# Load disaggregation Challenge: Energy use in buildings

## 1. Executive Summary

* **Objective:** This study aims to disaggregate the heating and/or cooling loads of buildings using smart meter data. The competition addresses a critical issue in the construction sector, which accounts for a significant portion of global energy consumption. Effective disaggregation of these loads is essential for achieving global carbon emission reduction targets, implementing targeted energy efficiency measures, and reducing operational costs for building owners and operators. The data provided in the competition includes main energy meter readings and weather data, while the sub-metering data for heating and cooling is kept confidential for the evaluation of results.
* **Key Findings:** The developed algorithm demonstrated high accuracy in disaggregating heating and cooling loads, achieving a Normalized Mean Absolute Error (NMAE) of 0.2557 across all analyzed buildings in public train leaderboard of the competition. The algorithm stands out for its simplicity, as it does not require additional temperature or weather data and needs minimal data cleaning. Additionally, the methodology was designed to be scalable, using reasonable computational resources and ensuring its reproducibility by domain experts, as defined by the competition's qualitative criteria.

## 2. Introduction

**Context about energy disaggregation:** Energy disaggregation in buildings is a critical field of research and development that seeks to identify and quantify the energy consumption of individual services within a building, such as heating, cooling, ventilation, lighting, and appliance loads. This process is essential for better understanding how energy is consumed and for implementing more targeted energy efficiency strategies. However, many existing buildings are equipped only with main energy meters that record total energy consumption, without detailing specific services. This makes disaggregation a significant challenge, especially in a context where installing detailed sub-metering systems is often unfeasible due to costs and logistical complexities.

**Project Objectives:** The main goal of this project is to develop an energy load disaggregation algorithm that can operate unsupervised and at various time resolutions, which can range from data collected every 5 minutes to data collected hourly, depending on the building in question. The purpose is to accurately disaggregate heating and/or cooling loads from the main meter data, thus addressing the challenges associated with the lack of specific device measurements and the diversity of building practices and standards. This project also seeks to promote generalization across different types of buildings to improve the applicability and efficiency of the algorithms developed in a wide range of construction scenarios.

## 3. Methodology

To disaggregate heating and cooling loads in buildings, an Adjusted STL algorithm (Seasonal-Trend Decomposition using LOESS) was developed. This method addresses the challenges associated with traditional decomposition techniques when applied to complex and noisy energy consumption data. The Adjusted STL combines the stability of classical decomposition with the adaptability of the STL method, resulting in accurate and robust predictions.

### Initial Approaches and Challenges

*Classical Decomposition:*

Initially, we employed classical decomposition methods to analyze the energy consumption data. Although this approach effectively determined the overall trend (height) of the data, it failed to accurately capture the baseline level. The baseline was significantly below the ideal point, and the seasonal component remained constant, unable to adapt to periods of noisy or rapidly changing seasonality (Figure 1). This rigidity made it difficult to adjust in situations where the data were imperfect or contained noise, especially when seasonality was unstable.

Gráfico, Histograma

Descrição gerada automaticamenteGráfico, Gráfico de barras

Descrição gerada automaticamente

Figure 1: Inflexibility of Classical Decomposition

* + Demonstrates how the classical decomposition method fails to adapt to changing seasonality, resulting in a misaligned baseline and constant seasonal patterns.

*Multiplicative Models:*

We also experimented with multiplicative models to address these issues. However, these models did not produce satisfactory results, as they still could not handle the complexities of noisy and changing seasonal patterns.

**Ref1 –** Article that generated the initial idea of the decomposition model and explains how this method functions with classical decomposition.

In Appendix B, I show how the calculations for classical decomposition were done and the automatic method for adjusting the size of height and baseline that ultimately became the main idea and culminated in the adjusted STL. The classical decomposition method greatly aided in developing the final model and in some buildings where the data were of high quality, it even generated the best results, making it interesting to test and even use depending on the quality of the building's data.

*STL Decomposition:*

Subsequently, we explored the standard STL decomposition method. The STL approach proved robust in adapting to noisy seasonal patterns and handling outliers. It provided a flexible seasonal component, capable of adjusting to complex and evolving seasonal trends. However, when seasonality changed drastically, the STL method tended to excessively amplify the baseline level and seasonal values, leading to distortions in the representation of the data (Figure 2). This overfitting made the predictions unreliable, particularly in scenarios with significant variations in energy consumption.

Gráfico, Histograma

Descrição gerada automaticamenteGráfico, Gráfico de barras

Descrição gerada automaticamente

Figure 2: Overfitting in Standard STL Decomposition

* + Shows the excessive amplification of the baseline level and seasonal values when using standard STL decomposition on data with rapidly changing seasonality.

### Development of the Adjusted STL Method

To overcome the limitations of previous methods, we developed the Adjusted STL algorithm, which integrates the stable baseline and trend components of classical decomposition with the adaptable seasonal component of the STL method. The main innovation lies in how the decomposition values are applied to the forecasting periods, ensuring stability and adaptability without distorting the baseline and trend data.

Gráfico, Histograma

Descrição gerada automaticamenteGráfico, Gráfico de barras

Descrição gerada automaticamente

Figure 3: Control and Adaptability of the Adjusted STL

* The figure illustrates how the Adjusted STL effectively combines seasonal adaptability with precise control of the height of the forecasts.

Main Steps of the Adjusted STL Pipeline

1. Data Preprocessing and Cleaning:
   * We apply data cleaning techniques to remove anomalies and ensure data quality.
   * The Adjusted STL method remains robust even with imperfect or noisy data, reducing the need for rigorous data cleaning, unlike the classical decomposition model, which requires nearly perfect quality data to produce satisfactory results.
2. Selection of Stable Reference Weeks:
   * We identify "stable reference weeks" where is suggested there was no energy consumption associated with heating or cooling (Figure 4, left).
   * These weeks serve as a basis for adjusting the seasonal and trend patterns in future forecasts.
3. Selection of Baseline and Height:
   * We calculate the median of the baseline and height from the reference weeks.
4. Selection of Multiplier for Heating and Cooling:
   * In buildings with heating and cooling systems, we identify the need for a global multiplicative factor to adjust the discrepancy between the reference weeks and actual consumption, as the reference weeks do not adequately capture the consumption of these systems. For further details, please refer to Appendix C
5. Baseline and Height Adjustment Using Multipliers:
   * Multipliers are applied to adjust the baseline and height levels in forecasts, particularly for heating and cooling scenarios.
6. STL Decomposition:
   * STL decomposition is performed on the energy consumption data.
   * Seasonal, trend, and residual components are extracted.
7. Forecasting Method for Plugs and Electrical Components:
   * The adjusted STL decomposition is used to forecast loads for plugs and electrical components.
   * Normalization is applied, and seasonal trend, baseline, and height are adjusted as necessary (Figure 4, center).
8. Forecasting Method for Temperature-Dependent Energy Consumption:
   * Temperature-dependent energy consumption is disaggregated by comparing total consumption to adjusted forecasts.
   * Energy used for heating and cooling is isolated (Figure 4, right).

Interface gráfica do usuário, Gráfico, Histograma

Descrição gerada automaticamente

* Figure 4: Forecasts Using the Adjusted STL Method
  + ***Left (Fine-Tuning of Forecasts):*** *Illustrates the use of stable reference weeks to finely tune forecasts, dynamically adjusting seasonal and trend components based on weekly quantiles.*
  + ***Center (Forecasting from Decomposition):*** *Compares total energy consumption with forecasts for plugs and electrical components, highlighting the model’s ability to adapt to data dynamics while controlling the sizes of the forecasts.*
  + ***Right (Forecasting Temperature-Dependent Consumption Disaggregation):*** *Presents the disaggregation of temperature-dependent energy consumption, showing precise isolation of heating and cooling loads.*

The Adjusted STL algorithm effectively addresses the shortcomings of classical decomposition and standard STL methods by combining their strengths. It ensures precise, stable, and robust forecasts for energy consumption disaggregation, particularly in datasets with complex or noisy patterns. This method not only improves adaptability to changing seasonal trends but also maintains necessary control over baseline and trend components, making it highly suitable for practical applications in energy consumption analysis.

### Data Requirements and Sources

The model utilized total energy consumption data (kw\_total) obtained from main meters and photovoltaic batteries installed in various buildings. These data, recorded at regular intervals, served as the primary time series for decomposition. The kw\_total encompasses both the consumption from main sources and energy usage from photovoltaic batteries.

data\_energy['kw\_total'] = data\_energy['main\_meter(kW)'] + data\_energy['PV\_battery\_system(kW)']

A significant advantage of the Adjusted STL algorithm is its independence from external variables such as temperature or weather. This allows its application in scenarios where such data are not available or where seasonal energy consumption does not solely depend on climatic variations, simplifying its implementation. The main variables used in the algorithm include:

The main variables used in the algorithm include:

* **Total energy consumption (kw\_total):** Basis for analysis and decomposition.

These data allowed for the identification of stable reference weeks—periods when the heating/cooling load was not used or was at its minimum state in the time series—facilitating the disaggregation of loads associated with electrical appliances and lighting.

The primary requirement for the algorithm is that the time series data be synchronized without null energy data. This simplicity, combined with the robustness of the method, allows for efficiently adjusting reference periods without relying on additional external information, such as meteorological data.

### Training and Testing Procedures

The training of the Adjusted STL algorithm was carried out by initially optimizing the size of weeks through classical decomposition, a procedure described in Appendix B. This was followed by manual adjustments using the visualization of the time series for each building to identify the best data points to be selected for the stable reference weeks—periods without energy consumption related to heating or cooling or the periods of lowest possible consumption related to temperature.

During the training, key parameters of the model were optimized:

* **median\_height and median\_base:** These parameters control the height and baseline of the plug and lighting components on the total energy chart. The optimization of these parameters aimed to find the ideal size to allow optimal disaggregation of seasonal components using the STL.

After adjusting the parameters, the testing phase involved comparing the model’s forecasts with results obtained on the public training leaderboard of the competition. This practical validation allowed for additional refinements, ensuring the best possible disaggregation of energy loads and the accuracy of the forecasts.

## 4. Data

**Data Sources**: The data used were provided by the competition for training, which includes the main energy meters of buildings (main meters) and meteorological data.

**Data Aggregation**: Initially, the data were aggregated using an aggregate\_data function, which combined all the data into a single DataFrame and performed a merge with the temporal data corresponding to each building.

**Data Cleaning and Preparation Process:** For each building, identified by its real ID (e.g., L14.B04\_1H), the following data cleaning pipeline was executed:

1. **Data Loading:** The data are loaded into the aggregated DataFrame through the load\_dataframe function. A synchronization is performed, if necessary, using the check\_sync\_and\_fill\_gaps function.
2. **Removal of Intervals and Holidays:** Cleaning variables such as intervals, holidays, years, and local information (country\_pt) are collected in the parameters. The remove\_intervals\_and\_holidays function is applied to remove the selected intervals and holidays, transforming these data into null values.
3. **Data Imputation:** The null values are then imputed using the fill\_na function, which groups the time series by hour and day of the week, allowing the imputation of data in a way that respects the seasonality and the nearest data.
4. **Results Verification:** The data are visualized for manual verification of the final result.
5. **Data Storage:** Two DataFrames are saved: one with the cleaned data and another with the original data, organized by each building and granularity.

The preprocessing\_and\_load function executes the pipeline.

Texto

Descrição gerada automaticamente

**Figure 1: Code for Preprocessing Pipeline Function**  
This figure displays the code of the function that executes the preprocessing pipeline, detailing the steps involved in data cleaning, synchronization, and preparation for analysis.

**Data Characteristics:** The available data are detailed in the attached table, which highlights the diversity of building types and the varied temporal resolutions of data collection, ranging from 5 minutes to 1 hour. These resolutions cover a wide range of buildings, including offices, schools, shopping centers, and senior living facilities, each with specific energy characteristics and demands. The complexity is further increased by the presence of battery systems or photovoltaic (PV) systems in some buildings, adding an extra layer to the analysis of the energy load. In total, there are 14 datasets, each representing a different building or a distinct granularity.

Tabela

Descrição gerada automaticamente

**Figure 2: List of Buildings and Key Information**  
This figure provides a detailed list of buildings along with important information such as country, area, and data granularity, illustrating the diversity of the dataset used in the study.

**Seasonality as a Key Data Feature:** Seasonality emerges as the most relevant characteristic of the data, primarily manifesting by hour and by day of the week. For instance, in an hourly chart, the seasonality would correspond to 168 lines. Therefore, it is crucial that the data are synchronized and free of nulls to prevent distortions in the forecasts derived from this extended seasonality.

**Significance of Differentiating Workdays and Business Hours:** Many buildings exhibit distinct temporal variations, with periods of occupancy and vacancy that can be clearly demarcated. A technique that proved effective in this separation was the use of Hidden Markov Models, which accurately distinguished between working and non-working periods. This allowed for a more complex approach to data decomposition, although it was not adopted due to yielding inferior results compared to weekly decomposition.

Uma imagem contendo Gráfico

Descrição gerada automaticamente

**Figure 3: Application of the Hidden Markov Model (HMM)**  
This figure illustrates an example of using the Hidden Markov Model (HMM) to differentiate between the occupancy and unoccupied periods in a building. The model effectively segments the data, highlighting the distinct patterns of energy use during occupied versus unoccupied times.

## Algorithm Implementation

**Software Used**

* Python: The implementation uses Python version 3.12.4, chosen for its stability and support for scientific libraries.
* Specific Libraries: The necessary libraries are listed in the requirements.txt file. This file includes packages such as numpy, pandas, matplotlib, scipy, and statsmodels, which are essential for data manipulation, statistical analysis, and graphical visualization.

**Project Repository:**

* The source code of the project is hosted in a private repository on GitHub. To access the repository <https://github.com/rafaelsudbrackzimmermann/REP_ADRENALIN>, interested parties should request access by sending an email to [rafaelszimmermann@gmail.com](mailto:rafaelszimmermann@gmail.com)
* <https://github.com/rafaelsudbrackzimmermann/REP_ADRENALIN>

**Optimization Parameters:**

* All parameters used for the optimization and adaptation of the algorithm to the data are available in Appendix A of this report; or can also be viewed in the \_0\_parameters.py file located on GitHub.
* The execution pipeline used to adjust and apply these parameters to the data can be seen in the development of the adjusted STL method in 3. Methodology or can also be viewed in the \_2\_model.py file located on GitHub.

## 6. Results and Evaluation

**Performance Metric:** The employed algorithm is an unsupervised disaggregation method. Therefore, the training evaluation metric adopted is the Normalised Mean Absolute Error (NMAE), which is used in the competition to consolidate all datasets into a single error result. In the case of the adjusted STL model implemented in this submission, the training result was 0.255.

**Comparative Analysis with Other Methods:** The algorithm's performance was compared with other traditional techniques for energy disaggregation in buildings, such as classical decomposition and decomposition using adjusted STL.

* **Classical Decomposition:** In buildings where the analysis involved only heating, the adjusted classical decomposition proved quite promising, as presented in Appendix B, achieving an NMAE of 0.320 in the competition’s training submissions. However, when applied to buildings that required heating and cooling analysis, the results were merely acceptable.
* STL Decomposition: This technique showed unsatisfactory results in most buildings, mainly due to the distortion that occurred when trying to apply the same methodology as classical decomposition. The results were particularly poor, with NMAE exceeding 0.400.
* Adjusted STL: The model that used adjusted STL exceeded expectations, managing to harmonize the data behavior without distortion in predictions. This model achieved a result of 0.255.

In some buildings, variations of the adjusted STL and classical decomposition were combined and achieved an NMAE of 2.430 in training, suggesting a possible future submission integrating both methods where there was success.

A filter that adjusts the predictions to the temperature data was developed, providing an improvement in results by reducing the best NMAE to 0.240. Depending on the results of the final submissions, this method may be added in future submissions. However, its complexity and the increased data requirement are seen as disadvantages, especially considering the small reduction in the score.

Moreover, various other methods were tested, including regression and several unsupervised approaches to NILM, deep learning and data clustering. However, none of them provided better results than the adjusted STL, and in most cases, they added excessive complexity to the process, leading to their exclusion.

**Error Analysis and Observed Limitations:** A significant limitation found is the need for data to be clean, synchronized, and imputed for a data decomposition to be done without errors or distortions. This process demands a cleaning work or, as was chosen, the selection of reference weeks, which, in my opinion, makes more sense and is more advantageous than cleaning the data.

Depending on the building and the specific scenario, opting for data cleaning or the selection of reference weeks may result in better or worse performances, but in the majority of cases, the selection of reference weeks wins this dispute.

In this study, the pipeline that offers the best results for each building was chosen, despite being more labor-intensive and possibly requiring additional adjustments. However, this pipeline could be fully automated, maintaining consistency in the results, especially if the data are of high quality and are well synchronized and clean. This is the main limitation of the algorithm: when these conditions are not met, it becomes necessary to undergo a cleaning process or adjust the reference weeks for each building with specific parameters.

## 7. Reproducibility

* **Step-by-Step Instructions:**

1. **Clone the Repository**: Start by cloning the GitHub repository where the project code is hosted. The link can be found in the 'Algorithm Implementation' section of this report.
2. **Install Dependencies**: Install all necessary libraries using the command pip install -r requirements.txt, or use the environment.yml file if you prefer, which is included in the repository. This file contains all the Python libraries required to run the project.
3. **Data Setup**: Download the data as described in the 'Data' section and place it in the data/train\_public folder.
4. **Variable Adjustment**: If necessary, adjust the PATH variable in the code/\_4\_main.py file.
5. **Execute the Script**: Run the script from the \_4\_main.py file.
6. **Evaluate the Results**: Use the generated charts to evaluate the decomposition of each building based on its time series, plugs, and temperature-dependent variables.

* **Configuração do Ambiente:**
  1. **Hardware:** Idealmente, um computador com pelo menos 16 GB de RAM e um processador moderno (i5 ou superior) para processamento eficiente.
  2. **Sistema Operacional:** O código é compatível com Windows, macOS e Linux.
  3. **Dependências:** Python 3.8 ou superior, Pandas, NumPy, Statsmodels, Bokeh, e Metpy. Detalhes específicos e versões estão listados no arquivo requirements.txt ou no arquivo environment.yml.
* Environment Setup:
  1. **Hardware:** Ideally, a computer with at least 16 GB of RAM and a modern processor (i5 or better) for efficient processing.
  2. **Operating System**: The code is compatible with Windows, macOS, and Linux.
  3. **Dependencies**: Python 3.8 or higher, Pandas, NumPy, Statsmodels, Bokeh, and Metpy. Specific details and versions are listed in the requirements.txt or environment.yml file.
* Reproducibility Checklist:
  + Source code cloned from the GitHub repository.
  + All dependencies installed as per requirements.txt or environment.yml.
  + Data downloaded and stored in the correct location.
  + Scripts executed in the correct order.
  + Hardware and software environment set up as per specifications.
  + Results evaluated by analyzing the decomposition charts.

## 8. Discussion

**Interpretation of Results**: The results demonstrate that the adjusted STL model provides consistent and robust energy disaggregation. The STL's ability to adjust to the trends and seasonality of the data allows for effective disaggregation even when the data quality is not ideal. This is particularly true in cases where only heating needs to be disaggregated, and the seasons play a crucial role, identifying weeks when heating is predominantly turned off, serving as excellent references. The accuracy achieved, with an NMAE of 0.255, highlights the model's effectiveness in capturing complex consumption patterns, despite data limitations. This success reinforces the potential of unsupervised modeling as a valuable tool in energy management in buildings, even when only main meter data are available.

**Study Limitations**: Despite promising results, the study faced some limitations that could impact the generalization of the results:

* The model may encounter difficulties mainly if the building has a system that stays on in seasons that do not require cooling or heating. In these cases, it is more challenging to perform the disaggregation of the reference weeks, necessitating a new step, described in Appendix C, which involves a global disaggregation using multipliers from the reference weeks to find the ideal disaggregation value for those weeks, increasing the algorithm's complexity.
* If the data are of low quality, especially if they are not synchronized or show weeks of atypical behavior, such as total absence of occupation during vacations, this can distort the reference weeks and substantially impair the algorithm's disaggregation. In these cases, it becomes necessary to implement a new step to ignore these atypical data.

**Suggestions for Future Work**: In future research, it would be beneficial to try incorporating weather condition data into the disaggregation. It would also be productive to develop an automatic data quality assessment model, especially to identify atypical weeks, as this could significantly improve the algorithm's results in scenarios of complete pipeline automation.

Investing more time in analyzing seasonality, trying to divide the data between workday and weekends and performing separate disaggregations for each scenario, is another promising area. Although previous attempts have not been fruitful, persisting in this approach might be worthwhile and could potentially elevate the quality of the algorithm. Finally, a test with a larger number of buildings would be interesting to determine in which scenarios the model performs best and to identify where improvements are needed, as suggested in Appendix C.

## 9. Conclusion

**Summary of Findings**: This study details the successful application of the STL model for energy disaggregation in buildings, even under data quality limitations. The model's ability to adapt to diverse trends and seasonality stood out as a central element for achieving robust and consistent disaggregations, particularly effective in scenarios where only heating is disaggregated and seasonal variations play a crucial role. The accuracy demonstrated by the NMAE of 0.255 emphasizes the model’s competence in capturing complex consumption patterns, offering a promising method for efficient energy management.

However, some limitations were identified, such as disaggregation difficulties in buildings with systems that remain on in seasons not dependent on heating or cooling and the potential distortion caused by low-quality data or atypical weeks. These challenges suggest the need for adjustments in the algorithms, such as the introduction of multipliers from the reference weeks to improve the accuracy of disaggregation under these particular conditions.

**Recommendations Based on Results**: Based on the results obtained, it is recommended to incorporate climatic data to enhance the accuracy of disaggregations and to create an automatic model for data quality assessment, focusing on the identification of atypical periods. Additionally, it is suggested to persist in exploring the division of data between periods of occupancy and vacancy, performing specific disaggregations for each scenario, which could significantly improve the model's efficacy.

Further studies involving multiple buildings could elucidate in which scenarios the model is most efficient. For example, in buildings that during seasons like summer and winter have reference weeks that well represent the consumption of plugs and lighting, the model can perform a very effective disaggregation of heating or cooling energy. This would allow a better understanding of which building configurations and uses benefit most from the application of the STL model.

For scenarios where clear reference weeks are not obtained, in which the energy of plugs and lighting are not adequately disaggregated in the decomposition, the development of additional steps in the model would be necessary. One of these steps could be the application of multipliers adjusted to the specific consumption characteristics of each building, obtained through an analysis of peaks and valleys, or global maxima and minima of consumption as detailed in Appendix C.

## 10. References

1. Narges Zaeri, H. Burak Gunay, and Araz Ashouri. 2022. Unsupervised energy disaggregation using time series decomposition for commercial buildings. In Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '22). Association for Computing Machinery, New York, NY, USA, 373–377.  
   <https://doi.org/10.1145/3563357.3566155>   
   https://nilmworkshop.org/2022/proceedings/nilm22-final97.pdf
2. Barsim, Karim Said & Streubel, Roman & Bin, Yang. (2014). Unsupervised Adaptive Event Detection for Building-Level Energy Disaggregation.
3. José Alcalá, Jesús Ureña, Álvaro Hernández, and David Gualda. 2017. Event-Based Energy Disaggregation Algorithm for Activity Monitoring From a Single-Point Sensor. IEEE Transactions on Instrumentation and Measurement 66, 10 (October 2017)
4. Jichuan Yan, Xinxin Ge, Xiaoxing Lu, Fei Wang, Kangping Li, Hongtao Shen, and Peng Tao. 2021. Joint Energy Disaggregation of Behind-the-Meter PV and Battery Storage: A Contextually Supervised Source Separation Approach.
5. Ying Ji, Peng Xu, and Yunyang Ye. 2015. HVAC Terminal Hourly End-Use Disaggregation in Commercial Buildings with Fourier Series Model. *Energy and Buildings* (March 2015). DOI: <https://doi.org/10.1016/j.enbuild.2015.03.048>.

## 11. Appendices

### Appendix A - Parameters Used

**Parameters Used for Setting and Optimizing the Model by Building: Detailed Description**This appendix aims to explain in detail the parameters used to set up and optimize the model applied to each building.

Decomposition ParametersThe following parameters are employed to decompose the time series and determine the best fit for the data:

Texto

Descrição gerada automaticamente

* **Seasonality** (Int 1-9999):
  + **Description:** Defines the seasonal period considered in the analysis.
  + **How to Set:** Select the period that corresponds to the expected seasonality (e.g., 168 for weekly, 24 for daily).
* **Multiplier** (Float 0-1):
  + **Description:** Applies a multiplication factor to adjust the data size amplitude, aiming to find the ideal factor for specific components like plugs and electricity.
  + **How to Set:** If necessary, values between 0.5 to 0.7; otherwise, multiply by 1.
* **STL** (Decomposition STL) (String ‘stl’, ‘stl\_adjusted’, ‘classic’):
  + **Description:** Specifies the type of decomposition to be applied to the data.
  + **How to Set:** Select the most suitable method based on eda (exploratory data analysis) studies.
* **STL Parameters:**
  + **Description:** Allows for detailed customization of the STL decomposition through specific settings.
    - **Trend Smoothing** (stl\_trend) (Int 1-9999): Controls the degree of smoothing applied to the trend component.
    - **Seasonality** (stl\_seasonal) (Int 1-9999): Adjusts the smoothing window for the seasonal component.
    - **Low-Pass Filter** (stl\_low\_pass) (Int 1-9999): Determines additional filtering to separate high-frequency components.
    - **Robustness** (stl\_robust) (Bool): Enables robust decomposition to reduce the impact of outliers.

#### Week plug Selection:

Texto

Descrição gerada automaticamente Gráfico, Diagrama

Descrição gerada automaticamente

**Week plug Selection** is a set of parameters used in the decomposition algorithm to robustly select reference weeks. These parameters effectively handle outliers and noise in time series data, such as holiday periods, without the need for exhaustive data treatment.

The algorithm uses six main parameters, provided in list format, that control how reference weeks are selected and how data is normalized.

* + **Parameter 0 – Minimum Quantile** (week\_lower\_quantile) (Float 0 to 1)
    - **Description:** Defines the quantile used to determine the minimum value considered for each reference week. For example, a value of 0.01 indicates that the 1% quantile of the data grouped by year and week of the building will be used.
    - **Purpose:** Adjust the calculation of the weekly minimum to reduce the impact of lower outliers.
  + **Parameter 1 – Maximum Quantile (week\_upper\_quantile) (Float 0 to 1)**
    - **Description:** Defines the quantile used to determine the maximum value considered for each reference week. For example, a value of 0.99 indicates that the 99% quantile of the data grouped by year and week of the building will be used.
    - **Purpose:** Similar to Parameter 0, but focused on upper outliers, adjusting the weekly maximum to more accurately reflect the data distribution.
  + **Weekly Median Calculation (week\_median):** The week\_median is a central adjusted measure that considers both the upper and lower quantiles.   
    week\_median = (week\_upper\_quantile + week\_lower\_quantile) / 2
  + **Parameter 2 – Minimum Quantile for Week Selection (Float 0 to 1)**
    - **Description:** Establishes the minimum threshold of the week\_median for a week to be included as a reference. For example, a value of 0.05 selects weeks whose week\_median is above the 5% quantile.
    - **Purpose:** Filter out weeks with abnormally low values, possibly due to noise or atypical periods.
  + **Parameter 3 – Maximum Quantile for Week Selection (Float 0 to 1)**
    - **Description:** Defines the maximum threshold of the week\_median for including reference weeks. A value of 0.25 includes weeks with a week\_median below the 25% quantile.
    - **Purpose:** Exclude weeks with values higher than typical for a reference week, selecting only those with little or no energy use for temperature adjustments.
  + **Note:** The settings for Parameters 2 and 3 should be adjusted according to the specific behavior of the data from each building. It is recommended to conduct an Exploratory Data Analysis (EDA) and visual adjustments to determine the most appropriate quantiles.
  + **Parameter 4 – Minimum Quantile for Normalization (Float 0 to 1)**
    - **Description:** Defines how outliers will be treated during the normalization of forecasts, specifically the minimum value considered for each week of forecast. A value of 0.01 indicates that the 1% quantile will be used as the minimum.
    - **Purpose:** Assist in normalizing data that is highly noisy or has lower outliers, making the process more effective.
  + **Parameter 5 – Maximum Quantile for Normalization (Float 0 to 1)**
    - **Description:** Similar to Parameter 4, but pertaining to the maximum value in the normalization of forecasts. A value of 0.99 uses the 99% quantile as the maximum.
    - **Purpose:** Adjust the normalization in the presence of upper outliers, ensuring consistency in the data scale.
  + **Practical Considerations:** Although it is possible to use default values (e.g., 0.01 and 0.25 for parameters 2 and 3, and 0.01 and 0.99 for parameters 0, 1, 4, 5) with generally good results, it is beneficial to adapt the parameters to the specific characteristics of the available data to optimize the performance of the algorithm.

#### Seasonal Trend Power:

**** Gráfico, Histograma

Descrição gerada automaticamente

**The Seasonal Trend Power** is an essential component of our model that adjusts the influence of seasonality on the analyzed data. This parameter allows for a more precise adaptation of the seasonality height to the observed data and modifies how the calculation base is established, optimizing the representation of seasonal fluctuations in forecasts.

The adjustment of the seasonal trend power is carried out through two main parameters that configure the adaptation and the base for the normalization of the data calculation.

* **Parameter 0 – Trend Adaptation Value (Float 0 to 1)**
  + **Description:** Defines the intensity with which the height of the data adapts relative to the weeks that show greater fluctuations than the reference weeks. For example, a value of 0.1 means that 10% of the intensity of the most expressive weeks will be added to the maximum values of the reference weeks.
  + **Purpose:** This parameter allows the model to flexibly adjust the height of the seasonal trend in a smoothed manner.
* **Parameter 1 – Type of Data Base to Use (String ‘norm\_min’ or ‘Base\_min’)**
  + **Description**: Specifies the type of minimum base that will be used for normalizing the reference weeks in the construction of the forecasts.
    - **norm\_min**: Uses the minimum of each week as its base.
    - **base\_min**: Uses the minimum of the reference weeks.
  + **Purpose**: The choice of base type allows for adapting the forecasts in a way that more accurately and adaptively reflects the minimum normalization variation of the data.

#### Data Cleaning and Preparation

Texto

Descrição gerada automaticamente

To ensure that the data are suitable for decomposition and analysis, the following cleaning parameters are used:

**Country (String of the country)**:

* Specifies the geographical context of the data in the holiday cleaning function.

**Year (list of years)**:

* Indicates the years to be used in the holiday cleaning functions and intervals.

**Clear Holidays (Bool)**:

* Whether or not to use the holiday cleaning function from the holidays library.

**Interval (List of tuples)**:

* Tuples of intervals to be removed from the time series.

**Holidays (Specific Holidays) (List of tuples)**:

* Tuples with holidays to be removed from the data.

### Appendix B - Classical Decomposition

This appendix aims to explain the formula used in classical decomposition and the method used to optimize the height and base of the decomposition.

#### Classical Decomposition:

The classical decomposition model is described in detail in article ref1; it essentially decomposes the time series and uses seasonality + minimal trend to generate the forecast for plugs and electric lights. plugs = seasonality + trend.min()   
Desagregation = total energy - plugs

Gráfico, Histograma

Descrição gerada automaticamente

Fig 1: Classical Decomposition without Any Adjustment of Height and Base

This model generates interesting results; however, the fact that it uses the minimal trend means that the data must be perfect concerning outliers and of high quality, otherwise, a manual adjustment of the seasonality height and especially the base location of the forecasts is necessary, as shown in the image above.

#### Optimization Model and Adjustment of the Height and Base of Classical Optimization

The first idea to solve this problem was an optimization system, which selected the reference weeks and then adjusted the height and base of the data to this reference week using optimization.

Reference week = Data from weeks between the 0 to 0.25 quartile of the data.

Optimization that generates the adjustment of height and base:

error\_exponent = 10

error = np.mean(np.abs((Reference week) - prediction) \*\* (1 / error\_exponent))

Using this algorithm described in the code by this function (optimize\_parameters), the first idea of optimization using the reference weeks is implemented, which was later evolved into the system that performs the normalization and size adjustment of the data from the stl decomposition, within the adjusted stl model.

Gráfico, Histograma

Descrição gerada automaticamente

**Fig 2: Classical Decomposition with Optimization that Finds Better Adjustment Values for Height and Base of Forecasts Based on the Reference Week.**

In this image, we can see the optimization procedure that adjusts the size of the predictions to match the data size in the reference week, reducing the error and generating very well-fitted plug forecasts as well as a beautiful and elegant disaggregation of the energy used for temperature.

Tela de computador

Descrição gerada automaticamente

**Fig 3: Shows the optimization of classical decomposition from reference weeks to find the best fit.**

This image displays the optimization process and its final result, which finds an adjustment multiplication factor of 0.6999, and a height adjustment using the 0.5801 quantile of the data, as shown in the prediction from the decomposition of plugs in the middle image of the figure.

This method has proven extremely efficient in situations where the data are of excellent quality and the seasonality is stable. It also proved to be a great way to discover and optimize the ideal size of height and base of the data, to then apply the adjusted stl with similar height and base parameters.

An example of the final version of this building using the adjusted stl, based on the sizes optimized by the model.

Gráfico, Histograma

Descrição gerada automaticamente

**Fig 4: Shows the decomposition using adjusted STL, very similar to classical decomposition, because it uses its adjustment parameters to find the ideal height and base points in the reference week.**

Notice how the height of the reference week and the base have very similar values, which generate forecasts very similar to those of the classical decomposition, but in the model of the adjusted STL, we have greater control over how seasonality will be used in the forecasts.

The calculation that generates the plug forecasts is found in the function (calculate\_prediction) using classical decomposition.

Texto

Descrição gerada automaticamente

Fig 5: Shows how the trend adjustment and quantile data will be implemented in classical decomposition data to generate plug forecasts.

### Appendix C - Multiplier Selection - Ok for Heating and Cooling

#### Day Type – Occupation and Vacancy

The first part of the algorithm makes an important distinction between occupation and vacancy periods using the hhm. Below, we can see images of the hourly separation into occupation and vacancy for buildings L10 and L09.  
This idea was suggested in reference article 5, where it is proposed that, if data are separated by days of occupation/work and days of vacancy/weekend, instead of using weekly seasonality, daily seasonality could be used. This idea really makes a lot of sense. I have worked on it, but the weekly results were still better in my studies. However, I still have my doubts whether, with more effort on this idea, I could find a way to improve the results or create new ways to solve the problem.  
Some attempts to apply this approach are detailed below. This separation method can reveal several ways to perform disaggregation, providing insights into the ideal value of the global disaggregation multiplier in complex heating and cooling contexts. An example of this are the reference weeks in buildings L09 and L10, which have potentially more complex systems. Due to the lack of complete information, I can only speculate that these buildings operate with an HVAC system that remains intermittently on, even under ideal temperature conditions. It is worth noting that this assumption is based on speculation, as specific data on the operation of the systems were not provided. This hypothesis suggests that the systems are often active, integrated into ventilation and other energy demands, as described in reference article 5, which details the components of an HVAC terminal.

This analysis follows my line of reasoning in exploring the possible dynamics of energy consumption in the mentioned buildings.

Gráfico, Gráfico de dispersão

Descrição gerada automaticamenteGráfico, Gráfico de dispersão

Descrição gerada automaticamente

Fig1: It is observed that the separation made by the hhm considered the occupation hours, which, in this scenario, proved more appropriate than the separation by days of occupation. On the other hand, I tested an algorithm that added an extra step, separating the days of occupation from the days of vacancy. However, this approach was not as effective as the separation by occupation hours. Additionally, separating by days may introduce errors on holidays and atypical days, where the occupation pattern does not follow the conventional, compromising the accuracy of the model.

#### Model 1 - Searching for Multiplier Values Using Energy Consumption Quantiles Highly Correlated with Air Temperature

A model that proved logically viable for finding a global disaggregation multiplier value, i.e., a global average of plug use disaggregation for subsequent analysis of heating or cooling usage, was the strategy of separating data by time, season, and year. Next, it was checked whether, at that time and season, there was a high correlation with temperature and total energy consumed. If such a correlation existed, we could take the range of these data, thus determining the percentage of energy used for plugs and lighting.

On GitHub, there is a file called appendix c that demonstrates exactly how the code was applied.

Now, I will provide a brief explanation of the logic and how the model is calculated:

1. We expect a certain time to have a high correlation with temperature and take its minimum and maximum values to generate the range.
2. Then, we divide this value by the total range and subtract one to arrive at the value for plugs and lighting.
3. Quantiles were used, instead of maxima and minima, to determine the extreme values, considering distortions caused by outliers.
4. The formula used was: Plugs = 1 - ((quantile09 - quantile01) / quantile09)

This procedure was applied to both building L9 and building L10, with Plug values varying between 0.5 and 0.6, suggesting that, on average, the heating and cooling system consumes 40% to 50% of the energy, depending on the time, occupation, and building.

We acknowledge that this is not the ideal way to perform the calculation and that, for example, during periods of vacancy or nighttime, these values are likely lower. However, the goal here is to find an average for all data and not a complete disaggregation model, although it is possible to try to create a complete disaggregation model from this logic in the future. I have made some attempts, but they have not produced results as good as disaggregation from adjusted STL decomposition.

L09

Interface gráfica do usuário, Gráfico, Aplicativo

Descrição gerada automaticamente

For example, in building L9, we can observe that only certain times reached a correlation higher than 70%. For these times, the average analysis revealed a plug value of 0.60, with a standard deviation of 0.03, a minimum value of 0.55, and a maximum of 0.65. Broadly speaking, we could say that during periods of occupation, about 60% of the energy consumption is used for plugs and lighting, varying between 55% and 65%.

L10

Interface gráfica do usuário, Gráfico

Descrição gerada automaticamente

In building L10, we were able to identify a significantly greater number of times with high correlation with temperature, with most of these correlations exceeding 0.8. On average, the value for plugs and lighting was 0.50, with a standard deviation of 0.09, a minimum value of 0.37, and a maximum of 0.63. Although we have a higher correlation, we also observe a higher standard deviation and a greater variation between the minimum and maximum values. We could say that, on average, the plugs and lighting in this building are in the range of 40% to 60%. However, the value that proved to be really effective was the average of 0.50.

#### Model 2 - Searching for Peaks and Valleys to Try to Find the Value of Plugs and Lighting

Another approach to finding a median value for reference weeks, when working with an HVAC system, is to look for events from peaks and valleys. In my opinion, this is an interesting technique for looking for moments when it is possible that some equipment or system is turned on or off. For example, it is useful for identifying quick moments when the HVAC system is turned off or when some important plug or lighting is deactivated. However, it is difficult to know exactly what type of equipment is being turned on or off at a frequency of 5 minutes or 15 minutes, which requires careful work with this type of model to ensure more consistent results.

With this in mind, in both building L9 and building L10, we looked for patterns that could indicate moments of turning on or off some important system. Preferably, I searched for heating or cooling systems, but not exclusively, as it is not possible to say for sure what is being turned on or off, especially working with such low frequencies. Each of the buildings presents distinct situations; for example, when there is vacancy in building L9, there are valleys in the 5-minute chart that may indicate the shutdown of an important system.

The calculations are detailed in code on GitHub. Below, I will show some images and basically demonstrate how the calculations are done.

L09

Gráfico

Descrição gerada automaticamente

Gráfico

Descrição gerada automaticamente

As mentioned earlier, here we can observe the valleys which are actually represented as peaks. This is because the scipy.signal.find\_peaks function only detects peaks, so it was necessary to invert the valleys to enable identification. This is an interesting pattern that apparently seems to indicate a quick shutdown of an important system. I sought to find all these valleys and, from them, generate an average amplitude very similar to Model 1, but now using these moments of rapid shutdown. At the end of the calculation, we arrived at a median for the plugs of 0.600.

The formula used for calculating the plugs is as follows:

properties['plugs'] = properties['prominences'] / ((properties['peak\_heights'] - properties['prominences']) \* -1)

L10

Gráfico, Histograma

Descrição gerada automaticamente

Gráfico

Descrição gerada automaticamente

In building L10, we observed a distinct pattern of peaks that suggests moments of rapid activation and deactivation of an important system. For example, on days of vacancy, such as weekends or nighttime hours, we noticed a significant 15-minute peak, followed by a rapid return of energy values to normal. This peak suggests a moment when some system is activated and, for some reason, quickly deactivated, although it is difficult to speculate the causes without additional information.

In this scenario, it is possible to calculate the amplitude of these peaks to better understand the variation in energy. The median of the plugs, calculated for these events, was 0.502.   
The formula used to calculate the plugs was:   
properties['plugs'] = properties['prominences'] / (properties['peak\_heights'] - properties['prominences'])