Modeling and Predicting Household Energy Consumption

Tapas Kumar Maiti
tapas_kumar@dau.ac.in
Mentor, Dhirubhai Ambani University
Gandhinagar, India

Chandan Pandit
202418043@dau.ac.in
Dhirubhai Ambani University
Gandhinagar, India

Vivek Gautam
202418053@dau.ac.in
Dhirubhai Ambani University
Gandhinagar, India

1 Project Description

This project uses the `pycity_base` framework to generate synthetic household electricity consumption data for a single residential building. The tool models building properties, weather conditions, and occupancy behavior to create realistic hourly load values for the year 2024, resulting in a dataset of 8760 data points.

Based on the generated household demand data, time series forecasting techniques are applied to predict future electricity consumption. The dataset is further enriched with preprocessing steps such as lag features, rolling statistics, and calendar-based attributes to improve model performance. This combination of synthetic data generation and time series forecasting provides a structured approach for analyzing and predicting residential energy usage patterns.

2 Project Objective

The objective of this study is to model, analyze, and forecast household electricity consumption using synthetic consumption profiles generated through the `pycity_base` framework. This approach enables early experimentation with data preprocessing, feature engineering, and forecasting model development before scaling the analysis to multi-building or district-level simulations using `pycity_calc`.

3 Dataset Description

The generated synthetic dataset contains the following variables:

Column	Description
DATE	Calendar date (e.g., 2024-01-01)
HOUR	Hour of the day (0–23)
DEMAND (kW)	Simulated electric demand for the building

Table 1: Synthetic Dataset Variables (Generated Using `pycity_base`)

	date	hour	Demand
0	01-01-2024	0	355.341456
1	01-01-2024	1	326.427360
2	01-01-2024	2	302.551524
3	01-01-2024	3	279.794880
4	01-01-2024	4	257.784348

Figure 1: Sample of the generated dataset

4 Data Generation Framework

The pyCity ecosystem is an open-source urban energy modelling framework designed for simulating, analyzing, and forecasting building and district-level energy consumption. It provides modular tools for generating synthetic electricity, heating, and cooling consumption profiles, making it highly useful for smart grid research, urban energy studies, and forecasting applications.

4.1 pycity_base

`pycity_base` is the core library used for building-level and district-level modelling. It includes:

- Building electricity consumption models,
- Electricity consumption generation on an hourly or sub-hourly basis.

In this project, `pycity_base` was used to generate a synthetic hourly electricity consumption dataset for a single building for the year 2024, providing the foundation for all forecasting experiments.

5 Exploratory Data Analysis (EDA)

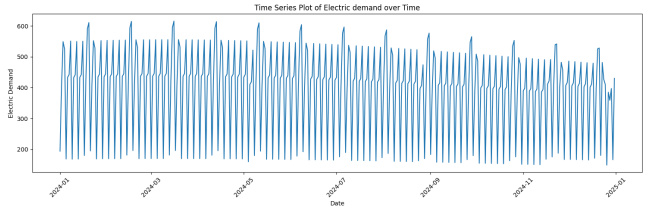

Figure 2: Daily electricity consumption pattern for the building. The load shows regular daily variation without major spikes.

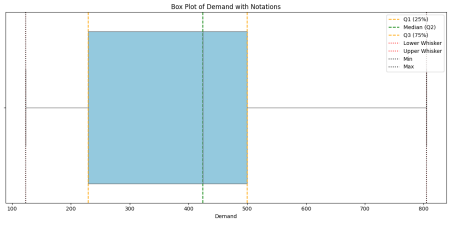

Figure 3: Box plot of electricity consumption. The distribution is right-skewed with no outliers.

5.1 Time Series Decomposition:

Decomposition is the process of separating a time series into its constituent components: *Trend*, *Seasonal*, and *Residual*. This technique helps understand the underlying patterns in the data and prepares it for forecasting or anomaly detection.

- **Actual:** The original observed time series data.
- **Trend:** The long-term direction or movement in the data, indicating increasing or decreasing patterns over time.
- **Seasonal:** Repeating short-term cycles or periodic fluctuations (e.g., daily, weekly, yearly).
- **Residual:** The random noise or irregular component left after removing trend and seasonality.

By analyzing these components individually, we can select appropriate forecasting models (e.g., SARIMA for seasonal data, Prophet for additive seasonality and holidays) and better interpret the behavior of the system being monitored.

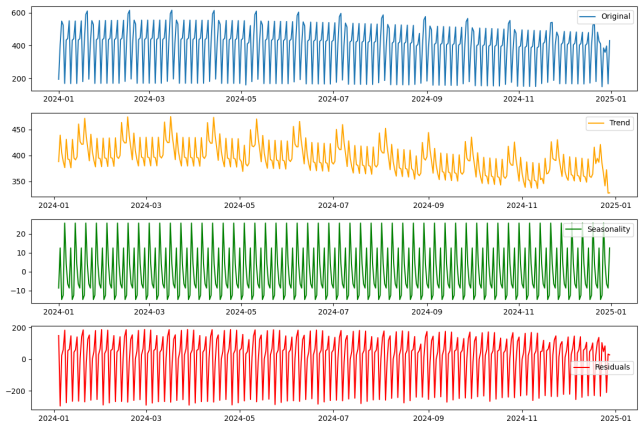

Figure 4: Time series decomposition of electricity consumption showing clear trend and weekly seasonality.

5.2 Augmented Dickey-Fuller (ADF) Test

The Augmented Dickey-Fuller (ADF) test is used to determine whether a time series is stationary. A stationary series has constant statistical properties such as mean and variance over time, which is essential for building reliable forecasting models.

Why use the ADF test?

- To test the null hypothesis that the series contains a unit root (i.e., it is non-stationary).
- To assess whether differencing or transformations are required.
- To guide the selection of appropriate time series models such as AR, ARIMA, or RNN-based models.

ADF Test Results:

- ADF Statistic: **-18.413410**
- p-value: **0.000000**
- Critical Values:
  - 1%: -3.431
  - 5%: -2.862
  - 10%: -2.567
- **Conclusion:** Since the p-value is effectively zero and the ADF statistic is far below all critical values, we reject the null hypothesis (H_0). Therefore, the data is **stationary**.

6 Naive Forecasting

Naive Forecasting

The Naive Forecasting model is one of the simplest time series prediction methods. It assumes that the next value in the series is equal to the most recently observed value. Despite its simplicity, it is commonly used as a baseline for comparing more advanced forecasting models.

Working Principle:

$$\hat{y}_{t+1} = y_t$$

This means that the predicted value at time $t + 1$ is simply the last available observation at time t.

Limitation: The Naive method does not capture any trend, seasonality, or temporal patterns in the data. Therefore, its accuracy is limited for datasets with strong daily or seasonal structure.

Evaluation: The Naive model was evaluated on the train, validation, and test sets. While it provides a straightforward baseline, the moderate R^2 values indicate that more advanced models are needed for accurate forecasting.

Table 2: Naive Forecasting Evaluation Metrics

Dataset	MAE	MSE	RMSE	R^2
Train	49.48	9092.76	95.36	0.7022
Validation	45.47	7818.94	88.42	0.6971
Test	43.16	6688.65	81.78	0.7074

Naive Forecast for the Next 24 Hours

The Naive model assumes that all future values remain equal to the last observed consumption value.

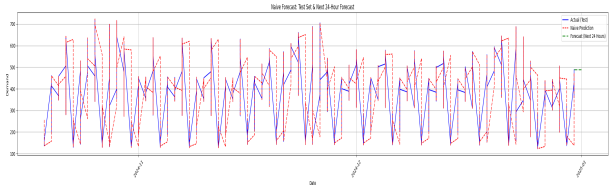

Figure 5: Naive forecast vs actual consumption

Date & Time	Predicted Consumption (kW)
2024-12-31 01:00:00	488.3438
2024-12-31 02:00:00	488.3438
2024-12-31 03:00:00	488.3438
2024-12-31 04:00:00	488.3438
2024-12-31 05:00:00	488.3438
⋮	⋮
2024-12-31 20:00:00	488.3438
2024-12-31 21:00:00	488.3438
2024-12-31 22:00:00	488.3438
2024-12-31 23:00:00	488.3438
2025-01-01 00:00:00	488.3438

Table 3: Naive forecast for the next 24 hours (first and last 5 records).

6.1 Autocorrelation and Partial Autocorrelation

6.1.1 Definition of ACF and PACF. **Autocorrelation Function (ACF)** The ACF measures the correlation between the time series and its own lagged values.

$$\text{ACF}(k) = \frac{\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2} \tag{1}$$

Partial Autocorrelation Function (PACF) The PACF measures the correlation between the series and its lagged values after removing the influence of all shorter lags.

$$\text{PACF}(k) = \phi_{kk} \tag{2}$$

6.1.2 Interpretation of ACF and PACF Plots. **ACF Plot:**

- The ACF shows strong positive spikes at regular intervals (every 4 lags), indicating a clear **seasonal pattern with period 4**.
- Significant autocorrelation appears at lags 1, 4, 8, 12, 16, and so on, confirming repeating demand cycles.
- Between these seasonal peaks, the autocorrelation values are negative, showing an oscillatory or cyclic behavior in the demand pattern.
- The overall slow decay of the ACF indicates that current demand is influenced by long-term past observations, suggesting strong persistence in the series.

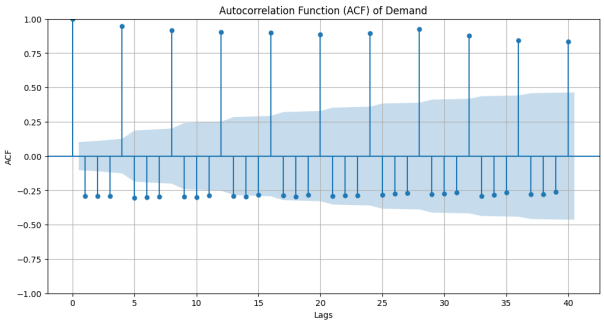

Figure 6: Autocorrelation Function (ACF) Plot

PACF Plot:

- The PACF shows a very strong and significant spike at lag 1, indicating that the current demand value is strongly influenced by its immediate previous value. This suggests the presence of a strong **AR(1)** component.
- Another significant spike appears at lag 4, aligning with the seasonal pattern observed in the ACF. This indicates a **seasonal AR component with period 4**.
- Beyond lag 4, the partial autocorrelations fall mostly within the confidence interval, suggesting no additional strong AR terms are required.
- The drop to negative values at intermediate lags indicates short-term fluctuations but no consistent higher-order AR structure.

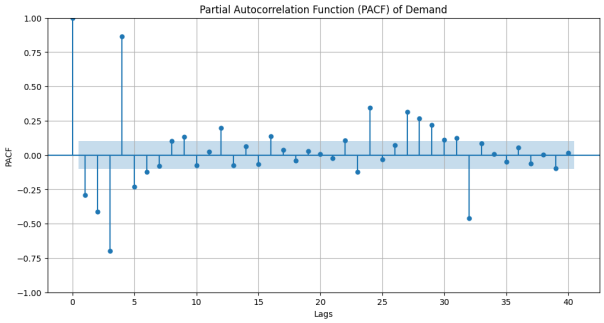

Figure 7: Partial Autocorrelation Function (PACF) Plot

7 AR(1) Model

The AR(1) model assumes that the current consumption value depends on the previous value. It is simple, fast, and works well when strong autocorrelation is present.

$$\hat{y}_t = \phi y_{t-1} + \epsilon_t$$

Strengths:

- Captures short-term relationships.
- Computationally efficient.

Limitations:

- Cannot model seasonal or long-term patterns.

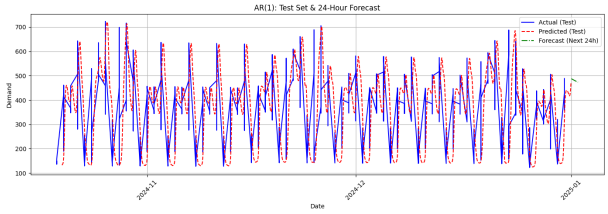

Figure 8: AR(1) forecast vs actual consumption (next 24 hours).

Model Evaluation

The AR(1) model achieved low error and high R^2 0.9859 on all splits.

Table 4: AR(1) Model Evaluation

Dataset	MAE	RMSE	R^2
Train	16.4393	21.0423	0.9855
Validation	15.1612	19.4485	0.9852
Test	14.0015	17.9624	0.9859

AR(1) Forecast for the Next 24 Hours

Date & Time	Predicted Consumption (kW)
2025-01-01 00:00:00	487.6564
2025-01-01 01:00:00	486.9741
2025-01-01 02:00:00	486.2968
2025-01-01 03:00:00	485.6245
2025-01-01 04:00:00	484.9572
⋮	⋮
2025-01-01 19:00:00	475.5201
2025-01-01 20:00:00	474.9275
2025-01-01 21:00:00	474.3393
2025-01-01 22:00:00	473.7555
2025-01-01 23:00:00	473.1760

Table: AR(1) Forecast for the Next 24 Hours

8 Vanilla RNN Model

A Vanilla RNN is designed for sequential data, making it suitable for load forecasting. It maintains a hidden state that stores past information and learns short-term temporal dependencies.

How It Works

At each timestep, the hidden state is updated as:

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

The final prediction is:

$$\hat{y}_t = W_{hy}h_t + b_y$$

Strength: Learns short-term sequence patterns. **Limitation:** Struggles with long-term memory (vanishing gradients).

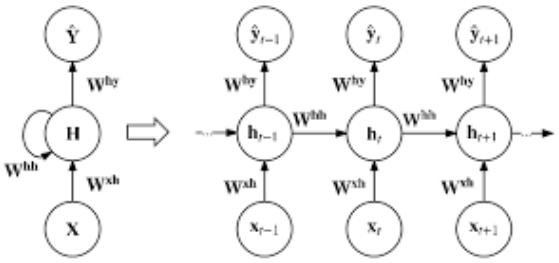

Figure 9: Vanilla RNN Architecture for Load Forecasting

Best Hyperparameters (Keras Tuner)

- Units: 192 (Layer 1), 128 (Layer 2)
- Dropout: 0.2 (L1), 0.4 (L2)
- L2 Regularization: 0.005
- Learning Rate: 0.001

Model Evaluation

The RNN achieved low prediction error and high R^2 scores.

Table 5: Vanilla RNN Model Evaluation

Dataset	MAE	MSE	R^2
Train	14.4468	378.1258	0.9875
Validation	13.2164	300.7877	0.9882
Test	13.5974	300.7407	0.9870

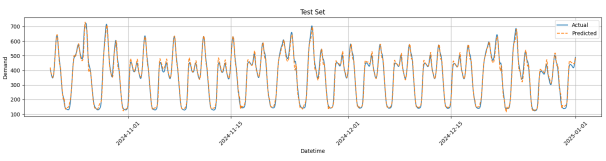

Figure 10: RNN Forecast vs Actual Demand (Next 24 Hours)

Forecast for the Next 24 Hours

Date & Time	Predicted Demand (kW)
2025-01-01 00:00:00	485.19
2025-01-01 01:00:00	503.31
2025-01-01 02:00:00	513.43
2025-01-01 03:00:00	511.85
2025-01-01 04:00:00	505.08
⋮	⋮
2025-01-01 19:00:00	449.62
2025-01-01 20:00:00	462.29
2025-01-01 21:00:00	472.20
2025-01-01 22:00:00	476.38
2025-01-01 23:00:00	477.96

Table: Vanilla RNN Forecast (Next 24 Hours)

9 LSTM Model

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network designed to learn long-term dependencies in sequence data. They process input data in a single forward direction, making them suitable for time-series forecasting tasks such as electric load demand. LSTMs are effective at capturing patterns and trends over time, helping improve prediction accuracy in load forecasting applications.

How LSTM Works Internally

Each LSTM unit uses gating mechanisms to control the flow of information:

- **Forget Gate:**

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

- **Input Gate:**

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \qquad \tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$$

- **Cell State Update:**

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

- **Output Gate:**

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \qquad h_t = o_t \cdot \tanh(C_t)$$

Strength: Effectively captures long-term temporal dependencies in time-series data.
Limitation: Higher computational cost compared to simpler recurrent architectures like GRUs.

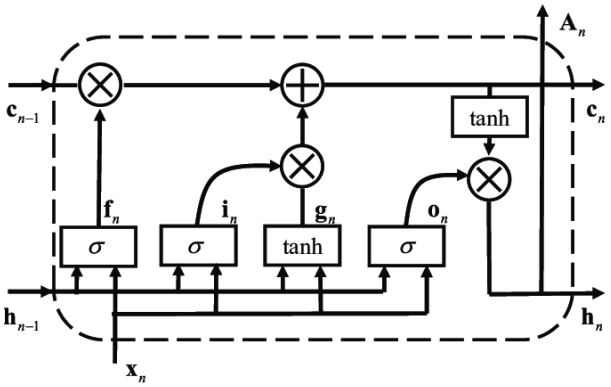

Figure 11: LSTM Architecture for Load Forecasting

9.1 Best Hyperparameters

The optimal hyperparameters identified using Keras Tuner are:

- **Units in LSTM Layer 1:** 128
- **L2 Regularization:** 0.001
- **Dropout (Layer 1):** 0.3
- **Units in LSTM Layer 2:** 64
- **Dropout (Layer 2):** 0.2
- **Learning Rate:** 0.0005

9.2 Performance Metrics

The LSTM model achieved strong performance with very high R^2 0.9818 scores on all data splits.

Table 6: LSTM Model Evaluation

Dataset	MAE	MSE	R^2
Train	10.6868	192.3571	0.9937
Validation	12.1057	236.4821	0.9908
Test	15.8230	421.5775	0.9818

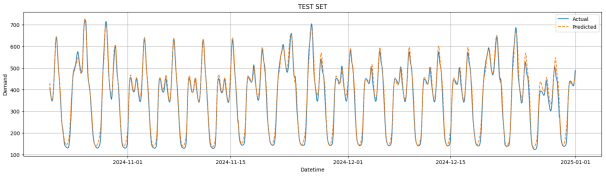

Figure 12: LSTM Forecast vs Actual Demand

9.3 Forecast for the Next 24 Hours

Table 7: LSTM Forecast (Next 24 Hours)

Date & Time	Predicted Demand (kW)
2025-01-01 00:00:00	471.04
2025-01-01 01:00:00	478.75
2025-01-01 02:00:00	485.22
2025-01-01 03:00:00	490.43
2025-01-01 04:00:00	494.44
⋮	⋮
2025-01-01 19:00:00	594.99
2025-01-01 20:00:00	610.42
2025-01-01 21:00:00	625.21
2025-01-01 22:00:00	638.83
2025-01-01 23:00:00	650.95

10 GRU Model

Gated Recurrent Unit (GRU) networks are a lightweight type of RNN designed to model sequential dependencies efficiently. They use fewer parameters than LSTMs while maintaining strong performance on time-series forecasting tasks such as electricity consumption.

How GRU Works

A GRU contains two gates controlling information flow:

- **Update Gate:** $z_t = \sigma(W_z[h_{t-1}, x_t] + b_z)$ — controls how much past information is carried forward.
- **Reset Gate:** $r_t = \sigma(W_r[h_{t-1}, x_t] + b_r)$ — decides how much previous information to forget.
- **Candidate Activation:** $\tilde{h}_t = \tanh(W_h[r_t * h_{t-1}, x_t] + b_h)$
- **Final State:** $h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$

Strength: Efficient and captures sequential patterns. **Limitation:** Slightly less expressive than LSTM for very long-term dependencies.

Best Hyperparameters

- **Units in LSTM Layer 1:** 128
- **L2 Regularization:** 0.001
- **Dropout (Layer 1):** 0.3
- **Units in LSTM Layer 2:** 64
- **Dropout (Layer 2):** 0.2
- **Learning Rate:** 0.0005

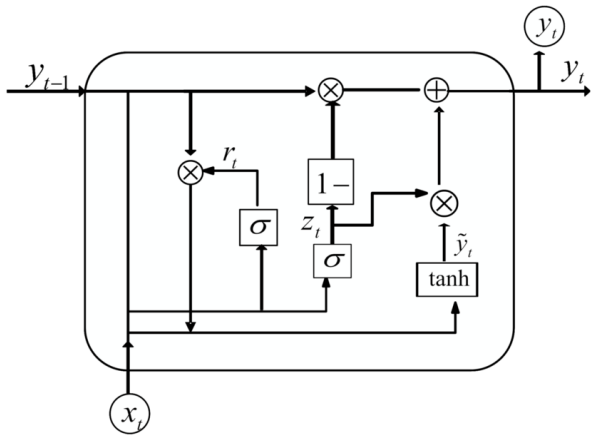

Figure 13: GRU Architecture for Load Forecasting

Model Performance

The GRU achieved excellent performance across all splits ($R^2$0.995).

Table 8: GRU Model Evaluation

Dataset	MAE	MSE	R^2
Train	8.4190	124.0355	0.9959
Validation	8.4652	112.2594	0.9956
Test	8.5926	112.9609	0.9951

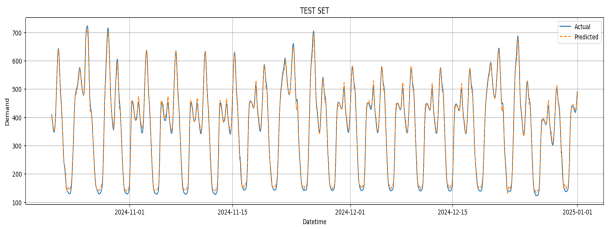

Figure 14: GRU Forecast vs Actual Consumption

Forecast for the Next 24 Hours

Date & Time	Predicted Consumption (kW)
2025-01-01 00:00:00	491.91
2025-01-01 01:00:00	507.70
2025-01-01 02:00:00	516.66
2025-01-01 03:00:00	517.34
2025-01-01 04:00:00	509.53
⋮	⋮
2025-01-01 19:00:00	473.57
2025-01-01 20:00:00	490.47
2025-01-01 21:00:00	503.08
2025-01-01 22:00:00	510.20
2025-01-01 23:00:00	511.12

Table: GRU Forecast (Next 24 Hours)

11 Conclusion

This project demonstrated an end-to-end workflow for electricity consumption forecasting using a synthetic dataset generated with `pycity_base`. The dataset contained hourly consumption values for 2024, providing a structured foundation for analysis and model development. EDA revealed clear daily and weekly seasonal patterns, while the box plot confirmed a stable, moderately right-skewed distribution with no outliers. Time series decomposition showed strong weekly seasonality and mid-year fluctuations, and the series was stationary, suitable for autoregressive modeling.

Autocorrelation and partial autocorrelation analyses indicated strong short-term dependencies and repeating cycles, supporting the use of both statistical and deep learning models. The Naive model served as a baseline but had limited predictive power. The AR(1) model improved accuracy significantly, with $R^2 > 0.98$ across all splits. Among deep learning models, Vanilla RNN, LSTM, and GRU captured nonlinear temporal patterns and seasonality effectively, with the GRU model achieving the highest accuracy ($R^2 > 0.995$).

Overall, simple autoregressive models can perform well on stationary consumption data, but GRU and other recurrent neural networks provide superior accuracy and robustness. This workflow lays the foundation for future work, including multi-building forecasting, integration of weather or environmental features, and district-level energy management using `pycity_calc`.

12 Future Scope

This project initially focused on forecasting electricity consumption for a single building using synthetic data from `pycity_base`. Future work aims to generate data for multiple buildings or entire cities and perform electric load forecasting at larger scales, improving accuracy, scalability, and practical relevance.

12.1 1. City-Scale Synthetic Data Generation

Using the full pyCity ecosystem (pycity_base, pycity_calc, pycity_street), it is possible to generate detailed consumption profiles for multiple buildings or entire districts/cities:

- Extract building footprints and street networks from OpenStreetMap,
- Assign building types, occupancy, and appliance models,
- Simulate hourly loads for all buildings for a full year,
- Aggregate data for forecasting at building, feeder, or city level.

12.2 4. Handling Missing or Incomplete Data

Synthetic data can supplement real datasets that often contain missing values or sensor noise, providing:

- Additional training data for models,
- Benchmarks for comparison and validation,
- Filler for corrupted or incomplete periods,
- Long sequences required for deep learning models.

12.3 5. Multi-Building and City-Level Forecasting

The workflow can scale from single buildings to:

- Multi-building or residential colony-level forecasting,
- Feeder-level or district-level aggregated forecasting,
- City-wide forecasting using hierarchical modeling with pyCity.

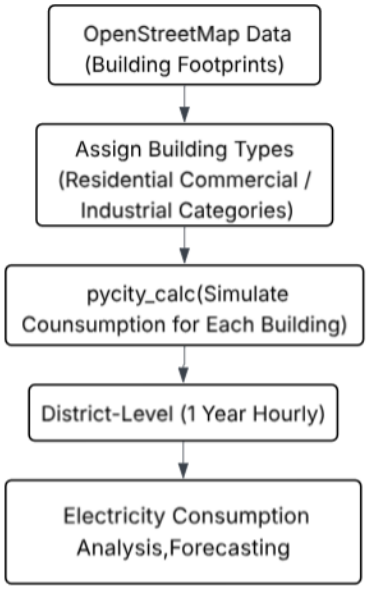

Figure 15: Workflow for District-Level electricity consumption Simulation Using pycity_calc

12.4 Summary

Future work will expand from single-building forecasts to urban-scale electricity consumption modeling. Generating synthetic data for multiple buildings or entire cities and combining it with real measurements will enhance forecasting accuracy, robustness, and practical applicability for smart grids and urban energy planning.

References

[1] I.Yazici et al., "Deep-learning-based short-term electricity load forecasting," *Applied Energy*, 2022. https://www.sciencedirect.com/science/article/abs/pii/S0952197621004516 :contentReference[oaicite:0]index=0

[2] Q.Hua, Z.Fan, W.Mu, J.Cui, R.Xing, H.Liu J.Gao, "Short-Term Power Load Forecasting Using CNN-GRU with Attention Mechanism," *Energies*, 2024. https://www.mdpi.com/1996-1073/18/1/106 :contentReference[oaicite:1]index=1

[3] A.Pai co-authors, "Enhanced household energy consumption forecasting using multivariate LSTM with weather data integration," 2025. https://doi.org/10.1016/j.rineng.2025.106512 :contentReference[oaicite:2]index=2

[4] S.Muzaffar, M.E.El-Hawary, "Short-Term Load Forecasts Using LSTM Networks," 2019. https://www.sciencedirect.com/science/article/pii/S1876610219310008 :contentReference[oaicite:3]index=3

[5] Z.Chen et al., "Load Forecasting Based on LSTM Neural Network," 2021. https://link.springer.com/article/10.1007/s42835-021-00768-8 :contentReference[oaicite:4]index=4

[6] H.Qiu et al., "Short-Term Electricity Load Forecasting: A Comprehensive Review," 2025. https://www.mdpi.com/2227-7390/13/5/813 :contentReference[oaicite:5]index=5

[7] N.Mounir et al., "Short-Term Electric Load Forecasting Based on EMD and BI-LSTM," 2023. https://www.sciencedirect.com/science/article/abs/pii/S0378778823002529 :contentReference[oaicite:6]index=6

[8] Q.Zhang et al., "A novel short-term electricity net load forecasting model based on GCN-Transformer," 2023. https://www.sciencedirect.com/science/article/pii/S2352484723000604 :contentReference[oaicite:7]index=7

[9] "Forecasting: Principles and Practice," 2nd (free) edition — classic time-series forecasting book (online). https://otexts.com/fpp2/ :contentReference[oaicite:8]index=8

[10] C.Deb, M.Sinha co-authors, "A review on time series forecasting techniques for building energy consumption," 2017. https://www.sciencedirect.com/science/article/abs/pii/S1364032117303155 :contentReference[oaicite:9]index=9

[11] M.Abumohsen et al., "Electrical Load Forecasting Using LSTM, GRU and RNN Algorithms," 2023. https://www.mdpi.com/1996-1073/16/5/2283 :contentReference[oaicite:10]index=10

[12] I.Ullah et al., "Multi-horizon short-term load forecasting using hybrid LSTM architectures," 2023. https://peerj.com/articles/cs-1487/ :contentReference[oaicite:11]index=11

[13] "Load forecasting using a hybrid Prophet+LSTM model," 2022. https://www.sciencedirect.com/science/article/pii/S2352484721015067 :contentReference[oaicite:12]index=12

[14] T.Ouyang, Y.He, H.Li, Z.Sun S.Baek, "A Deep Learning Framework for Short-term Power Load Forecasting," 2017. https://arxiv.org/abs/1711.11519 :contentReference[oaicite:13]index=13

[15] F.Bayram, B.S.Ahmed, P.Aupke co-authors, "DA-LSTM: A Dynamic Drift-Adaptive Learning Framework for Load Forecasting," 2023. https://arxiv.org/abs/2305.08767 :contentReference[oaicite:14]index=14

[16] Y.Zhou, A.Sukumaran Nair, D.Ganger, A.Tripathi, C.Baone, H.Zhu, "Appliance-level short-term load forecasting via RNN," 2021. https://arxiv.org/abs/2111.11998 :contentReference[oaicite:15]index=15

[17] DhawaniShah ManishkumarThaker, "A Review of Time Series Forecasting Methods," 2024. https://www.ijrar.org/viewfull.php?&p_id=IJRAR24B1376 :contentReference[oaicite:16]index=16

[18] S.Salehimehr H.Niknam, "Short-term load forecasting in smart grids using AI methods," 2022. https://digital-library.theiet.org/doi/full/10.1049/tje2.12183 :contentReference[oaicite:17]index=17

[19] L.Zhang et al., "LSTM-based short-term electrical load forecasting and simulation," 2020. https://www.e3s-conferences.org/articles/e3sconf/pdf/2020/42/e3sconf_cpeee2020_01004.pdf :contentReference[oaicite:18]index=18

[20] "Machine Learning for Time Series Forecasting" — book covering time series deep learning methods. https://www.manning.com/books/time-series-forecasting-in-python-book :contentReference[oaicite:19]index=19

[21] B.Oreshkin, G.Dudek P.Pełka, "N-BEATS neural network for mid-term electricity load forecasting," 2020. https://arxiv.org/abs/2009.11961 :contentReference[oaicite:20]index=20

[22] R.Gao, L.Du, P.N.Suganthan, Q.Zhou K.F.Yuen, "Random vector functional link NN based ensemble deep learning for short-term load forecasting," 2021. https://arxiv.org/abs/2107.14385 :contentReference[oaicite:21]index=21

[23] S.Abdul co-authors, "Hybrid Energy Demand Forecasting with CNN-GRU for Smart Grids," 2024. https://www.sciencedirect.com/science/article/pii/S2352484724006346 :contentReference[oaicite:22]index=22

[24] M.P.Raju et al., "IoT-based online load forecasting using machine learning," 2020. https://www.sciencedirect.com/science/article/pii/S1877050920310267 :contentReference[oaicite:23]index=23

[25] A.Almalaq G.Edwards, "Electricity demand forecasting using EMD and LSTM," 2021. https://www.sciencedirect.com/science/article/pii/S2095927321001247 :contentReference[oaicite:24]index=24

[26] C.Deb S.K.Sinha, "Time-Series Forecasting Methods for Building Energy Consumption: Survey," 2017. https://www.sciencedirect.com/science/article/abs/pii/S1364032117303155 :contentReference[oaicite:25]index=25

[27] M.Hasan et al., "Comparative review: Load forecasting techniques for power systems," 2025. https://www.sciencedirect.com/science/article/pii/S2590174525000546 :contentReference[oaicite:26]index=26

[28] "International Journal of Forecasting" — journal page. http://www.forecasters.org/ijf :contentReference[oaicite:27]index=27

[29] "Journal of Forecasting" — journal information page. https://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1099-131X :contentReference[oaicite:28]index=28

[30] T.Bashir et al., "Hybrid Prophet-LSTM method for short-term electricity load forecasting," 2022. https://www.sciencedirect.com/science/article/pii/S2352484721015067 :contentReference[oaicite:29]index=29