Project Report

Created	@May 10, 2025 9:05 PM
Class	Data Mining

🌐 Energy Demand Analysis and Forecasting – Project Summary

🔎 Introduction

This project focuses on analyzing energy demand data using **clustering** and **forecasting techniques**. The implementation includes:

- Python scripts for preprocessing, EDA, clustering, and predictive modeling
- A web application for **interactive visualization and analysis**

🧩 Project Components

Component	Description
Clustering Analysis (1.py)	Identifies patterns/segments in energy demand data
Predictive Modeling (2.py)	Forecasts future energy demand using ML techniques
Web Application (app.py)	Interactive interface for visualization and analysis
Jupyter Notebook	Educational combo of clustering & forecasting (Notebook.ipynb)

🗂️ Data Description

Dataset: `dataset_cleaned.csv`

Key Features:

- `demand`: *Energy consumption/demand (target)*
- **Weather features:** `precipIntensity`, `precipProbability`, `temperature`, `apparentTemperature`, etc.
- **Time features**: `hour`, `day`, `month`, `day_of_week` (either present or derived)

⚙️ Methods

🛠️ 1. Data Preprocessing

- **Time Indexing**: Convert timestamps → datetime format
- **Missing Values**: Numeric → mean | Categorical → mode
- **Feature Engineering**:
  - Time-based: `hour`, `day_of_week`, `month`
  - Lag: `demand_lag1`, `demand_lag24`, `demand_lag168`
  - Rolling stats: mean, std over windows
  - Cyclical encoding: sine/cosine of time

🌀 2. Clustering Analysis

- **Feature Selection**: Weather variables
- **Dimensionality Reduction**: PCA
- **Algorithms:**
  - K-Means (with silhouette optimization)

- DBSCAN
- Hierarchical Clustering
- **Evaluation**:
  - Silhouette scores
  - Cluster stability
  - PCA & t-SNE visualizations

📈 3. Predictive Modeling

Model Type	Models
Baseline	Naive forecast (previous day's same hour)
Linear Models	Linear Regression, Ridge Regression
Tree-Based	Random Forest, Gradient Boosting, XGBoost
Deep Learning	Custom PyTorch NN (batch norm, dropout)
Time Series	ARIMA (1,1,1)
Ensemble	Stacking of top 3 performing models

🌐 4. Web Application (Flask)

Features:

- Parameter selection (city, date range, cluster count, model)
- Cluster visualization (PCA projections)
- Forecast comparison (actual vs predicted)
- Model metrics: MAE, RMSE, R^2, MAPE

📊 Results

🔍 Clustering Analysis

- **Optimal Clusters**: Typically 3–5 clusters (silhouette-based)
- **Cluster Types**:
  - High demand / low temp → *Winter heating*
  - High demand / high temp → *Summer cooling*
  - Moderate demand → *Neutral weather*
- **Feature Importance:** `temperature` / `apparentTemperature` most impactful

🔮 Forecasting Results

Key Findings:

- **Baseline**: Naive forecast sets benchmark
- **Best Models**: XGBoost & Ensembles
- **Good Models**: Random Forest, Gradient Boosting, Neural Network
- **Moderate Models**: Linear models, ARIMA

Top Features:

- `demand_lag24` (previous day's demand)
- `demand_lag168` (previous week's demand)
- `hour` (especially cyclically encoded)
- `temperature`

📏 Key Metrics

Metric	Improvement (vs Baseline)
MAE	15–30% improvement
RMSE	10–25% improvement
MAPE	5–15% (best models)

💡 Discussion

🔑 Clustering Insights

- **Segmentation**: Clear patterns based on weather & time
- **Usage**: Helps in capacity planning, anomaly detection, demand driver analysis

---

⚖️ Forecasting Tradeoffs

Model	Pros	Cons
XGBoost/Ensemble	Highest accuracy	Complex to deploy
Random Forest	Balanced performance & interpretability	Slower for large datasets
Linear Models	Simple & interpretable	Less accurate
ARIMA	Lightweight, needs less data	Least accurate of advanced models

Feature Engineering Impact:

- Lag features ➔ capture history
- Cyclical encoding ➔ represents time naturally
- Rolling stats ➔ recent trends & volatility

🖥️ Web App Benefits

- **Accessibility**: For non-technical users
- **Interactivity**: Explore different setups easily
- **Visualization**: Turns complex results into simple visuals

---

✅ Conclusion

The project successfully combines clustering + forecasting for **comprehensive energy demand analysis**.

Key Takeaways:

1. ✅ *Feature engineering is crucial*
2. ✅ *Ensemble methods give best accuracy*
3. ✅ *Combining clustering & forecasting ➔ deeper insights*
4. ✅ *Interactive apps make results accessible*

✅ **Tip:** In Notion, you can make sections collapsible (toggle lists) for a clean look and add icons/emoji to headings for better visual appeal!