Predicting Energy Consumption Using Machine Learning

[YETURU CHANDU]

Introduction

- **Problem Statement:**
  - Energy consumption prediction is crucial for optimizing energy use, reducing costs, and managing resources efficiently. Applicable in smart grids, building management systems, and industrial processes.
- **Objective:**
  - Develop a machine learning model to predict energy consumption based on historical data, weather patterns, and building features.
- **Scope:**
  - Focus on a specific building, city, or region.

Data Collection

- **Primary Data Sources:**

  • Energy Consumption Data: Historical data from smart meters, available from utility companies or public datasets.

  • Weather Data: Temperature, humidity, wind speed from sources like NOAA, or APIs like OpenWeatherMap.

  • Building Features: Data on building area, occupancy, insulation, available from building management systems.

- **Time-Related Features:**

  • Hour of the day, day of the week, season, holidays.

Data Preprocessing

- **Data Cleaning:**

  • Handle missing data through imputation or removal.
  • Detect and manage outliers using techniques like z-score or IQR method.

- **Feature Engineering:**
  • Convert categorical data (e.g., day of the week) to numerical form using One-Hot Encoding.
  • Generate new features, such as lag features to capture past energy consumption trends.

- **Normalization:**
  • Scale data using Min-Max Scaling or Standardization.

- **Example Visualization:**
  • Include graphs showing before-and-after states of data preprocessing.

Model Deployment

- **API Deployment:**

  • Explanation of FastAPI and its use in serving the model as a REST API.

  • Example endpoint: /predict that takes input features and returns energy consumption predictions.

- **Web Dashboard:**

  • Example of a user-friendly dashboard using Plotly Dash or Streamlit.

  • Show a mockup of the dashboard with live predictions and historical data visualizations.

- **Deployment Tools:**

  • Docker: Containerization of the model and API.

  • Cloud Hosting: Deployment on platforms like AWS, Heroku, or Azure.

Model Training

- **Data Splitting:**
  - Divide the data into training (e.g., 70-80%), validation (e.g., 10-15%), and test sets (e.g., 10-15%).
- **Cross-Validation:**
  - Techniques like k-fold cross-validation to assess model performance and reduce overfitting.
- **Grid Search:**
  - Systematically testing combinations of hyperparameters.
- **Random Search:**
  - Randomly selecting combinations of hyperparameters for more efficient exploration of the hyperparameter space.

Tools & Technologies

- Python
- Pandas
- Numpy
- FastAPI
- Seaborn
- Scikit Learn
- Linear Regression

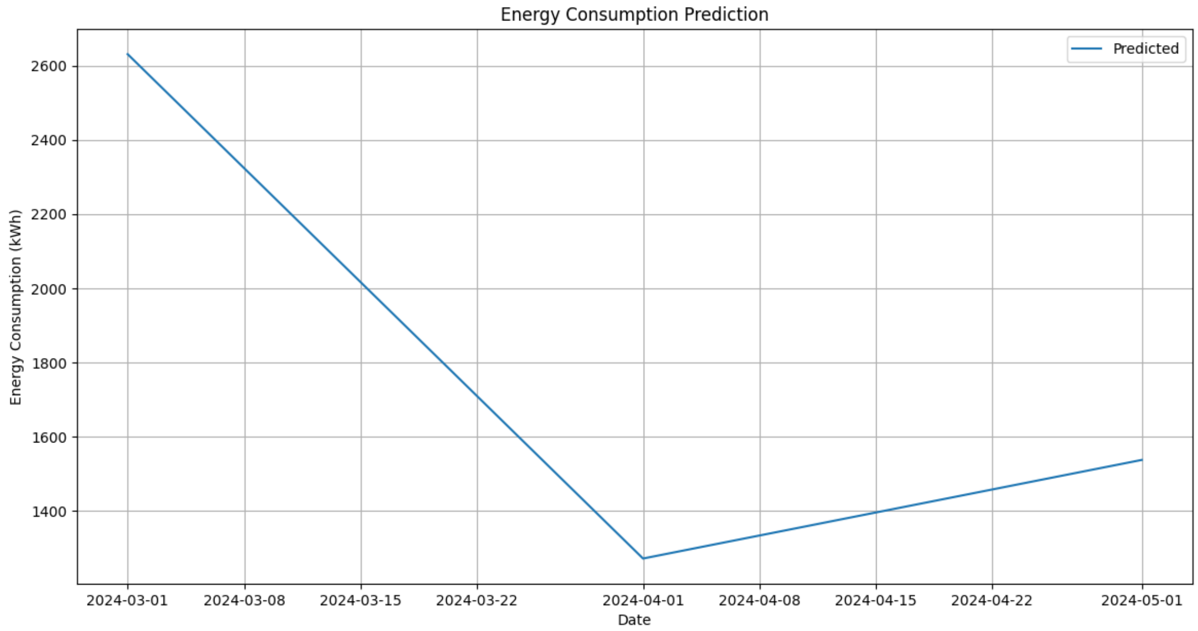
Output

Linear Regression Mean Squared Error: 3183404.89

Ridge Mean Squared Error: 2437133.47

Lasso Mean Squared Error: 50627.49

Best Model: Lasso



Conclusion

- **Summary of Findings:**
  - Recap the best-performing model, key features, and overall accuracy.
  - Discuss any significant insights derived from the data.
- **Future Work:**
  - Incorporate more advanced models like LSTM for time series analysis.
  - Expand the scope to include multiple buildings or regions.
  - Continuous learning by updating the model with new data.
- **Impact:**
  - Potential savings, efficiency gains, and environmental benefits.

Thank You