Dhirubai Ambani Institute of Information and Communication Technology

Zone-1 Power Consumption of Tetouan City

*Student :*Krishna Tank (202418056)   Teacher: Dr. Pritam Anand

Contents

1 Abstract

This project aims to develop a reliable model for forecasting hourly electricity consumption in Zone 1 of Tetuan City for the month of January 2017. The dataset includes environmental features such as temperature, humidity, wind speed, and solar radiation. After conducting exploratory data analysis and seasonal decomposition, we apply a variety of forecasting models—ARIMA, SARIMA, SARIMAX, XGBoost, Random Forest, LSTM, and GRU—to capture both linear and nonlinear patterns in the data.

Each model's performance is evaluated using metrics like RMSE, MAPE, and R^2. Results show that while traditional models provide strong baselines, deep learning models like LSTM and GRU are also good at capturing complex temporal dependencies. The study highlights the effectiveness of combining statistical and machine learning approaches to build robust forecasting systems for smart energy planning.

2 Problem Statement

Effective energy planning in urban areas requires precise forecasting of electricity demand. In Tetuan City, managing power consumption in Zone 1 is particularly challenging due to the strong influence of unpredictable environmental variables such as temperature, humidity, and wind speed. These fluctuations complicate short-term energy forecasting, potentially leading to inefficient resource allocation, increased operational costs, or energy shortages. The absence of a robust predictive system limits the ability of energy providers to proactively manage supply and demand. This project addresses the problem by developing a forecasting model that accurately predicts hourly electricity consumption in Zone 1. While the study focuses on the month of January, the solution is designed to be scalable and applicable to any month, offering a flexible framework for improving energy management.

3 Data Description

- The dataset for this project contains time-stamped records of environmental conditions and electricity consumption in Tetuan City from January 1, 2017, to December 30, 2017, with data recorded every 10 minutes.
- It includes the column DateTime as the time index, along with variables such as Temperature (in ° C), Humidity (percent- age) and Wind Speed (m / s), which influence power consumption. The data sett also features solar radiation measures, such as General Diffuse Flows and Diffuse Flows, which impactthe powerpower demand. Finally, power consumption data are provided for Zone 1, Zone 2, and Zone 3, possibly measured in watts
- From this data set, we perform temporal analysis, seasonal decomposition, and forecasting power demand based on weather and environmental variables.

Data Preprocessing

- The data set initially contained time-stamped records of environmental conditions and electricity consumption in three distinct zones of the city of Tetuan. To prepare the data for analysis, the DateTime column was first parsed into a standard datetime format to facilitate temporal manipulation.
- Entries with missing or invalid timestamps were subsequently removed to ensure data integrity. After cleaning, the DateTime column was set as the index, enabling effective time-series operations and ensuring chronological alignment across variables.
- The dataset was then sorted in ascending order of time, which is crucial for accurate forecasting and visualization. The power consumption data for Zone 1, Zone 2, and Zone 3 was resampled at an hourly frequency to reduce noise and better observe overarching trends.
- This resampling involved calculating the hourly mean power usage for each zone. To provide a visual understanding of these consumption patterns, a line plot was created that compared hourly electricity usage across the three zones over the entire year

The original dataset consisted of power consumption measurements recorded at 10minute intervals throughout an entire year. This resulted in a total of:

Total data points $= \frac{60 \text{ minutes}}{10 \text{ minutes}} \times 24$ hours ×365 days = 6×24×365 = 52,416

In order to obtain a time series of hourly power consumption, we aggregated every 6 consecutive 10-minute readings by summing them, resulting in:

Hourly data points $= \frac{52,416}{6} = 8,736$

Let the original power consumption time series be denoted as:

$$P = \{p_1, p_2, p_3, \ldots, p_{52416}\}$$

We define the new hourly aggregated time series P' as:

$$P'_t = \sum_{i=1}^{6} p_{6(t-1)+i} \qquad \text{for } t = 1,2,\ldots,8736$$

This transformation provides the total power consumption for each hour and ensures the time series is more suitable for hourly forecasting models.

- Gradual decline toward year-end.

4 Data Analysis

4.1 Data Analysis Introduction

In this section, we performed a comprehensive exploratory data analysis (EDA) to understand the temporal dynamics and environmental factors influencing power consumption in Tetuan City. Hourly consumption trends across the three zones revealed consistent patterns, with Zone 1 showing the highest usage and Zone 3 displaying greater volatility. A significant mid-year spike in consumption, especially in Zone 3, pointed toward possible seasonal influences.

Meteorological variables such as temperature, humidity, wind speed, and solar radiation were analyzed. Temperature followed a clear seasonal trend, peaking in summer, while humidity and wind speed showed irregular fluctuations—highlighting their potential impact on indoor energy demands.

We conducted seasonal decomposition on power consumption and meteorological variables for January. The trend components revealed underlying consumption growth, while seasonal components showed regular daily cycles. Residuals captured short-term, random fluctuations, affirming the complexity of energy usage behavior.

Correlation heatmaps and scatter plots confirmed strong relationships between temperature and power usage, especially in Zones 1 and 2. These insights laid a strong foundation for selecting relevant features and developing effective forecasting models.

4.2 Zone 1, Zone 2 and Zone 3 Power Consumption

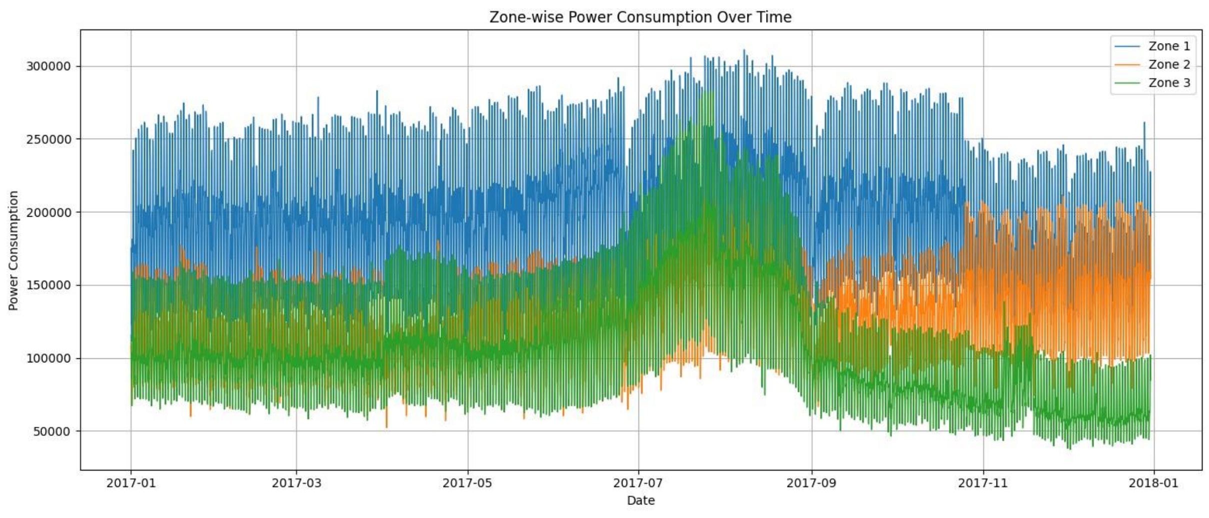


Figure 1: Zone 1, Zone 2, and Zone 3 Power Consumption

The power consumption across all three zones exhibits clear temporal trends with Zone 1 consistently showing the highest usage. A notable surge is observed around mid-2017, especially in Zone 3, possibly indicating seasonal peaks or policy changes. Consumption gradually stabilizes towards the end of the year, reflecting a return to baseline demand levels.

4.3 Hourly General Diffuse Flows

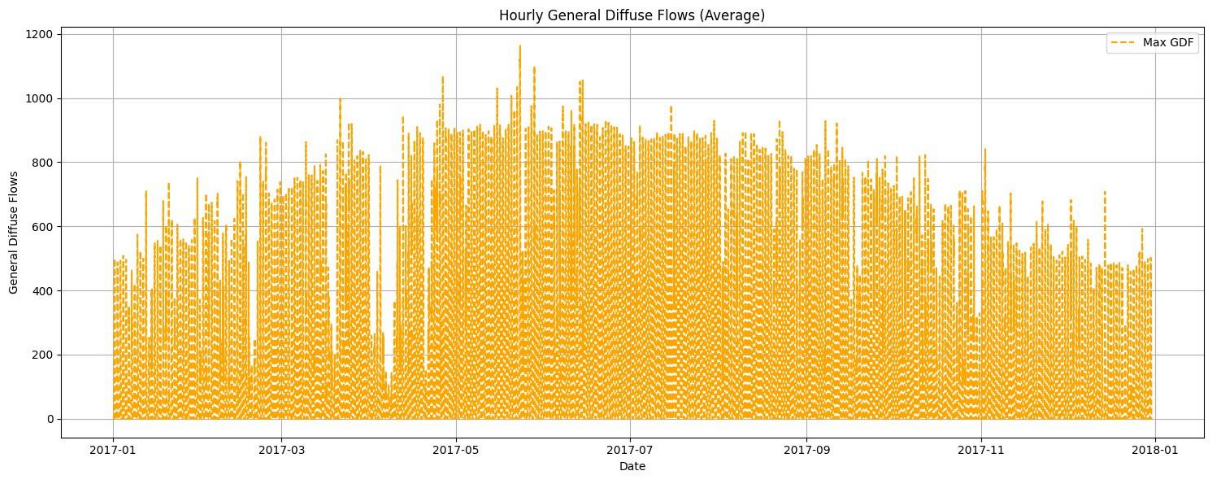


Figure 2: Hourly General Diffuse Flows

General diffuse flow increases gradually from the start of the year and peaks between mid of the year, aligning with summer months. This pattern reflects elevated solar radiation during this period, which can influence cooling demands and power usage indirectly.

4.4 Hourly Humidity

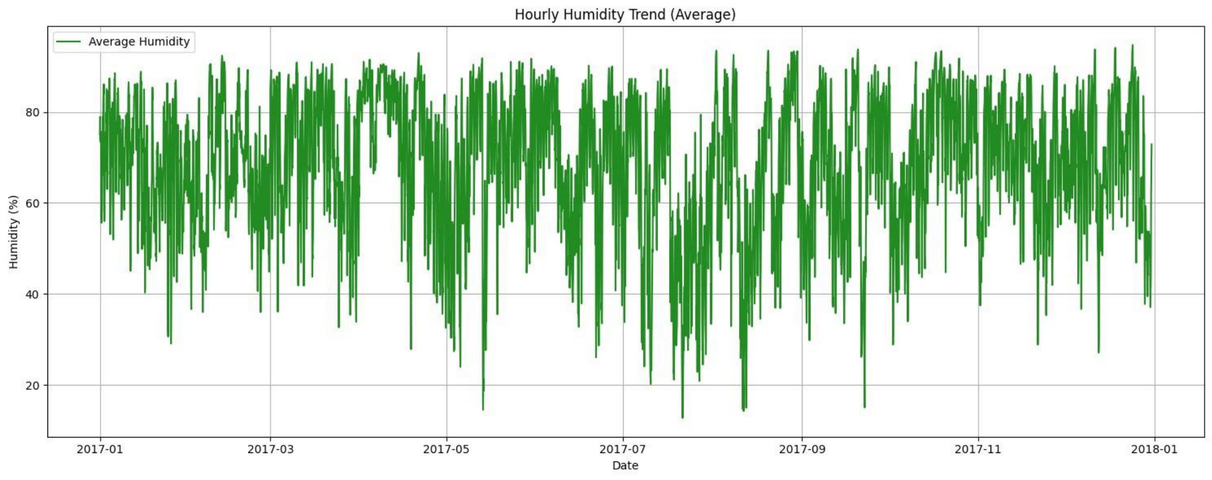


Figure 3: Hourly Humidity Levels

Humidity levels fluctuate throughout the year, ranging roughly between 40% and 90%. Mid-year months exhibit higher humidity variance, potentially tied to seasonal monsoon transitions. Elevated humidity can increase indoor cooling needs, influencing electricity consumption.

4.5 Hourly Wind Speed

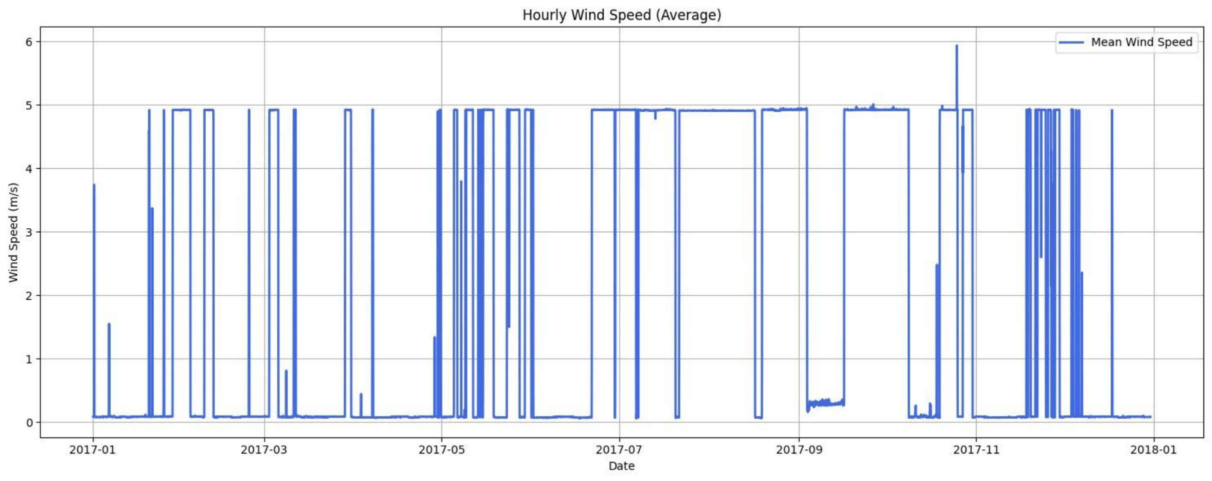


Figure 4: Hourly Wind Speed Trends

Wind speed shows frequent fluctuations with sharp spikes observed in spring and autumn months.

4.6 Hourly Temperature Trend

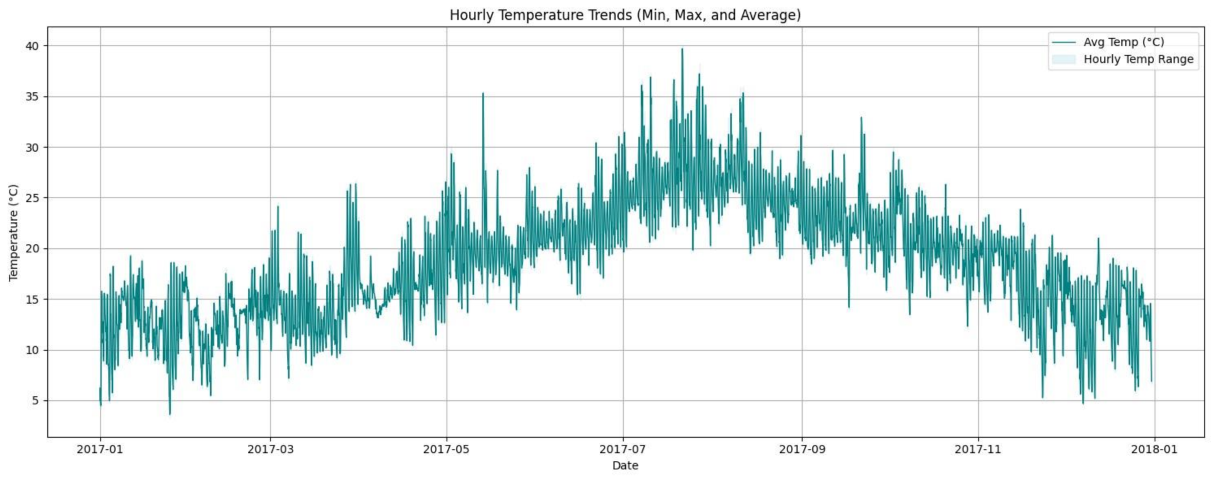


Figure 5: Hourly Temperature Trend

Temperature follows a seasonal sinusoidal trend, with summer peaks during July–August and winter lows in January and December. This variation plays a significant role in driving electricity consumption due to temperature-sensitive appliances like air conditioning and heating.

4.7 Correlation Heatmap

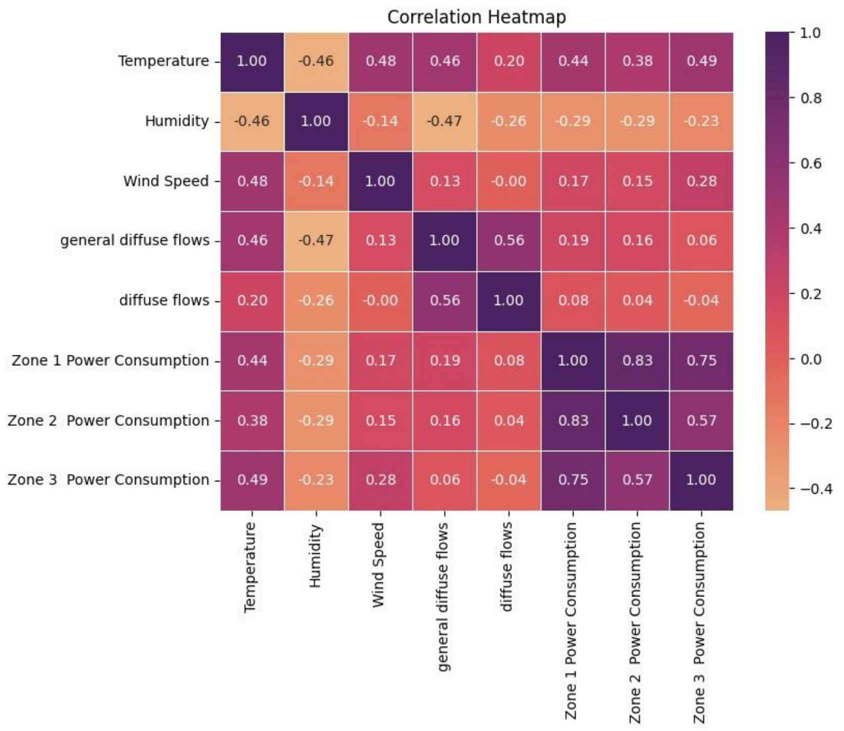


Figure 6: Correlation Heatmap

The heatmap reveals strong positive correlations between temperature and power consumption, especially in Zone 1. Humidity shows a weaker, yet noticeable, inverse relationship with temperature. These interactions highlight key variables influencing energy usage and justify their inclusion in forecasting models.

4.8 Zone 1 vs Zone 2 Power Consumption

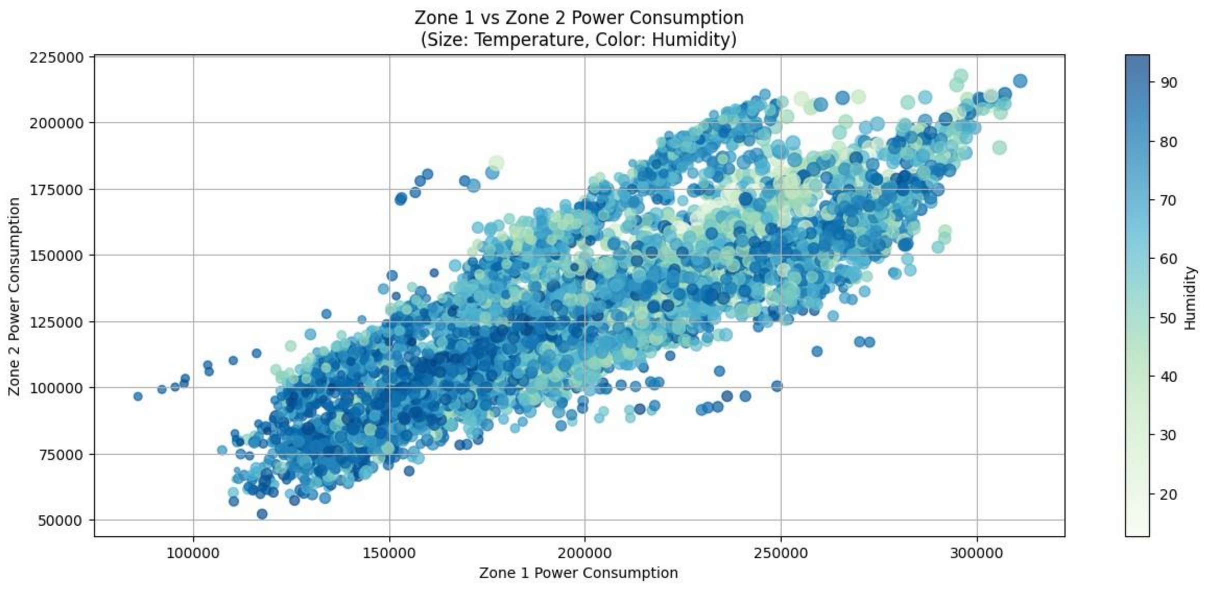


Figure 7: Zone 1 vs Zone 2 Power Consumption

Power usage patterns in Zone 1 and Zone 2 are relatively synchronized, suggesting similar usage behavior or infrastructure demands. However, Zone 1 consistently exhibits higher consumption, which may indicate denser population or greater industrial activity. Here we add two more factore size and the color saturation size shows the Temperature as the size increase the temperature is increase and the color saturation shows the humidity

4.9 Zone 1 vs Zone 3 Power Consumption

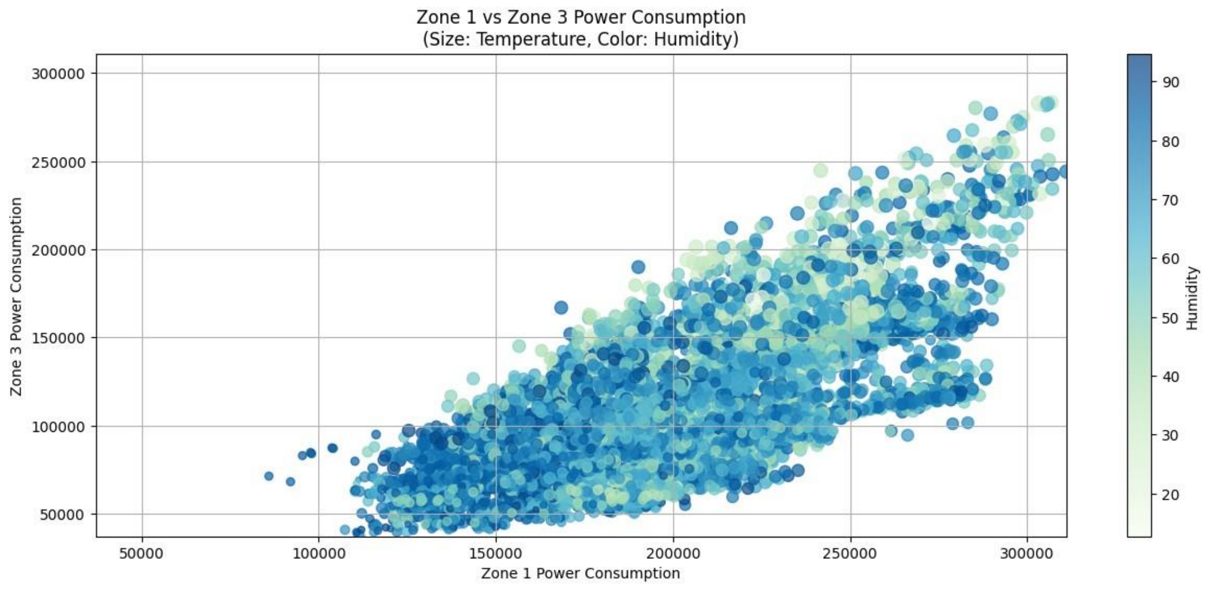


Figure 8: Zone 1 vs Zone 3 Power Consumption

Zone 3 shows more fluctuation compared to Zone 1 and experiences a sharper mid-year peak. This may reflect different usage patterns or external environmental factors impacting energy consumption more significantly in Zone 3.Here we follow the same pattern as Temperature is shows by the size and the Humidity is show by the Color .

4.10 Temperature vs Zone 1 Power Consumption

On X axis we take the Temperature and on Y-Axis we take the Zone wise power consumption and we add size and Color saturation where size shows the General Diffuse flow and the color shows the Humidity .

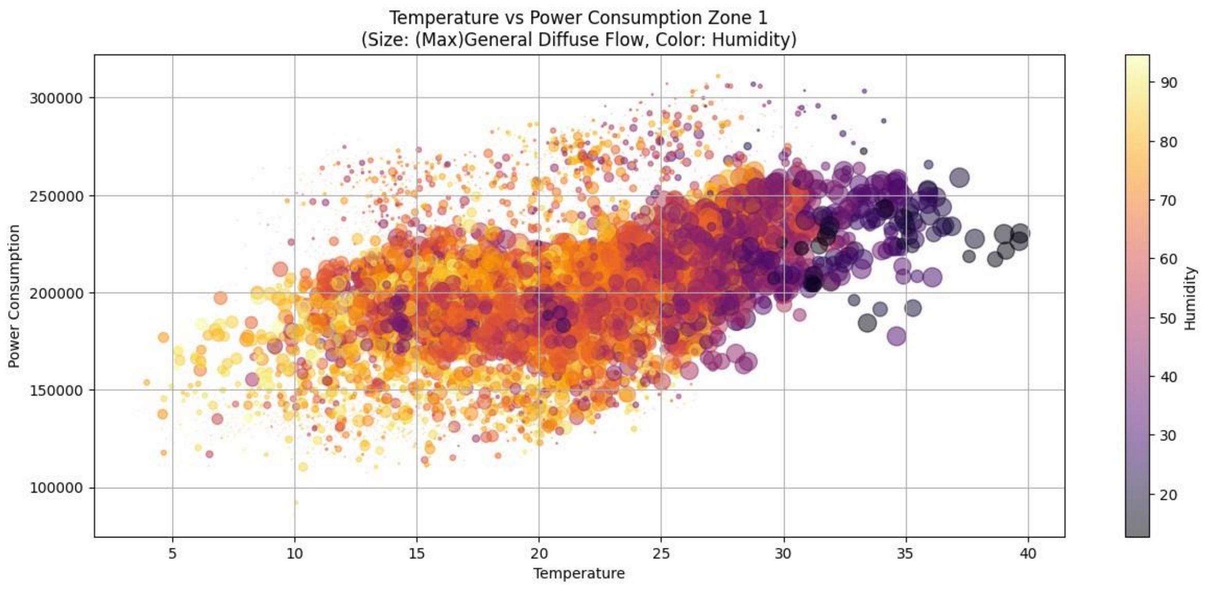


Figure 9: Temperature vs Zone 1 Power Consumption

A strong positive trend is evident—higher temperatures coincide with higher power usage, likely due to increased cooling requirements. This indicates that temperature is a major driver of electricity demand in Zone 1.

4.11 Temperature vs Zone 2 Power Consumption

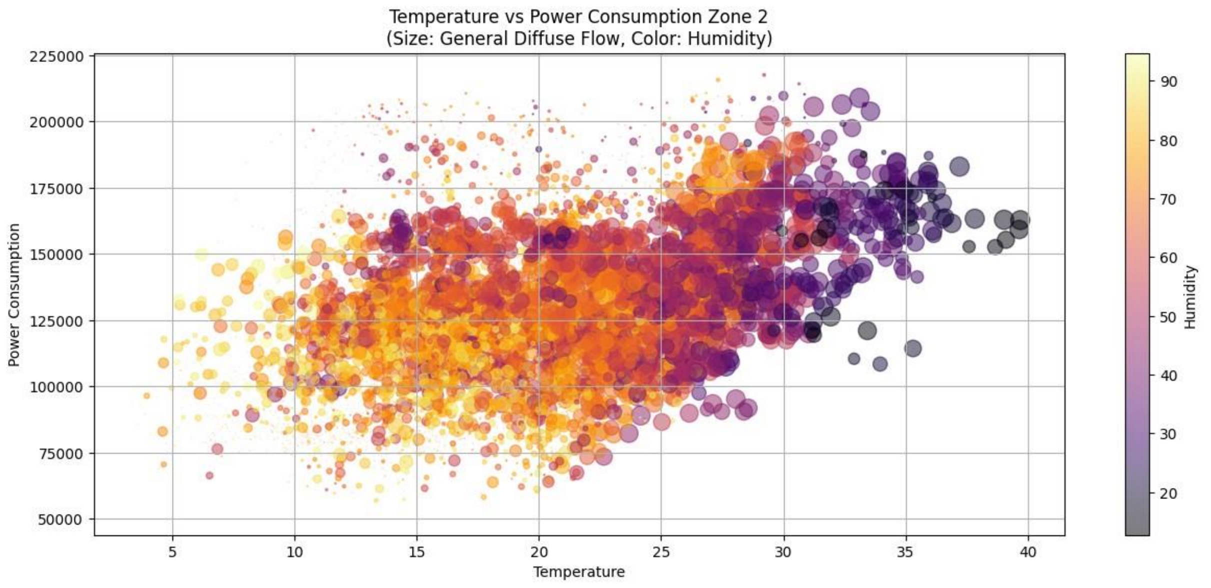


Figure 10: Temperature vs Zone 2 Power Consumption

Zone 2 follows a similar trend as Zone 1, but with slightly lower sensitivity. This might imply a lesser dependence on temperature-driven appliances or more efficient insulation and climate control systems in that zone.

4.12 Temperature vs Zone 3 Power Consumption

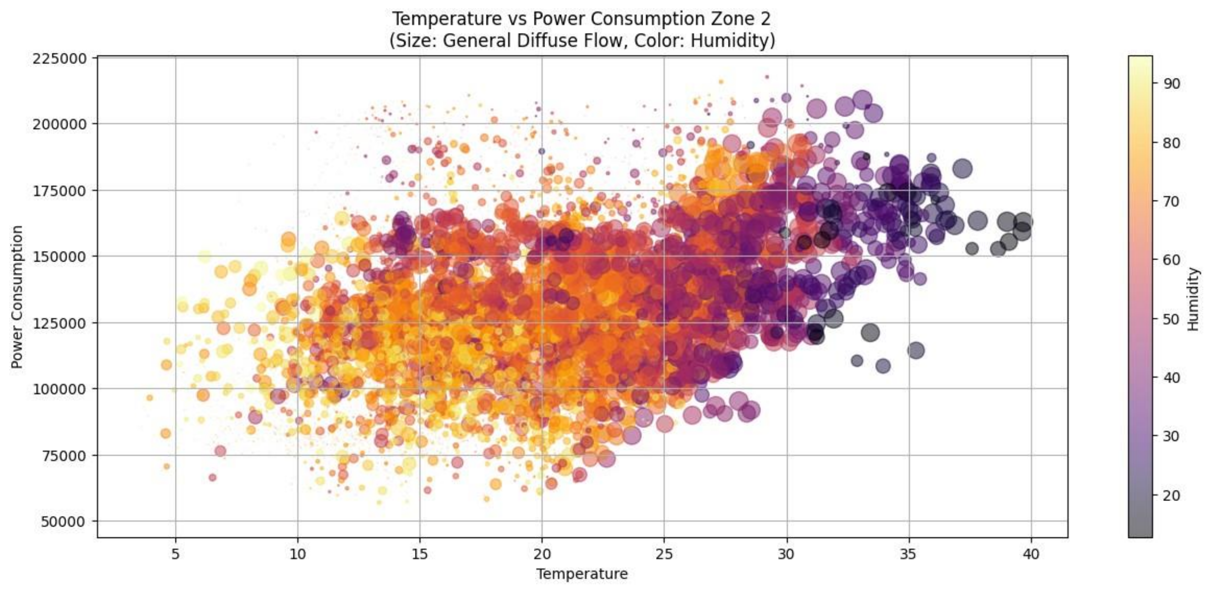


Figure 11: Temperature vs Zone 3 Power Consumption

Zone 3 demonstrates a more volatile relationship with temperature, reflecting inconsistent usage patterns or diverse consumer profiles. Nonetheless, higher temperatures still generally lead to increased energy demand

4.13 Diffuse Flow vs Humidity

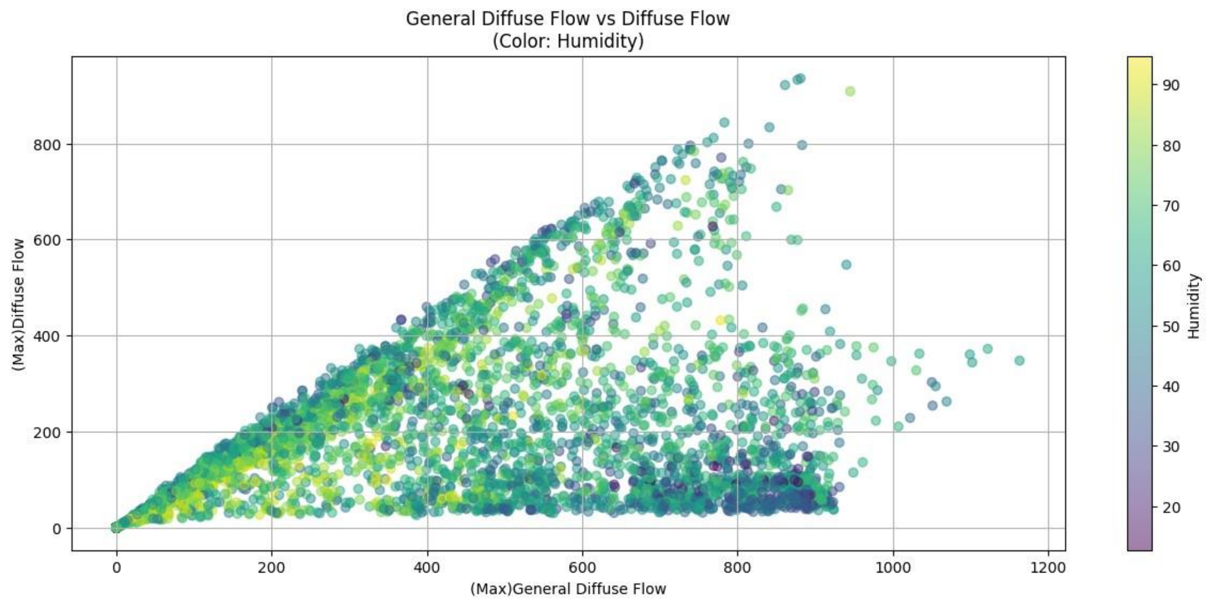


Figure 12: Scatter Plot: Diffuse Flow vs Humidity

The scatter plot suggests a moderately inverse relationship—higher humidity often coincides with lower diffuse solar radiation. This interaction may affect indoor lighting and cooling needs, thereby impacting power consumption indirectly.

4.14 Decomposition of all the Attribute

Here we decompose All the Attribute into the tree part Trend Seasonality and residual to show that the how the overall trend of the all the attribute have.Is there any strong seasonality? or to Show the residual.

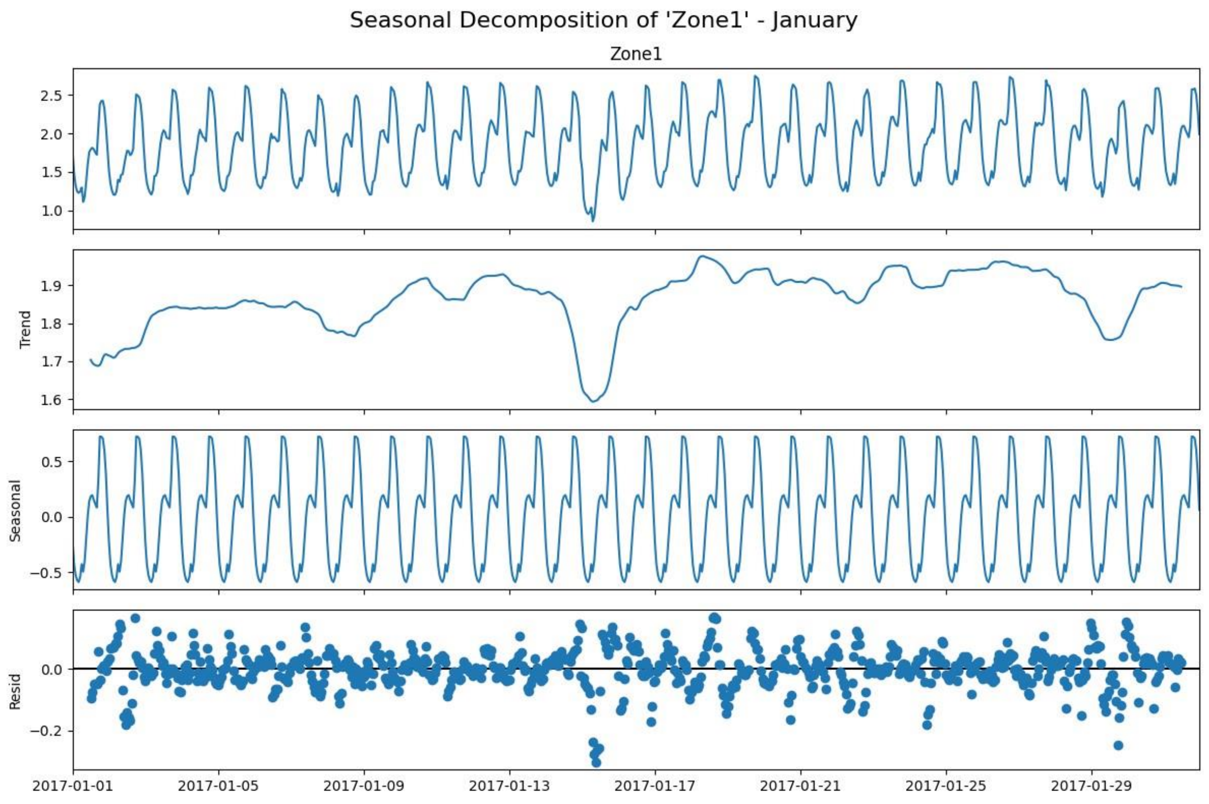


Figure 13: Seasonal Decomposition of Zone 1 Power Consumption (January)

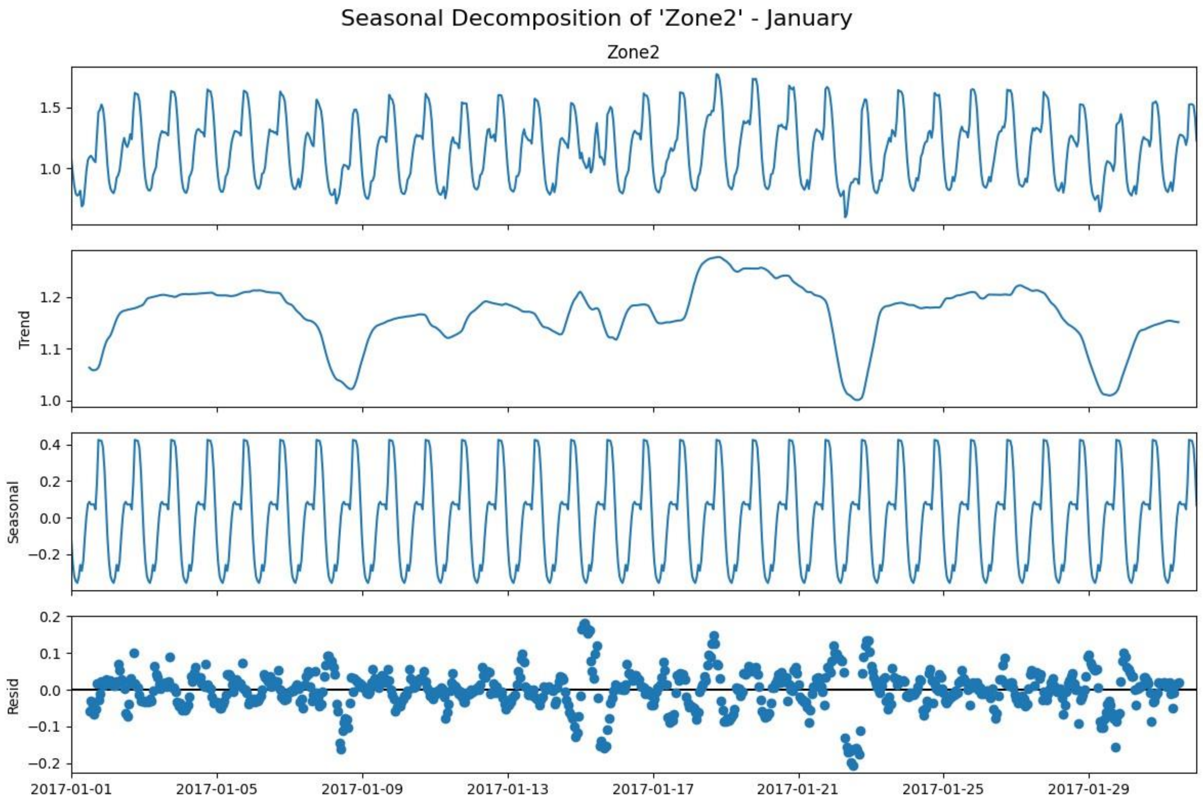


Figure 14: Seasonal Decomposition of Zone 2 Power Consumption (January)

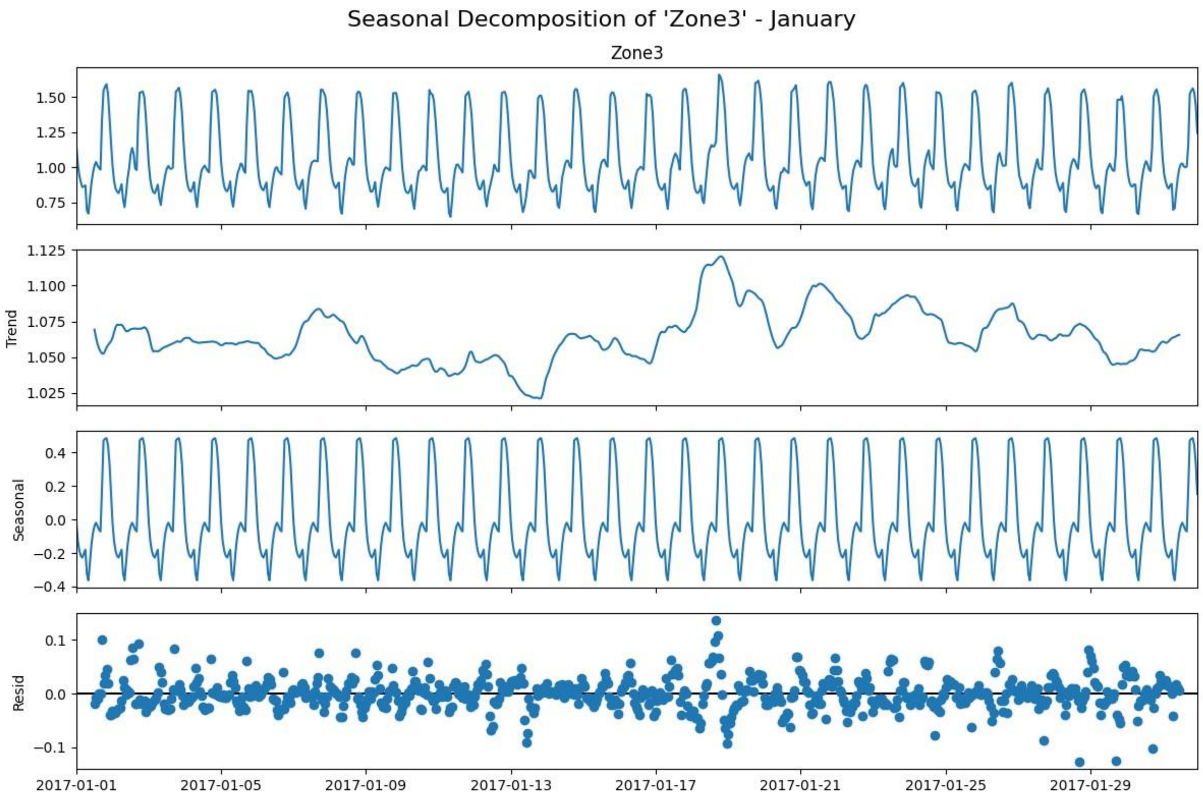


Figure 15: Seasonal Decomposition of Zone 3 Power Consumption (January)

Insight on Seasonal Decomposition:
All three zones display a strong seasonal component with clear hourly cycles, reflecting consistent daily power usage patterns. The trend components in Zone 1 and Zone 2 are relatively stable with a slight upward direction, suggesting gradual demand growth. In contrast, Zone 3 shows more fluctuation in the trend, hinting at more volatile usage behavior. Across all zones, residuals remain low, indicating minimal irregular activity and a well-fitted decomposition model.

5 Methodology

Stationarity Testing using Augmented Dickey-Fuller (ADF) Test

To ensure the reliability of time series forecasting models, it is essential to verify whether the data is stationary. A stationary time series has constant statistical properties over time, such as mean and variance. The Augmented Dickey-Fuller (ADF) test is a widely used statistical test to check for stationarity.

The ADF test examines the null hypothesis that a unit root is present in the time series, indicating non-stationarity. The test statistic is compared against critical values, and the corresponding p-value helps determine the result:

- If the p-value is <0.05, we reject the null hypothesis and conclude that the series is stationary.
- If the p-value is >0.05, we fail to reject the null hypothesis and conclude that the series is non-stationary.

In our analysis, the ADF test statistic for Zone 1 was -4.45 with a p-value of 0.0002, indicating that the data is stationary at a 5% significance level. Therefore, differencing was not required before modeling for Zone 1. This simplifies the preprocessing step and allows us to directly apply time series models to the original data.

Table 1: Augmented Dickey-Fuller (ADF) Test Results

Variable	ADF Statistic	p-value	Stationarity
Zone1	-4.4459	0.0002	Stationary
Zone2	-4.6853	0.0001	Stationary
Zone3	-1.2612	0.6467	Not Stationary
GDF_max	-5.4609	2.52e-06	Stationary
DF_max	-5.4609	2.52e-06	Stationary
Wind_mean	-6.5301	9.92e-09	Stationary
Temp_mean	-3.0812	0.0280	Stationary
Humidity_mean	-10.1607	7.52e-18	Stationary

5.1 Monthly Data Preparation

To prepare the data for time series forecasting, we created a custom function get_month_data() that extracts data for a specific month. This function separates the exogenous variables from the target variable (Zone1) and also drops irrelevant or less informative columns such as GDF_max, DF_max, and Wind_mean.

```
X, y = get_month_data( final_df,
        target_column='Zone1',
        month=1,
```

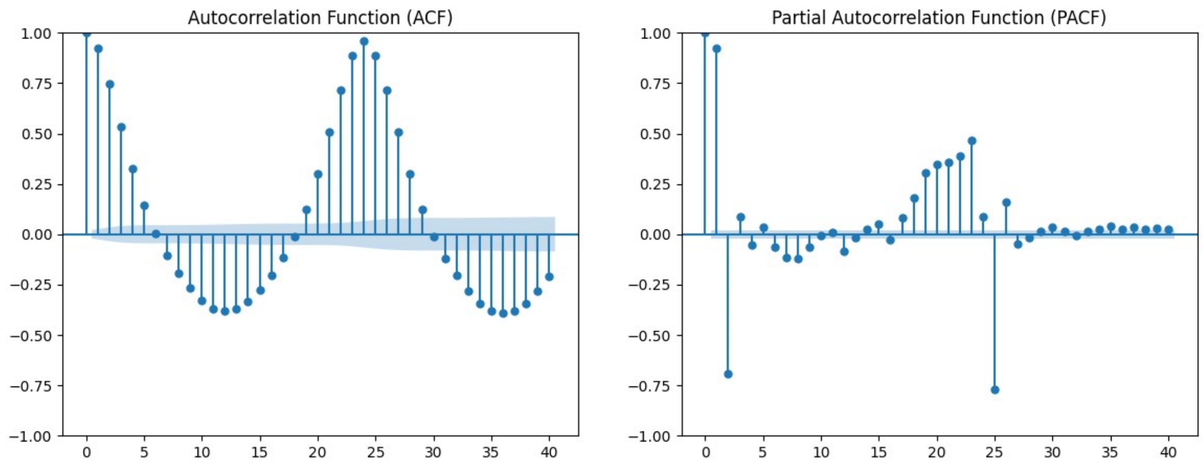


Figure 16: Zone 1 - without differencing

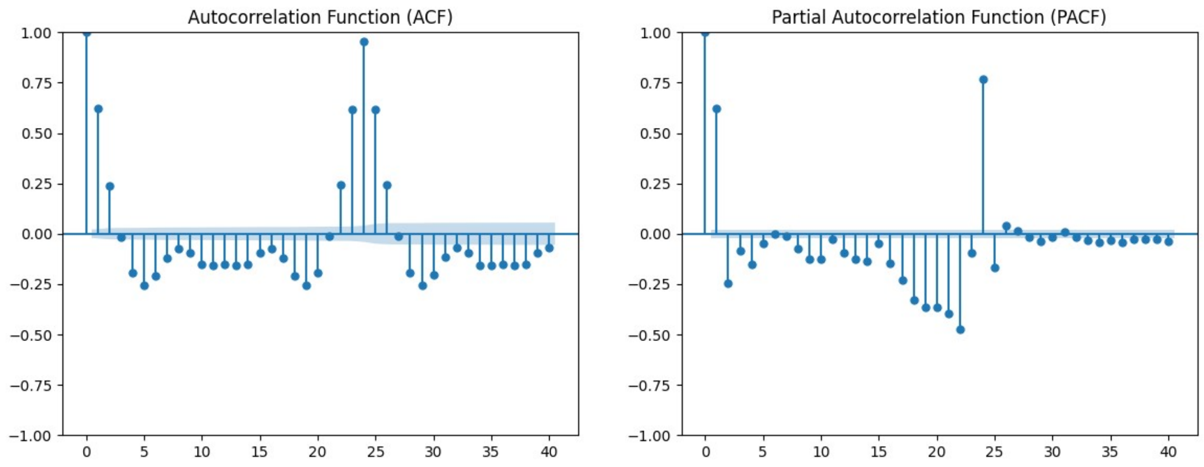


Figure 17: Zone 1 - with differencing

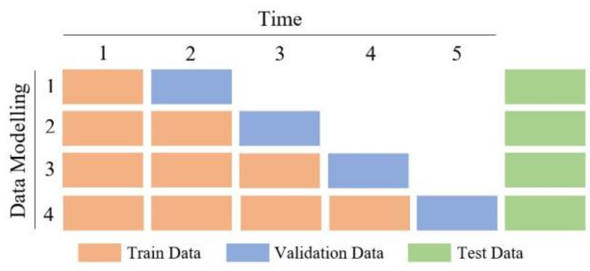


Figure 18: TimeSeriesSplit

```
        drop_columns=['GDF_max', 'DF_max', 'Wind_mean']
)
```

The motivation behind using monthly data is to narrow down the forecasting window to a manageable size, enabling us to model the consumption pattern of a few consecutive days—

typically 3 to 4 days. This setup allows the model to learn short-term temporal dependencies more effectively and predict daily power consumption patterns within a month.

5.2 Train-Test Split and Validation Strategy

Initially, the dataset was divided into training and test sets using an 85:15 ratio. This ensures that the model has sufficient data to learn from while retaining a portion for unbiased performance evaluation. Initially, the dataset was divided into training and test sets using an 85:15 ratio. The training set was used to fit the model, while the test set was kept for final performance evaluation on unseen data.
To respect the temporal nature of the data, validation within the training set was performed using a TimeSeriesSplit strategy. Unlike K-Fold cross-validation, which randomly splits the data, TimeSeriesSplit preserves the time order by creating training/validation sets where the training set is always prior in time to the validation set.

- Each split expands the training set and rolls forward the validation set, mimicking a real-world forecasting scenario.
- This strategy prevents data leakage and allows us to tune the model while maintaining the chronological integrity of the data.

This method helps ensure that the model generalizes well to future data and avoids overfitting by validating on temporally future subsets.

5.3 Model Implementation

5.4 Time Series Modeling Flow

The typical steps to build a classical time series model (e.g., ARIMA/SARIMA) are as follows:

1. Check Stationarity:
   Use the Augmented Dickey-Fuller (ADF) test. If p-value > 0.05, the series is likely a random walk and non-stationary.
2. Differencing:
   Apply differencing to remove trends:
   $$Y_t' = Y_t - Y_{t-1}$$
   If seasonal effects exist, use seasonal differencing:
   $$Y_t'' = Y_t - Y_{t-s}$$
3. Model Selection:
   Fit different combinations of (p,d,q) and seasonal (P,D,Q,s), and choose the model with the lowest AIC.
4. Model Fitting and Evaluation:
   Fit the selected model and evaluate on test data using:

- RMSE
- MAPE
- R^2 Score

5.5 ARIMA(24,0,24)

The ARIMA model, with parameters $p = 24$, $d = 0$, and $q = 24$, is an autoregressive moving average model given by the equation:

$$y_t = \mu + \sum_{i=1}^{24} \phi_i y_{t-i} + \sum_{j=1}^{24} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

Where:

- y_t is the observed value at time t,
- μ is a constant term (intercept),
- $\phi_1,\ldots,\phi_{24}$ are the autoregressive coefficients,
- $\theta_1,\ldots,\theta_{24}$ are the moving average coefficients,
- ε_t is the error term (white noise) at time t.

This equation models the time series data based on both the past values (y_{t-i}) and the errors from previous time points (ε_{t-j}).

5.5.1 RMSE (Root Mean Squared Error)

The RMSE for the ARIMA(24,0,24) model on the test set is:

$$\text{RMSE} = 0.11$$

This indicates that, on average, the model's one-step-ahead predictions deviate from the true values by approximately 0.11 units of power consumption. This is a relatively small error, reflecting the model's accuracy in predicting the target variable.

5.5.2 MAPE (Mean Absolute Percentage Error) The MAPE for

the ARIMA model on the test set is:

$$\text{MAPE} = 4.01\%$$

This means that, on average, the model's forecast errors are within 4.01

5.5.3 R-Squared (R^2)

The R^2 value for the ARIMA model on the test set is:

$$R^2 = 0.9375$$

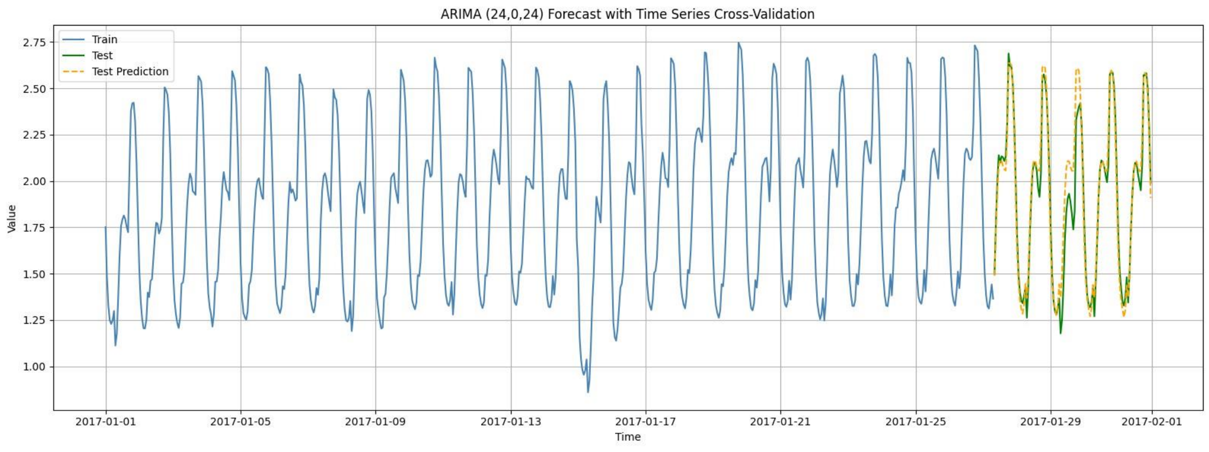
This indicates that approximately 93.75

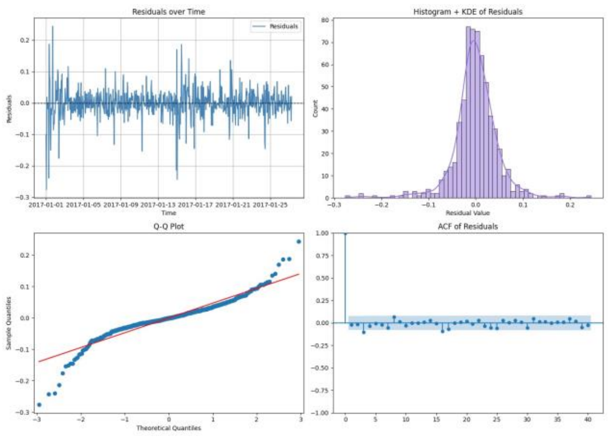
5.5.4 Comparison with Cross-Validation Results

During the time-series cross-validation, the ARIMA(24,0,24) model produced the following metrics for each fold:

- Fold 1: RMSE = 0.20, MAPE = 9.00%, $R^2 = 0.8221$
- Fold 2: RMSE = 0.12, MAPE = 5.66%, $R^2 = 0.9266$
- Fold 3: RMSE = 0.10, MAPE = 4.66%, $R^2 = 0.9512$

The test set metrics of RMSE = 0.11, MAPE = 4.01%, and $R^2 = 0.9375$ are consistent with the cross-validation results, indicating that the model performs well on unseen data and is not overfitting. The metrics show that the model generalizes well and is effective for forecasting power consumption in the Tetuan city zones.





5.6 SARIMA Model

The SARIMA (Seasonal AutoRegressive Integrated Moving Average) model extends the ARIMA model by incorporating seasonality. The specific configuration used is:

SARIMA$(2,1,2)(1,2,2)_{24}$ Where:

- $p = 2$, the order of the autoregressive part,

- $d = 1$, the degree of differencing to make the series stationary,
- $q = 2$, the order of the moving average part,
- $P = 1$, the seasonal autoregressive order,
- $D = 2$, the seasonal differencing order,
- $Q = 2$, the seasonal moving average order,
- $s = 24$, indicating a seasonal period of 24 hours (daily seasonality).

The general SARIMA model can be represented as:

$$\Phi_P(B^s)\phi_p(B)(1-B)^d(1-B^s)^D y_t = \Theta_Q(B^s)\theta_q(B)\varepsilon_t$$ Where:

- B is the backshift operator, such that $B^k y_t = y_{t-k}$,
- $\phi_p(B)$ and $\theta_q(B)$ are the non-seasonal AR and MA polynomials,
- $\Phi_P(B^s)$ and $\Theta_Q(B^s)$ are the seasonal AR and MA polynomials.

5.6.1 Evaluation Metrics

Time Series Cross-Validation Results:

- Split 1: RMSE = 0.27, MAPE = 14.06%, $R^2 = 0.6758$
- Split 2: RMSE = 0.10, MAPE = 4.03%, $R^2 = 0.9555$
- Split 3: RMSE = 0.08, MAPE = 4.05%, $R^2 = 0.9672$ Test Set Metrics:
- RMSE = 0.12
- MAPE = 4.57%
- $R^2 = 0.9199$

5.6.2 Analysis

The SARIMA model performs quite well on the test data, with a low RMSE of 0.12 and MAPE of 4.57%, and a high R^2 score of 0.9199. The cross-validation results show some variation, particularly in Split 1, where MAPE and RMSE are significantly higher. This suggests the presence of some anomalies or structural changes in that fold, possibly due to weather variations or changes in consumption behavior during that time window. Despite this, the overall performance across folds is strong, and the consistency in test metrics validates the model’s generalization ability on unseen data.

5.7 SARIMA with Exogenous Variables (SARIMAX)

5.7.1 Model Description:

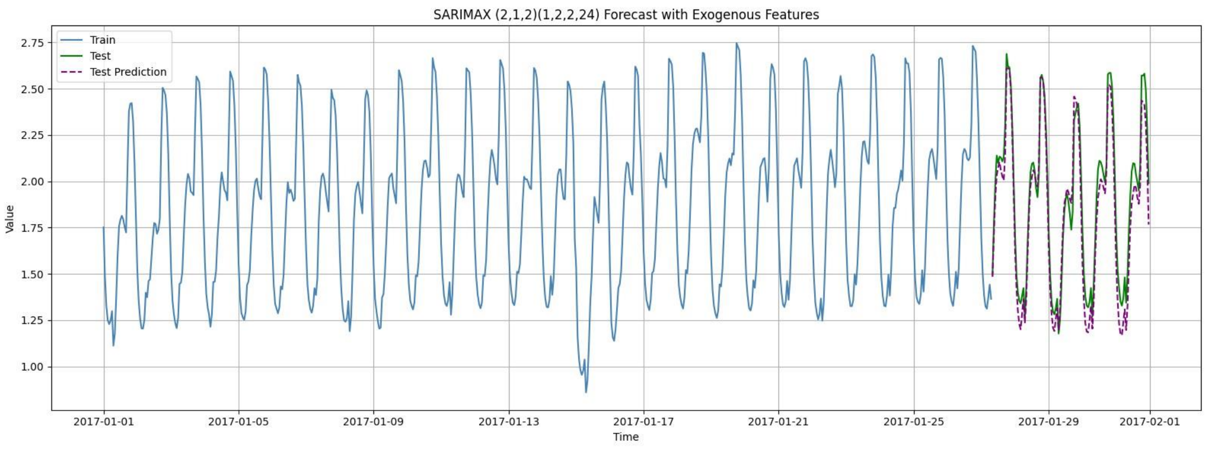
To enhance the forecasting capability of SARIMA, we incorporated exogenous features (SARIMAX model), which allows the model to leverage additional external variables that influence the target variable. In this case, weather-related features such as temperature, humidity, and others were included as exogenous inputs to better capture the variability in power consumption.

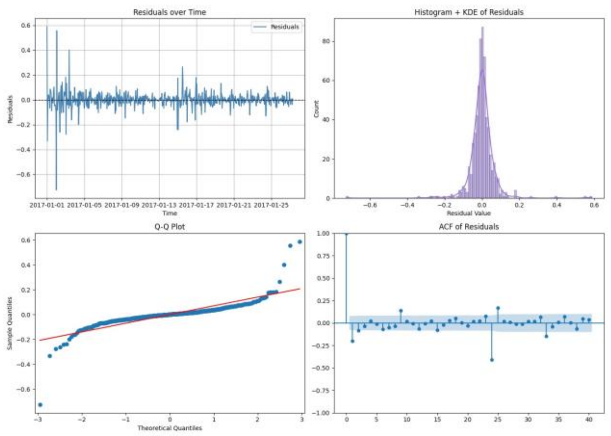
5.7.2 Model Notation:

The SARIMAX model is represented similarly to SARIMA but includes an exogenous component:

$$\Phi_P(B^s)\phi_p(B)(1-B)^d(1-B^s)^D y_t = \Theta_Q(B^s)\theta_q(B)\varepsilon_t + \beta X_t$$ Where:

- X_t is the vector of exogenous variables at time t,
- β is the coefficient vector associated with the exogenous variables.





5.7.3 Time Series Cross-Validation Results with Exogenous Features:

- Split 1: RMSE = 0.22, MAPE = 9.76%, $R^2 = 0.7811$
- Split 2: RMSE = 0.10, MAPE = 4.65%, $R^2 = 0.9527$
- Split 3: RMSE = 0.16, MAPE = 7.56%, $R^2 = 0.8795$

Test Set Metrics:

- RMSE = 0.11
- MAPE = 5.26%

- $R^2 = 0.9354$

5.7.4 Analysis:

Incorporating exogenous features improved the model's ability to explain variance in power consumption. Compared to the basic SARIMA model, SARIMAX showed better consistency across validation splits and improved test performance. The high R^2 value and low RMSE/MAPE indicate that the model effectively utilizes external influences like weather to enhance prediction accuracy.

5.8 Deep Learning Models: Simple RNN and LSTM

For the deep learning models (Simple RNN and LSTM), a rolling window strategy was employed instead of a time series cross-validation method. In this approach, the input sequence for each training sample was constructed using a sliding window of previous time steps, which helps capture the temporal dependencies of the power consumption data. This method enables the model to learn patterns across consecutive days, effectively mapping a sequence of prior consumption values (and optionally, exogenous features) to the next target value.

5.8.1 Mathematical Formulation of RNN and LSTM

Recurrent Neural Network (RNN): A basic Recurrent Neural Network (RNN) processes sequential data by maintaining a hidden state that captures information about previous time steps. The mathematical formulation for RNN is as follows:

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)\ \hat{y}_t$$

$= W_{hy}h_t + b_y$ Where:

- x_t is the input at time step t
- h_t is the hidden state at time step t
- W_{xh}, W_{hh} are weight matrices for input-to-hidden and hidden-to-hidden transitions respectively
- b_h, b_y are bias terms
- $\hat{y}_t$ is the predicted output

RNNs can struggle to capture long-term dependencies due to vanishing gradients, which is why more advanced architectures like LSTM are preferred for time series modeling.

Long Short-Term Memory (LSTM): LSTM networks mitigate the vanishing gradient problem by introducing gating mechanisms to control information flow. The LSTM cell uses the following equations:

$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$ (forget gate)

$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$ (input gate)

$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$ (candidate cell state) $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$ (new cell state) $o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$ (output gate) $h_t = o_t \odot \tanh(c_t)$ (new hidden state) Where:

- σ is the sigmoid activation function
- $\tanh$ is the hyperbolic tangent activation
- $\odot$ denotes element-wise multiplication
- x_t is the input at time t
- h_t is the hidden state at time t
- c_t is the cell state at time t
- W_*, U_* are weight matrices
- b_* are bias terms

The LSTM architecture enables learning from both short-term and long-term dependencies, which is critical in capturing complex temporal patterns in time series data like power consumption.

5.8.2 Training Summary:

- Simple RNN:
  - Epoch 100 Loss: 0.0187
  - Epoch 200 Loss: 0.0128
  - Epoch 300 Loss: 0.0102
  - Epoch 400 Loss: 0.0105
  - Epoch 500 Loss: 0.0099
- LSTM:
  - Epoch 100 Loss: 0.0216
  - Epoch 200 Loss: 0.0107
  - Epoch 300 Loss: 0.0067
  - Epoch 400 Loss: 0.0048
  - Epoch 500 Loss: 0.0030

5.8.3 Evaluation Metrics (on Test Set):

- Simple RNN:
  - RMSE: 0.1534
  - MAPE: 6.06%
  - R^2: 0.8726

- LSTM:
  - RMSE: 0.1122
  - MAPE: 4.83%
  - R^2: 0.9318

5.8.4 Analysis:

RNN: For the Simple RNN, the training loss gradually decreased from 0.0187 at epoch 100 to 0.0099 at epoch 500. This shows that the model was able to minimize the error on the training set steadily. However, the final performance on the test set revealed an RMSE of 0.1534, a MAPE of 6.06 percent, and an R^2 score of 0.8726. These metrics suggest that while the RNN learned the patterns fairly well, it might not have fully captured the complex temporal dependencies present in the data. The higher RMSE compared to LSTM also indicates slightly larger prediction errors. This is consistent with the known limitation of RNNs in modeling long sequences due to the vanishing gradient problem.

LSTM: In contrast, the LSTM model showed a more significant reduction in training loss, going from 0.0216 at epoch 100 to a much lower 0.0030 by epoch 500. The LSTM was able to learn long-term dependencies better and more efficiently due to its memory cell and gating mechanisms. This improvement is clearly reflected in the test performance: RMSE = 0.1122, MAPE = 4.83 percent, and R^2 = 0.9318. The lower RMSE and MAPE values suggest that LSTM produced more accurate predictions, and the higher R^2 indicates a better fit to the actual values.

Interpretation of Loss Trends and Metrics: The consistent decline in training loss for both models indicates that they were not underfitting. Additionally, the relatively low error values on the test set imply that neither model overfit severely either. However, the LSTM's significantly better performance suggests that the underlying data had temporal patterns that required more memory or context to model accurately—something that LSTM architectures are better suited for.

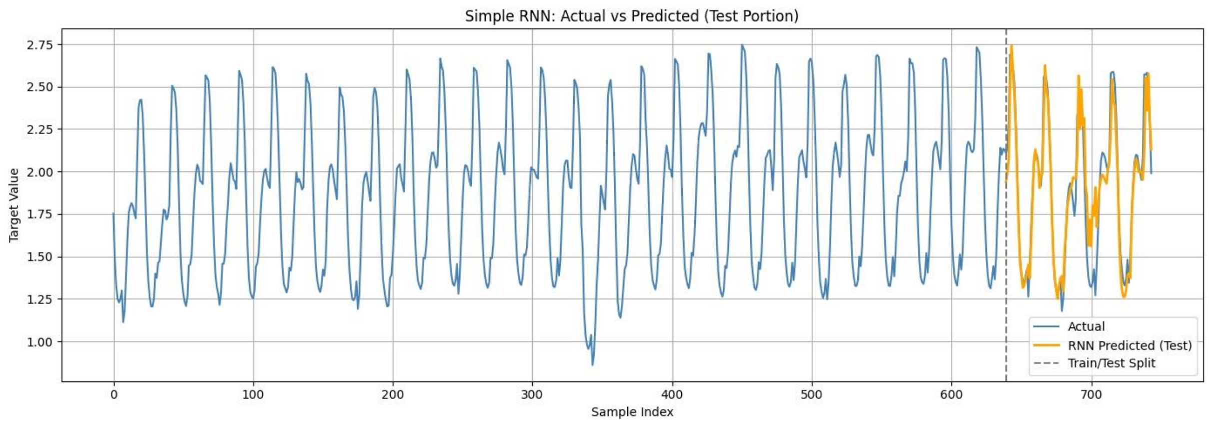


Figure 19: RNN Results

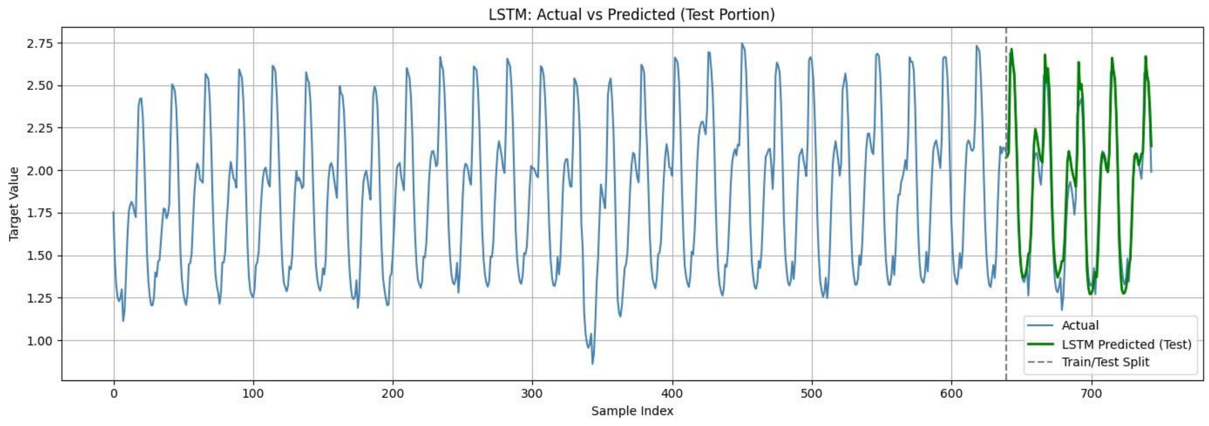


Figure 20: LSTM Results

5.9 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) is a simplified variant of the LSTM model that also aims to solve the vanishing gradient problem and capture long-term dependencies in sequential data. GRU combines the forget and input gates into a single update gate and merges the cell state and hidden state.

5.9.1 Mathematical Formulation:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \text{ (update gate) } r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \text{ (reset gate)}$$

$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \text{ (candidate hidden state)}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad \text{(new hidden state)}$$ Where:

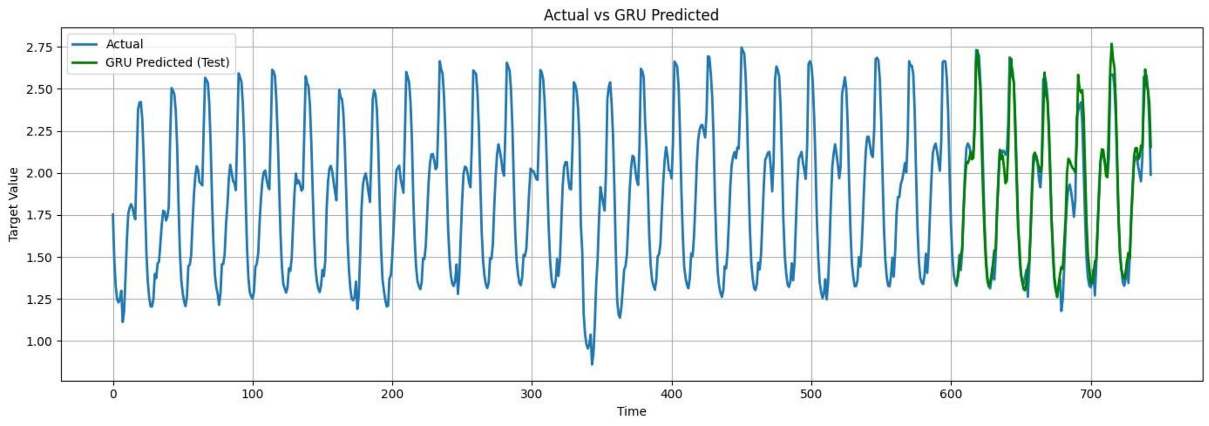
- x_t is the input at time t
- h_t is the hidden state at time t
- z_t is the update gate controlling memory retention
- r_t is the reset gate controlling past information relevance
- $\tilde{h}_t$ is the candidate hidden state
- σ is the sigmoid activation function
- $\odot$ represents element-wise multiplication • W_*, U_* are weight matrices, and b_* are biases

5.9.2 Training Loss:

During training, the GRU model showed a consistent decline in loss values:

- Epoch 100: Loss = 0.0091
- Epoch 200: Loss = 0.0037
- Epoch 300: Loss = 0.0029
- Epoch 400: Loss = 0.0021
- Epoch 500: Loss = 0.0013

This decreasing trend in training loss indicates the model is effectively learning the temporal patterns in the data over epochs without overfitting.



5.9.3 Evaluation Metrics:

- RMSE (Root Mean Squared Error): 0.11
  Indicates low average prediction error magnitude; the model has learned general structure well.
- MAPE (Mean Absolute Percentage Error): 4.19%
  A low percentage error suggests good relative prediction accuracy, especially important when the magnitude of consumption varies.
- R^2 Score: 0.9400
  Shows that the GRU model explains 94% of the variance in the target variable, a strong performance indicating effective learning of temporal dependencies.

Overall, the GRU model performs comparably to the LSTM while being computationally less expensive, making it a suitable alternative for sequence modeling tasks.

5.10 Extreme Gradient Boosting (XGBoost)

XGBoost is a powerful and scalable implementation of gradient-boosted decision trees. It is widely used in regression and classification tasks due to its regularization capabilities, efficient handling of missing data, and ability to model complex nonlinear relationships.

5.10.1 Model Configuration:

XGBoost is a powerful and scalable implementation of gradient-boosted decision trees. It is widely used in classification and regression problems due to its regularization capabilities, efficient handling of missing data, and ability to model complex nonlinear relationships.

XGBoost for Time Series Forecasting: Although XGBoost is traditionally applied to non-sequential tasks like classification and regression, it can be adapted for time series forecasting by transforming the sequential data into a supervised learning problem. This involves:

- Creating lag-based features (e.g., consumption at previous time steps).
- Including rolling window statistics such as moving averages or standard deviations.
- Incorporating exogenous variables (e.g., weather data, date-time features).

By training the model on such engineered features, XGBoost learns to map past patterns to future values, making it suitable for time-dependent predictions.

- n_estimators = 50
- max_depth = 6
- learning_rate = 0.05

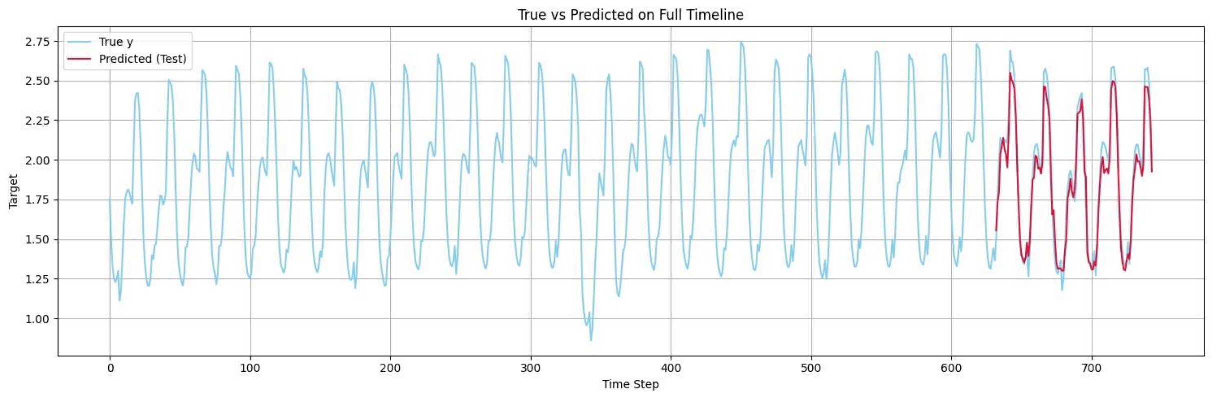
5.10.2 Cross-Validation Metrics:

The model was validated using a time series split strategy. The metrics across the three folds are:

- Fold 1: RMSE = 0.2731, MAPE = 14.28%, R^2 = 0.6782
- Fold 2: RMSE = 0.1246, MAPE = 5.07%, R^2 = 0.9241
- Fold 3: RMSE = 0.1488, MAPE = 4.90%, R^2 = 0.8910

5.10.3 Test Set Performance:

- RMSE: 0.0873
- MAPE: 3.79%
- R^2 Score: 0.9568



5.10.4 Analysis:

The XGBoost model shows a strong generalization ability, as evidenced by a low RMSE and MAPE on the test set. While Fold 1 has a relatively higher error, Folds 2 and 3 demonstrate robust learning performance. The final test R^2 score of 0.9568 implies that the model captures the variance in the target variable very well. The low MAPE of 3.79% highlights its accuracy even when the consumption levels vary. This suggests that XGBoost is particularly effective in modeling the underlying trends in the data and makes it a competitive model for this forecasting task.

5.11 Random Forest Regression

5.11.1 Model Description:

Random Forest is an ensemble learning method that constructs a multitude of decision trees and aggregates their predictions to improve generalization and reduce overfitting. Although it is commonly used for non-sequential regression and classification tasks, Random Forest can be applied to time series forecasting by transforming the series into a supervised problem using lagged features and exogenous variables.

5.11.2 Feature Engineering for Time Series:

To adapt Random Forest for forecasting, the input data was framed as follows:

- Lag Features: Power consumption values from previous time steps (e.g., past 48 hours).
- Exogenous Variables: Weather metrics and other relevant covariates aligned with each time step.
- Sliding Window: Each training sample uses a fixed-length window of past observations to predict the next value.

5.11.3 Hyperparameters:

```
rf_params = {
    'n_estimators': 100,
    'min_samples_split': 2,
    'max_depth': None,
    'random_state': 0
}
```

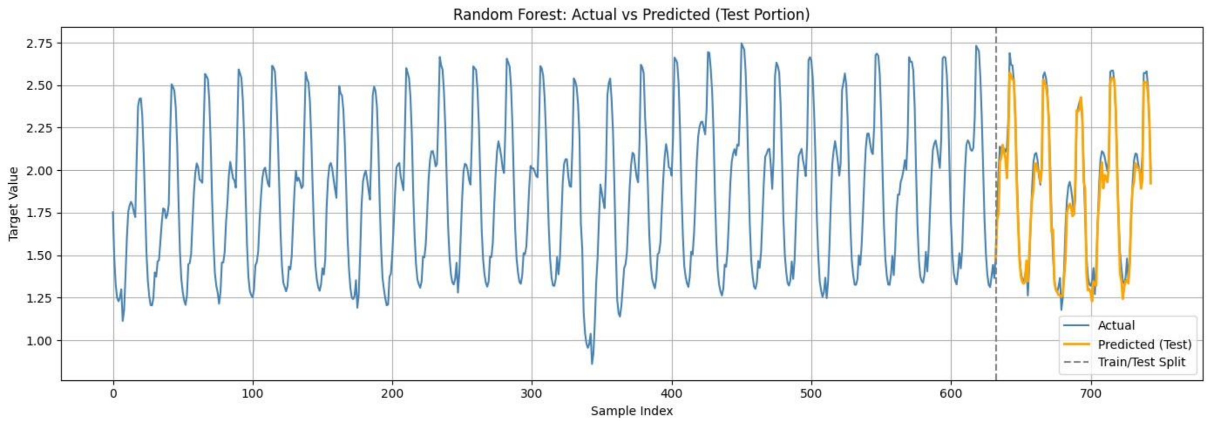
5.11.4 Cross-Validation Metrics:

Using a time-series-aware split strategy with three folds, the Random Forest model achieved:

- Fold 1: RMSE = 0.1412, MAPE = 6.27%, $R^2 = 0.8897$
- Fold 2: RMSE = 0.1954, MAPE = 8.35%, $R^2 = 0.8237$
- Fold 3: RMSE = 0.1386, MAPE = 4.44%, $R^2 = 0.9069$

5.11.5 Test Set Performance:

- RMSE: 0.0808
- MAPE: 3.51%
- R^2: 0.9630



5.11.6 Analysis

The Random Forest model demonstrates robust performance on the test set, achieving a low RMSE of 0.0808 and MAPE of 3.51%. Its high R^2 of 0.9630 indicates that it explains over 96% of the variance in the unseen data. While Fold 2 showed a slightly higher error (RMSE = 0.1954), overall cross-validation results are consistent, suggesting the model generalizes well. The ensemble nature of Random Forest effectively captures both linear and nonlinear patterns in the time series when provided with appropriately engineered lag and exogenous features.

By converting the time series forecasting problem into a supervised learning format, Random Forest becomes a strong predictive tool. Its ease of implementation, resilience to overfitting, and competitive accuracy make it a valuable component of the forecasting model suite for Tetuan city's power consumption.

6 Results and Observations

In this study, we compared a range of statistical and machine learning approaches to forecast Zone 1 power consumption in Tetuan city. Table 2 summarizes the test-set performance (RMSE, MAPE, and R^2) of each model:

6.1 Model Performance

Model	RMSE	MAPE	R^2
ARIMA (24,0,24)	0.11	4.01%	0.9375
SARIMA (2,1,2)(1,2,2,24)	0.12	4.57%	0.9199
SARIMAX	0.11	5.26%	0.9354
RNN	0.1534	6.06%	0.8726
LSTM	0.1122	4.83%	0.9318
GRU	0.11	4.19%	0.9400
XGBoost	0.0873	3.79%	0.9568
Random Forest	0.0808	3.51%	0.9630

Table 2: Comparison of Test Metrics for Different Models

1. Statistical vs. Machine Learning Models: While ARIMA and SARIMA provided solid baselines (explaining over 90% of variance), incorporating exogenous weather features (SARIMAX) did not substantially outperform the simpler ARIMA specification, suggesting that the hourly seasonality and lag information alone capture most of the signal in Zone1 consumption.
2. Deep Learning Models: The RNN family showed progressively better performance as the architecture became more sophisticated: LSTM outperformed Simple RNN, and GRU improved marginally over LSTM (achieving $R^2 = 0.94$). However, all deep models required extensive sliding-window preprocessing and proved more sensitive to hyperparameter settings and training stability.
3. Ensembles Excel: Tree-based ensembles (XGBoost and Random Forest) achieved the lowest errors overall. Random Forest slightly edged out XGBoost, with RMSE = 0.0808 and MAPE = 3.51%, capturing both linear and nonlinear dependencies effectively—even without explicit temporal differencing or seasonal parameters.
4. Computational Considerations: Deep learning models demanded significant hyperparameter tuning (e.g., hidden sizes, learning rates, batch sizes) and longer training times on GPU–equipped hardware. Due to resource constraints, extensive

model architecture searches and large-scale grid searches were out of scope. In contrast, tree-based models trained quickly on CPU and required only modest tuning.

Future Work

- Advanced Deep Architectures: Investigate attention-based transformers or temporal convolutional networks (TCNs) that can capture long-range dependencies more efficiently than RNNs/LSTMs/GRUs.
- Ensemble of Heterogeneous Models: Combine statistical (SARIMA) and machine-learning forecasts via stacking or blending to potentially reduce bias and variance.
- Feature Engineering: Incorporate additional exogenous variables such as calendar effects (holidays, weekdays vs. weekends), special events, and real-time environmental indicators.

Overall, while classical methods remain strong baselines for hourly consumption forecasting, modern ensemble and deep learning approaches offer complementary strengths. Future efforts that integrate these paradigms, backed by robust compute resources and richer feature sets, are likely to yield further gains in forecasting accuracy and reliability.

References

[1] Mouad Enna. *Time Series Splitting Techniques: Ensuring Accurate Model Validation*, Medium, 2024. https://medium.com/@mouadenna/ time-series-splitting-techniques-ensuring-accurate-model-validation-5a3146db3088

[2] Konrad Banachewicz. *10 Validation Methods for Time Series*, Kaggle, 2024. https://www.kaggle.com/code/konradb/ts-10-validation-methods-for-time-series

[3] Analytics Vidhya. *XGBoost for Time Series Forecasting*, 2024. https://www.analyticsvidhya.com/blog/2024/01/xgboost-for-time-series-forecasting/

[4] Analytics Vidhya. *Random Forest for Time Series Forecasting*, 2021. https://www.analyticsvidhya.com/blog/2021/06/ random-forest-for-time-series-forecasting/

[5] Marco Peixoto. *Time Series Forecasting in Python*, GitHub Repository. https://github.com/marcopeix/TimeSeriesForecastingInPython

[6] GeeksforGeeks. *Time Series Analysis and Forecasting*. https://www.geeksforgeeks.org/time-series-analysis-and-forecasting/

[7] *Power Consumption of Tetouan City Dataset*. UCI Machine Learning Repository. Available at: https://archive.ics.uci.edu/dataset/849/power+consumption+of+tetouan+city