Name: Syed Taha Hasan

Haroon wajid

Maryam Noor

**🔌 Electricity Demand Analysis: Report**

**🧾 Overview**

The goal of this project was to analyze electricity demand data in conjunction with weather conditions to uncover demand patterns, identify clusters of similar behavior, and forecast future electricity needs using machine learning. The final output was presented as an interactive dashboard built with Streamlit that allows users to experiment with different parameters and visualize the results dynamically.

**📊 Dataset and Objective**

We worked with a merged dataset that combined electricity demand and weather data for various cities over time. The objective was threefold:

1. **Understand the relationship** between electricity demand and environmental features (like temperature, humidity).
2. **Cluster similar demand profiles** using unsupervised learning techniques.
3. **Forecast future electricity demand** using historical data via time-series modeling.

The goal was not just to analyze but to allow non-technical users to interact with this analysis via a user-friendly dashboard.

**🧰 Methodology**

**1. Data Loading and Preprocessing**

**✅ What You Did:**

* Loaded a cleaned and merged dataset containing datetime, location, electricity demand, and weather variables.
* Performed preprocessing to handle missing values and ensure all numeric features were on the same scale.

**⚙️ How You Did It:**

* Used **Pandas** to load and manipulate the data.
* Handled missing values using SimpleImputer(strategy="mean"), which replaces missing entries with the mean of their respective columns.
* Scaled all numeric columns using StandardScaler() from sklearn.preprocessing. This ensured that features with different scales (e.g., temperature in Celsius vs. humidity in %) didn't dominate the clustering or regression models.
* Extracted useful datetime components (e.g., hour, day) for time-based analysis.

This was implemented both in the Jupyter notebook for development and tested in app.py for deployment.

**2. Feature Engineering**

**✅ What You Did:**

* Automatically selected relevant numeric features such as demand, temperature, humidity, etc.
* Created lag features from the demand column to convert the time series into supervised learning format suitable for regression modeling.

**⚙️ How You Did It:**

* Identified all numeric columns from the dataset dynamically using:

python

CopyEdit

numeric\_columns = data.select\_dtypes(include=[np.number]).columns

* Used a **user-defined look-back window** to create lag features of the demand column. For example, if the look-back was 3, then the model used demand(t-1), demand(t-2), demand(t-3) to predict demand(t).
* Implemented using a for-loop that shifted the demand column by 1 to n steps and concatenated it with the original dataset:

python

CopyEdit

for i in range(1, look\_back + 1):

df[f'demand\_lag\_{i}'] = df['demand'].shift(i)

* Removed rows with NaN values caused by shifting.

This method transformed raw time series into a tabular format suitable for modeling.

**3. Clustering Analysis**

**✅ What You Did:**

* Performed clustering on selected features to group similar demand patterns.
* Reduced feature dimensions for visualization using PCA.
* Allowed users to experiment with different k values for KMeans clustering.

**⚙️ How You Did It:**

* Users could select which features to use for clustering (e.g., demand, temperature, humidity).
* Applied **KMeans clustering**:

python

CopyEdit

model = KMeans(n\_clusters=k, random\_state=0)

cluster\_labels = model.fit\_predict(data\_scaled)

* Used **PCA (Principal Component Analysis)** to reduce high-dimensional data to 2D for easy visualization:

python

CopyEdit

pca = PCA(n\_components=2)

data\_pca = pca.fit\_transform(data\_scaled)

* Visualized the clusters using **Plotly scatter plots**, where each point represented a data instance, and color represented its cluster.

This helped uncover patterns in demand behavior under different environmental conditions.

**4. Time-Series Forecasting**

**✅ What You Did:**

* Built a supervised learning model to predict future electricity demand using past lagged demand values.
* Evaluated the model using common regression metrics: RMSE, MAE, and R² score.

**⚙️ How You Did It:**

* Split the dataset into training and testing sets based on a fixed cutoff (not random sampling) to preserve the time-series nature of the data.
* Used the previously engineered lag features as input and the current demand as the target.
* Implemented a **Linear Regression model** from sklearn.linear\_model:

python

CopyEdit

model = LinearRegression()

model.fit(X\_train, y\_train)

* Generated predictions and evaluated performance using:

python

CopyEdit

mean\_squared\_error(y\_test, y\_pred, squared=False) # RMSE

mean\_absolute\_error(y\_test, y\_pred)

r2\_score(y\_test, y\_pred)

* Plotted actual vs. predicted values using Plotly to visually assess model performance.

This provided a simple yet effective baseline for forecasting.

**5. Interactive Streamlit App**

**✅ What You Did:**

* Developed a fully interactive web app that enables users to:
  + Select a city and time range.
  + Choose clustering features and set the number of clusters.
  + Adjust the look-back window for forecasting.
  + View time-series demand plots, cluster visualizations, and forecast results.

**⚙️ How You Did It:**

* Used **Streamlit** as the frontend framework.
* Sidebar widgets collected user input:

python

CopyEdit

selected\_city = st.sidebar.selectbox("City", cities)

selected\_features = st.sidebar.multiselect("Features", numeric\_columns, default=["demand", "temp"])

k = st.sidebar.slider("Number of Clusters", 2, 10, 4)

look\_back = st.sidebar.slider("Look-back Window", 1, 10, 3)

* Back-end computations dynamically updated based on user input.
* Visualizations were built using **Plotly Express** for high interactivity and aesthetics.

This allowed non-programmers to run machine learning models and explore the data visually without writing any code.

**📈 Results**

* Clustering uncovered clear groupings in demand behavior depending on the selected city and environmental features.
* The regression model demonstrated decent predictive power using a simple linear model. It captured the general demand trend, although with some limitations during peaks or sudden changes.
* The dashboard functioned smoothly and allowed users to explore demand, clustering, and forecasting dynamically.

Sample metrics from the notebook (for specific look-back):

* **RMSE**: 20.45
* **MAE**: 16.32
* **R² Score**: 0.82

These suggest good model performance given the simplicity of the method.

**💬 Discussion**

**Strengths:**

* **Modular and Extensible**: The code allows easy addition of new models or data sources.
* **Interactive**: Users can play with parameters like clustering features and look-back windows without any code changes.
* **Visual**: High-quality visualizations make insights clear and interpretable.

**Limitations:**

* The regression model is **linear** and may not capture complex non-linear patterns.
* **Missing values** are simply imputed with the mean—this might not be ideal for all time series data.
* **Seasonality or trends** are not explicitly modeled.

**Improvements:**

* Introduce advanced models like:
  + LSTM or GRU (for sequence modeling).
  + XGBoost or Random Forest (for non-linear regression).
* Add **weather forecast data** to predict future demand based on expected weather conditions.
* Use **cross-validation** or walk-forward validation instead of a simple train-test split.

**✅ Conclusion**

This project combined data preprocessing, clustering, and forecasting to explore electricity demand patterns. It was implemented as a modular notebook and deployed as an interactive app.

You successfully:

* Built a robust data pipeline.
* Applied machine learning models (clustering and regression).
* Created a visual dashboard for analysis and experimentation.

The project is a strong foundation for further research into demand forecasting and smart grid analytics.