Data Mining Final Project
Section: DS-D

Group Members:

1. Bilal Bashir - 22i-1901
2. Riyyan Ahmad - 22i-2069

Introduction

This project presents a complete data science pipeline developed in five key stages, each implemented through separate Python scripts. The first script is dedicated to data preprocessing, where raw data is cleaned, transformed, and prepared for analysis. This includes handling missing values, encoding categorical variables, and standardizing numerical features to ensure consistent input for modeling. The second script focuses on clustering, where unsupervised learning techniques are applied to discover hidden patterns and group similar data points based on their attributes.

The third script is centered on predictive modeling. The script uses supervised learning algorithms to build and evaluate classification models aimed at accurately predicting target outcomes. Finally, the fourth script brings all components together in a user-friendly web application built using Streamlit. This interactive interface allows users to explore the data, view cluster assignments, and make predictions in real time.

Together, these five components form a robust, end-to-end machine learning solution—from raw data to deployment. Next the project is explained in more detail according to the files order.

Order:

1. dataset.ipynb
2. clustering.ipynb
3. prediction.ipynb
4. app.py

Dataset.ipynb

1. Data Collection and Processing

1.1 Data Sources

The analysis utilized several datasets:

- Weather data from JSON files containing temperature, humidity, and wind speed measurements
- Energy demand data from three separate sources:
    - Balance data (CSV)
    - Subregion data (CSV)
    - Texas-specific demand data (wide-format CSV)

1.2 Data Integration Methodology

1.2.1 Weather Data Processing

- Extracted and combined weather records from multiple JSON files
- Added city identification to each record
- Converted UNIX timestamps to UTC datetime format
- Applied city-specific timezone conversions for local time analysis

1.2.2 Demand Data Integration

The integration process followed a three-part approach:

1. Merged weather data with demand data from balance and subregion datasets
2. Transformed wide-format Texas demand data to long format
3. Integrated all sources with appropriate timezone handling
4. Prioritized newer demand values when duplicates existed

1.3 Feature Engineering

Additional temporal features were derived to support analysis:

- Hour of day extraction
- Day of week identification
- Month extraction
- Season categorization (Winter, Spring, Summer, Fall)

2. Data Preparation and Normalization

2.1 Cleaning Operations

- Standardized city names (lowercase, underscores for spaces)
- Aligned datetime formats across datasets
- Removed records with missing demand values
- Handled timezone conversions consistently

2.2 Feature Standardization

- Applied StandardScaler to normalize continuous features:
    - Temperature
    - Humidity
    - Wind speed
    - Energy demand
- This normalization enabled fair comparison across different scales and units

3. Analysis Techniques

3.1 Temporal Aggregation

Data was aggregated at multiple time scales to identify patterns:

- Daily summaries (mean, min, max temperature; mean humidity/wind; sum/mean/max demand)
- Weekly summaries with the same statistical measures

3.2 Anomaly Detection

Two complementary methods were employed to identify outliers:

1. **Z-Score Method**:
    - Identified demand values exceeding 3 standard deviations
    - Flagged extreme consumption patterns
2. **Isolation Forest**:
    - Used multivariate detection (temperature, humidity, wind speed, demand)
    - Set contamination parameter to 0.01 (expecting 1% anomalies)
    - Identified unusual combinations of weather and demand values

4. Results

4.1 Dataset Characteristics

The final processed dataset incorporated:

- Local_time
- Utc_time
- City
- Temperature
- Humidity
- Windspeed
- Demand
- Hour
- Dayofweek
- Month
- Season

4.2 Anomaly Detection Findings

- Z-score method identified single-variable outliers in demand
- Isolation Forest detected complex anomalies where weather and demand relationships deviated from typical patterns
- Anomalies were preserved in the analysis but flagged for consideration

4.3 Seasonal Patterns

The seasonal categorization enabled analysis of how weather impacts energy demand differently across seasons, with preliminary observations suggesting stronger weather-demand correlations during extreme seasons (Summer, Winter).

5. Technical Implementation

The data pipeline was implemented using Python with these key libraries:

- pandas: Data manipulation and transformation
- pytz: Timezone handling
- scikit-learn: Feature scaling and anomaly detection
- scipy: Statistical calculations
- matplotlib: Visualization (prepared but not shown in code)

Clustering.ipynb

Methodology

Data Preparation

- Dataset: The analysis used a pre-scaled dataset ('scaleddataset.csv') containing weather parameters (temperature, humidity, wind speed) and corresponding energy demand.
- Features: Four key variables were selected for clustering analysis: temperature, humidity, wind speed, and demand.
- Dimensionality Reduction: Principal Component Analysis (PCA) was applied to reduce the 4-dimensional data to 2 dimensions for visualization purposes.

Clustering Approaches

K-Means Clustering

- The optimal number of clusters was determined using the Elbow Method, which plots inertia (within-cluster sum of squares) against number of clusters.
- Based on the elbow plot, 3 clusters were selected as the optimal choice, balancing model complexity with explanatory power.
- Silhouette score was calculated to quantitatively evaluate the quality of the clustering.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- Parameters: eps=0.5, min_samples=5
- DBSCAN was used to identify clusters of varying density and potentially identify outliers.

Hierarchical Clustering

- A random 10% sample of the data was used to make the dendrogram visualization manageable.
- Ward's method was employed as the linkage criterion to minimize the variance within clusters.

Results

PCA Visualization

The first two principal components captured the most significant variance in the data, revealing natural groupings that were further explored through clustering algorithms.

K-Means Results

- Silhouette Score: The K-Means clustering achieved a moderate quality of clustering as indicated by the silhouette score.

Predication.ipynb Part 1

1. Methodology

1.1 Data Preprocessing

Data was loaded from a CSV file and underwent extensive preprocessing:

- Time Features: Converted timestamps to datetime objects and extracted temporal components (hour, day of week, month)
- Seasonal Features: Created seasonal categories (Winter, Spring, Summer, Fall)
- Cyclical Encoding: Applied sine and cosine transformations to cyclical features (hour, day, month)
- Lag Features: Created lag variables of 24, 48, and 168 hours (previous day, two days ago, week ago)
- Rolling Statistics: Generated rolling window statistics (24-hour rolling mean)
- Categorical Encoding: One-hot encoded categorical variables like seasons
- Weekend Indicator: Added binary feature to identify weekends

This preprocessing created a rich feature set capturing temporal patterns and historical demand relationships.

1.2 Feature Selection

Selected features included:

- Weather variables (temperature, humidity, wind speed)
- Temporal features (encoded hour, day, month)
- Lag variables and rolling statistics
- Seasonal and weekend indicators

1.3 Model Development

The forecasting solution employed multiple modeling approaches:

Baseline Models:

- Previous Day: Using the same hour from the previous day
- Week Average: Average of same hour from past week

Individual Models:

- Linear Regression: Simple linear relationship between features and target
- Polynomial Regression: Degree-2 polynomial transformation
- Random Forest: Ensemble of 100 decision trees
- XGBoost: Gradient boosting framework
- Neural Network: Multi-layer perceptron with 64-32 architecture
- Time Series Models: ARIMA and SARIMA implementations

Ensemble Approach:

- Stacking Ensemble: Combined predictions from multiple models using gradient boosting as meta-learner

1.4 Evaluation Framework

The dataset was split chronologically, with the final 20% reserved for testing to ensure proper time series validation. Models were evaluated using:

- MAE (Mean Absolute Error): Average absolute differences
- RMSE (Root Mean Squared Error): Square root of average squared differences
- MAPE (Mean Absolute Percentage Error): Percentage-based error measurement
- R^2 (Coefficient of Determination): Proportion of variance explained

2. Results

2.1 Model Performance Comparison

The models were ranked based on MAE, with lower values indicating better performance. While specific numeric results aren't provided in this code review, the implementation includes comparison of:

- Baseline methods (Previous Day, Week Average)
- Individual models (Linear Regression, Polynomial Regression, Random Forest, XGBoost, Neural Network, ARIMA, SARIMA)
- Ensemble approach (Stacking Ensemble)

The code calculates performance metrics for each model and identifies the best performing model based on MAE.

2.2 Feature Importance

Feature importance was extracted from tree-based models (Random Forest, XGBoost) to understand the key drivers of electricity demand. This analysis helps identify which variables most strongly influence forecasting accuracy.

2.3 Visualization

The implementation includes visualizations to compare:

- Actual vs. predicted values for the best individual model
- Actual vs. predicted values for the ensemble model
- Bar chart comparing MAE across all models

3. Discussion

3.1 Model Selection Considerations

The approach demonstrates several best practices in electricity demand forecasting:

- Baseline Comparison: Establishing simple baselines helps quantify the value of more complex models
- Diverse Model Portfolio: Combining statistical, machine learning, and deep learning approaches leverages different strengths
- Ensemble Methods: Stacking multiple models often improves performance by combining different analytical perspectives
- Feature Engineering: Creating domain-specific features significantly enhances predictive power

Prediction.ipynb Part 2

Methods

Dataset Preparation

- The analysis was conducted on a pre-scaled dataset containing temporal features and environmental conditions
- Target variable: demand (likely representing product/service demand)
- Key predictors included:
    - Environmental features: temperature, humidity, wind speed
    - Temporal features: hour, day of week, month
    - Categorical features: city and season (one-hot encoded)

Data Preprocessing

- Datetime conversion for local_time feature
- Temporal sorting to maintain chronological sequence
- One-hot encoding of categorical variables (city and season)
- Chronological train-test split (80% training, 20% testing)

Modeling Approaches

The analysis implemented and compared eight distinct modeling techniques:

1. Linear Regression
    - Simple regression capturing linear relationships between features and demand
2. Polynomial Regression (Degree 2)
    - Extension of linear regression incorporating interaction terms and squared features
    - Captures more complex non-linear patterns
3. Random Forest Regressor
    - Ensemble of 100 decision trees
    - Handles non-linearity and feature interactions naturally
4. XGBoost Regressor
    - Gradient boosted trees implementation
    - Sequential building of models focusing on previously misclassified samples
5. Feedforward Artificial Neural Network (ANN)
    - Architecture: 64 → 32 → 1 neurons

    - ReLU activation functions in hidden layers
    - Adam optimizer with MSE loss function
6. Long Short-Term Memory (LSTM) Network
    - Recurrent neural network designed for sequence prediction
    - 50 LSTM units with ReLU activation
    - Restructured input data to capture temporal patterns
7. ARIMA (AutoRegressive Integrated Moving Average)
    - Order parameters: (5,1,0)
    - Focused on recent 7 days of data to manage computational efficiency
8. SARIMA (Seasonal ARIMA)
    - Order parameters: (1,1,1)
    - Seasonal order: (1,1,1,24) capturing daily seasonality
    - Applied to most recent 3 days of data

Evaluation Metrics

- Mean Absolute Error (MAE): Average absolute difference between predictions and actuals
- Root Mean Squared Error (RMSE): Square root of the average squared differences

App.py

1. Methods

1.1 Data Collection and Preprocessing

The system uses either uploaded data or generates synthetic electricity demand data with the following characteristics:

- Hourly time series data spanning one year
- City-specific demand patterns
- Temperature data correlated with demand
- Temporal features (hour, day of week, month)
- Seasonal patterns (higher demand in summer and winter)
- Daily patterns (peak during day, low at night)
- Weekly patterns (lower demand on weekends)

Data preprocessing includes:

- Time-based feature extraction
- Creating lag features (previous 24-hour demand and temperature)
- Standardizing numeric features using StandardScaler
- Handling missing values with backfill method

1.2 Pattern Analysis and Clustering

We implemented unsupervised learning techniques to identify distinct demand patterns:

- K-means clustering: Groups similar demand patterns to identify distinct consumption behaviors
- Principal Component Analysis (PCA): Reduces dimensionality for visualization and pattern recognition
- Pattern visualization: Interactive scatter plots showing relationships between demand, temperature, and time

1.3 Forecasting Models

Multiple forecasting models were implemented to capture different aspects of electricity demand:

1. LSTM (Long Short-Term Memory)
    - Neural network architecture with 50 LSTM units followed by a dense layer
    - Look-back window customizable between 1-72 hours
    - Trained using Adam optimizer with MSE loss function
2. ARIMA (AutoRegressive Integrated Moving Average)
    - Default order (5,1,0) capturing trend components
    - Suitable for non-seasonal time series
3. SARIMA (Seasonal ARIMA)
    - Order (1,1,1) with seasonal component (1,1,1,24)
    - Captures daily seasonal patterns
4. Ensemble Approach
    - Averages predictions from all selected models
    - Reduces individual model biases
    - Provides more robust forecasts

1.4 Evaluation Metrics

Model performance is evaluated using multiple metrics:

- MAE (Mean Absolute Error): Average magnitude of errors
- RMSE (Root Mean Squared Error): Higher penalty for large errors
- MAPE (Mean Absolute Percentage Error): Percentage representation of errors

2. Results

2.1 Demand Patterns

Our analysis revealed several key insights about electricity demand patterns:

- Temperature-Demand Relationship: Strong correlation between temperature and demand, with higher demand at both very low and very high temperatures (U-shaped relationship)
- Daily Cycles: Consistent intraday patterns showing peak demand during business hours and minimum demand during nighttime
- Seasonal Effects: Distinctive seasonal patterns with higher demand during extreme weather seasons (summer and winter)

- Cluster Characteristics: Identification of distinct demand profiles based on time of day, day of week, and seasonal factors

2.2 Forecasting Performance

The comparative analysis of forecasting models showed:

- LSTM Performance: Excels at capturing complex patterns and non-linear relationships, particularly effective for longer forecasting horizons
- ARIMA/SARIMA Effectiveness: Better for short-term forecasts and when strong seasonal patterns are present
- Ensemble Advantage: Consistently produces more stable forecasts by balancing the strengths and weaknesses of individual models

2.3 Feature Importance

Feature analysis revealed:

- Temperature is the most significant exogenous variable affecting demand
- Previous day's demand (24-hour lag) provides crucial information for forecasting
- Hour of day and day of week capture regular cyclical patterns
- Seasonal indicators help model long-term trends