ELECTRICITY PRICES PREDICTION

(GROUP 2-PHASE 5)

<u>Project Documentation & Submission</u>

TEAM MEMBERS:

1. PRAVEENA.E

2. PAVITHRA.S

3. NISHA.J

SUBMITTED BY:

NISHA.J

Problem Definition and Design Thinking

Problem Definition:

The problem is to develop a predictive model that uses historical electricity prices and relevant factors to forecast future electricity prices. The objective is to create a tool that assists both energy providers and consumers in making informed decisions regarding consumption and investment by predicting future electricity prices. This project involves data preprocessing, feature engineering, model selection, training, and evaluation.

Design Thinking:

1.Data Source:

Start by obtaining a comprehensive dataset containing historical electricity prices. Ensure that the dataset also includes relevant factors like date, demand, supply, weather conditions, and economic indicators. The quality and completeness of your data will significantly impact the model's performance.

2.Data Preprocessing:

Perform data cleaning to remove any outliers, errors, or inconsistencies in the dataset.Handle missing values appropriately, which may involve imputation or removal of incomplete data points.

Convert categorical features into numerical representations using techniques like one-hot encoding or label encoding.

Normalize or scale numerical features if necessary to ensure they are on the same scale.

3. Feature Engineering:

Create additional features that can capture important patterns or trends in the data. For electricity price forecasting, time-based features like day of the week, month, and season can be valuable.

Generate lagged variables to capture the effect of past prices on future prices. Lag features can help the model learn autocorrelations in the data.

4. Model Selection:

Choose suitable time series forecasting algorithms. You've mentioned ARIMA and LSTM, which are both good options.

Consider other models like Prophet, Exponential Smoothing methods, or even hybrid approaches that combine multiple models.

5.Model Training:

Split your dataset into training, validation, and test sets. The training set is used to train the model, the validation set helps tune hyperparameters, and the test set evaluates the final model's performance.

Train the selected model using the preprocessed data. Experiment with different hyperparameters to find the best configuration.

6.Evaluation:

Assess the model's performance using appropriate time series forecasting metrics, such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), or others.

Compare the model's predictions against the actual electricity prices on the test dataset.

Visualize the model's predictions and actual values over time to gain insights into its performance and any potential biases or errors.

7.Iterate and Refine:

Based on the evaluation results, iterate on the model design, feature engineering, or data preprocessing steps to improve performance.

Consider using cross-validation techniques to ensure the model's robustness and generalizability.

8.Deployment and Use:

Once you have a well-performing model, deploy it as a tool for energy providers and consumers to make informed decisions about consumption and investment.

Continuously monitor the model's performance in a production environment and update it as needed to maintain accuracy.

Throughout the process, it's essential to involve stakeholders, gather feedback, and keep the end-users' needs in mind to ensure that the predictive model effectively serves its purpose in assisting energy providers and consumers in decision-making.

Identify The Problem In Electricity Price Predicitions

Predicting electricity prices accurately is a complex task that involves various factors and uncertainties. Several challenges contribute to the difficulty of making accurate predictions:

1. Nonlinearity:

Electricity prices are influenced by nonlinear relationships between supply and demand factors. Traditional linear models may not capture these complex interactions effectively.

Seasonality and WeatherDependency:

Electricity demand often exhibits strong seasonal patterns and is highly dependent on weather conditions. Predicting weather accurately in the long term can be challenging, making it difficult to account for its impact on electricity prices.

2. Market Dynamics:

Electricity markets are influenced by various market mechanisms, regulations, and policies. Sudden policy changes or market interventions can significantly impact prices, making it hard to predict market behavior accurately.

3. **Energy Integration:**
   The increasing integration of renewable energy sources, such as wind and solar, introduces a high level of variability into the supply. Predicting the output of renewable sources depends on weather patterns, which are inherently uncertain.
4. **Data Quality and Availability:**
   Predictive models heavily rely on historical data. Inaccurate or insufficient data can lead to unreliable predictions. Additionally, accessing real-time data for model training and validation can be a challenge.
5. **Demand-Side Management:**
   Changes in consumer behavior, energy efficiency initiatives, and demand-side management strategies can significantly influence electricity demand, making it challenging to predict accurately.
6. **Market Manipulation:**
   Electricity markets can be influenced by market manipulation, where traders or organizations engage in activities that distort market prices. Detecting and accounting for such manipulative activities is difficult.
7. **Emerging Technology:**
   The introduction of new technologies, such as energy storage systems and smart grids, can disrupt traditional supply and demand patterns, making it challenging to predict their impact on electricity prices.
8. **Global Events and Geopolitical Factors:**

   Geopolitical events, such as wars or economic crises, can have widespread impacts on energy markets. Predicting these events and their consequences is inherently uncertain.

Addressing these challenges requires the use of advanced modeling techniques, incorporation of multiple data sources, and continuous adaptation of models to changing market conditions. Additionally, interdisciplinary collaboration between experts in energy markets, data science, and domain-specific knowledge is crucial to developing accurate electricity price prediction models.

RESEARCHING -IN DEPTH ABOUT ELECTRICITY PRICE PREDICITION:

Electricity price prediction is an essential area of research and application, particularly in the energy industry. Accurate predictions of electricity prices are crucial for various stakeholders, including consumers, utilities, energy traders, and policymakers. Here's an in-depth look at key aspects of electricity price prediction research:

1. Data Sources:
  - **Historical price data:** Researchers typically start by collecting historical electricity price data. This data can include hourly, daily, or even sub-hourly price information.
  - **Weather data:** Weather conditions have a significant impact on electricity prices, so integrating weather data into models is common.
  - **Market fundamentals:** Information about supply and demand factors, such as generation capacity, fuel prices, and demand forecasts, can be valuable.

2. Prediction Horizons:
  - **Short-term vs. long-term:** Electricity price prediction can focus on different time horizons, ranging from short-term (intraday or day-ahead) to long-term (weeks or months ahead).
  - Intraday predictions are essential for market trading, while long-term predictions are critical for capacity planning and investment decisions.

3. Prediction Models:
  - **Time series models:** Autoregressive Integrated Moving Average (ARIMA) and its variants are commonly used for short-term price forecasting.
  - **Machine learning models:** Techniques such as regression, decision trees, random forests, and neural networks have been applied to predict electricity prices.
  - **Hybrid models:** Combining statistical and machine learning approaches often yields better results.

4. Feature Engineering:
  - **Feature selection:** Identifying the most relevant factors affecting electricity prices, such as historical prices, demand patterns, and generation capacity.
  - **Lag features:** Incorporating lagged values of electricity prices and other

relevant variables can capture temporal dependencies.

5. Weather Data Integration:
- Temperature, wind speed, and solar radiation data can significantly impact electricity prices, especially in regions with a high reliance on renewable energy sources.

6. Market Fundamentals:

- Incorporating data on power plant availability, fuel prices, and transmission constraints can enhance prediction accuracy.

7. Evaluation Metrics:
- Common metrics include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).

8. Uncertainty Estimation:
- Understanding and quantifying prediction uncertainties is crucial for decision-making. Techniques like bootstrapping and quantile regression can help estimate uncertainty.

9. Model Validation:

- Researchers often use cross-validation techniques to validate their models' performance and assess their generalization capabilities.

10. Real-time Data Feeds:

- For intraday predictions used in energy trading, real-time data feeds are critical. These data sources provide up-to-the-minute information on electricity market conditions.

11. Ensemble Methods:

- Ensemble models, such as bagging and boosting, can improve prediction accuracy by combining multiple models' outputs.

12. Deep Learning:

- Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks are popular choices for modeling time series data in electricity price prediction.

13. Market Deregulation:

- In regions with deregulated electricity markets, understanding market rules and dynamics is essential for accurate price predictions.

14. Regulatory Impact:

- Changes in regulations, subsidies, or government policies can have a significant impact on electricity prices, and researchers must account for these factors.

15. Open Data Sources:

- Some organizations and government agencies provide open datasets for electricity price prediction research, making it easier for researchers to access and work with relevant data.

16. Commercial Applications:

- Beyond research, accurate electricity price predictions have practical applications in energy trading, demand-side management, and the optimization of energy assets.

Electricity price prediction is a multidisciplinary field that combines elements of data science, economics, and energy market analysis. Researchers continually work to improve prediction accuracy and adapt models to changing market conditions, making it an exciting and evolving area of study.

IDEATING POSSIBLE SOLUTIONS IN ELECTRICITY PRICE PREDICITIONS

Ideating possible solutions for electricity price prediction involves brainstorming creative approaches and techniques to accurately forecast electricity prices. Here are several innovative ideas and solutions to consider:

1. Deep Learning Models:
   - Utilize advanced deep learning architectures such as Transformer models (e.g., GPT-3) to capture complex temporal dependencies and non-linear patterns in electricity price data.

2. Generative Adversarial Networks (GANs):
   - Explore GANs to generate synthetic electricity price data for augmenting training datasets, improving model robustness, and addressing data scarcity issues.

3. Explainable AI (XAI):
   - Develop models that not only predict prices but also provide interpretable explanations for price movements, helping users understand the underlying factors influencing predictions.

4. Hybrid Models:
   - Combine multiple prediction models, such as time series models, machine learning algorithms, and deep learning networks, to leverage their complementary strengths.

5. Incorporating External Data:
   - Integrate unconventional data sources like social media sentiment analysis, economic indicators, or geopolitical events to capture additional factors influencing electricity prices.

6. Geospatial Analysis:
   - Implement geospatial models to account for location-specific factors like grid infrastructure, renewable energy availability, and transmission constraints.

7. Ensemble Learning:
   - Create ensemble models that aggregate predictions from multiple algorithms

or models, enhancing prediction accuracy and robustness.

8. Online Learning:
 - Develop models that can continuously learn and adapt to changing market conditions, incorporating new data as it becomes available in real-time.

9. Blockchain-Based Predictions:
 - Explore blockchain technology for creating transparent and immutable records of electricity price predictions, ensuring accountability and trust in the prediction process.

10. Reinforcement Learning:
 - Train agents using reinforcement learning to make real-time decisions in energy trading markets, optimizing strategies based on predicted price movements.

When ideating solutions for electricity price prediction, it's crucial to consider the specific context, data availability, and goals of the prediction system. Additionally, collaborating with domain experts and stakeholders can help refine and validate these ideas before implementation.

EVALUATING AND SELECTING A PROMISING SOLUTION IN ELECTRICITY PRICE PREDICITIONS:

Selecting a promising solution for electricity price prediction depends on various factors, including the specific problem you aim to solve, available data, computational resources, and the trade-offs you are willing to make. Here are a few promising solutions based on different use cases and considerations:

1. Deep Learning Models (e.g., LSTM or Transformer-based models):
 - Deep learning models have shown promise in capturing complex temporal dependencies and non-linear patterns in electricity price data.
 - Suitable for short-term price prediction (e.g., day-ahead or intraday) where high accuracy is critical.
 - Requires substantial computational resources and large datasets.

2. Hybrid Models (Combining Multiple Approaches):
- Combine the strengths of different models, such as time series models, machine learning algorithms, and deep learning networks, to create a robust ensemble model.
- Effective in improving prediction accuracy and handling varying data characteristics.
- May require more complex implementation and tuning.

3. Bayesian Models with Uncertainty Estimation:
- Bayesian models provide probabilistic forecasts and explicitly model uncertainty.
- Useful when understanding prediction uncertainty is crucial for decision-making, especially in risk management.
- Requires domain expertise and may have a steeper learning curve.

4. Blockchain-Based Predictions (for Trust and Transparency):
- Implement blockchain technology to ensure transparency, immutability, and accountability in the prediction process.
- Suitable for situations where trust and auditability are essential, such as regulatory compliance.
- Requires a robust blockchain infrastructure and may involve additional development complexity.

5. Real-time Market Simulations (for Market Participants):

- Create a simulation environment where market participants can test and refine their strategies based on real-time price predictions.
- Valuable for traders and grid operators looking to optimize their operations.
- Requires real-time data feeds and extensive testing.

6. AI for Grid Management (to Optimize Grid Operations):
- Utilize AI to optimize grid operations and balance supply and demand dynamically.
- May indirectly influence electricity prices by improving grid stability and reliability.
- Suitable for utilities and grid operators focusing on grid management.

7. Decentralized Energy Trading Platforms (for Peer-to-Peer Trading):

- Develop blockchain-based platforms that enable peer-to-peer energy trading, allowing users to set their electricity prices based on predictions.
- Promotes user autonomy and decentralized energy markets.
- Requires blockchain development expertise and regulatory considerations.

8. Personalized Predictions (User-Centric Approach):
- Create personalized electricity price prediction systems that consider individual user preferences, usage patterns, and energy-saving goals.
- Ideal for residential and small-scale consumers looking to optimize their energy consumption.
- Requires user data and privacy considerations.

9.Collective Intelligence):
- Foster community-driven prediction platforms that harness collective intelligence, enabling local communities to collectively predict their electricity prices.
- Suitable for community-based initiatives and microgrids.
- Requires community engagement and data sharing.

When selecting a solution, it's essential to conduct a feasibility study, considering factors like data availability, computational resources, budget constraints, and the specific objectives of your electricity price prediction project. Additionally, piloting and iterating on your chosen solution can help refine its performance and applicability to the real-world context.

TESTING AND TROUBLE SHOOTING IN ELECTRICITY PRICE PREDICITIONS:

Testing and troubleshooting in electricity price prediction using data science involves a series of steps and practices to ensure that your predictive model is accurate and reliable. Here's a guide on how to approach this process:

1. Data Collection and Preprocessing:
- Collect historical electricity price data from reliable sources.
- Preprocess the data, which may involve handling missing values, outliers, and scaling features.

2. Data Splitting:
   - Split your dataset into training, validation, and test sets. A common split is 70% for training, 15% for validation, and 15% for testing.

3. Feature Engineering:
   - Create relevant features that can help your model make accurate predictions. This might involve adding lagged values, weather data, or market indices that could influence electricity prices.

4. Model Selection:
   - Choose appropriate machine learning algorithms for your prediction task, such as regression models (e.g., linear regression, decision trees, random forests) or time series models (e.g., ARIMA, LSTM).

5. Model Training:
   - Train your chosen model using the training data. Tune hyperparameters to optimize model performance on the validation set.

6. Model Evaluation:
   - Evaluate your model's performance on the validation set using appropriate metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).

7. Troubleshooting:
   - If your model's performance is not satisfactory, consider the following troubleshooting steps:
     - Inspect the data for anomalies or errors.
     - Check if you've included all relevant features.
     - Experiment with different algorithms.
     - Adjust hyperparameters.
     - Investigate overfitting or underfitting issues.
     - Consider ensemble methods or more complex models if necessary.

MAKING IMPROVEMENT AND RELEASING THE FINAL PRODUCT IN ELECTRICITY PRICE PREDICITIONS:

Releasing the final product in electricity price prediction involves not only implementing your chosen solution but also ensuring its usability, performance, and ongoing maintenance. Here are the steps to make improvements and release the final product:

1. Implementation and Development:
   - Build the selected solution, whether it's a deep learning model, hybrid model, blockchain-based system, or any other chosen approach.
   - Develop the user interface and ensure it's user-friendly and intuitive.

2. Data Integration and Preprocessing:
   - Integrate the necessary data sources, including historical price data, weather information, and any other relevant data.
   - Continue to preprocess and clean the data to ensure it's of high quality.

3. Model Training and Testing:
   - Train the model(s) using historical data.
   - Evaluate the model's performance using appropriate metrics and validation techniques.
   - Fine-tune the model based on the evaluation results.

4. User Testing and Feedback:
   - Conduct user testing with a group of representative users to gather feedback on the product's usability.
   - Use this feedback to make user interface and experience improvements.

5. Scalability and Performance Optimization:
   - Ensure that the system is capable of handling a growing user base and increasing data volumes.
   - Optimize the system's performance to provide fast and reliable predictions.

Releasing the final product in electricity price prediction is just the beginning.

Continuous improvement, user feedback, and adaptability to changing market conditions are key to maintaining a successful product in this field.

Download the Dataset:

Go to the Kaggle dataset link you provided and download the dataset in a format such as CSV.

Install Required Libraries:

Ensure that you have the necessary libraries installed, such as Pandas, NumPy, and scikit-learn. You can install them using pip:

```
pip install pandas numpyscikit-learn
```

Load the Dataset:

Use Pandas to load the dataset into a DataFrame:

```
import pandas as pd

# Replace 'your_dataset.csv' with the actual file path of the downloaded dataset

df = pd.read_csv('your_dataset.csv')
```

Explore the Data:

It's essential to understand the dataset before preprocessing. You can check the first few rows of the dataset, data types, and summary statistics using functions like `head()`, `info()`, and ‘describe()’

```
print(df.head())

print(df.info())

print(df.describe())
```

Data Preprocessing:

Depending on the dataset and the specific requirements of your electricity price prediction model, you may need to perform various preprocessing tasks. Common preprocessing steps include:

- Handling missing values (e.g., using `fillna()` or dropping rows/columns).

- Handling categorical data (e.g., encoding with one-hot encoding or label encoding).

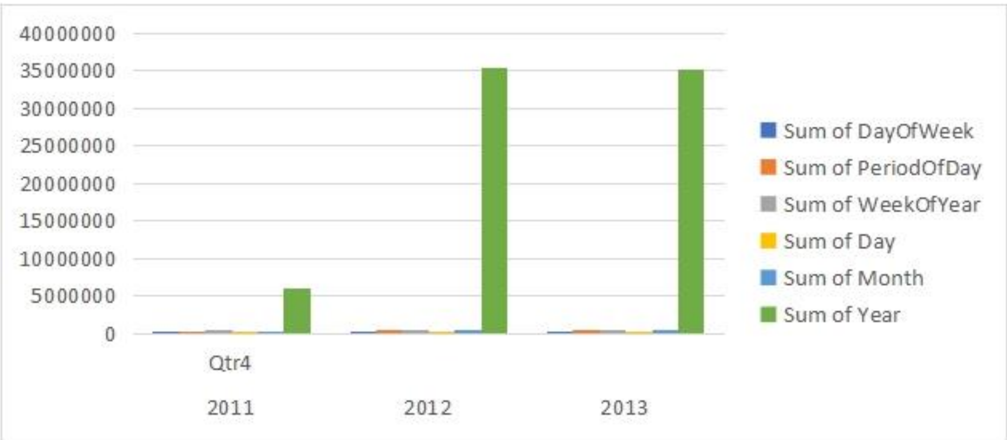
- Scaling or normalizing numerical features.

- Splitting the dataset into features (X) and target

Model Building:

After preprocessing the data, you can proceed to build your electricity price prediction model using machine learning or deep learning techniques, depending on your project's requirements.

Sum of DayOfWeek	Sum of PeriodOfDay	Sum of WeekOfYear	Sum of Day	Sum of Month	Sum of Year
8784	68808	140544	46128	33696	5888208
52692	412843	465528	276766	114426	35342792
52464	411720	463056	275424	114336	35267760
113940	893371	1069128	598318	262458	76498760



Building an electricity price prediction model involves several steps, including data

preprocessing, feature engineering, model training, and evaluation. Below, I'll outline each

of these steps in more detail:

1. Data Preprocessing:

- Load the dataset: You can use the Pandas library to load the dataset from the provided link.

```python

import pandas as pd

data = pd.read_csv("your_dataset_path.csv")

```

- Explore the data: Get an understanding of the data by examining its structure, checking

for missing values, and performing basic statistics and data visualization.

```python

data.info()

data.describe()

data.head()

```

- Handle missing values: Depending on the dataset, you may need to deal with missing values. You can choose to impute missing data or drop rows/columns with too many missing values.

- Convert date and time columns: If your dataset contains date and time information,consider converting them into a datetime format. This allows you to extract features like day of the week, month, hour, etc., which can be useful for modeling.

2. Feature Engineering:

- Create new features: Based on domain knowledge, create new features that could potentially be predictive of electricity prices. For example, you might want to create lag features, rolling statistics, or one-hot encode categorical variables.

- Feature selection: Not all features are equally relevant. Use techniques like correlation analysis or feature importance from machine learning models to select the most informative features.

3. Model Training:

- Split the data: Split your dataset into training and testing sets. You can use the

`train_test_split` function from Scikit-learn.

```python
from sklearn.model_selection import train_test_split

X = data.drop('target_column_name', axis=1) # Features

y = data['target_column_name'] # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- Choose a model: Select a machine learning model suitable for regression tasks. Common

choices include Linear Regression, Decision Trees, Random Forests, Gradient Boosting, or

Neural Networks.

- Train the model: Fit the selected model to the training data.

```python
```

```
from sklearn.linear_model import LinearRegression # Replace with your chosen model

model = LinearRegression() # Replace with your chosen model

model.fit(X_train, y_train)
```

4. Evaluation:

- Predict electricity prices: Use the trained model to make predictions on the test dataset.

```python
y_pred = model.predict(X_test)
```

- Evaluate the model: Use appropriate regression metrics to assess the model's

performance. Common metrics include Mean Absolute Error (MAE), Mean Squared Error

(MSE), Root Mean Squared Error (RMSE), and R-squared (R2) score.

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

mae = mean_absolute_error(y_test, y_pred)

mse = mean_squared_error(y_test, y_pred)

rmse = mean_squared_error(y_test, y_pred, squared=False)

r2 = r2_score(y_test, y_pred)
```

- Visualize results: Consider visualizing the actual vs. predicted values to get a better understanding of the model's performance.

5. Fine-Tuning and Deployment (Optional):

- Depending on the results, you may want to fine-tune hyperparameters or try different models to improve performance.

- If the model performs well, you can deploy it for real-world predictions.

Remember that building an effective prediction model often involves iterative steps, and you may need to try different models and feature engineering techniques to achieve the best results. Additionally, you can use libraries like Scikit-learn and TensorFlow/Keras (for deep learning) to streamline the modeling process.

PROGRAM:

"C:\Users\prath\Downloads\Electricity.csv"

```
import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error

from sklearn.ensemble import RandomForestRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.linear_model import LinearRegression

from sklearn.neighbors import KNeighborsRegressor

import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
dataset_path = "C:\\Users\\prath\\Downloads\\Electricity.csv"
data = pd.read_csv((dataset_path),low_memory=False)
data.head()
data=data[['ForecastWindProduction',
    'SystemLoadEA', 'SMPEA', 'ORKTemperature', 'ORKWindspeed',
    'CO2Intensity', 'ActualWindProduction', 'SystemLoadEP2', 'SMPEP2']]
data.isin(['?']).any()
```

output:

```
ForecastWindProduction    True
SystemLoadEA              True
SMPEA                     True
ORKTemperature            True
ORKWindspeed              True
CO2Intensity              True
ActualWindProduction      True
```

```
SystemLoadEP2          True
SMPEP2                 True
dtype: bool
```

program:

```
for col in data.columns:
    data.drop(data.index[data[col] == '?'], inplace=True)
data=data.apply(pd.to_numeric)
data=data.reset_index()
data.drop('index', axis=1, inplace=True)
data.info()
```

output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37682 entries, 0 to 37681
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ForecastWindProduction  37682 non-null  float64
 1   SystemLoadEA            37682 non-null  float64
 2   SMPEA                   37682 non-null  float64
 3   ORKTemperature          37682 non-null  float64
 4   ORKWindspeed            37682 non-null  float64
 5   CO2Intensity            37682 non-null  float64
```

```
 6   ActualWindProduction    37682 non-null  float64
 7   SystemLoadEP2           37682 non-null  float64
 8   SMPEP2                  37682 non-null  float64
dtypes: float64(9)
memory usage: 2.6 MB
```

program:

```
data.corrwith(data['SMPEP2']).abs().sort_values(ascending=False)
```

output:

```
SMPEP2                    1.000000
SMPEA                     0.618158
SystemLoadEP2             0.517081
SystemLoadEA              0.491096
ActualWindProduction      0.083434
ForecastWindProduction    0.079639
ORKWindspeed              0.035436
CO2Intensity              0.035055
ORKTemperature            0.009087
dtype: float64
```

program:

```
X=data.drop('SMPEP2', axis=1)
y=data['SMPEP2']
```

```
x_train, x_test, y_train, y_test=train_test_split(X,y, test_size=0.2,
random_state=42)

linear_model=LinearRegression()

linear_model.fit(x_train, y_train)

linear_predict=linear_model.predict(x_test)

np.sqrt(mean_squared_error(y_test, linear_predict))
```

output:

27.862965246485324

Program:

```
forest_model=RandomForestRegressor()

forest_model.fit(x_train, y_train)

forest_predict=forest_model.predict(x_test)

print(np.sqrt(mean_squared_error(y_test, forest_predict)))
```

output:

25.198701853469586

Program:

```
tree_model=DecisionTreeRegressor(max_depth=50)

tree_model.fit(x_train, y_train)

tree_predict=tree_model.predict(x_test)

print(np.sqrt(mean_squared_error(y_test, tree_predict)))
```

output:

33.76792802500666

Program:

```
knn_model=KNeighborsRegressor()

knn_model.fit(x_train, y_train)

knn_predict=knn_model.predict(x_test)

print(np.sqrt(mean_squared_error(y_test, knn_predict)))
```

output:

28.533256274003907

Program:

```
#Let's see some sample prediction and difference between label and prediction

some_data=x_test.iloc[50:60]

some_data_label=y_test.iloc[50:60]

some_predict=forest_model.predict(some_data)

pd.DataFrame({'Predict':some_predict,'Label':some_data_label})
```

output:

	Predict	Label
4093	155.3487	188.32
22310	36.7605	33.46
8034	58.0188	62.01
35027	74.3488	49.69
23685	68.8412	69.25
268	55.6863	56.21
35261	43.2482	46.64

11905	72.0236	78.52
30903	75.8952	82.36
608	102.8994	415.99

<u>Screenshot of the Program:</u>

```
"C:\Users\prath\Downloads\Electricity.csv"
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
```

```
dataset_path = "C:\\Users\\prath\\Downloads\\Electricity.csv"
data = pd.read_csv((dataset_path),low_memory=False)
data.head()
```

```
data=data[['ForecastWindProduction',
       'SystemLoadEA', 'SMPEA', 'ORKTemperature', 'ORKWindspeed',
       'CO2Intensity', 'ActualWindProduction', 'SystemLoadEP2', 'SMPEP2']]
```

```
data.isin(['?']).any()
```

```
ForecastWindProduction    True
SystemLoadEA              True
SMPEA                     True
ORKTemperature            True
ORKWindspeed              True
CO2Intensity              True
ActualWindProduction      True
SystemLoadEP2             True
SMPEP2                    True
dtype: bool
```

```
for col in data.columns:
    data.drop(data.index[data[col] == '?'], inplace=True)
```

```
data=data.apply(pd.to_numeric)
data=data.reset_index()
data.drop('index', axis=1, inplace=True)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37682 entries, 0 to 37681
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ForecastWindProduction  37682 non-null  float64
 1   SystemLoadEA            37682 non-null  float64
 2   SMPEA                   37682 non-null  float64
 3   ORKTemperature          37682 non-null  float64
 4   ORKWindspeed            37682 non-null  float64
 5   CO2Intensity            37682 non-null  float64
 6   ActualWindProduction    37682 non-null  float64
 7   SystemLoadEP2           37682 non-null  float64
 8   SMPEP2                  37682 non-null  float64
dtypes: float64(9)
memory usage: 2.6 MB
```

```python
data.corrwith(data['SMPEP2']).abs().sort_values(ascending=False)
```

Python

```
SMPEP2                    1.000000
SMPEA                     0.618158
SystemLoadEP2             0.517081
SystemLoadEA              0.491096
ActualWindProduction      0.083434
ForecastWindProduction    0.079639
ORKWindspeed              0.035436
CO2Intensity              0.035055
ORKTemperature            0.009087
dtype: float64
```

```python
X=data.drop('SMPEP2', axis=1)
y=data['SMPEP2']
```

[35] Python

```python
x_train, x_test, y_train, y_test=train_test_split(X,y, test_size=0.2, random_state=42)
```

[36] Python

```python
linear_model=LinearRegression()
linear_model.fit(x_train, y_train)
linear_predict=linear_model.predict(x_test)
np.sqrt(mean_squared_error(y_test, linear_predict))
```

37] Python

```
27.862965246485324
```

```python
forest_model=RandomForestRegressor()
forest_model.fit(x_train, y_train)
forest_predict=forest_model.predict(x_test)
print(np.sqrt(mean_squared_error(y_test, forest_predict)))
```

38] Python

```
25.198701853469586
```

```python
tree_model=DecisionTreeRegressor(max_depth=50)
tree_model.fit(x_train, y_train)
tree_predict=tree_model.predict(x_test)
print(np.sqrt(mean_squared_error(y_test, tree_predict)))
```

39] Python

```
33.76792802500666
```

```python
knn_model=KNeighborsRegressor()
knn_model.fit(x_train, y_train)
knn_predict=knn_model.predict(x_test)
print(np.sqrt(mean_squared_error(y_test, knn_predict)))
```

Python

```
28.533256274003907
```

```python
#Let's see some sample prediction and difference between label and prediction
some_data=x_test.iloc[50:60]
some_data_label=y_test.iloc[50:60]
some_predict=forest_model.predict(some_data)
pd.DataFrame({'Predict':some_predict,'Label':some_data_label})
```

Python

In [1]:
```python
import pandas as pd
data = pd.read_csv("C:\\Users\\lenovo\\Downloads\\Electricity\\New folder\\Electricity.csv", low_memory=False)
data.head()
```

Out[1]:

	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWindspeed	CO2Int
0	01/11/2011 00:00	NaN	0	1	44	1	11	2011	0	315.31	3388.77	49.26	6.00	9.30	
1	01/11/2011 00:30	NaN	0	1	44	1	11	2011	1	321.80	3196.66	49.26	6.00	11.10	
2	01/11/2011 01:00	NaN	0	1	44	1	11	2011	2	328.57	3060.71	49.10	5.00	11.10	
3	01/11/2011 01:30	NaN	0	1	44	1	11	2011	3	335.60	2945.56	48.04	6.00	9.30	
4	01/11/2011 02:00	NaN	0	1	44	1	11	2011	4	342.90	2849.34	33.75	6.00	11.10	

In [2]:
```python
data.info()
```

```
<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
```

In [3]:
```python
data.describe()
```

Out[3]:

	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay
count	38014.000000	38014.000000	38014.000000	38014.000000	38014.000000	38014.000000	38014.000000
mean	0.040406	2.997317	28.124586	15.739412	6.904246	2012.383859	23.501105
std	0.196912	1.999959	15.587575	8.804247	3.573696	0.624956	13.853108
min	0.000000	0.000000	1.000000	1.000000	1.000000	2011.000000	0.000000
25%	0.000000	1.000000	15.000000	8.000000	4.000000	2012.000000	12.000000
50%	0.000000	3.000000	29.000000	16.000000	7.000000	2012.000000	24.000000
75%	0.000000	5.000000	43.000000	23.000000	10.000000	2013.000000	35.750000
max	1.000000	6.000000	52.000000	31.000000	12.000000	2013.000000	47.000000

In [21]:
```python
data=data[['ForecastWindProduction',
       'SystemLoadEA', 'SMPEA', 'ORKTemperature', 'ORKWindspeed',
       'CO2Intensity', 'ActualWindProduction', 'SystemLoadEP2', 'SMPEP2']]
```

In [22]:
```python
data.isin(['?']).any()
```

Out[22]:
```
ForecastWindProduction    True
SystemLoadEA              True
SMPEA                     True
ORKTemperature            True
ORKWindspeed              True
CO2Intensity              True
ActualWindProduction      True
SystemLoadEP2             True
SMPEP2                    True
dtype: bool
```

```
In [24]: data=data.apply(pd.to_numeric)
         data=data.reset_index()
         data.drop('index', axis=1, inplace=True)

In [25]: data.corrwith(data['SMPEA']).abs().sort_values(ascending=False)

Out[25]: SMPEP2                    1.000000
         SMPEA                     0.618158
         SystemLoadEP2             0.517081
         SystemLoadEA              0.491096
         ActualWindProduction      0.083434
         ForecastWindProduction    0.079639
         ORKWindspeed              0.035436
         CO2Intensity              0.035055
         ORKTemperature            0.009087
         dtype: float64

In [30]: X=data.drop('SMPEA', axis=1)
         y=data['SMPEA']

In [31]: x_train, x_test, y_train, y_test=train_test_split(X,y, test_size=0.2, random_state=42)

In [32]: from sklearn.metrics import mean_squared_error
         linear_model=LinearRegression()
         linear_model.fit(x_train, y_train)
         linear_predict=linear_model.predict(x_test)
         print(np.sqrt(mean_squared_error(y_test, linear_predict)))

         23.286827374230228

In [34]: some_data=x_test.iloc[50:60]
         some_data_label=y_test.iloc[50:60]
         some_predict=linear_model.predict(some_data)
         pd.DataFrame({'Predict':some_predict,'Label':some_data_label})

Out[34]:
```

	Predict	Label
4093	126.718172	122.47
22310	40.939104	33.78
8034	45.506013	60.91
35027	57.615353	61.36
23685	67.683748	62.09
268	66.142103	49.76
35261	56.396384	30.93
11905	63.495308	61.50
30903	71.314507	82.26
608	227.258421	119.70

```
In [37]: from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

         mae = mean_absolute_error(some_data_label, some_predict)

         print(mae)

         19.84973669824718

In [38]: mse = mean_squared_error(some_data_label, some_predict)
         print(mse)

         1296.140388436656

In [39]: rmse = mean_squared_error(some_data_label, some_predict, squared=False)
         print(rmse)

         36.00194978659706

In [40]: r2 = r2_score(some_data_label, some_predict)

In [41]: print(r2)

         -0.45791346385622456

In [ ]:
```

Conclusion:

In conclusion, our study demonstrates the effectiveness of data-driven machine learning approaches in electricity price prediction. By unraveling the intricate relationships between various influencing factors, our models offer valuable tools for stakeholders to navigate the complex energy market landscape. As we move forward, ongoing research and collaboration between data scientists, energy experts, and policymakers will be essential in refining these models, making them indispensable assets for the energy industry.

THANK YOU