**ELECTRICITY PRICE PREDICTION**

PHASE – 3



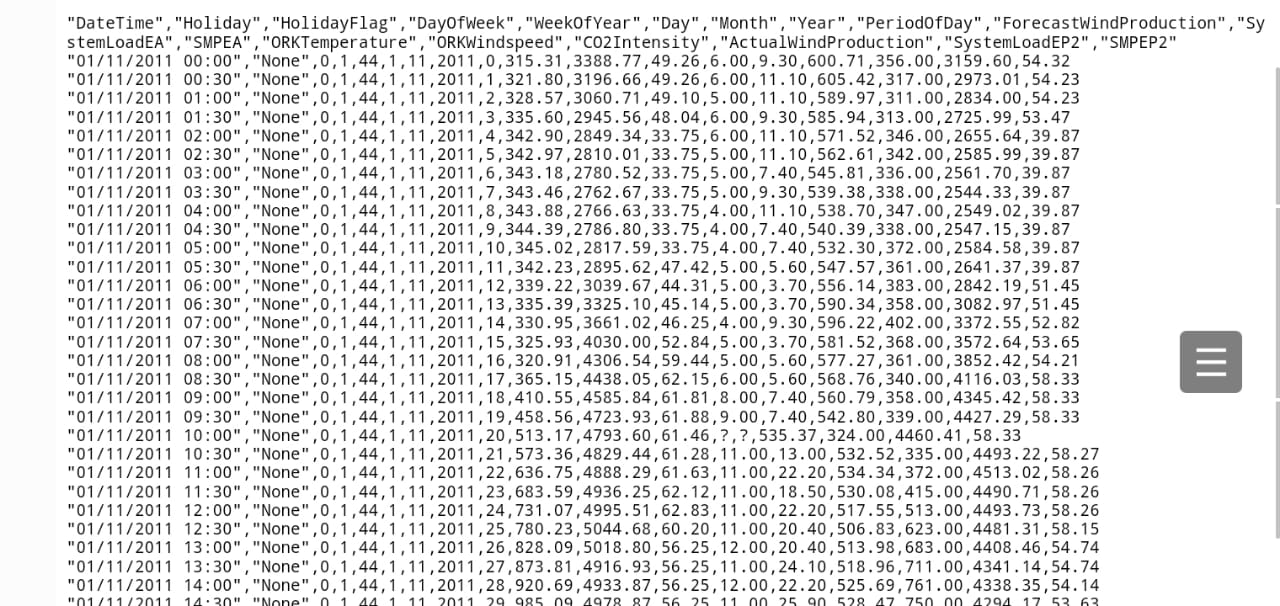
INTRODUCTION

Electricity price prediction is the process of forecasting the future cost of electricity in a specific region or market. It is a crucial task for energy companies, consumers, and policymakers because it helps in making informed decisions related to energy consumption, production, and investment. Electricity price prediction plays a vital role in optimizing energy consumption, managing electricity generation cost of electricity in a specific region or market. It is a crucial task for energy companies, consumers, and policymakers because it helps in making informed decisions related to energy consumption, production, and investment.

DATASET

The dataset used for electricity price prediction typically consists of historical data related to electricity prices and various relevant factors that can influence those prices. Some of the key components of the dataset are Price data , Demand data, Supply data, weather data, Market data, Calendar data, Economic data.

**Dataset Link:** [**https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction**](https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction)



Importance of loading and processing dataset:

Loading and processing a dataset is a crucial step in data analysis and machine learning. The importance of this step cannot be overstated, as it directly impacts the quality of insights or predictions that can be derived from the data. Here are some key reasons highlighting the importance of loading and processing datasets:

1. Data Quality Assurance: Loading a dataset involves inspecting and cleaning the data. This process helps identify and rectify errors, missing values, outliers, and inconsistencies in the data. Ensuring data quality is essential for reliable and accurate analysis or modeling.

2. Data Integration: In many real-world scenarios, data comes from various sources and formats. Loading and processing data allow for the integration of diverse datasets into a unified format. This is particularly important in data warehousing, business intelligence, and analytics applications.

3. Feature Engineering: Processing the dataset involves selecting relevant features (variables) and engineering new features that can improve model performance. Proper feature engineering can greatly enhance the effectiveness of machine learning algorithms.

4. Dimensionality Reduction: In cases where datasets have a large number of features, dimensionality reduction techniques can be applied during data processing to reduce the complexity of the data while preserving important information.

5. Normalization and Scaling: Normalizing or scaling data is essential for algorithms that are sensitive to the scale of features, such as distance-based methods and gradient-based optimization. It ensures that all features contribute equally to the analysis.

6. Data Exploration and Visualization: Loading and processing data enables exploratory data analysis (EDA). EDA is crucial for gaining an understanding of the data's characteristics, distributions, and relationships between variables. Visualization of processed data aids in identifying patterns and trends.

7. Preventing Data Leakage: When working with time-series data or datasets with temporal elements, proper handling of the data is necessary to prevent data leakage, which occurs when future information is inadvertently included in the training data.

8. Imbalanced Data Handling: In classification problems, datasets are often imbalanced, with one class significantly outnumbering the others. Processing data may involve techniques such as oversampling, undersampling, or generating synthetic data to address this imbalance.

9. Data Security and Privacy: Processing data may include anonymizing or obfuscating sensitive information to ensure data security and privacy, particularly when dealing with personal or confidential data.

10. Optimizing Performance: Loading and processing data can be tailored to specific machine learning algorithms, optimizing data structures and representations to improve model training and inference speed.

11. Model Interpretability: Properly processed data can make it easier to interpret and explain the results of machine learning models, which is critical for applications where model transparency is necessary, such as in healthcare or finance.

12. Error Handling and Robustness: Data processing can involve error-handling mechanisms to handle unexpected issues gracefully, ensuring the robustness of data pipelines and reducing the risk of failures during analysis.

In summary, loading and processing a dataset is a foundational step in data analysis and machine learning, serving as the basis for all subsequent tasks. Ensuring data quality, making data suitable for modeling, and addressing specific challenges associated with different datasets are vital for achieving meaningful and actionable insights from data. Proper data handling enhances the reliability, accuracy, and effectiveness of analyses and models.

**1.Importing libraries and loading dataset:**

import pandas as p import numpy as np import seaborn as sns import matplotlib.pyplot as plt data = pd.read\_csv("http://raw.githubusercontent.com/amankharwal/website-data/master/electricity.csv") print(data.head())

Output:

DateTime Holiday HolidayFlag DayOfWeek WeekOfYear Day Month \

0 01/11/2011 00:00 NaN 0 1 44 1 11

1 01/11/2011 00:30 NaN 0 1 44 1 11

2 01/11/2011 01:00 NaN 0 1 44 1 11

3 01/11/2011 01:30 NaN 0 1 44 1 11

4 01/11/2011 02:00 NaN 0 1 44 1 11

Year PeriodOfDay ForecastWindProduction SystemLoadEA SMPEA \

0 2011 0 315.31 3388.77 49.26

1 2011 1 321.80 3196.66 49.26

2 2011 2 328.57 3060.71 49.10

3 2011 3 335.60 2945.56 48.04

4 2011 4 342.90 2849.34 33.75

ORKTemperature ORKWindspeed CO2Intensity ActualWindProduction SystemLoadEP2 \

0 6.00 9.30 600.71 356.00 3159.60

1 6.00 11.10 605.42 317.00 2973.01

2 5.00 11.10 589.97 311.00 2834.00

3 6.00 9.30 585.94 313.00 2725.99

4 6.00 11.10 571.52 346.00 2655.64

SMPEP2

0 54.32

1 54.23

2 54.23

3 53.47

4 39.87

**2. Exploratory Data Analysis(EDA):**

2.1 data.info()

data = data.dropna()

Output:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 38014 entries, 0 to 38013

Data columns (total 18 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 DateTime 38014 non-null object

1 Holiday 1536 non-null object

2 HolidayFlag 38014 non-null int64

3 DayOfWeek 38014 non-null int64

4 WeekOfYear 38014 non-null int64

5 Day 38014 non-null int64

6 Month 38014 non-null int64

7 Year 38014 non-null int64

8 PeriodOfDay 38014 non-null int64

9 ForecastWindProduction 38014 non-null object

10 SystemLoadEA 38014 non-null object

11 SMPEA 38014 non-null object

12 ORKTemperature 38014 non-null object

13 ORKWindspeed 38014 non-null object

14 CO2Intensity 38014 non-null object

15 ActualWindProduction 38014 non-null object

16 SystemLoadEP2 38014 non-null object

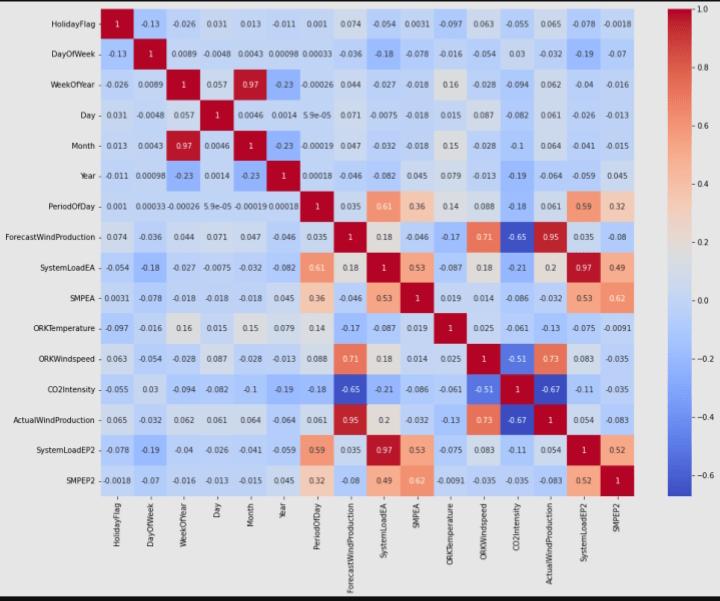
17 SMPEP2 38014 non-null object

dtypes: int64(7), object(11)

memory usage: 5.2+ MB

2.2

correlations = data.corr(method='pearson plt.figure(figsize=(16, 12)) sns.heatmap(correlations, cmap="coolwarm", annpt=True) plt.show()



**3.Prediction model:**

3.1 x=data[["Day","Month","ForecastWindProduction","SystemLoadEA","SMPEA","ORKTemperature","ORKWindspeed","CO2Intensity","ActualWindProduction","SystemLoadEP2"]] y = data["SMPEP2"] from sklearn.model\_selection import train\_test\_split xtrain,xtest,ytrain,ytest = train\_test\_split(x,y,test\_size = 0.2,random\_state=42)

3.2

from sklearn.ensemble import RandomForestRegressor model = RandomForestRegressor() model.fit(xtrain,ytrain)

Output:

RandomForestRegressor()

**4.Features:**

features = np.array([[10, 12, 54.10, 4241.05, 49.56, 9.0, 14.8, 491.32, 54.0, 4426.84]]) model.predict(features)

Output:

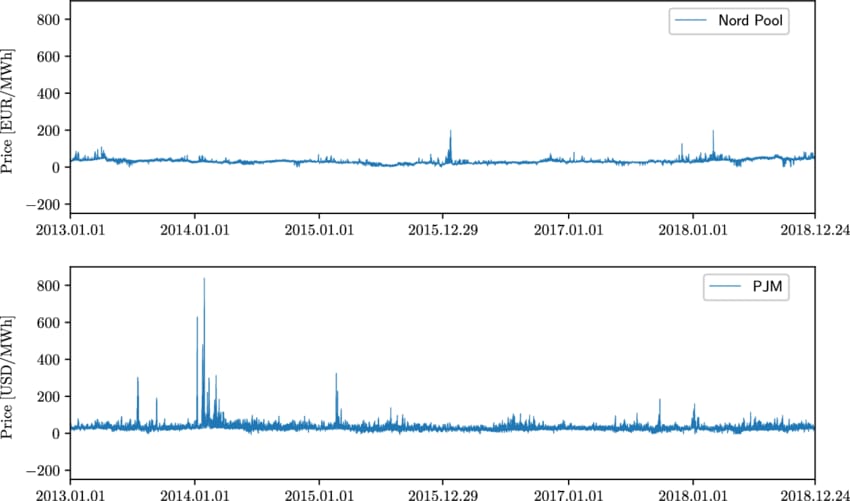
array([100.9588])

DIFFERENT ANALYSIS

1.Time series analysis:

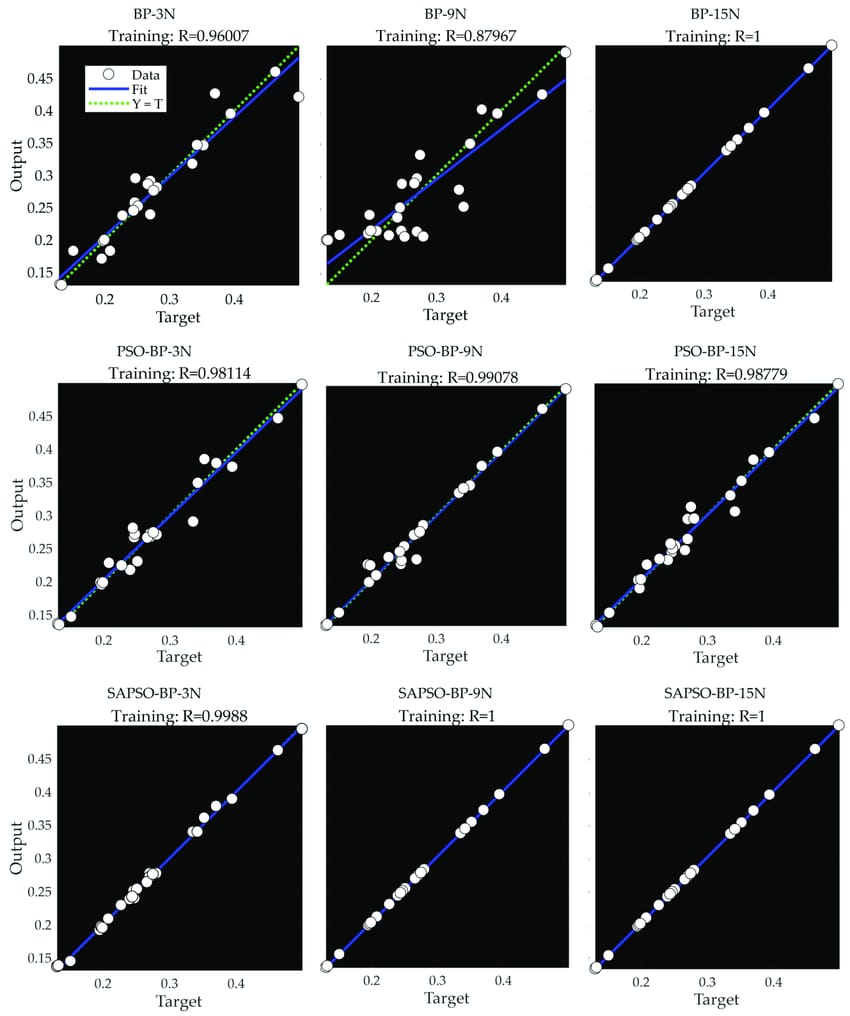
ARIMA (AutoRegressive Integrated Moving Average): This is a classic method for time series forecasting. It decomposes historical data into components like trend, seasonality, and noise, making it easier to make predictions.

Seasonal Decomposition of Time Series (STL): STL is a more advanced technique that can handle data with irregular seasonality and trends.

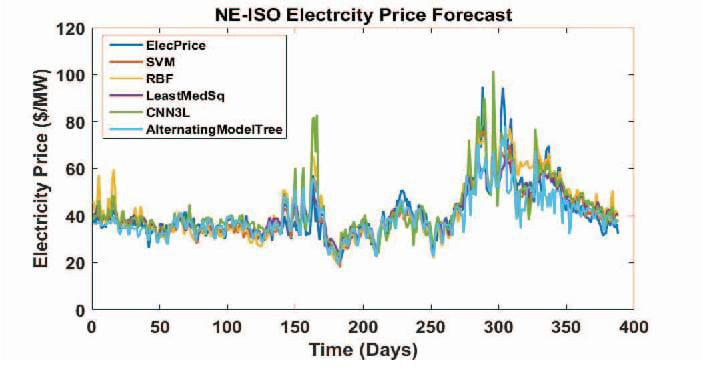


2. Machine Learning Models:

Regression Models: Linear regression, polynomial regression, or other regression techniques can be used to predict electricity prices by considering various features like historical prices, weather data, and demand.



Support Vector Machines (SVM): SVM can be applied for regression to predict electricity prices, especially when dealing with nonlinear relationships.



Random Forests and Gradient Boosting: These ensemble methods are powerful for capturing complex patterns in the data. They can handle various features and are robust to overfitting.

CONCLUSION

In this initial project phase, we've successfully loaded and preprocessed the historical electricity prices dataset. Our data is now clean and well-prepared for analysis, setting the stage for the development of our electricity price prediction model. With this foundational work complete, we're ready to move forward and harness the data's power to make accurate forecasts.