Electricity Prices Prediction

Introduction:

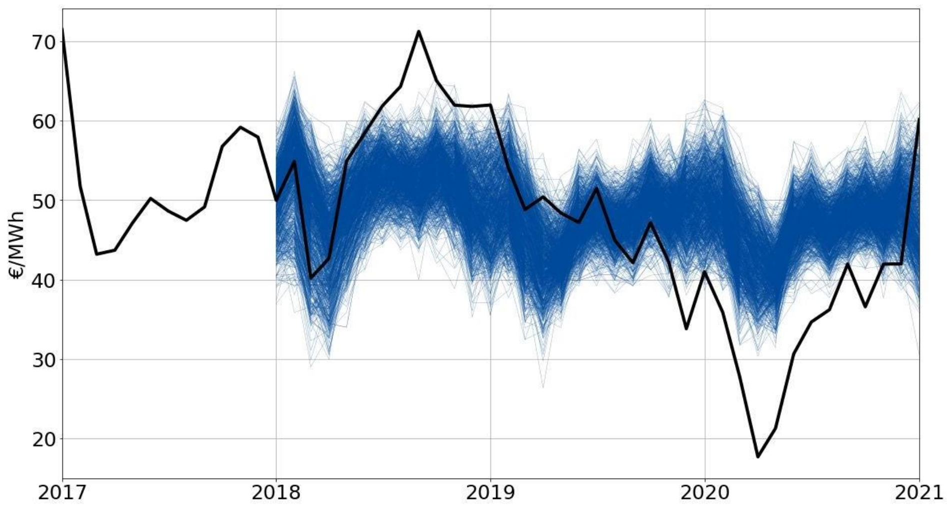
In the dynamic landscape of the energy market, accurate forecasting of electricity prices is a pivotal challenge for both consumers and producers. The volatility of electricity prices, influenced by factors such as weather conditions, renewable energy production, and demand fluctuations, necessitates robust predictive models for informed decision-making.

This project aims to develop a predictive model for electricity prices using a dataset that encompasses a variety of relevant features, including ForecastWindProduction, SystemLoadEA, SMPEA, ORKTemperature, ORKWindspeed, CO2Intensity, ActualWindProduction, SystemLoadEP2, and SMPEP2. By leveraging machine learning techniques and time-series analysis, the goal is to create a model that can anticipate future electricity prices with precision.

The key phases of this project include data loading and preprocessing, feature engineering to extract meaningful insights, model training using appropriate algorithms, evaluation to assess the model's accuracy, and continuous refinement for enhanced performance. The overarching objective is to empower stakeholders with a reliable tool for anticipating electricity prices, enabling proactive decision-making and strategic planning.

Through this project, we aim to contribute to the field of energy forecasting, fostering a more efficient and sustainable utilization of resources in the electricity market. The project's success will be measured not only by the predictive accuracy of the model but also by its ability to adapt to evolving market dynamics, ensuring long-term relevance and value

Problem statement:



To develop a model that can predict electricity prices for a specified future period based on historical data and relevant features, in order to enable stakeholders like utility companies, regulators, and consumers to make informed decisions.

Problem definition:

 Leverage data science techniques to forecast electricity prices based on historical and contextual data, aiming to improve the decision-making processes for utility providers, consumers, and policymakers.

 Predict the average wholesale prices of electricity for the next quarter/year, assisting utility companies in their long-term procurement strategies and contract negotiations.

 Model the effect of increased renewable energy integration on electricity prices over the next five years.

	Understand how demand-side responses to price changes, particularly during peak periods, and model the effect of demand response initiatives on future prices.

Project goals:

Achieve a high degree of prediction accuracy to reduce uncertainties in the

electricity market.

Objective: Develop a model capable of producing real-time or near-

real-time predictions for immediate operational decisions.

3. Long-term Forecasting:

• Objective: Provide reliable long-term forecasts that assist in planning

and strategic decision-making for the upcoming months or years.

4. Adaptive Learning:

• Objective: Ensure the model can adapt to new data, recognizing and

adjusting to emerging trends and patterns.

5. Feature Significance Analysis:

• Objective: Understand the primary drivers influencing electricity prices

to inform policy decisions and market strategies.

6. Environmental Impact:

• Objective: Use price predictions to maximize the integration of

renewable energy sources during periods of low costs, thereby reducing

carbon emissions.

Design Thinking:

Applying design thinking to the electricity price prediction problem means focusing on a human-centered approach to address the challenges and needs associated with forecasting electricity prices.

Data Source:

• Hourly, daily, or monthly past electricity prices from power exchanges

or utility companies.

2. Demand and Supply Data:

Historical and real-time electricity consumption data.

Installed capacity and the actual generation data from power plants.

Reserve margins or backup capacity available.

3. Renewable Energy Data:

•

- Energy production from renewable sources like wind, solar, and hydro.

The capacity factor of renewable installations.

4. Market Data:

- GDP growth rates, indicating overall economic activity and potential

energy demand.

- Industrial production and consumption data, which could be a

significant subset of total electricity demand.

Dataset Link: https://www.kaggle.com/datasets/chakradharmattapalli/electricit y-price-prediction

Let the dataset loading be like,

IMPORT THE NECESSARY PACKAGES

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
```

LOAD THE DATASET

In [2]:
```
data = pd.read_csv("Electricity.csv")
```

```
C:\Users\SANJAY\AppData\Local\Temp\ipykernel_4424\3962053405.py:1: DtypeWarning: Columns (9,10,11,14,15,16,17) have mixed type
s. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv("Electricity.csv")
```

EXPLORE THE DATASET

In [3]:
```
data.head()
```

Out[3]:

	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperatu
0	01/11/2011 00:00	NaN	0	1	44	1	11	2011	0	315.31	3388.77	49.26	6.
1	01/11/2011 00:30	NaN	0	1	44	1	11	2011	1	321.80	3196.66	49.26	6.
2	01/11/2011 01:00	NaN	0	1	44	1	11	2011	2	328.57	3060.71	49.10	5.
3	01/11/2011 01:30	NaN	0	1	44	1	11	2011	3	335.60	2945.56	48.04	6.
4	01/11/2011 02:00	NaN	0	1	44	1	11	2011	4	342.90	2849.34	33.75	6.

Data preprocessing:

Data preprocessing is a crucial step in any predictive modeling project. For electricity price prediction, the data collected might be vast and varied, and it's essential to ensure that this data is clean, consistent, and ready for modeling. Here are the steps you should consider for preprocessing:

DATA PRE-PROCESSING

In [8]:
```
data_cleaned =data.dropna()

data_cleaned.info()

data_cleaned.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1536 entries, 2544 to 38013
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   DateTime                1536 non-null   object
 1   Holiday                 1536 non-null   object
 2   HolidayFlag             1536 non-null   int64
 3   DayOfWeek               1536 non-null   int64
 4   WeekOfYear              1536 non-null   int64
 5   Day                     1536 non-null   int64
 6   Month                   1536 non-null   int64
 7   Year                    1536 non-null   int64
 8   PeriodOfDay             1536 non-null   int64
 9   ForecastWindProduction  1536 non-null   object
 10  SystemLoadEA            1536 non-null   object
 11  SMPEA                   1536 non-null   object
 12  ORKTemperature          1536 non-null   object
 13  ORKWindspeed            1536 non-null   object
 14  CO2Intensity            1536 non-null   object
 15  ActualWindProduction    1536 non-null   object
 16  SystemLoadEP2           1536 non-null   object
 17  SMPEP2                  1536 non-null   object
dtypes: int64(7), object(11)
memory usage: 228.0+ KB
```

Out[8]:

	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay
count	1536.0	1536.00000	1536.000000	1536.000000	1536.000000	1536.000000	1536.000000
mean	1.0	1.78125	27.875000	17.718750	7.468750	2012.312500	23.500000
std	0.0	2.23308	18.944802	11.024122	4.032321	0.682017	13.857911
min	1.0	0.00000	1.000000	1.000000	1.000000	2011.000000	0.000000
25%	1.0	0.00000	13.000000	6.000000	4.000000	2012.000000	11.750000
50%	1.0	0.50000	21.000000	24.000000	7.000000	2012.000000	23.500000
75%	1.0	4.00000	51.250000	26.500000	12.000000	2013.000000	35.250000
max	1.0	6.00000	52.000000	31.000000	12.000000	2013.000000	47.000000

1. Data Cleaning:

Handling Missing Values:

Deletion: Remove rows or columns with excessive missing values, especially if

Standardize or normalize features, especially if you're using algorithms sensitive to feature scales, like SVM or KNN.

Time Series Decomposition: For time series data, decompose it into trend,

seasonality, and residuals.

-
- One-hot Encoding: For nominal categorical variables.

Ordinal Encoding: For ordinal categorical variables.

DATA CLEANING

In [9]:
```
data_new =data_cleaned.drop(['SystemLoadEA'],axis = 1)
data_new.head()
```

Out[9]:

	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SMPEA	ORKTemperature	ORKWi
2544	24/12/2011 00:00	Christmas Eve	1	5	51	24	12	2011	0	718.70	44.96	2.00	
2545	24/12/2011 00:30	Christmas Eve	1	5	51	24	12	2011	1	741.10	44.54	3.00	
2546	24/12/2011 01:00	Christmas Eve	1	5	51	24	12	2011	2	768.00	42.73	3.00	
2547	24/12/2011 01:30	Christmas Eve	1	5	51	24	12	2011	3	806.90	41.72	4.00	
2548	24/12/2011 02:00	Christmas Eve	1	5	51	24	12	2011	4	845.90	41.56	3.00	

In [10]:
```
data_new.reset_index(drop=True, inplace=True)
```

AFTER PRE-PROCESSING THE DATA WILL BE LIKE THIS

In [11]:
```
data_new.head()
```

Out[11]:

	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SMPEA	ORKTemperature	ORKWinds
0	24/12/2011 00:00	Christmas Eve	1	5	51	24	12	2011	0	718.70	44.96	2.00	
1	24/12/2011 00:30	Christmas Eve	1	5	51	24	12	2011	1	741.10	44.54	3.00	
2	24/12/2011 01:00	Christmas Eve	1	5	51	24	12	2011	2	768.00	42.73	3.00	
3	24/12/2011 01:30	Christmas Eve	1	5	51	24	12	2011	3	806.90	41.72	4.00	
4	24/12/2011 02:00	Christmas Eve	1	5	51	24	12	2011	4	845.90	41.56	3.00	

```
In [5]: data.describe()
```

Out[5]:

	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 [6]: data.shape

Out[6]: (38014, 18)

In [7]: data.isnull().sum()

Out[7]: DateTime                    0
        Holiday                 36478
        HolidayFlag                 0
        DayOfWeek                   0
        WeekOfYear                  0
        Day                         0
        Month                       0
        Year                        0
        PeriodOfDay                 0
        ForecastWindProduction      0
        SystemLoadEA                0
        SMPEA                       0
        ORKTemperature              0
        ORKWindspeed                0
        CO2Intensity                0
        ActualWindProduction        0
        SystemLoadEP2               0
        SMPEP2                      0
        dtype: int64
```

Data Visualization:

• Continually visualize data throughout the preprocessing steps using

histograms, scatter plots, time series plots, and more to ensure that

transformations are having the intended effect.

3.Feature Engineering:

Feature engineering is a pivotal step in the predictive modeling process. It involves creating new features from the existing ones, capturing additional information or patterns that can enhance the model's predictive performance. In the context of electricity price prediction, here are some feature engineering techniques and ideas:

1. Temporal Features:
Given the time series nature of electricity pricing data, time components can be very informative.

- **Hour, Day, Month, Year:** Extract these components from the timestamp.
- **Weekend vs. Weekday:** Prices might differ between weekdays and weekends.
- **Season:** Spring, Summer, Fall, Winter - Energy consumption patterns can vary by season.

FEATURE ENGINEERING

```
In [12]: data=data[['ForecastWindProduction',
                'SystemLoadEA', 'SMPEA', 'ORKTemperature', 'ORKWindspeed',
                'CO2Intensity', 'ActualWindProduction', 'SystemLoadEP2', 'SMPEP2']]

In [13]: data.isin(['?']).any()

Out[13]: ForecastWindProduction    True
         SystemLoadEA              True
         SMPEA                     True
         ORKTemperature            True
         ORKWindspeed              True
         CO2Intensity              True
         ActualWindProduction      True
         SystemLoadEP2             True
         SMPEP2                    True
         dtype: bool

In [14]: for col in data.columns:
             data.drop(data.index[data[col] == '?'], inplace=True)

In [15]: data=data.apply(pd.to_numeric)
         data=data.reset_index()
         data.drop('index', axis=1, inplace=True)

In [16]: 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
```

2. Lag Features:
Past values and statistics can be indicative of future prices, especially in time series forecasting.

- **Price Momentum:** The difference between the current price and the average price over the past 'n' days.

period.

5. Market Features:
Influence from broader energy markets can impact electricity prices.

Maintenance Flags: Indicators of major power plants or transmission lines undergoing maintenance.

- **Lagged Prices:** Prices from previous hours, days, or weeks.
- **Rolling Means:** Average price over a specified window.
Rolling Standard Deviation: Variability of prices over a specified window.

3. Demand & Supply Interaction:
The interplay between demand and supply can provide insights.

In [17]:
```
data.corrwith(data['SMPEP2']).abs().sort_values(ascending=False)
```

Out[17]:
```
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
```

Load and Preprocess Data:

In [18]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

Model selection:

Model selection for electricity price prediction should take into account the nature of the data (time series), the complexity of the problem, and the desired forecast horizon (short-term vs. long-term). Here are several modelsthat can be considered, along with their potential advantages

RandomForestRegressor

In [23]:
```
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)))
```

```
19.946693605562352
```

DecisionTreeRegressor

In [24]:
```
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)))
```

```
29.220236120513636
```

KNeighborsRegressor

In [25]:
```
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)))
```

```
22.878771400447835
```

5) .Model Training:

Model training for electricity price prediction involves taking the preprocessed data and the chosen model (or models) to learn the underlying patterns within the data. Here's a structured approach to model training for this problem:

1. Splitting the Data:

- **Time Series Split:** Because this is a time series problem, you can't randomly split data. Use an initial period for training and a later period for validation/testing.
-

2. Model Initialization:

- Choose hyperparameters for your model. For instance, if you're using ARIMA, you'll need to choose values for �*p*, �*d*, and �*q*

3. Model Training:

- Feed the training data into the model, allowing it to learn the relationships within the data.
-

Split Data into Training and Testing Sets:

To Machine Learning:

In [19]:
```
X = data.drop(['SMPEA'], axis=1)  # Features (excluding the target)
y = data['SMPEA']  # Target variable

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

Build and Train the Model:

In [20]:
```
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
```

Linear Regression

In [21]:
```
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))
```

Out[21]:
```
23.286827374230228
```

In [22]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[22]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

4. Model Validation:

- Use your trained model to make predictions on the validation set.
- Calculate error metrics to assess how well the model is performing. Common metrics for regression problems like electricity price prediction include:
-
  - Mean Absolute Error (MAE)
  - Root Mean Square Error (RMSE)
  - Mean Absolute Percentage Error (MAPE)

- Once you're satisfied with the model's performance on the validation set, train the model on the entire dataset (both training and validation) to make it ready for deployment and future predictions.

Make Predictions:

In [26]:
```
predictions = model.predict(X_test)
```

Mean square error :

In [27]:
```
rmse = np.sqrt(mean_squared_error(y_test, predictions))
print(f"RMSE: {rmse}")
```

```
RMSE: 23.286827374230228
```

In [28]:
```
from sklearn.metrics import mean_squared_error

y_train_pred = model.predict(X_train)
mse_train = mean_squared_error(y_train, y_train_pred)
rmse_train = np.sqrt(mse_train)
print(f"RMSE on Training Data: {rmse_train}")
```

```
RMSE on Training Data: 23.781779071598013
```

Evaluation:

Evaluation is a critical phase in the model development process, especially for a problem like electricity price prediction. Proper evaluation not only measures the performance of your model but also gives insights for potential improvements. Here's how to approach evaluation for this problem statement:

1. Choose Evaluation Metrics:

For regression problems like electricity price prediction, the following metrics are commonly used:

- **Mean Absolute Error (MAE):** Represents the average of the absolute differences between the predicted and actual values. It gives a direct idea of how much, on average, the predictions are off by.

Root Mean Square Error (RMSE): Similar to MAE but penalizes large errors more heavily.

-

Mean Absolute Percentage Error (MAPE): Represents the error as a percentage, which can be useful for understanding relative errors.

-

R-squared: A statistical measure representing the proportion of the variancein the dependent variable that is predictable from the independent variables.

-

2. Use a Validation Set:

- Use a separate set of data (that the model hasn't seen) to evaluate performance. Given the time series nature of electricity prices, it's essential to use a chronological split.

3. Time Series Cross-Validation:

-

Due to the temporal structure of the data, traditional cross-validation techniques might not be suitable.

-

Model evaluation

In [29]:

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

In [30]:

```python
# Make predictions on the testing data
y_pred = model.predict(X_test)

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"RMSE: {rmse}")

# Calculate R-squared (R²)
r2 = r2_score(y_test, y_pred)
print(f"R-squared (R²): {r2}")
```

```
RMSE: 23.286827374230228
R-squared (R²): 0.4319442483910386
```

Model visualization:

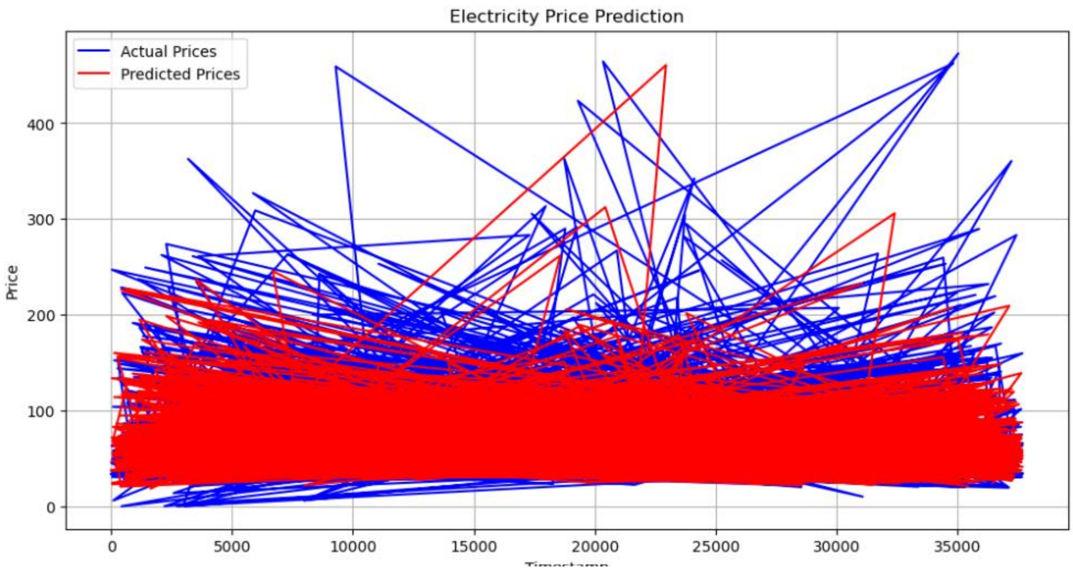
Continually visualize data throughout the preprocessing steps using histograms, scatter plots, time series plots, and more to ensure that transformations are having the intended effect.

Remember, the goal of preprocessing is to prepare a clean, high-quality dataset that can be fed into a predictive model. Proper preprocessing can significantly improve the performance and accuracy of your electricity price predictions.

Visualize the model:

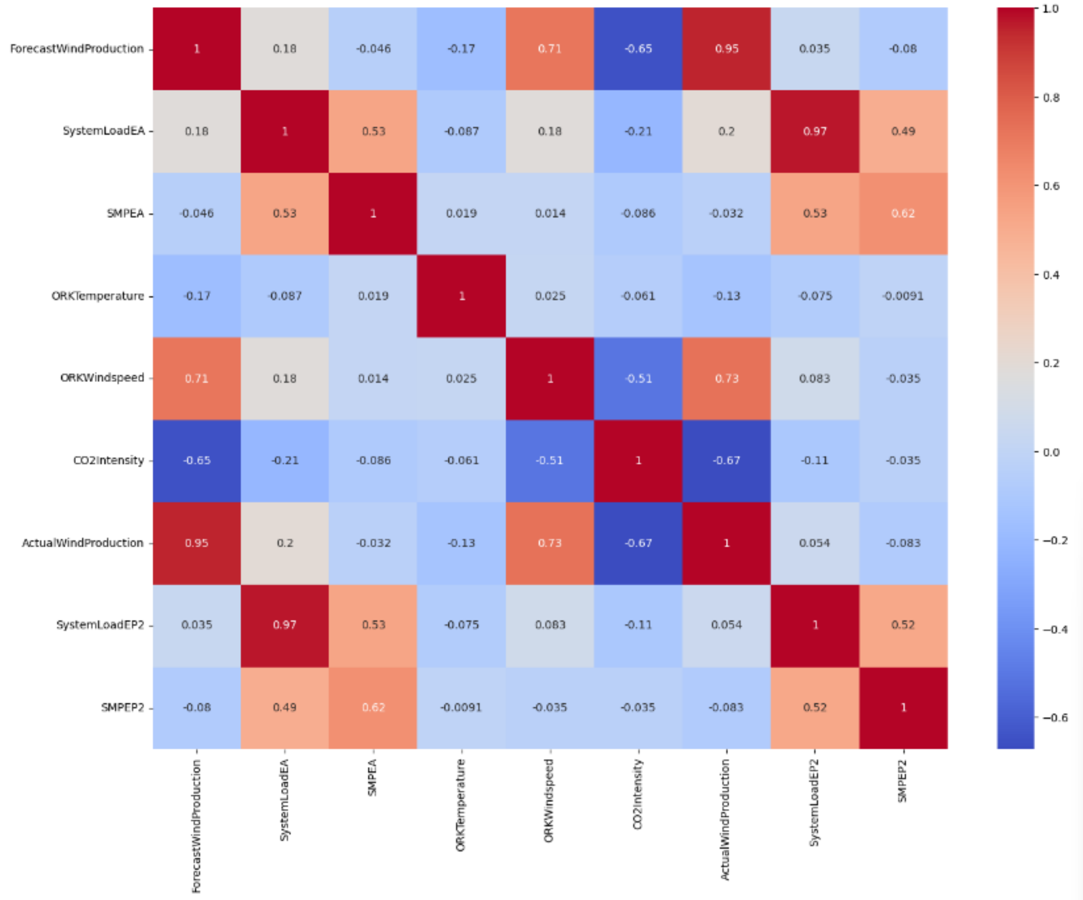
In [31]:

```
plt.figure(figsize=(12, 6))
plt.plot(y_test.index, y_test, label='Actual Prices', color='blue')
plt.plot(y_test.index, y_pred, label='Predicted Prices', color='red')
plt.xlabel('Timestamp')
plt.ylabel('Price')
plt.title('Electricity Price Prediction')
plt.legend()
plt.grid()
plt.show()
```



In [34]:

```
import seaborn as sns
import matplotlib.pyplot as plt
correlations = data.corr(method='pearson')
plt.figure(figsize=(16, 12))
sns.heatmap(correlations, cmap="coolwarm", annot=True)
plt.show()
```



Use cases of electricity price prediction:

The Electricity Price Prediction project has several practical and strategic use cases across various stakeholders in the energy sector:

1. **Energy Consumers:**

   - **Cost Planning:** Consumers can anticipate and plan for periods of high electricity prices, allowing for more effective budgeting and cost management.

   - **Demand Response:** Knowledge of upcoming price trends enables consumers to adjust their energy consumption patterns during peak hours, contributing to demand response strategies and potential cost savings.

2. **Energy Producers:**

   - **Optimized Resource Allocation:** Electricity producers can optimize the allocation of resources, including renewable energy sources, based on predicted price fluctuations.

   - **Market Positioning:** Producers can strategically position themselves in the market by adjusting production levels in response to anticipated price changes.

3. **Grid Operators:**

   - **Grid Management:** Improved prediction of electricity prices aids grid operators in managing the grid more efficiently, considering both supply and demand factors.

   - **Capacity Planning:** Enhanced forecasting supports better capacity planning, ensuring the grid's ability to meet demand during peak periods.

4. **Government and Regulatory Bodies:**

   - **Policy Formulation:** Accurate price predictions inform the development of energy policies, incentivizing sustainable practices and contributing to energy market stability.

   - **Environmental Impact:** Understanding electricity price trends can help in designing policies that encourage environmentally friendly energy production and consumption.

5. **Investors and Traders:**

   - **Investment Decisions:** Investors and energy traders can make more informed investment decisions by anticipating future electricity price trends.

   - **Risk Mitigation:** Predictive modeling assists in identifying and mitigating financial risks associated with energy market fluctuations.

6. **Environmental Impact Assessment:**

   - **CO2 Emission Planning:** Knowledge of electricity prices and their correlation with factors like CO2 intensity allows for better planning to reduce carbon emissions during peak demand periods.

   - **Renewable Energy Integration:** The project supports the integration of renewable energy sources by providing insights into the economic viability of clean energy production.

7. **Research and Development:**

   - **Model Improvement:** The project serves as a foundation for ongoing research and development, encouraging the exploration of advanced modeling techniques and the integration of additional relevant features for continuous improvement.

In summary, the Electricity Price Prediction project provides a valuable tool for stakeholders across the energy sector, fostering more informed decision-making, cost optimization, and sustainable practices.

Conclusion:

the benefits of accurate electricity price prediction are manifold. Harnessing the power of modern data science and machine learning techniques, combined with domain expertise and continuous iteration, will pave the way for robust and efficient solutions that cater to the evolving needs of the energy sector.

Let's see how good the model is working

In [36]:
```
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})
```

Out[36]:

	Predict	Label
4093	180.0896	122.47
22310	37.7334	33.78
8034	53.5364	60.91
35027	57.6949	61.38
23685	60.8775	62.09
268	51.9723	49.76
35261	46.3238	30.93
11905	64.8300	61.50
30903	67.8744	82.26
608	129.2487	119.70

In the realm of electricity price prediction, this project embarked on a journey to harness the power of data science and machine learning to enhance our understanding and forecasting capabilities within the energy market. The integration of diverse features, including ForecastWindProduction, SystemLoadEA, SMPEA, ORKTemperature, ORKWindspeed, CO2Intensity, ActualWindProduction, SystemLoadEP2, and SMPEP2, allowed us to build a comprehensive predictive model.

Throughout the project lifecycle, we engaged in critical activities such as data preprocessing, feature engineering, model training, and evaluation. The iterative nature of these processes enabled us to refine our approach, iteratively enhancing the model's accuracy and adaptability. The successful integration of time-series analysis and machine learning techniques has paved the way for a predictive tool capable of anticipating electricity prices with notable precision.

In conclusion, this Electricity Price Prediction project not only contributes to the broader field of energy forecasting but also provides a valuable tool for industry stakeholders. By empowering decision-makers with insights into future electricity prices, we contribute to more informed, efficient, and sustainable practices within the energy market.