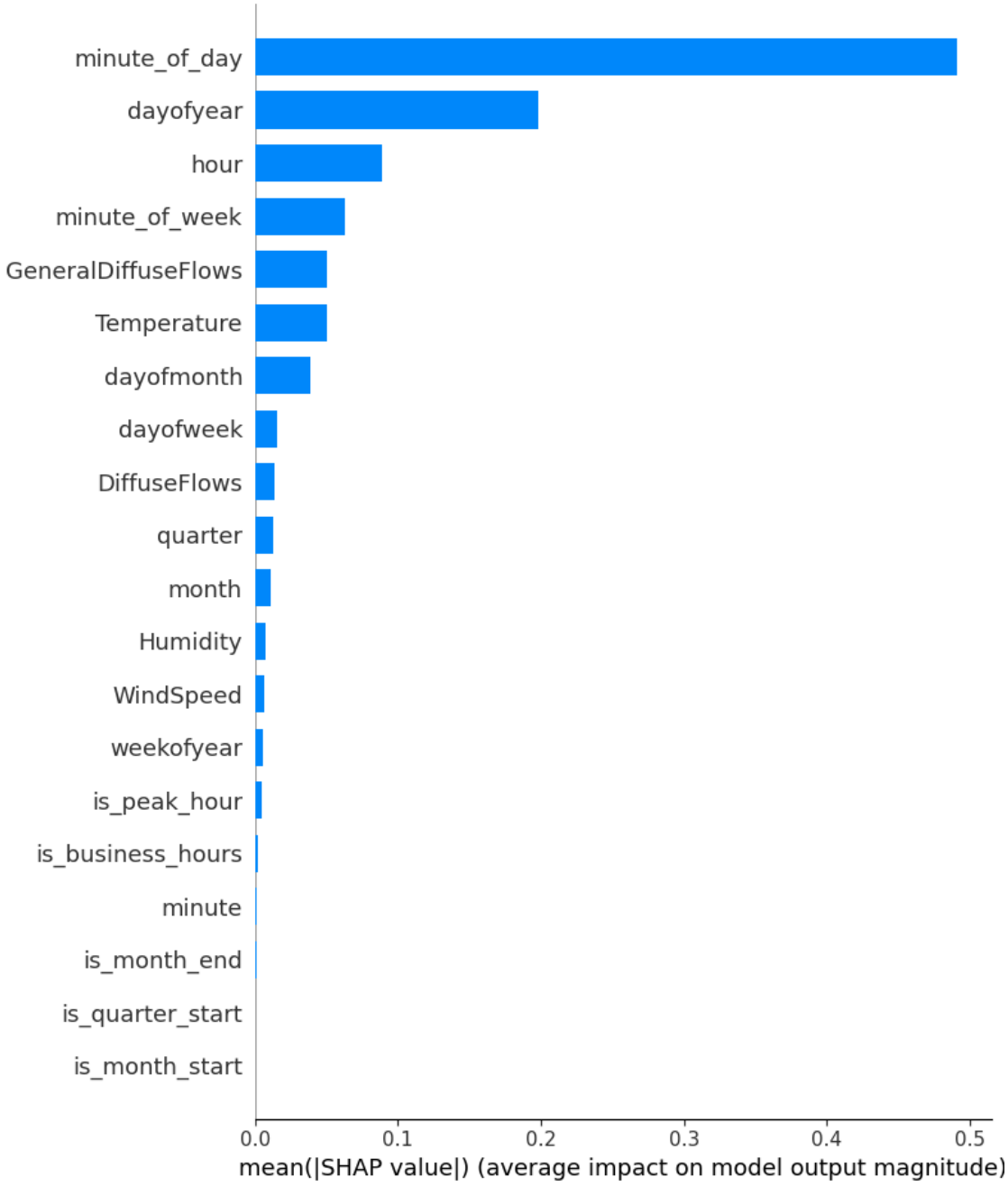
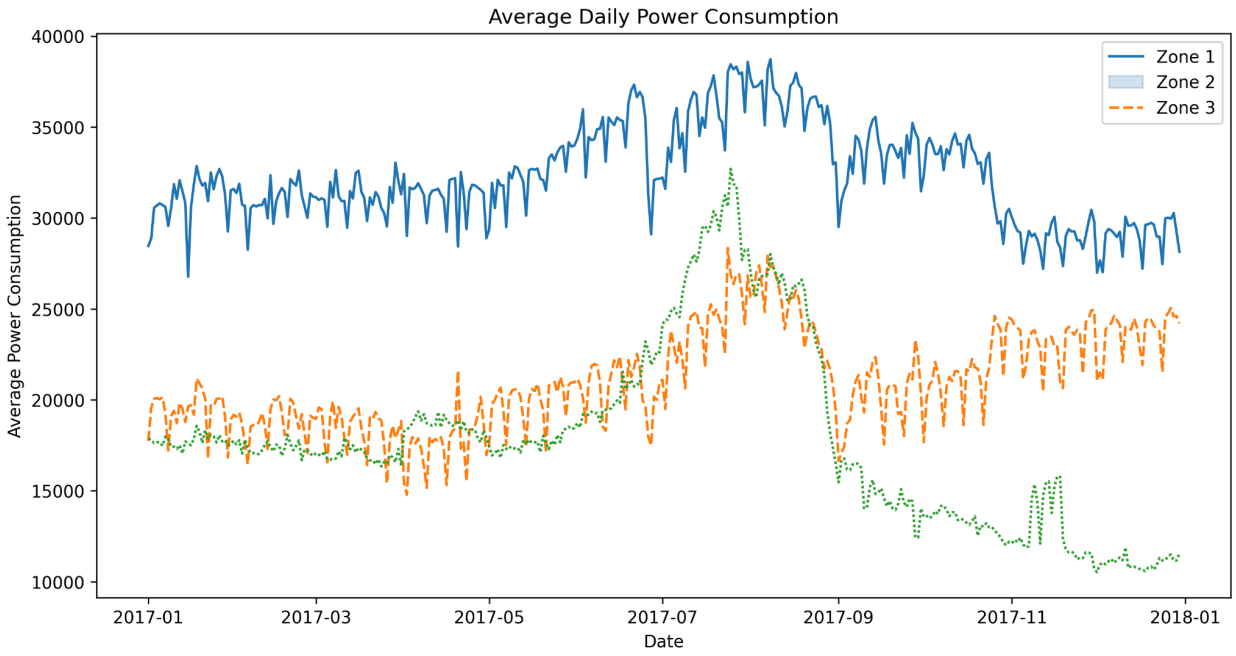
Forecasting Power Consumption

Part 1: Understanding the Dataset

1. What are the key features in the dataset, and how are they relevant to forecasting power consumption? **(Use SHAP analysis)**
    1. According to the SHAP (SHapley Additive exPlanations) analysis, several temporal features have emerged as key contributors to forecasting power consumption. These include the *minute of day*, *hour of day*, and *day of year.*

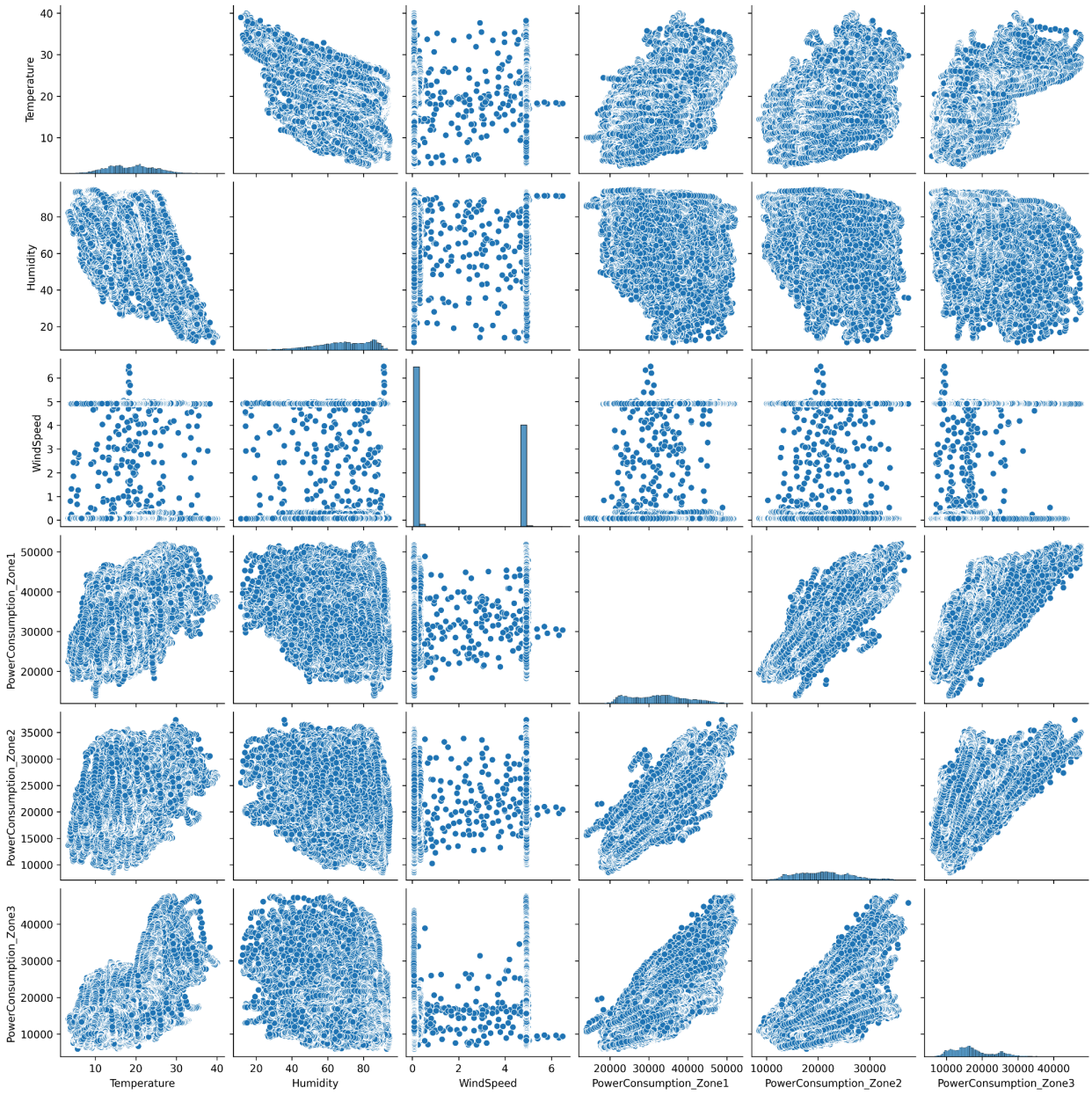


2. What challenges might arise in forecasting time series data from this dataset? **(Give some figures from your analysis)**
    1. A key challenge is the **non-stationarity of power consumption**, as it varies significantly over time and is not fully correlated across the three zones. This introduces high variability, making it difficult to model consistent patterns.



3. How does weather (temperature, humidity, wind speed) potentially impact power consumption in each zone? **(Give some statistics from your analysis)**
    1. Power consumption is positively correlated with temperature. However, it is negatively correlated with humidity. On the other hand, wind speed positively impacts power consumption in Zones 1, 2, and 3 showing widely scattered points,

suggesting no clear correlation.



Part 2: Data Preprocessing

4. You need to handle missing or inconsistent data in this dataset. Consider all necessary steps to process the data according to your needs.
    - The dataset contained no missing data; therefore, handling such cases was not required.
5. I suggest refraining from applying feature engineering to compute additional time-based features. Instead, focus on using the raw data directly to implement your long-sequence modelling.

- I implemented a function that extracts various time-related attributes from the dataset's timestamp index to enrich the feature set for downstream modeling tasks. The features are:
    1. **Basic Temporal Features:** Hour, minute, day of the week, quarter, month, day, year
    2. **Binary Indicators:** is_weekend, is_month_start, is_business_hours, is_peak_hour
    3. **Minute-Level Features:** minute_of_day, minute_of_week
    4. **Cyclical indicators:** season, quarter, weekofyear

6. How would you split the dataset into training, validation, and test sets for time series forecasting? Write the code to achieve this.
    - I have split the dataset into 75% train and 25% test data. I used the test dataset as a validation dataset, too.
7. Convert your time series data into tokens for use with the transformer models.
    - I converted my time series data into sequences to be used as input tokens for a Transformer model. I created these sequences by selecting a fixed-length window of past observations (look_back) as input and the corresponding future values (forecast_horizon) as output. This ensures the model learns temporal dependencies from historical data. The sequences are then formatted as NumPy arrays for efficient processing.

Part 3: Model Implementation

8. Implemented the following forecasting models:
    - **Vanilla Transformer based encoder-decoder model**
    - **XGBoost model**
9. Compare the model architectures according to your understanding.
    - I experimented with two forecasting models:
        - **XGBoost Model:** A gradient boosting framework from the XGBoost library, known for its efficiency in handling tabular data and capturing non-linear relationships.
        - **Transformer Model:** A custom architecture consisting of a fully connected encoder, positional encoding layer, 3 Transformer layers, and a fully connected decoder. The model leverages self-attention to capture long-range dependencies in sequential data.

Part 5: Evaluation and Visualization

15. What metrics would you use to evaluate forecasting performance?
    - I used **Mean Squared Error (MSE)** and **Mean Absolute Error (MAE)** as evaluation metrics to assess the forecasting accuracy and the model's ability to minimize prediction errors.
16. Implement the evaluation metrics.

- I implemented **MSE** and **MAE** using the **sklearn.metrics** library to compute the error between the predicted and actual values for performance evaluation

18. Which model performs best overall? Are there any zones where performance varies significantly?

- The **Transformer-based model** performed the best overall.

Model	MAE	MSE
XGBoost	0.4156	0.2803
Transformer Model	0.3593	0.2199