediction
Prediction
Frequency
Normalized CCI
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
25
50
75
100
125
150
175
200
87
88
89
90
91
92
93
94


hour averages
Actual
Prediction
Prediction
Frequency
hour
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100
120
140
160
12
18
0
6


Wind_Speed_Norm averages
Actual
Prediction
Prediction
Frequency
Normalized Wind_Speed_Norm
0.0
0.2
0.4
0.6
0.8
−1.0
−0.5
0.0
0.5
1.0
10^1
10^2


day averages
Actual
Prediction
Prediction
Frequency
day
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
20 26 2 19 17 18 24 25 15 16 13 31 5 21 23 9 11 8 22 27 28 14 12 29 4 10 30 1 7 3 6





In [ ]:
```
dependency = best_tft.predict_dependency(val_dataloader.dataset, 'relative_time_idx', np.linspace(0, 300, 3
# plotting median and 25% and 75% percentile
agg_dependency = dependency.groupby('relative_time_idx').normalized_prediction.agg(median="median", q25=lar
ax = agg_dependency.plot(y="median")
ax.fill_between(agg_dependency.index, agg_dependency.q25, agg_dependency.q75, alpha=0.01)
```

```
Predict:   0%|          | 0/30 [00:00<?, ? batches/s]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
```

```
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
```

```
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
```

```
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
```

```
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
```

```
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
```

```
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature nam
es, but StandardScaler was fitted with feature names
  warnings.warn(
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/usr/local/lib/python3.10/dist-packages/lightning/pytorch/trainer/connectors/data_connector.py:441: The 'pr
edict_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
```

Out[ ]: `<matplotlib.collections.PolyCollection at 0x7d0be1430fa0>`