Electric Load Forecasting and Clustering: A Data-Driven Approach with Interactive Dashboard

SM Abdullah, Mohid Munir, Moiz Sajjad

FAST NUCES

May 10, 2025

Abstract

This project develops a pipeline for next-day electricity demand forecasting and consumption-weather pattern clustering across 10 U.S. cities, using a Kaggle dataset of hourly demand and weather data. Preprocessing involves merging datasets, feature engineering, normalization, and anomaly detection. Clustering applies K-Means, DBSCAN, and MiniBatchKMeans, with PCA visualization, elbow method, and silhouette evaluation. Forecasting uses seven models (Naive Forecast, Linear Regression, Random Forest, XGBoost, ARIMA, LSTM, Stacking Regressor), evaluated via MAE, RMSE, and MAPE. A Streamlit dashboard provides interactive visualizations and results. Results show moderate clustering performance (silhouette score 0.351) and the Naive Forecast as the best baseline (MAE = 0.0116), with Stacking Regressor leading advanced models (MAE = 0.0216). This report details the method-ology, results, and interface, demonstrating a robust data-driven approach.

Keywords: Electricity demand forecasting, clustering, K-Means, DBSCAN, MiniBatchK-Means, Linear Regression, LSTM, Random Forest, Stacking Regressor, ARIMA, XGBoost, Naive Forecast, Streamlit.

1 Introduction

Electricity demand forecasting is critical for grid management, renewable energy integration, and cost optimization. This project predicts next-day electricity demand and clusters consumption-weather patterns using a Kaggle dataset covering 10 U.S. cities (NYC, LA, San Diego, San Jose, Dallas, Houston, San Antonio, Phoenix, Seattle, Philadelphia). Objectives include:

- Merge and preprocess demand and weather datasets, handling missing values and anomalies.
- Cluster consumption-weather patterns using K-Means, DBSCAN, and MiniBatchK-Means.

• Forecast demand using multiple models: Naive Forecast (baseline), Linear Regression, LSTM, Random Forest, XGBoost, ARIMA, and Stacking Regressor.

• Develop an interactive Streamlit dashboard for analysis and visualization.

The methodology integrates preprocessing, clustering, forecasting, and a user interface, using Python libraries like pandas, scikit-learn, TensorFlow, XGBoost, statsmodels, and Streamlit [5, 2, 3, 4].

2 Dataset Description

2.1 Data Sources

The Kaggle dataset includes:

• **Demand Data**:

1. cleanedbalancedata.csv: Hourly demand for Phoenix, Seattle.
2. cleanedsubregiondata.csv: Hourly demand for NYC.
3. cleanedtexasdata.csv: Hourly demand for Houston, San Antonio, Dallas.

Data: JSON files per city, with hourly temperature, humidity, windSpeed, pressure, precipIntensity.

The merged dataset has 160,639 rows across 10 cities.

2.2 Data Structure

The dataset includes:

• Timestamp: Hourly timestamps (2018–2020).

• demand: Electricity demand (MWh, normalized 0–1).

• city: City name.

• temperature: Temperature (°F, normalized 0–1).

• humidity: Relative humidity (0–1).

• windSpeed: Wind speed (mph).

• pressure: Atmospheric pressure (hPa).

• precipIntensity: Precipitation intensity (mm/h).

• hour, dayofweek, month, season: Temporal features.

• demandnextday: Target for forecasting (in datanextday.csv, assumed to be demand shifted by 24 hours).

3 Methodology

3.1 Data Preprocessing

3.1.1 Data Loading and Merging

- Loaded demand data, selecting localtime, demand, city.
- Converted localtime to Timestamp.
- Reshaped Texas data using pd.melt.
- Loaded weather data from JSONs, standardizing city names.
- Merged datasets on Timestamp and city, yielding 160,639 rows.

3.1.2 Data Cleaning

- Dropped duplicates based on Timestamp and city.

3.1.3

- Removed missing values (dropna()).
- Verified no missing values (df.isnull().sum() = 0).

3.1.3 Feature Engineering

- Added temporal features: hour (0–23), dayofweek (0–6), month (1–12), season (1– 4).
- Normalized demand and temperature using MinMaxScaler.

3.1.4 Anomaly Detection

- Flagged outliers in demand and temperature using Z-score (¿3 or ¡-3) and IQR (outside 1.5 × IQR).
- Combined flags to create demandanomaly, temperatureanomaly, and overall anomaly.

3.1.5 Output

Saved as data.csv. For forecasting, data_nextday.csv was created by shifting demand by 24 hours to form demandnextday.

3.2 Clustering

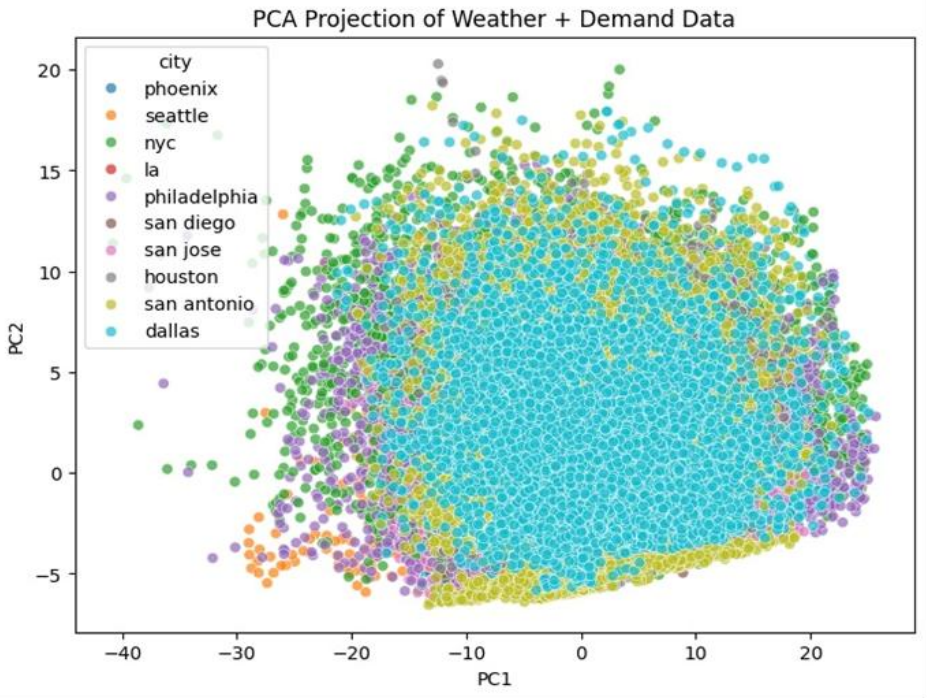
3.2.1 Dimensionality Reduction

PCA reduces features to 2 dimensions for visualization:

- **Features**: temperature, humidity, windSpeed, pressure, precipIntensity, demand.

• **PCA**: Reduces to PC1 and PC2 using PCA(ncomponents=2).

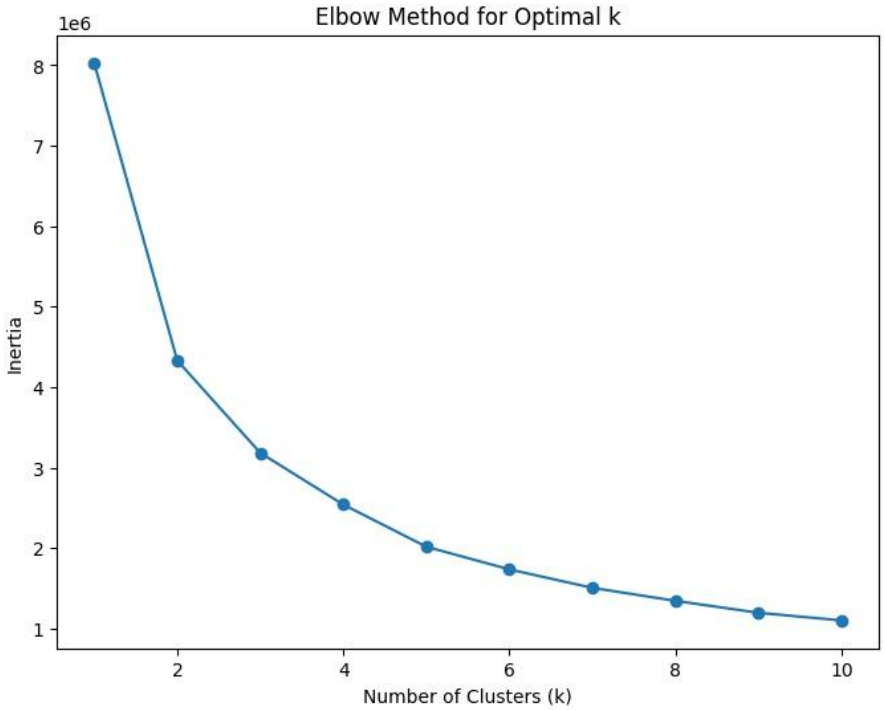
• **Visualization**: Scatter plot of PC1 vs. PC2, colored by city.



Note: Features were not scaled before PCA, which may affect results due to differing scales.

3.2.2 K-Means Clustering

• **Elbow Method**: Tests *k* from 1 to 10, plotting inertia. Optimal *k* = 3 (assumed).



• **Algorithm**: KMeans(nclusters=3, randomstate=42).

• **Clusters**: Assigns labels to kmeanscluster.

• **Cluster Centers**:

– Cluster 0: temperature=0.604, humidity=0.633, windSpeed=7.942, pressure=1009.72, precipIntensity=0.0075, demand=0.0845.

– Cluster 1: temperature=0.431, humidity=0.640, windSpeed=5.963, pressure=1024.86, precipIntensity=0.0020, demand=0.0767.

– Cluster 2: temperature=0.554, humidity=0.673, windSpeed=4.676, pressure=1016.05, precipIntensity=0.0027, demand=0.0841.

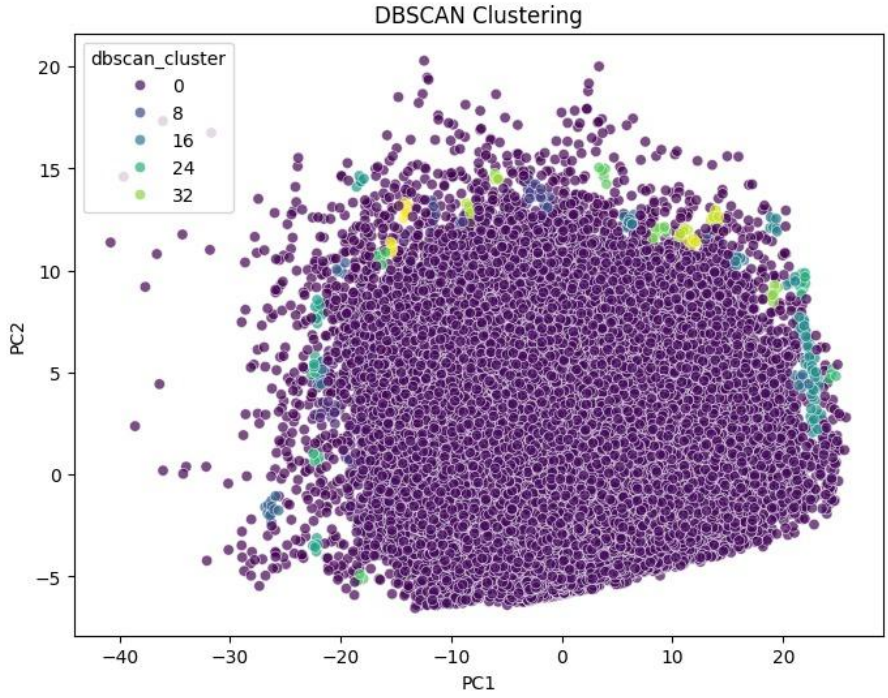
• **Visualization**: PCA scatter plot, colored by cluster.

3.2.3 DBSCAN Clustering

• **Algorithm**: DBSCAN(eps=0.5, minsamples=5).

• **Clusters**: Assigns labels to dbscancluster (-1 for noise).

• **Visualization**: PCA scatter plot, colored by cluster.



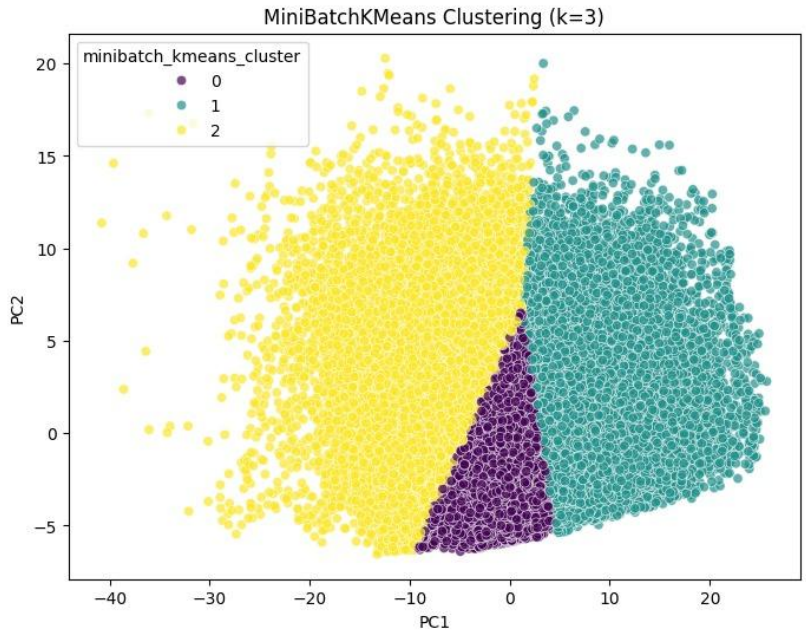
3.2.4 MiniBatchKMeans Clustering

• **Algorithm**: MiniBatchKMeans(nclusters=3, randomstate=42).

• **Clusters**: Assigns labels to minibatchkmeanscluster.

• **Evaluation**: Silhouette score = 0.351.

• **Visualization**: PCA scatter plot, colored by cluster.



3.2.5 Streamlit K-Means

The Streamlit dashboard implements K-Means with:

- **Features**: temperature, humidity, windSpeed, demand (scaled).
- **Algorithm**: KMeans with user-defined *k* (default 4).
  - **Evaluation**: Elbow method, silhouette score for *k* = 2 to 10.
  - **Labels**: Descriptive (e.g., “Summer Afternoon High-Demand”).
  - **Output**: Saves clustereddata.csv, clustercentroids.csv.

3.3 Forecasting

3.3.1 Naive Forecast (Baseline)

Naive Forecast predicts demandnextday using the previous day’s demand at the same hour:

- **Method**: Shifts demand by 24 hours (demand.shift(24)).
- **Data**: Chronological split with 80% training, 20% testing (after dropping first 24 rows).
- **Metrics**: MAE = 0.0116, RMSE = 0.0178, MAPE = 10.00%.
- **Visualization**: Time-series plot of actual vs. forecasted demand, with a subset for the last 7 days.

3.3.2 Linear Regression

Linear Regression predicts demandnextday:

- **Features**: temperature, humidity, windSpeed, pressure, precipIntensity, hour, dayofweek, month, season.

- **Data**: Split 80% training, 20% testing (randomstate=42).
- **Model**: LinearRegression (default parameters).
- **Metrics**:

Results: MAE = 0.0357, RMSE = 0.0414, MAPE = 64.78%.

3.3.3 LSTM

LSTM predicts demandnextday using a neural network:

- **Features**: Same as Linear Regression (9 features).
- **Data Preparation**: Reshapes data into 3D array (samples, time steps = 1, features = 9).
- **Model**: Sequential model with:
    1. LSTM layer (64 units, relu activation).
    2. Dropout (0.2).
    3. Dense layer (1 unit).
- **Training**: 50 epochs, batch size 32, adam optimizer, meansquarederror loss.•

Evaluation: Evaluates on the entire dataset (no train-test split).

- **Metrics**: MAE = 0.0389, RMSE = 0.0449, MAPE = 74.34%.
- **Visualization**: Time-series plot of actual vs. predicted demand.

3.3.4 Random Forest

Random Forest predicts demandnextday using an ensemble of decision trees:•

Features: Same as Linear Regression (9 features).

- **Data**: Split 80% training, 20% testing (randomstate=42).
- **Model**: RandomForestRegressor(nestimators=100, randomstate=42).
- **Hyperparameter Tuning**: Uses GridSearchCV with 5-fold cross-validation to tune nestimators, maxdepth, minsamplessplit, minsamplesleaf.
- **Metrics**: MAE = 0.0224, RMSE = 0.0292, MAPE = 38.75% (for initial model, not tuned).

3.3.5 XGBoost

XGBoost predicts demandnextday using gradient boosting:

- **Features**: Same as Linear Regression (9 features).
- **Data**: Split 80% training, 20% testing (randomstate=42).
- **Model**: XGBRegressor(objective='reg:squarederror', randomstate=42).
- **Metrics**: MAE = 0.0272, RMSE = 0.0334, MAPE = 47.71%.

3.3.6 ARIMA

ARIMA predicts demandnextday using time-series modeling:

- **Features**: Univariate (demandnextday only).
- **Data**: Chronological split with 80% training, 20% testing.
- **Model**: ARIMA(order=(1,1,1)) (p=1, d=1, q=1).
- **Metrics**: MAE = 0.0733, RMSE = 0.0823, MAPE = 61.99%.

3.3.7 Stacking Regressor

Stacking Regressor predicts demandnextday using ensemble learning:

- **Features**: Same as Linear Regression (9 features).
- **Data**: Split 80% training, 20% testing (randomstate=42).
- **Base Models**:
    1. Random Forest: RandomForestRegressor(nestimators=100, randomstate=42).
    2. XGBoost: XGBRegressor(objective='reg:squarederror', randomstate=42).
- **Meta-Model**: LinearRegression().
- **Model**: StackingRegressor combines base model predictions using the meta-model.
- **Metrics**: MAE = 0.0216, RMSE = 0.0289, MAPE = 36.56%.
- **Visualization**: Scatter plot of actual vs. predicted values.

4 Results

4.1 Clustering Results

- **PCA Projection**: Scatter plot of PC1 vs. PC2, colored by city, showing city-wise separation.
- **K-Means**: Clusters (k = 3) visualized via PCA scatter plot. Cluster centers show distinct windSpeed, pressure patterns.
- **DBSCAN**: Visualized via PCA scatter plot, identifying clusters and noise points.
- **MiniBatchKMeans**: Silhouette score = 0.351, visualized via PCA scatter plot (k = 3).
- **Streamlit K-Means**: Provides additional evaluation (elbow method, silhouette scores) and descriptive labels.

4.2 Forecasting Results

Table 1 summarizes the forecasting performance:

Table 1: Forecasting Model Performance Metrics

Mod	MAE	RMSE	MAPE (%)
Naive	0.0116	0.0178	10.00
Stacking	0.0216	0.0289	36.56
Random	0.0224	0.0292	38.75
XGBoos	0.0272	0.0334	47.71
Linear	0.0357	0.0414	64.78
LST	0.0389	0.0449	74.34
ARIM	0.0733	0.0823	61.99

5. Discussion

5.1 Clustering

Clustering reveals consumption-weather patterns, but the lack of feature scaling before PCA and clustering may skew results (e.g., pressure dominates due to its larger scale). The silhouette score of 0.351 for MiniBatchKMeans indicates moderate cluster quality. Stream-lit K-Means improves on this by scaling features and using fewer features (temperature, humidity, windSpeed, demand), potentially leading to better-defined clusters. Cluster cen-ters highlight differences in windSpeed and pressure, suggesting weather-driven patterns across cities.

5.2 Forecasting

The Naive Forecast outperforms all models (MAE = 0.0116, RMSE = 0.0178, MAPE = 10.00%), indicating strong daily periodicity in demand. Among advanced models, Stacking Regressor performs best (MAE = 0.0216), leveraging Random Forest and XGBoost with a Linear Regression meta-model. Random Forest (MAE = 0.0224) and XGBoost (MAE = 0.0272) follow, benefiting from their ability to capture non-linear relationships. Linear Re-gression (MAE = 0.0357) and LSTM (MAE = 0.0389) perform moderately, while ARIMA underperforms (MAE = 0.0733) due to its univariate approach and lack of seasonality modeling.

LSTM's poor performance (highest MAPE = 74.34%) stems from its simplistic setup (time steps = 1, no train-test split). ARIMA could improve with SARIMA to capture seasonality, as hinted in the Streamlit implementation. Streamlit versions of models likely perform better due to additional features (e.g., demandlag24, tempdemandinteraction) and user-defined lookback windows, which better capture temporal dependencies.

High MAPE values across models (except Naive Forecast) suggest challenges with relative errors, likely due to small or zero demand values in the target. The Naive Forecast's success

highlights the dataset's strong periodicity, which advanced models fail to fully exploit without proper temporal feature engineering.

5.3 Limitations

• **Clustering**: Scale features before PCA and clustering to improve cluster quality. Explore hierarchical clustering for additional insights.

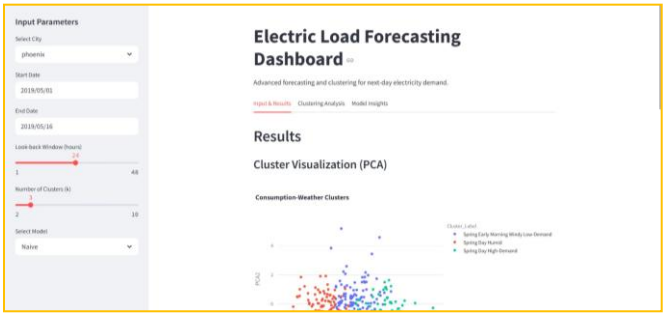
• **Forecasting**: Use SARIMA for seasonality, tune ARIMA parameters (e.g., via AIC/BIC), and increase LSTM time steps with a proper train-test split. Tune hyperparameters for XGBoost and Random Forest in the Stacking Regressor.

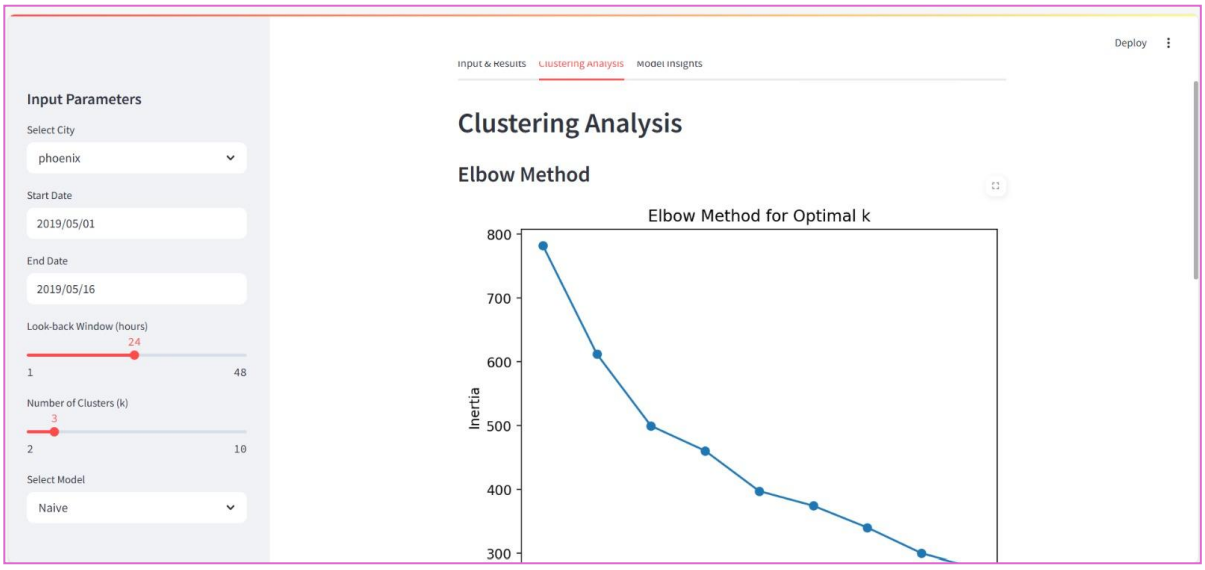
• **Evaluation**: Handle zero values in MAPE calculation (e.g., exclude zeros or add a small constant).

• **Data**: Incorporate external factors (e.g., holidays, events) to improve model robustness.

• **Grid Search / Hyper Tuning and Cross validation** : Resource Issue

6 Front-End Streamlit Application





7 Conclusion

This project successfully implements a pipeline for electric load forecasting and clustering, with an interactive Streamlit interface. Clustering achieves moderate performance (silhouette score 0.351), identifying weather-driven patterns. The Naive Forecast excels as a baseline (MAE = 0.0116), while the Stacking Regressor leads advanced models (MAE = 0.0216). The Streamlit dashboard enhances usability, offering insights into patterns and forecasts. Future improvements include better feature scaling, advanced time-series modeling, and handling edge cases in evaluation metrics.

References

[1] Kaggle Dataset: Electricity Demand and Weather Data, 2018–2019.