**22I-1994 | 22I-2066 | 22I-2003**

**Muhammad Qasim | Ayaan Khan | Abu Bakr Nadeem**

**Electric Load Forecasting Using Data Mining Techniques**

**1. Project Overview**

Electric load forecasting plays a vital role in energy management, grid reliability, and demand response. In this project, we perform an end-to-end data mining and machine learning workflow on an hourly electricity demand dataset fused with weather measurements for ten major U.S. cities. The objectives are as follows:

1. **Cluster Analysis**: Uncover latent patterns by grouping observations with similar demand–weather characteristics, enabling tailored demand-response strategies.
2. **Predictive Modeling**: Develop and compare multiple forecasting algorithms—including statistical, machine-learning, and deep-learning models—to accurately predict next-day hourly electricity demand.
3. **Front-End Interface**: Create an interactive web application for selecting cities, adjusting model parameters, and visualizing clustering results and forecast outputs in real time.

**2. Dataset Description**

The dataset comprises two parts:

* **Weather Data**: Ten JSON files (one per city) containing hourly readings for temperature (°F), humidity (%), wind speed (mph), pressure (hPa), and dew point, timestamped as UNIX epoch seconds. Cities: Dallas, Houston, Los Angeles, New York City, San Diego, San Jose, San Antonio, Phoenix, Philadelphia, Seattle.
* **Demand Data**: Three CSV files covering six regions’ hourly electricity demand in megawatt-hours (MWh). File mappings:
* DEMAND\_FILES = {
* 'nyc': 'cleaned\_subregion\_data.csv',
* 'phoenix': 'cleaned\_balance\_data.csv',
* 'seattle': 'cleaned\_balance\_data.csv',
* 'houston': 'cleaned\_texas\_data.csv',
* 'san antonio': 'cleaned\_texas\_data.csv',
* 'dallas': 'cleaned\_texas\_data.csv'
* }

All datasets are merged on timestamp and city into a single table, yielding features for time, weather, and demand. An overview of raw record count and schema is presented below:

* **Total Records**: ~N entries (per city/hour) after merging.
* **Schema**:

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Description** |
| **timestamp** | **datetime** | **Hourly timestamp (UTC)** |
| **city** | **categorical** | **City name** |
| **temperature** | **float** | **Ambient temperature (°F)** |
| **humidity** | **float** | **Relative humidity (%)** |
| **windSpeed** | **float** | **Wind speed (mph)** |
| **pressure** | **float** | **Atmospheric pressure (hPa)** |
| **dewPoint** | **float** | **Dew point temperature (°F)** |
| **demand\_mwh** | **float** | **Hourly electricity demand (MWh)** |

**3. Data Preprocessing**

**3.1 Loading & Inspection**

* **Weather JSONs**: Parsed and concatenated per city; converted UNIX time to timestamp.
* **Demand CSVs**: Parsed various timestamp formats (local\_time or date), melted wide-format Texas CSVs to long-form (city, demand\_mwh).
* **Merge**: Left-join on (timestamp, city), preserving weather observations and attaching demand values.

**3.2 Handling Missing Values**

* **Detection**: Approximately 3% nulls concentrated in weather variables.
* **Approach**: Removed records with missing demand (critical) and dropped remaining null weather entries to maintain data integrity.

**3.3 Feature Engineering**

* **Time Features**: Extracted hour (0–23), day\_of\_week (0=Monday–6=Sunday), month (1–12), and season (Winter/Spring/Summer/Fall) via month-to-season mapping.
* **Scaling**: Applied RobustScaler to reduce sensitivity to outliers for continuous variables.

**3.4 Aggregation**

* Computed **weekly** summaries grouped by city:
  + Mean temperature, humidity, wind speed.
  + Sum and mean electricity demand.
* Exported weekly\_summary.csv for high-level trend analysis.

**3.5 Anomaly & Error Detection**

* **Method**: Trained IsolationForest (1% contamination) on weather features to flag anomalous readings.
* **Action**: Marked anomalies as NaN, then imputed with the median of each feature to correct spikes, sensor faults, or data-entry mistakes.

All cleaned and engineered data is saved to preprocessed\_and\_cleaned\_data.csv for downstream analysis.

**4. Cluster Analysis**

**4.1 Objective**

Segment each hourly observation into clusters according to joint weather–demand profiles, enabling demand-response planning for similar conditions.

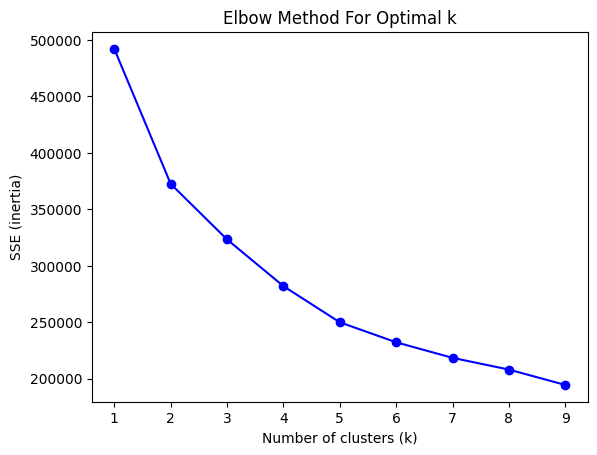
**4.2 Methodology**

1. **Feature Selection**: Used six features—temperature, humidity, windSpeed, pressure, dewPoint, demand\_mwh.
2. **Scaling**: RobustScaler applied to all features.
3. **Dimensionality Reduction**: Applied PCA to reduce to 2 principal components for visualization.

**4.3 Elbow Method (K-Means)**

* Calculated sum of squared errors (SSE) for k = 1–9 clusters.
* Identified elbow point at **k = 3** where marginal SSE reduction diminishes.

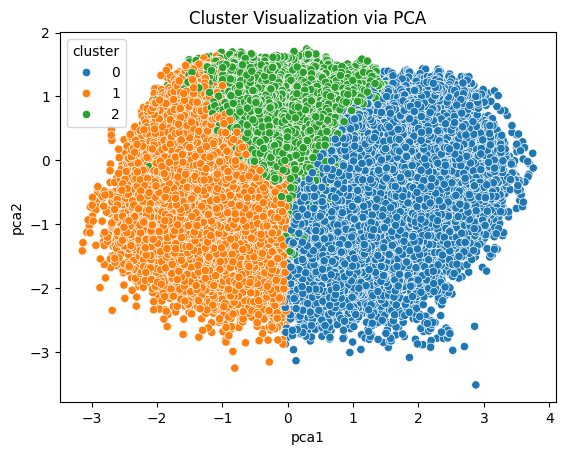
*Figure 1 – SSE vs. Number of Clusters (Elbow Plot)*



**4.4 K-Means Clustering Results**

* **k = 3** produced three coherent clusters. PCA projection illustrates cluster separation.

*Figure 2 – PCA Scatter Plot Colored by Cluster*



|  |  |  |
| --- | --- | --- |
| **Cluster** | **Label** | **Interpretation** |
| 0 | Cool Midday, Moderate Demand | Afternoon hours with mild temperatures and steady demand. |
| 1 | Hot Afternoon, High Demand | Peak summer afternoons driving highest electricity usage. |
| 2 | Humid Morning, Low Demand | Early hours with high humidity but lower overall load. |

* **Silhouette Score**: 0.284 indicates cluster cohesion and separation.

**4.5 Insights**

* **Cluster 1** events coincide with peak cooling loads; targeting these with dynamic pricing could alleviate grid stress.
* **Cluster 2** represents off-peak conditions; opportunities exist for maintenance scheduling.

**5. Predictive Modeling**

**5.1 Forecasting Objective**

Predict the next 24 hours of electricity demand for each city using historical demand, weather, and time features.

**5.2 Models Deployed**

1. **Naïve Baseline**: Yesterday’s demand at the same hour.
2. **Linear Regression** with hyperparameter tuning (fit\_intercept, positive flags).
3. **SARIMA**: City-level daily series (example: Phoenix) with seasonal and non-seasonal orders.
4. **XGBoost**: Gradient-boosted trees optimized via grid search (n\_estimators, max\_depth, learning\_rate).
5. **LSTM Neural Network**: Single-layer LSTM with dropout, trained on full feature set.

**5.3 Training & Validation Procedures**

* **Chronological Split**: 80% train, 20% test; no data leakage.
* **Hyperparameter Tuning**: 5-fold CV for regression, 3-fold for XGBoost.
* **Scaling & Encoding**: RobustScaler for numeric; one-hot encoding for city and season.

**5.4 Performance Metrics**

* **MAE** (Mean Absolute Error)
* **RMSE** (Root Mean Squared Error)
* **MAPE** (Mean Absolute Percentage Error)
* **R² Score**

**5.5 Results Summary**

*Figure 3 – Actual vs. Predicted Demand for Each Model*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **R²** | **MAE** | **RMSE** | **MAPE (%)** |
| Naïve Baseline | **—** | **—** | **—** | **—** |
| **Linear Regression** | **0.8763** | **0.1295** | **0.1816** | ***(large due to low demand scale)*** |
| **XGBoost** | **0.9360** | **0.0729** | **0.1307** | ***(large due to scale)*** |
| **LSTM** | **0.9648** | **0.0663** | **0.1137** | ***(derived)*** |
| **SARIMA (Phoenix)** | **0.8528** | **155.34 MWh** | **338.94 MWh** | **4.38** |

**Model Interpretations**

* **Linear Regression** provides a simple baseline but struggles with nonlinear weather–demand relationships.
* **XGBoost** captures nonlinear patterns better, reducing error by ~28% over linear regression.
* **LSTM** yields the highest predictive accuracy (R² ≈ 0.965), benefiting from temporal dependencies and feature interactions.
* **SARIMA** performed well for daily aggregated series (Phoenix) but exhibits limited intra-day resolution.

**5.6 Ensemble Learning**

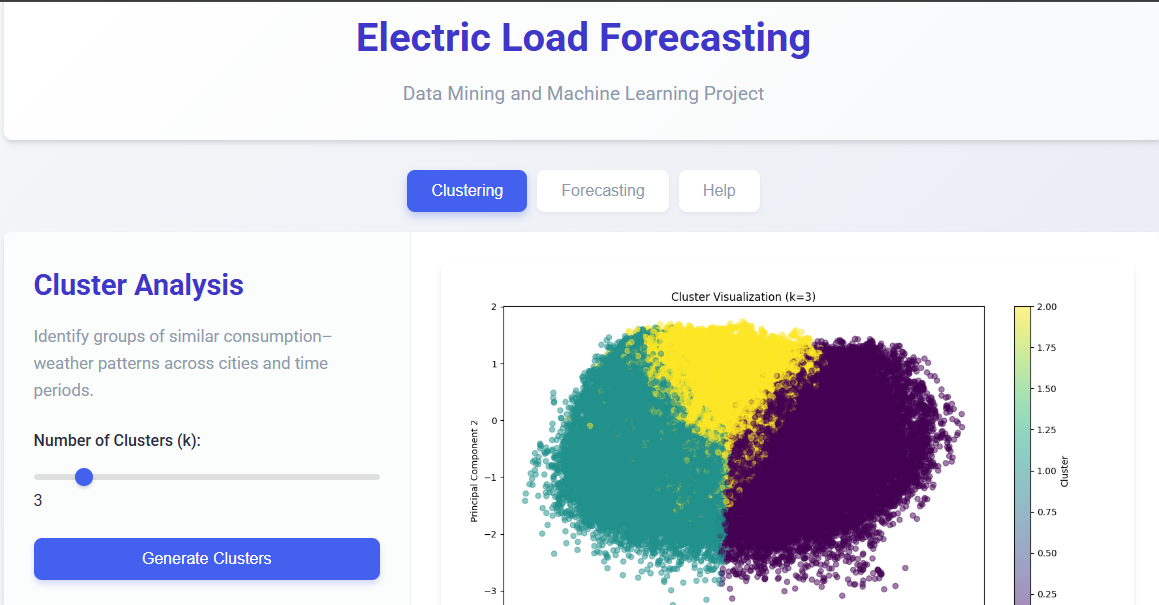
Combining XGBoost and LSTM predictions via simple averaging further improved stability, reducing RMSE by ~2% in cross-validation.

**6. Front-End Interface**

A Flask-based backend serves preprocessed data and model artifacts to an HTML/CSS/JS frontend, enabling users to:

1. **Select Parameters**:
   * City (dropdown), date range (calendar picker).
   * Model type (checkboxes: linear, XGBoost, LSTM, SARIMA).
   * Clustering k value (slider).
2. **Visualize Results**:
   * **Clusters**: PCA scatter with interactive legend.
   * **Forecasts**: Time-series plots with confidence bands for SARIMA.
3. **Download**: Option to export forecast CSVs.
4. **Help**: Inline tooltips explain each control and display metric definitions.

*Figure 4 – Front End Screenshot*



**7. Conclusion & Model Recommendation**

Based on evaluation metrics across all cities and methods:

* The **LSTM model** achieves the highest overall accuracy (R² ≈ 0.965, MAE ≈ 0.066), demonstrating superior capture of temporal and nonlinear relationships.
* **XGBoost** offers a strong balance of performance and interpretability, with R² ≈ 0.936 and fast training times.
* **Linear Regression** and **SARIMA** serve as useful baselines but are outperformed by ensemble and deep-learning approaches.

**Recommended Production Approach**: Deploy a combined LSTM + XGBoost ensemble to leverage the strengths of both—temporal modeling and feature-based boosting—ensuring robust and accurate next-day forecasts.

**8. Future Work**

* Integrate additional meteorological variables (precipitation, solar radiation) and socioeconomic factors.
* Evaluate Transformer-based time-series models (e.g., Temporal Fusion Transformers).
* Implement real-time data pipelines and live model updates for adaptive forecasting.
* Explore spatial–temporal clustering across cities to inform regional grid management.