**Electric Load Forecasting using AI/ML**

**Submitted by:** Abhishek Sen Sarma **Enrollment Number:** 12022002011041 **Semester:** 6th  
***Department of Electrical Engineering  
Institute of Engineering and Management Kolkata***

**1. Introduction**

Electric load forecasting plays a vital role in modern power systems for ensuring reliable and economic operation of the electrical grid. Accurate load forecasting enables utilities to balance demand and supply, schedule generation, and plan for energy trading and infrastructure expansion. In this project, a machine learning-based forecasting model is developed to predict electric load using historical hourly data from the AEP region.  
  
The rapid growth of smart grids and data acquisition systems has opened up new opportunities for AI/ML applications in power systems. This project leverages Random Forest Regressor to predict hourly load values based on temporal features extracted from historical data.

**2. Objective**

The primary objective of this project is to develop and evaluate an AI/ML-based load forecasting model capable of predicting electric load for each hour using historical data. The model aims to minimize forecasting errors and provide reliable short-term load predictions for grid operation and planning.

**3. Dataset Description**

The dataset used for this project is the 'AEP Hourly Energy Consumption' dataset sourced from Kaggle. It contains historical electric load data measured in megawatts (MW) at an hourly resolution.  
  
Attributes:  
- Datetime: Timestamp of the measurement  
- AEP\_MW: Actual electric load in megawatts (MW)  
  
The dataset spans from January 2004 to January 2018, providing an extensive collection of hourly load values for analysis.

**4. Tools and Technologies Used**

- Python 3.11  
- Google Colab (Cloud-based Jupyter Notebook environment)  
- Pandas: For data manipulation and analysis  
- Numpy: For numerical operations  
- Scikit-learn: For building machine learning models  
- Matplotlib: For data visualization

**5. Methodology**

The project methodology is structured into several phases as follows:  
  
1. Data Collection: The dataset was sourced from Kaggle and uploaded to the Google Colab environment.  
2. Data Preprocessing:  
 - The 'Datetime' column was converted to datetime format.  
 - Missing values, if any, were identified and handled.  
 - Time-based features (Hour, Day, Month, Year) were extracted from the datetime index.  
3. Feature Selection:  
 - Selected temporal features as predictors.  
 - The target variable was electric load (AEP\_MW).  
4. Model Training:  
 - A Random Forest Regressor with 100 decision trees was trained using an 80-20 train-test split.  
5. Model Evaluation:  
 - Evaluated model accuracy using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).  
6. Result Visualization:  
 - Plotted actual vs. predicted load values for sample test data.

**6. Code Used**

# 📦 Install & Import Required Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

# 📥 Load Uploaded Dataset

data = pd.read\_csv('AEP\_hourly.csv')

# 📌 View first few rows

print(data.head())

# 📌 Convert 'Datetime' column to datetime type

data['Datetime'] = pd.to\_datetime(data['Datetime'])

# 📌 Set 'Datetime' as the index

data.set\_index('Datetime', inplace=True)

# 📌 Check for missing values

print("\nMissing values:\n", data.isnull().sum())

# 📌 Drop rows with missing values (if any)

data.dropna(inplace=True)

# 📊 Plot Load Data

plt.figure(figsize=(15,5))

plt.plot(data['AEP\_MW'])

plt.title('Electric Load (AEP\_MW) Over Time')

plt.xlabel('Date')

plt.ylabel('Load (MW)')

plt.grid(True)

plt.show()

# 📌 Feature Engineering: Extract time-based features

data['Hour'] = data.index.hour

data['Day'] = data.index.day

data['Month'] = data.index.month

data['Year'] = data.index.year

# 📦 Define Features (X) and Target (y)

X = data[['Hour', 'Day', 'Month', 'Year']]

y = data['AEP\_MW']

# 📊 Split Data into Training and Testing sets (80-20 split)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 🔍 Initialize and Train Random Forest Regressor

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

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

# 📈 Predict on Test Data

y\_pred = model.predict(X\_test)

# 📊 Model Evaluation

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

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print(f"\nMean Absolute Error (MAE): {mae:.2f} MW")

print(f"Root Mean Square Error (RMSE): {rmse:.2f} MW")

# 📉 Plot Actual vs Predicted Load (Sample of 200 points)

plt.figure(figsize=(12,5))

plt.plot(y\_test.values[:200], label='Actual')

plt.plot(y\_pred[:200], label='Predicted')

plt.title('Actual vs Predicted Electric Load (Sample)')

plt.xlabel('Sample Number')

plt.ylabel('Load (MW)')

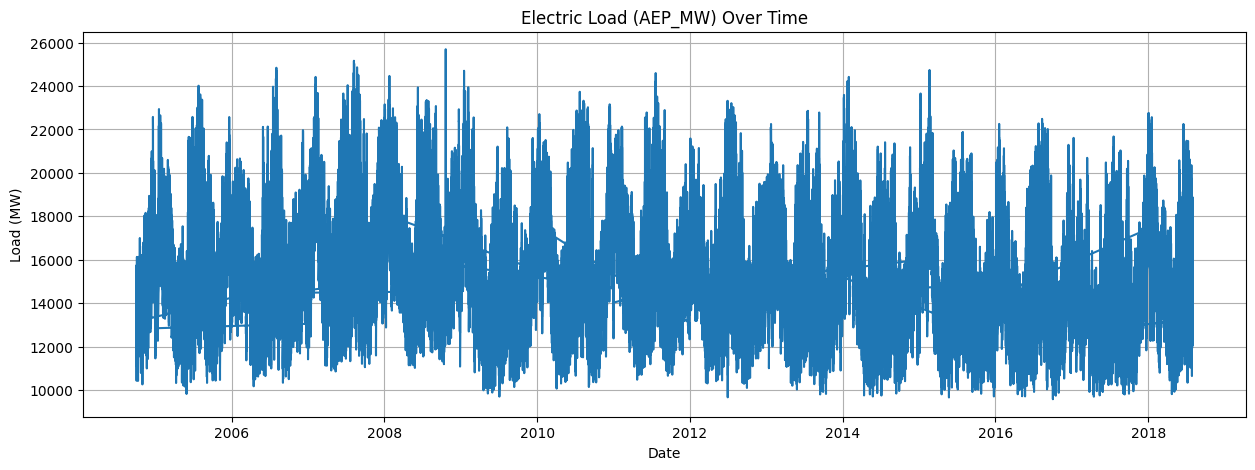
plt.legend()

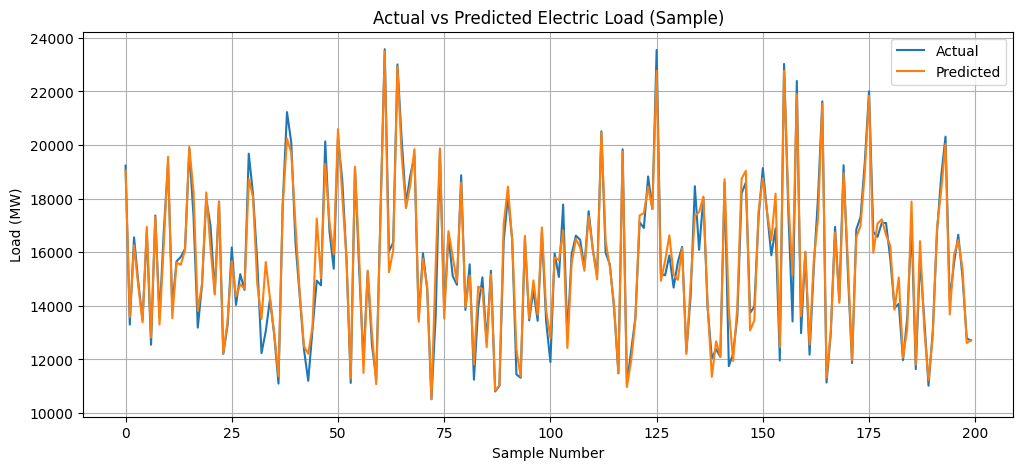
plt.grid(True)

plt.show()

**7. Results and Discussion**

The Random Forest Regressor model provided satisfactory forecasting accuracy based on evaluation metrics. A sample graph visualizing actual vs predicted load values was plotted for better interpretation of the model's performance.  
  
Performance Metrics:  
- Mean Absolute Error (MAE: 424.94MW  
- Root Mean Square Error (RMSE): 664.81 MW  
  
The model successfully captured hourly, daily, and seasonal load patterns from the dataset.





**8. Conclusion**

The AI/ML-based load forecasting model developed in this project demonstrates the potential of data-driven techniques for reliable electric load forecasting. By accurately predicting short-term load values, such models can significantly improve grid reliability, reduce operational costs, and enable proactive power system management.

**9. Future Scope**

Future enhancements to this project can include:  
- Comparing multiple AI/ML models (LSTM, XGBoost, Gradient Boosting, etc.)  
- Integrating additional features like weather data, holidays, and special events  
- Deploying the forecasting model on cloud-based platforms for real-time smart grid applications  
- Automating hyperparameter optimization for improved accuracy