From Raw HEAT VENTILATION AIR CONDITIONING Signals to Predictive Insights

Melahat Tayli

5 August 2025

Exploratory Data Analysis

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 408000 entries, 0 to 407999
Data columns (total 45 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   timestamp              408000 non-null  object
 1   indoor_temp            405616 non-null  float64
 2   supply_temp            405615 non-null  float64
 3   hvac_control           407974 non-null  float64
 4   airflow                407999 non-null  float64
 5   power_usage            407999 non-null  float64
 6   outdoor_temp           408000 non-null  float64
 7   solar_radiation        408000 non-null  float64
 8   occupancy              408000 non-null  float64
 9   price                  408000 non-null  float64
 10  temp_error             405616 non-null  float64
 11  cooling_demand         408000 non-null  float64
 12  heating_demand         405616 non-null  float64
 13  …….
memory usage: 140.1+ MB
```

❯ *As I began exploring the dataset, I quickly noticed its scale: it contains **408,000 data points** spread across **45 features**. This richness offers a great opportunity to uncover meaningful patterns but also calls for careful preprocessing to manage the complexity.*

Exploratory Data Analysis

Important Steps|

- Fix The Data Types (object->timestamp)
- Removing Non-Informative Features
- Analyze Correlations
- Feature Selection
- Remove Missing Values

Analysis indicate that the HVAC data is recorded at **15-minute intervals**, spanning from **2020-01-01 00:00:00** to **2031-08-20 23:45:00**. This suggests a consistent and regular sampling frequency throughout the dataset

Removing the following zero-variance columns:
['hvac_control', 'cooling_demand', 'hvac_contr
'cooling_demand_sma'.....]

8 columns have constant values

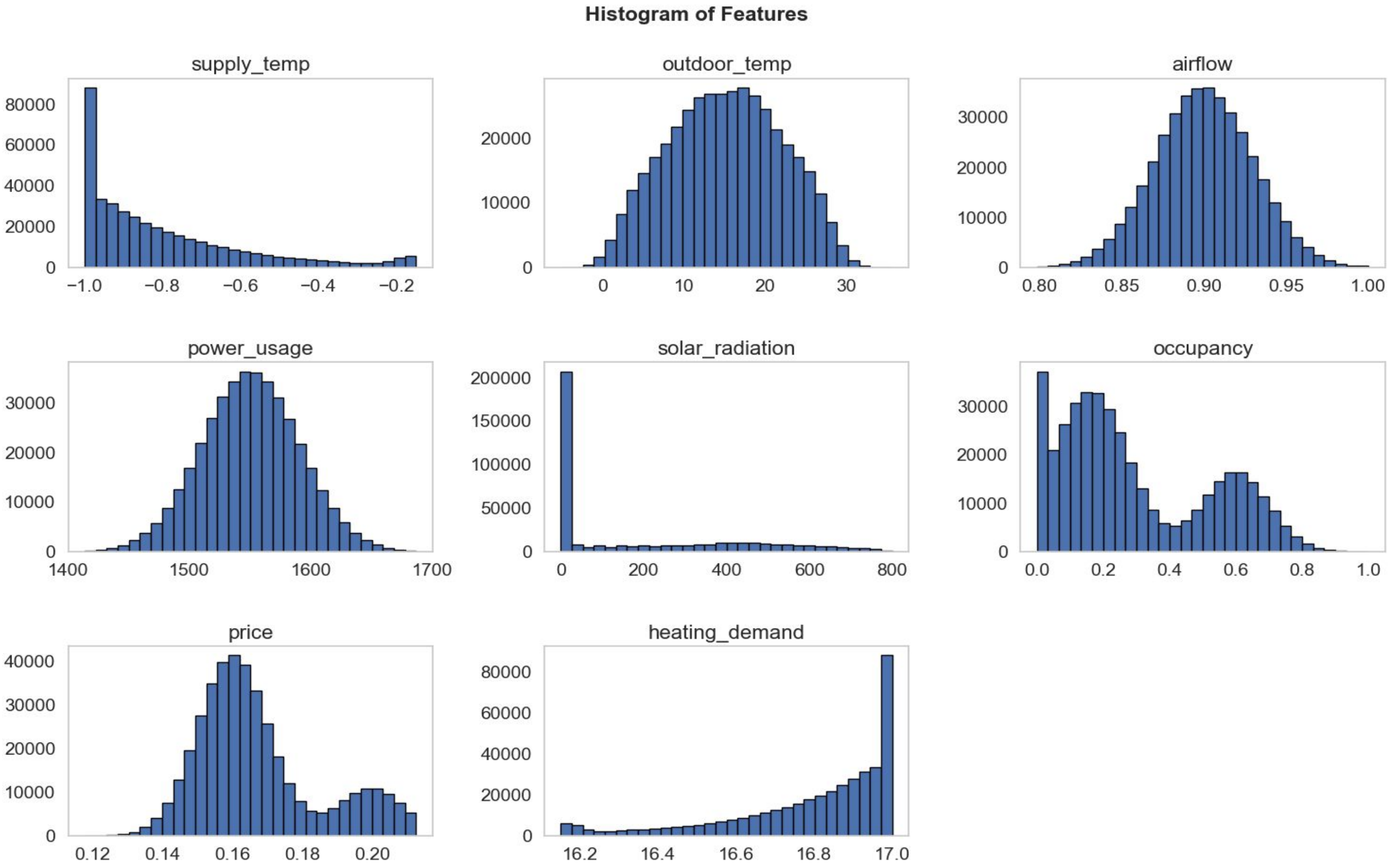
Perfectly correlated variable pairs:
temp_error <--> indoor_temp
heating_demand <--> indoor_tçemp
heating_demand <--> temp_error...........

More than 30 columns correlated

Most variables are **multiple transformed versions of the same original features**, including **Savitzky-Golay, Simple Moving Average, Exponential Moving Average), Robust filtering, and Min-Max scaling.**

Consecutive missing (NaN) values are dropped!

Exploratory Data Analysis



Exploratory Data Analysis

Exploring the Target Variable: Power Usage

- **Trend**
  Is there a long-term increase or decrease in power consumption over time?
- **Seasonality**
  Are there recurring patterns (e.g., daily, weekly, yearly cycles)?
- **Autocorrelation**
  Does current power usage depend on previous values?

Exploring the Predictor variable: Seasonality

- Gives us hints about how to treat them: categorical, numerical....

Modeling Strategy Based on Data Characteristics

🔹 **No Autocorrelation in Power Usage**
→ Use standard models:
· Linear Regression, Gradient Descent, Tree-Based Models...

🔹 **Autocorrelation Detected**
→ **Time Series Models**

Seasonality in Power Usage?

➤ Use models designed for seasonal patterns:
→ SARIMA, Facebook Prophet.......

Non-Stationarity in the Series?

➤ Address trends or variance shifts using:
→ Differencing , Log / Power Transformations..

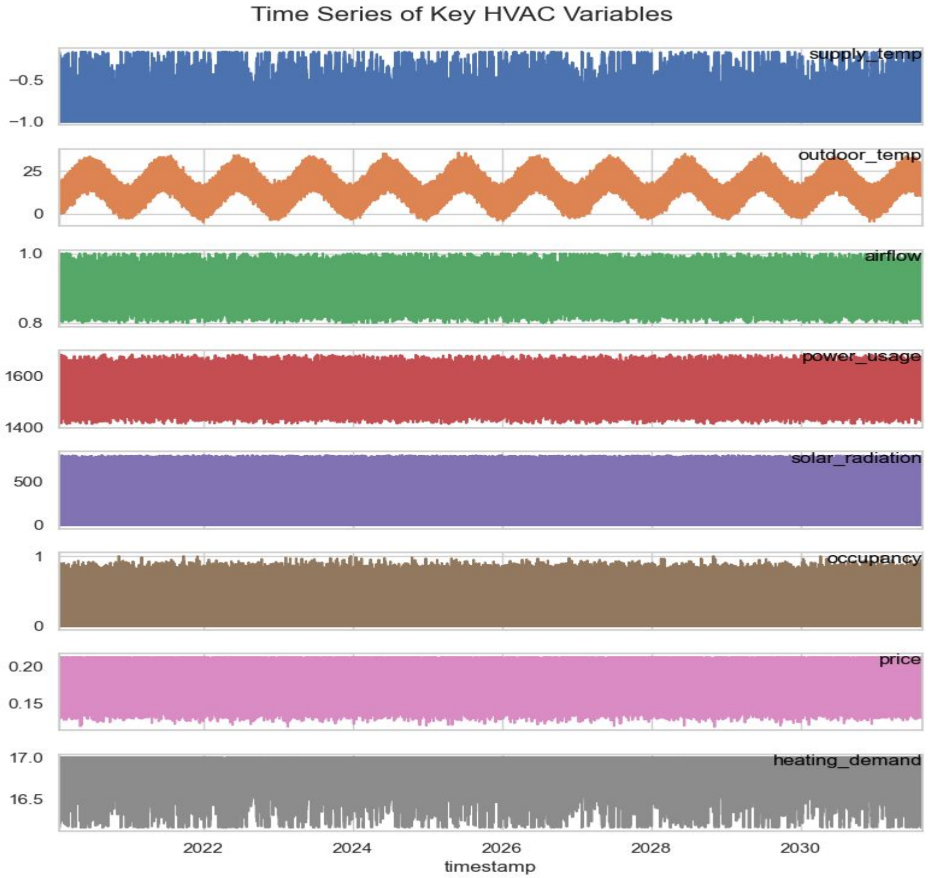
Exploratory Data Analysis

🔷 Seasonality in Outdoor Temperature

- Outdoor temperature exhibits **clear annual seasonality** :
  - It **rises** during the first half of the year
  - And **falls** during the second half

- This pattern is **less apparent** in other variables (e.g., heating_demand)

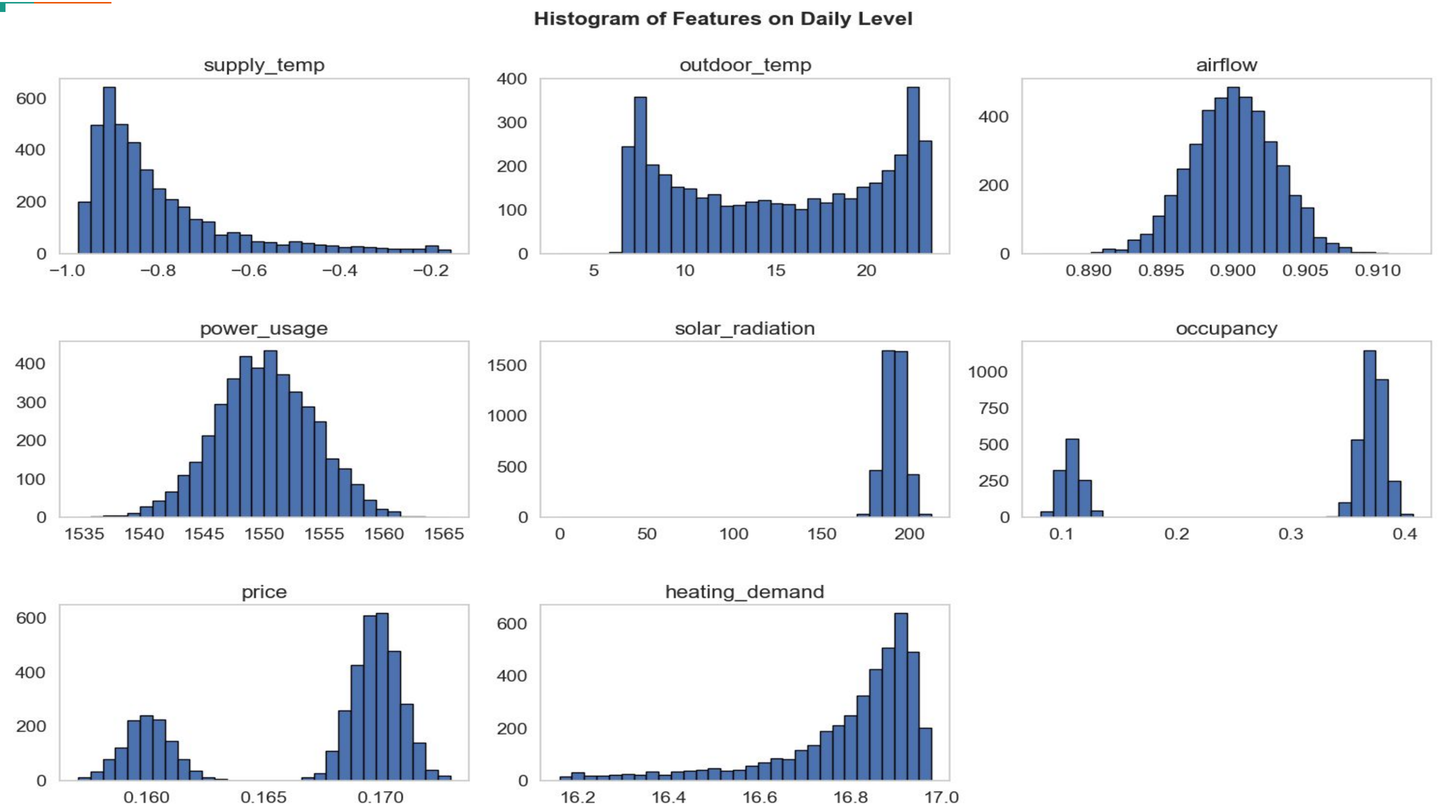
- To reveal potential seasonal effects in those variables,
  consider analyzing data at a **coarser time granularity** :
  - **Daily** or **hourly averages**



Exploratory Data Analysis

Seasonal Decomposition: Shifting to Daily Resolution for Analysis

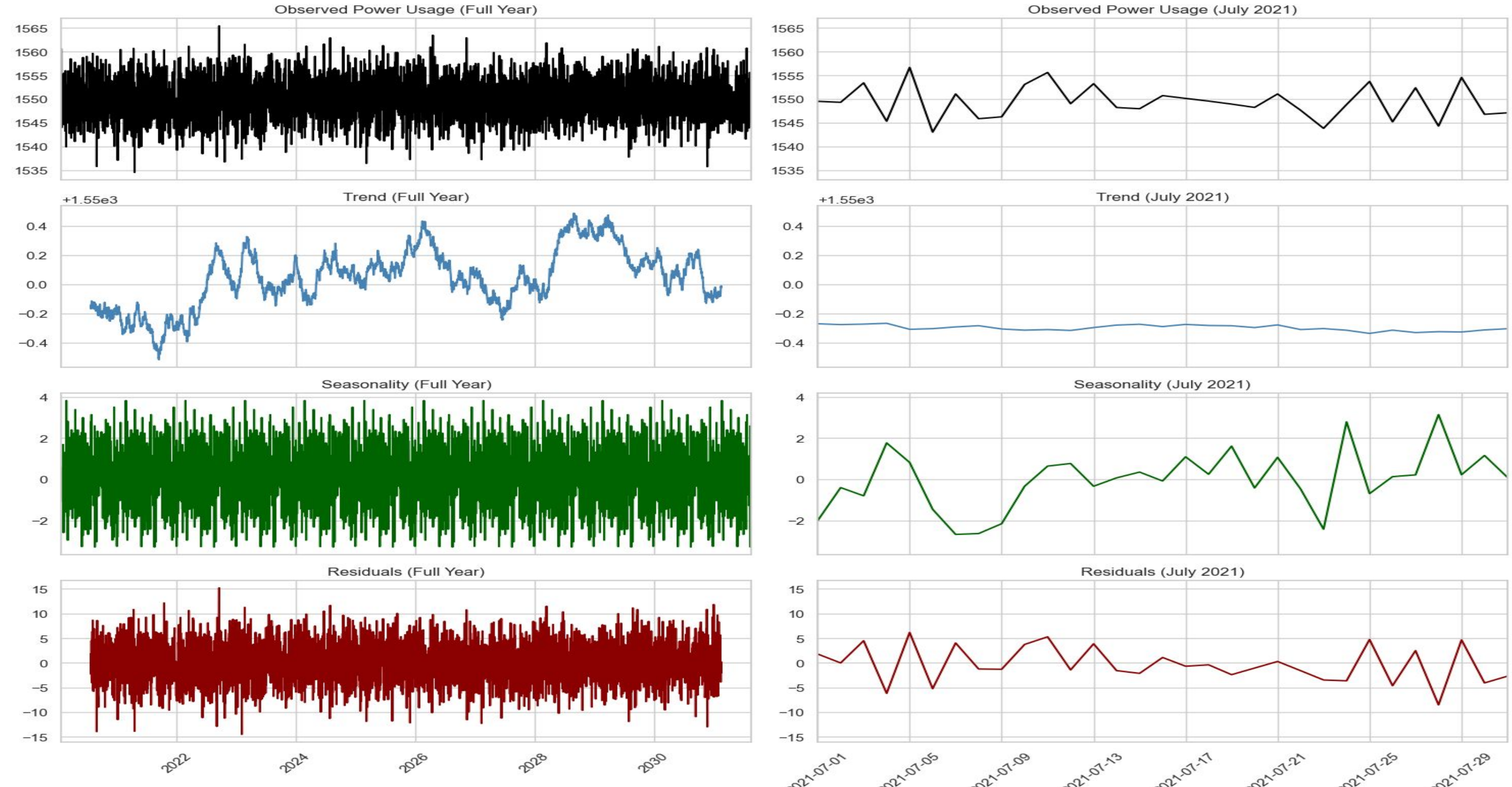
```
daily_data = hvac_raw_data.resample('D').mean()
```



Exploratory Data Analysis

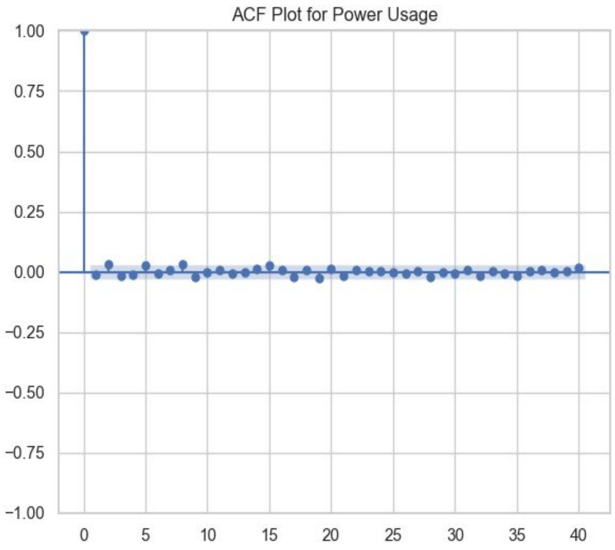
Seasonal Decomposition: Shifting to Daily Resolution for Analysis

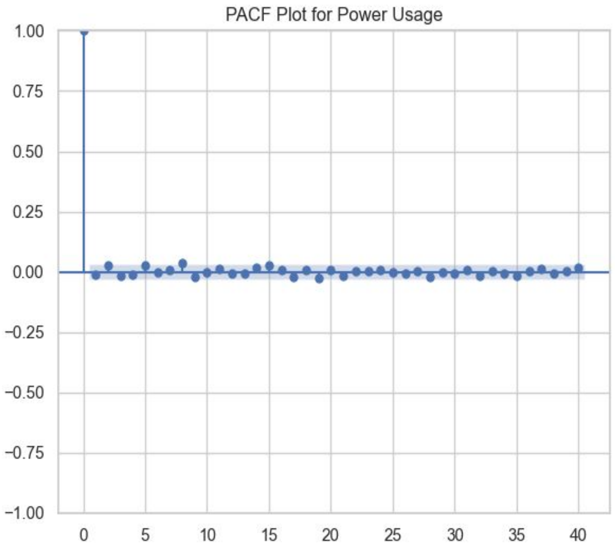
```
daily_data = hvac_raw_data.resample('D').mean()
```



Exploratory Data Analysis

Diagnostic Tests for Autocorrelation





PACF, ACF, and the Ljung-Box test indicate **no significant autocorrelation or partial autocorrelation** in the data.

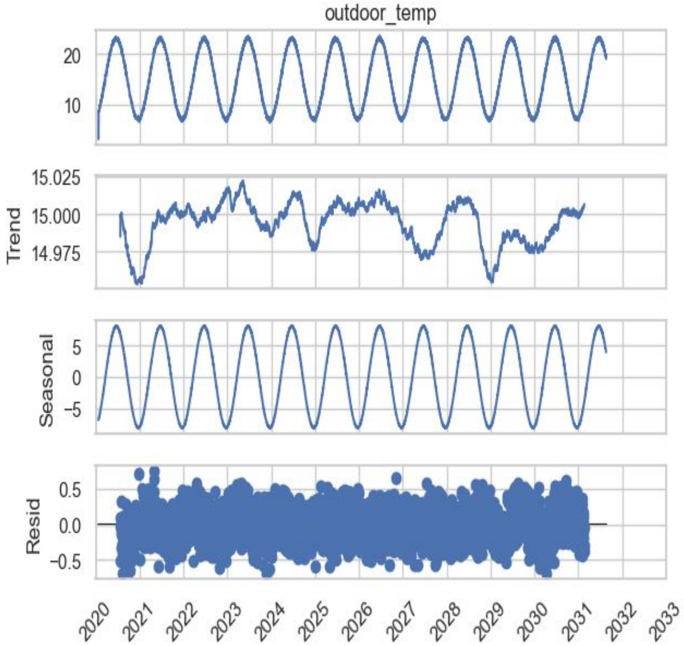
The Ljung-Box test

	lb_stat	lb_pvalue
40	34.87208	0.699994

The Ljung-Box test returned a **p-value of 0.70**, which is well above the 0.05 threshold, suggesting that the null hypothesis of no autocorrelation cannot be rejected.

Exploratory Data Analysis

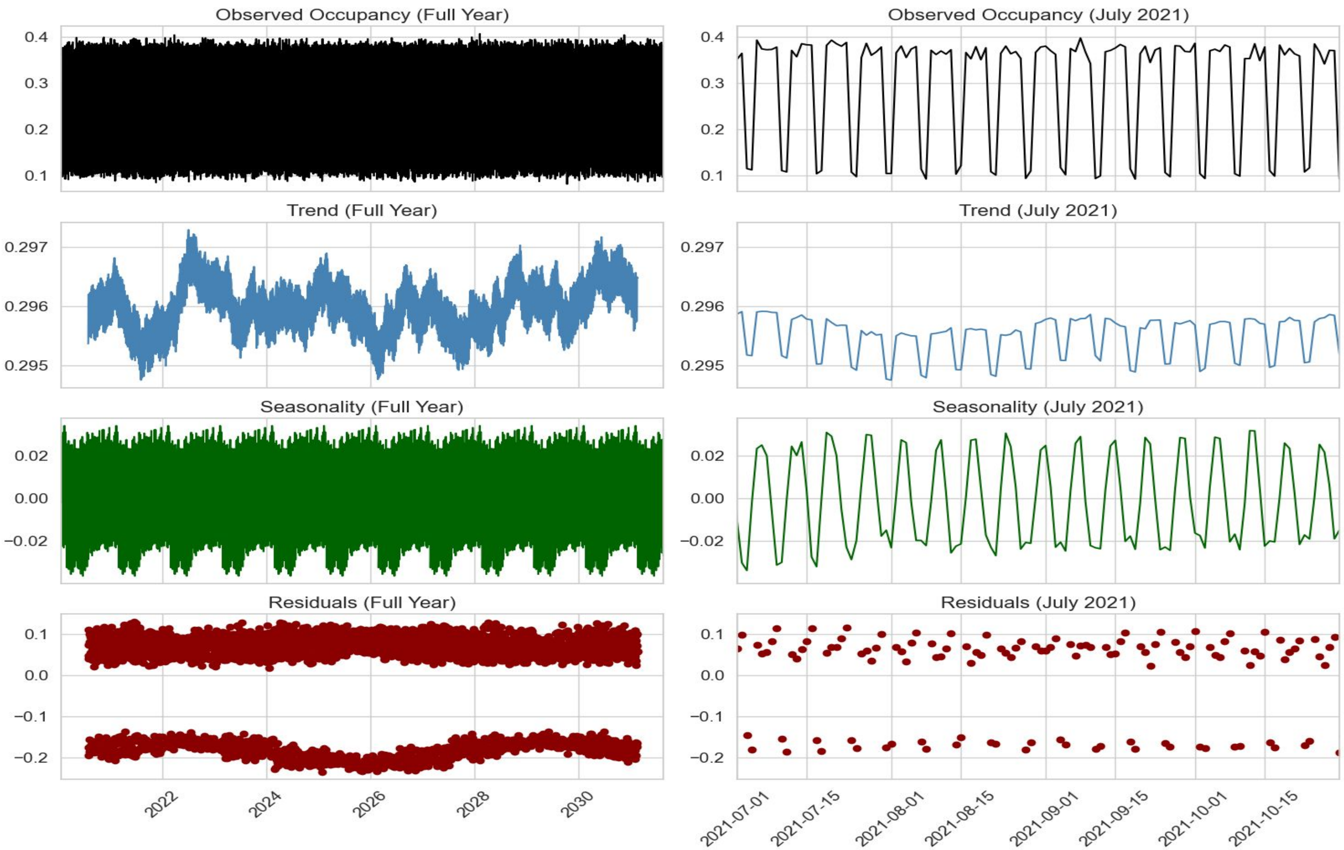
Seasonality in predictors



Outdoor temperature exhibits a clear **annual seasonal pattern**, characterized by **high values in summer and low values in winter**.

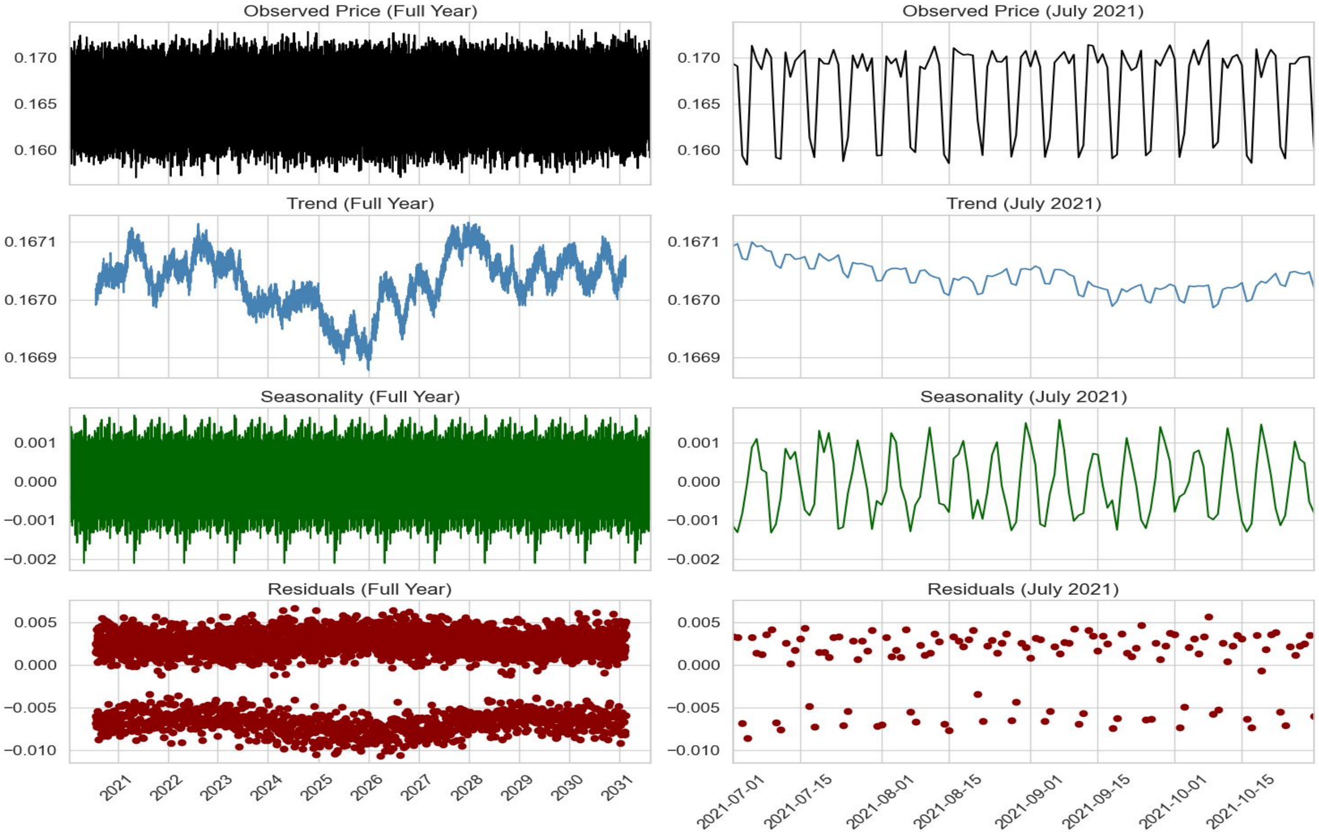
Exploratory Data Analysis

Seasonality in predictors



Exploratory Data Analysis

Seasonality in predictors



Exploratory Data Analysis

Seasonality in predictors

Stable Long-Term Trends Across Variables

- **Supply Temperature:** Very stable, ranging from -0.825 to -0.775
- **Airflow:** Minimal trend variation (0.9000 to 0.9005)
- **Solar Radiation:** Steady trend between 190.5 and 191.5
- **Heating Demand:** Consistently between 16 and 17; trend remains around 16.8

Exploratory Data Analysis

Conclusion: Daily Power Usage Analysis

- No clear trend or strong seasonality in daily power usage; minor fluctuations (~±2.5 units) likely reflect HVAC response to temperature extremes
- ACF, PACF, and Ljung-Box tests show no significant autocorrelation — daily values largely independent
- Predictors like supply temperature, solar_radiation and airflow are stable; only outdoor temperature shows annual seasonality. Price and occupancy shows slight weekly seasonality.
- Next step: analyze **hourly data** to detect intra-day patterns and short-term dependencies

Feature Engineering

```
# Seasonal encoding for outdoor temperature (annual cycle)

daily_data['day_of_year'] = daily_data.index.dayofyear

daily_data['sin_day'] = np.sin(2 * np.pi * daily_data['day_of_year'] / 365)

daily_data['cos_day'] = np.cos(2 * np.pi * daily_data['day_of_year'] / 365)
```

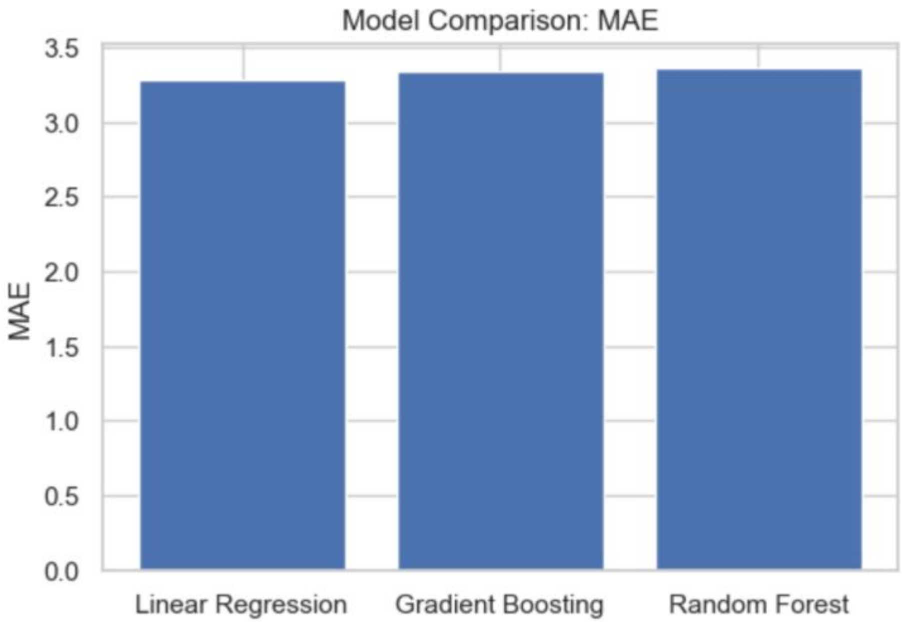
```
# Rolling mean of price and occupancy over last 7 days

daily_data['price_roll_7'] = daily_data['price'].rolling(window=7).mean()

daily_data['occupancy_roll_7'] = daily_data['occupancy'].rolling(window=7).mean()
```

Modelling

Linear Regression, Gradient Boosting Regressor, Random Forest



Models could detect the daily power consumption with MAE of ~3.4 !

What is next?

What insights emerge when the dataset is analyzed at multiple granularities?

……

Questions & Discussion

Any Questions?

Thank you for Listening!

Melahat

Tayli