Assignment 1

Code Explanation

1. **is_nrec_holiday function**

- ✧ This function determines whether a given date is a NERC holiday.
- ✧ It uses a set of known NERC holiday dates (e.g., New Year's Day, Memorial Day, etc.).

2. **get_hours function**

- ✧ This function takes three parameters: iso (name of the Independent System Operator), peak_type (type of peak), and period (time period).
- ✧ It defines the peak hours for different peak types including onpeak, offpeak, flat, 2x16H, and 7x8.

3. **Period Parsing**

- ✧ The function parses the period to determine the start and end dates.
- ✧ It supports annual, quarterly, monthly, and daily time periods.
- ✧ For annual periods, the function constructs the start and end dates based on the year.
- ✧ For quarterly periods, the function sets the dates based on the specified quarter.
- ✧ For monthly periods, the function converts the month text to a month number and calculates the start and end dates.
- ✧ For daily periods, the function simply sets the start and end dates to the specified day.

4. **Hour Calculation**

- ✧ The function initializes num_hours to 0.
- ✧ It iterates over each day in the specified period, checking the peak type and the current date.
- ✧ For onpeak type, it checks if the current date is a weekday and not a NERC holiday, and adds the onpeak hours (HE7 to HE22).
- ✧ For offpeak type, it adds 24 hours for weekends and NERC holidays, and adds offpeak hours for weekdays.
- ✧ For flat type, it adds 24 hours for every day.
- ✧ For 2x16H type, it adds the hours for weekends and NERC holidays.
- ✧ For 7x8 type, it adds the hours for weekdays and non-NERC holidays.
- ✧ The function increments the current_date by one day in each iteration.

5. **Return Result**

- ✧ The function returns a dictionary containing iso, peak_type, startdate, enddate, and num_hours.

6. **Verification**

- ✧ Data was downloaded from https://www.energygps.com/HomeTools/PowerCalendar to test, data filter was set as below

- ✧ The sample run calls the function get_hours("ERCOT", "onpeak", "2019May") and prints the result of 352.

Assignment2

Code Explanation

1. Load the Data:

- Convert the time column to datetime.
- Convert power consumption from watts to kilowatts (kW_min).
- Resample the data to hourly intervals, summing the values.
- Adjust the date format to include the year and handle edge cases (e.g., converting '24:00:00' to '00:00:00' and incrementing the day).
- Set the time column as the index.

2. Data Merging and Analysis

- The two datasets were merged on the time column, keeping only the overlapping periods.
- A new column, total_kW, was added to the merged dataset, representing the total hourly electricity consumption by summing all the numeric columns.

Finding

Average Weekly Electricity Consumption

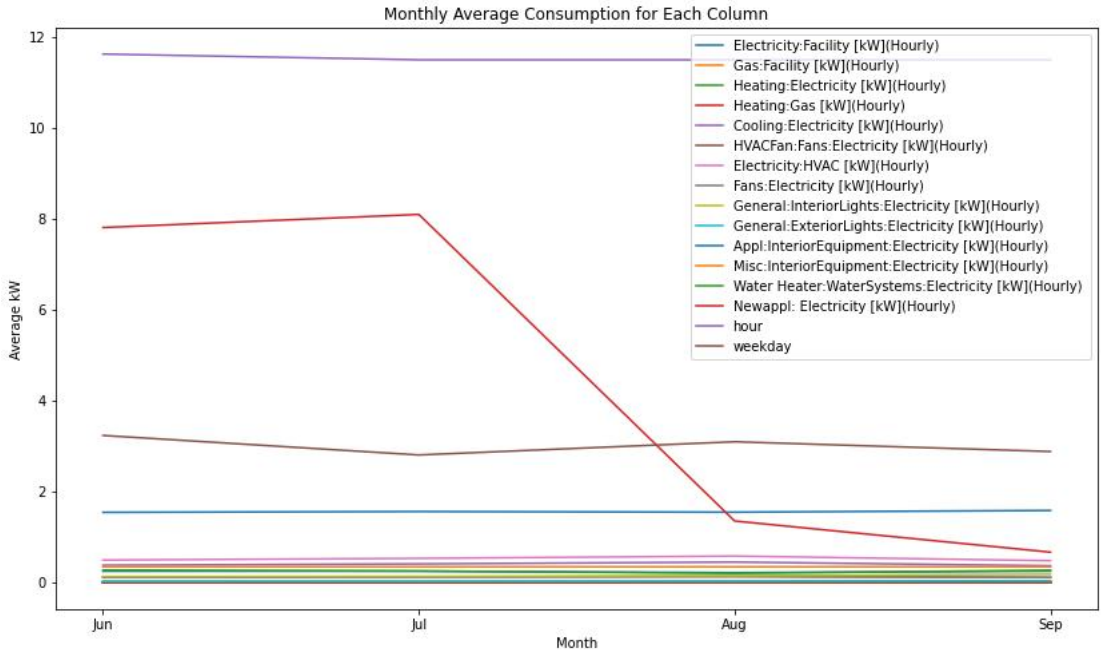
- The consumption is relatively stable throughout the week.
- There are no significant spikes or drops in electricity usage on any particular day.
- This suggests consistent electricity usage patterns across different weekdays.

Average Monthly Electricity Consumption:

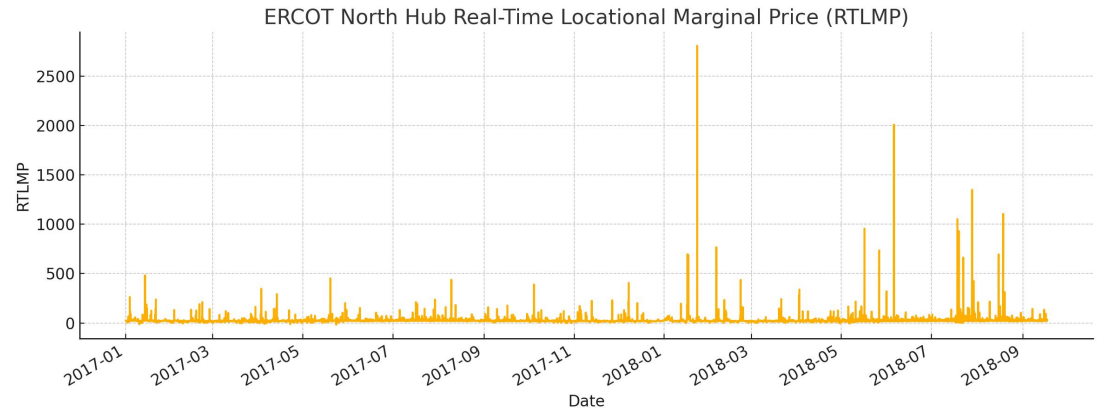
- There is a noticeable peak in electricity consumption in the month of July.
- June also shows higher consumption compared to August and September.
- This pattern suggests increased electricity usage during the summer months, likely due to cooling needs.

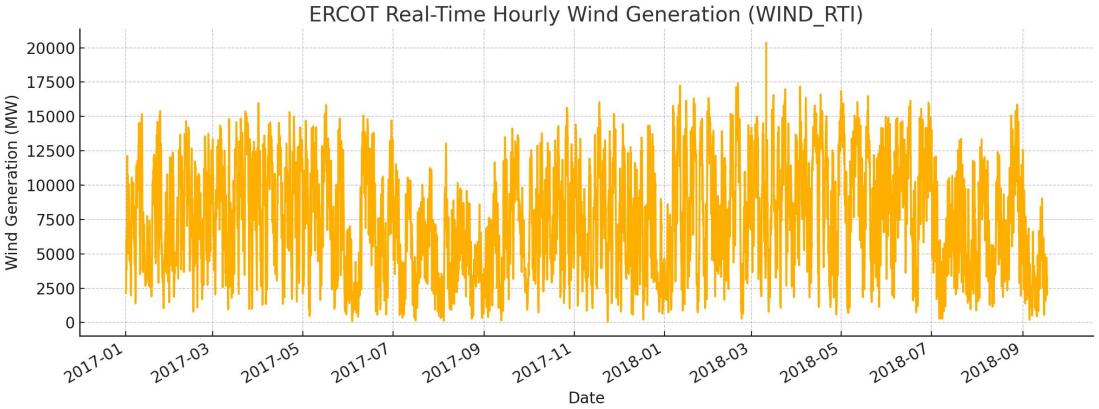
Abnormality

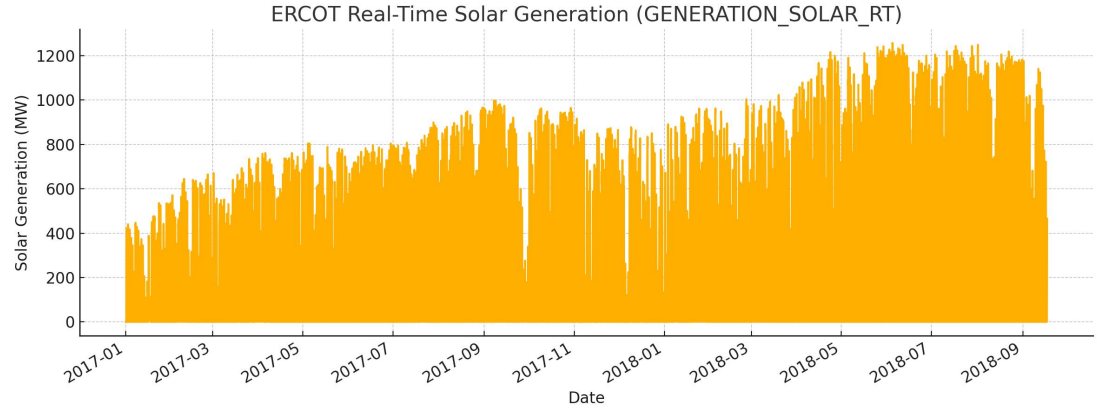
There is an obvious drop from July to August, which is caused by the significant drop from new appliance.

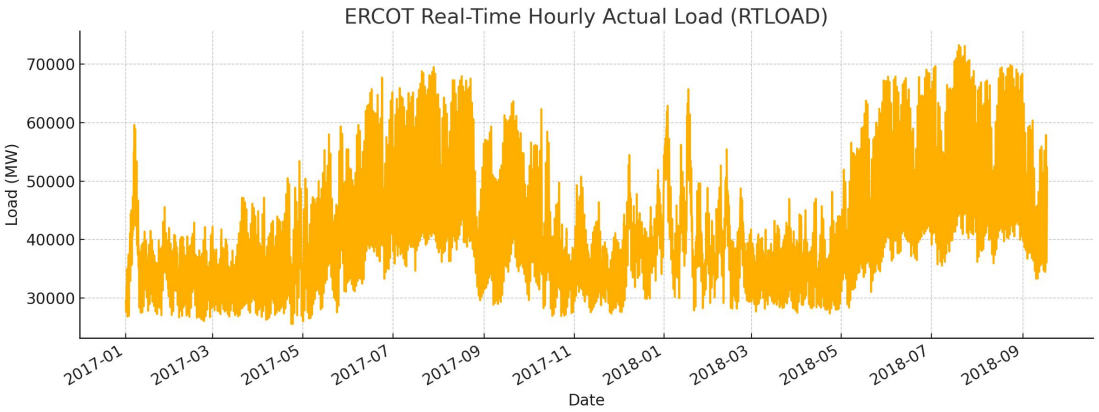


Assignment3









EDA

North Hub Real-Time Locational Marginal Price (RTLMP):

The RTLMP shows considerable spikes, indicating volatility with significant peaks occurring sporadically.

Real-Time Hourly Wind Generation (WIND_RTI):
Wind generation varies significantly, with visible seasonal patterns and high variability within short periods.

Real-Time Solar Generation (GENERATION_SOLAR_RT):
Solar generation shows a clear seasonal pattern, increasing during certain times of the year and dropping during others.

Real-Time Hourly Actual Load (RTLOAD):
The load follows a strong seasonal pattern, with peaks during certain months indicating higher demand periods.

Forecast model

Process

lagged features are generated to capture the time-dependent nature of the variables. Specifically, it creates lagged versions of the following columns: HB_NORTH (RTLMP), ERCOT (WIND_RTI), ERCOT (GENERATION_SOLAR_RT), and ERCOT (RTLOAD). The lagged features represent the values of these variables at previous time steps, which can help the model to understand temporal dependencies. The process is repeated for lags from 1 to 24 hours, creating new columns for each lag. For example, RTLMP_LAG_1 represents the value of HB_NORTH (RTLMP) one hour ago, RTLMP_LAG_2 represents the value two hours ago, and so on up to 24 hours.

New columns extracted from DATETIME are created to capture temporal information. The mutate function is used to add these new features: These additional features help the model understand the temporal patterns and dependencies in the data, such as daily, weekly, and monthly cycles.

Result

Feature: According to correlation analysis and grep function, regarding time factors, “hour ending” and ”hour” are specified as predictors in a machine learning model.

Performance:

Train MAE (Mean Absolute Error): 10.28128
Train RMSE (Root Mean Squared Error): 48.23403
Test MAE (Mean Absolute Error): 9.964333
Test RMSE (Root Mean Squared Error): 35.26201
These metrics indicate the model's error rates on the training and testing datasets. Lower values of MAE and RMSE suggest better model performance. The slight difference between training and testing errors indicates that the model generalizes well to new data.