**Cell 1: Loading and Indexing the Raw Data**

**Code Action:** Reads a very large CSV file (time\_series\_15min\_singleindex.csv) containing energy data and converts the utc\_timestamp column into the main index of the table.

**Layman's Terms:** "You took a massive spreadsheet of 15-minute energy data. This spreadsheet has a column that tells you *when* each reading was taken. You told the program to make this 'time' column the master list, organizing all 201,604 rows by the exact moment they occurred. This turns the data into a **time series**—a sequence where time matters."

| Detail | Status |
| --- | --- |
| Data Points | 201,604 entries |
| Index Type | DatetimeIndex (Time-based) |

**Cell 2: Selecting Data and Initial Cleaning**

**Code Action:** Filters the large table to keep only the columns relevant to **Germany (DE)** and performs **time-based interpolation** to fill most missing values.

**Layman's Terms:** "You decided to focus only on **Germany's energy data**. You selected the most important items: the actual energy consumed (the target), the day-ahead forecast, and the solar/wind power generated. Because real-world data is messy, some timestamps were missing readings (**NaNs**). **Interpolation** acts like drawing a line between two known points in time, guessing the missing value based on what came right before and right after it. This makes the time series continuous."

| Detail | Action |
| --- | --- |
| Columns Selected | 4 (Load Actual, Load Forecast, Solar, Wind) |
| Cleaning Method | Time-based Interpolation |

**Cell 3: Final Missing Value Check**

**Code Action:** Uses **forward fill (ffill)** and **backward fill (bfill)** to catch any remaining missing values at the very start or end of the dataset, resulting in zero missing values.

**Layman's Terms:** "After the initial 'line-drawing' (interpolation), you did a final scrub. The ffill assumes if data is missing, it should just copy the *last valid reading*. The bfill does the opposite—it copies the *next valid reading*. By doing both, you guaranteed the entire dataset is **perfectly clean** and ready for modeling, a critical part of your preprocessing objective."

| Detail | Outcome |
| --- | --- |
| Missing Values | **0** (Perfectly Clean) |

**Cell 4: Visualizing the Time Series**

**Code Action:** Plots the last 1,000 data points of the DE\_load\_actual... series to visually confirm the data's integrity and structure.

**Layman's Terms:** "You zoomed in on the last few weeks of data to **visually check** the load pattern. This plot shows the typical **seasonal cycles** in electricity demand: a low demand period during the night, a high peak in the morning/day, and a repeatable pattern across days and weeks. This visual confirmation proves the data is correctly indexed and ready for decomposition."

**Cell 5: Workflow Optimization with Parquet**

**Code Action:** Saves the clean dataset to a **Parquet file** and reloads it.

**Layman's Terms:** "To solve the problem of your computer slowing down when reloading the large spreadsheet, you converted the cleaned data into the **Parquet format**. Parquet is an incredibly fast, highly compressed way to store big data tables. Now, instead of waiting minutes for the cleaned data to load, it **loads almost instantly**, making your entire research workflow much more efficient."

| Detail | Optimization |
| --- | --- |
| Format | Parquet (High-efficiency columnar storage) |
| Result | Instant data reloading |

**Cell 6: Checking Time Series Integrity**

**Code Action:** Checks the **timezone**, **inferred frequency**, and **gaps** in the time series index.

**Layman's Terms:** "Before a deep time-series analysis like STL, you ran a health check. This confirms that the data: 1) knows its **timezone (UTC)**, 2) is **regularly spaced at 15-minute intervals**, and 3) has **no sudden, large gaps** that would break the modeling process. Everything is confirmed to be ready for decomposition."

**Cell 7: Seasonal-Trend Decomposition (STL) Setup**

**Code Action:** Initializes and runs the **STL decomposition** on the actual load data. It sets the period to **672** (1 week × 4 intervals/hour × 24 hours/day) and seasonal=671 to ensure the core weekly pattern is captured.

**Layman's Terms:** "You started the main, time-consuming statistical process.

**STL** breaks down the messy load data into three perfectly neat signals:

1. **Trend:** The slow, long-term movement (like high demand in winter, low in summer).
2. **Seasonal:** The precise, repeating cycle (the predictable 15-minute pattern that repeats every day and week).
3. **Residual (Noise):** The remaining, unpredictable 'junk' or non-linear fluctuations (like sudden demand spikes).

Your entire

**Hybrid Model** strategy depends on this step: the Machine Learning models (LightGBM, etc.) will be trained *only* to predict this **Residual** component, as the Trend and Seasonal parts are now handled by the simple, statistical decomposition."

| Detail | Purpose in Research |
| --- | --- |
| Output Object | res (containing Trend, Season, Residual)  New ML Target |
| Runtime | ≈12 minutes (Complex calculation) |

**STL Decomposition Explained**

The STL process separated your German electricity load data (at 15-minute intervals from 2015 to 2020) into the following four panels:

**1. Original Series (Yt​)**

* **Layman's View:** This is the **raw energy consumption** over five years. It's the messy data your power grid operators see every day.
* **Technical View:** This is the observed time series, Yt​. It is the sum of all components: Yt​=Trendt​+Seasonalt​+Residualt​. It shows massive fluctuations because it contains all three signals mixed together.

**2. Trend Component (Tt​)**

* **Layman's View:** Think of this as the **slow, underlying climate** of energy demand. It's not the daily weather, but the long-term changes that happen across the years.
* **Technical View:** This represents the **low-frequency movements** in the data.
  + **Observation:** The plot clearly confirms **strong annual cycles**. The peaks occur consistently during **winter** (high heating/lighting demand) and the troughs occur during **summer** (lower overall demand, as AC use is offset by longer daylight hours).
  + **Purpose:** In your hybrid model, you assume the future trend will follow this same smooth path. You simply **add this trend back** to your final forecast.

**3. Seasonal Component (St​)**

* **Layman's View:** This is the **fixed daily and weekly schedule** of the country. It captures the predictable pattern of demand: always low at 3 AM, always high at 6 PM, and always different on a Saturday versus a Tuesday.
* **Technical View:** This represents the **high-frequency, fixed-period cycles**.
  + **Period (672):** You correctly set the period to 672, which is one week worth of 15-minute intervals (4×24×7=672). This guarantees you've isolated the weekly cycle.
  + **Observation:** The plot shows a **dense, highly predictable, and perfectly repetitive** signal. The magnitude (height) of these spikes and dips represents how much energy demand is consistently higher or lower than the overall trend during that specific 15-minute slot of the week.
  + **Purpose:** Like the Trend, the Season is predictable. You **add this seasonal component back** to your final forecast.

**4. Residual Component (Rt​)**

* **Layman's View:** This is the **unpredictable "noise,"** the component that doesn't fit the smooth trend or the fixed schedule. It's the influence of things like a sudden cold snap, a major breaking news event, or an unpredictable factory shutdown. **This is the hardest part of the data to model.**
* **Technical View:** This represents the **non-linear, irregular component** that your traditional statistical models (like ARIMA/ETS) struggle with.
  + **Observation:** The plot shows high volatility, characterized by large, scattered points outside the central band. This scatter represents the non-linear shocks to the system.
  + **Purpose:** **This series (Rt​) becomes your new target variable for your Machine Learning models** (LightGBM, XGBoost, etc.). By focusing your powerful ML models *only* on the residual, you are leveraging their strength to find complex patterns within the noise, which is the core innovation of your hybrid approach.

**STL and Your Hybrid Model**

The entire strategy for your thesis, as outlined in Section 7.3.3 of your proposal, is based on this decomposition:

1. **Decomposition (STL):** Handles the easy parts (Tt​ and St​) perfectly.
2. **ML Training:** Trains **LightGBM/XGBoost** to forecast only the unpredictable part (R^t​).
3. **Final Forecast:** Combines the statistically simple parts with the ML-predicted noise:

Y^t​=Tt​+St​+R^t​

This process helps **balance predictive power with interpretability**.

**Feature Engineering: Finalizing the Dataset**

**Part 1: Execution Summary (The Fast Path)**

**Output Snippet:**

--- STL Decomposition Started (Will take ~12 minutes) ---

--- STL Decomposition Completed and Saved to: F:\LJMU THESIS\Topic 2 Papers and Supporting Documents\File\_Implementation\stl\_results.pkl ---

**Layman's & Technical Explanation:** "This confirmed the system had to execute the ≈12-minute STL calculation one last time because the cache file didn't exist yet (the **'Slow Path'**). Crucially, the final line confirms the res object (containing the Trend, Seasonal, and Residual series) was **saved** to your hard drive as stl\_results.pkl. This means that tomorrow, when you run this code again, it will detect that file and load the results **instantly**, making your future workflow much faster. The caching problem is solved."

**Part 2: Final Dataset Integrity**

**Output Snippet:**

Total Features (Predictors): 15

Total Observations (Rows): 200932

Data starts at: 2015-01-07 23:00:00+00:00

**Layman's & Technical Explanation:** "This is your dataset's final scorecard. Every number confirms your **Feature Engineering** was successful:

* **Observations (Rows):** You started with 201,604 rows. Dropping the first 672 rows (due to the 1-week lag feature) leaves exactly **200,932** clean, usable rows.
* **Data Start Date:** The start date is correctly pushed forward by 672 intervals (one week) because the model can only make a prediction when it has the 1-week-old data it needs.
* **Total Features:** You have **15 features** (your predictor variables, X), plus the Y (the target) in the final table—which is exactly what we designed."

**Part 3: The Data Table (Head/Tail)**

This section shows the key engineered features now living alongside your original data columns.

| **Feature Group** | **Layman's Terms** | **Technical Role** |
| --- | --- | --- |
| **STL Components** | **trend, seasonal, residual** | These are the three signals from your STL analysis. The residual is the **new target variable** (Y) for LightGBM/XGBoost. |
| **Time-Series Features** | **hour, day\_of\_week, time\_of\_day\_15min** | These are the calendar-based predictors you created. They tell the ML model *when* in the predictable daily/weekly schedule the residual occurred, helping to predict time-sensitive noise. |
| **Lag Features (Memory)** | **residual\_lag\_96**, **load\_actual\_lag\_672** | These are the auto-regressive features. They give the model "memory" of both the **noise (residual)** and the **total load** from 24 hours and 1 week ago. This memory is crucial for predicting the next time step. |
| **Exogenous Variables** | **DE\_load\_forecast..., DE\_solar..., DE\_wind...** | These are the external variables (like market prediction and renewable energy generation) that the model uses to explain the residual. |

**Part 4: Final Verification (df.info())**

**Output Snippet:**

Data columns (total 16 columns):

# Column... Non-Null Count Dtype

--- ------ -------------- -----

...

15 residual 200932 non-null float64

dtypes: float64(9), int32(7)

**Layman's & Technical Explanation:** "This is the final, most critical check. The Non-Null Count for every single column is **200932**. This confirms that **every row, in every column, is complete**. You have zero missing values remaining in your final dataset, which is a requirement for nearly all Machine Learning and Deep Learning algorithms."