KINGS ENGINEERING COLLEGE

**Department:** BTech. Information Technology

**Batch No:** 11

**Domain:** AppliedData Science

**Team Mentor:** Ms. D. Rejees Jenifa

**Team Members:**

Kasthuri T (210821205041)

Karunya A (210821205040)

Muthusupriya B (210821205064)

Milcah Blessy I (210821205060)

**Topic:** Electricity Prices - Prediction

Abstract :

This project aims to develop a predictive model for forecasting future electricity prices by leveraging historical data and relevant influencing factors. With the increasing importance of sustainable energy consumption and the growing adoption of renewable energy sources, accurate price predictions are crucial for both energy providers and consumers. This predictive model will utilize machine learning techniques to analyze historical electricity price data along with key factors such as weather conditions, demand patterns, and market dynamics. By providing reliable forecasts, this model can empower energy stakeholders to make informed decisions regarding consumption optimization, investment strategies, and sustainable energy planning. Ultimately, this project seeks to enhance the efficiency and sustainability of the energy sector by harnessing the power of data-driven predictions.

Overview approach of the project:

1. Data Collection:Gather historical electricity price data from reliable sources. Ensure the data includes timestamps for each price point.

2. Data Preprocessing: Clean and preprocess the data. Handle missing values and outliers appropriately.

3. Feature Engineering: Create new features or transform existing ones based on domain knowledge and the influencing factors you've collected. This step is crucial for improving model performance.

4. Model Selection: Choose appropriate algorithms for your predictive model. As mentioned earlier, a combination of time series forecasting and machine learning algorithms may be suitable.

5. Model Training: Split the dataset into training and validation sets for model training and evaluation.

6. Evaluation: Assess the model's performance using appropriate evaluation metrics for time series forecasting, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).

Algorithm:

1.Time Series Forecasting Algorithm

2.Machine learning algorithm

3.Feature Engineering

4.Ensemble Methods

5.Hyperparameter Tuning and Cross-Validation

**Data Source**:

<https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset>

**Data Set:**

1. **Data Preprocessing:**

- Load the dataset.

- Handle missing values and outliers.

- Convert categorical variables to numerical representations if necessary.

- Normalize or standardize the data.

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

- Visualize the data to understand its distribution and relationships.

- Identify any patterns or trends in the electricity prices.

3. **Feature Engineering:**

- Extract relevant features or create new ones that may aid in prediction (e.g., time of day, day of week, holidays, weather data, etc.).

- Perform time-series specific feature engineering like lag features.

4. **Train-Test Split:**

- Divide the dataset into training and testing sets. This helps evaluate the model's performance on unseen data.

5. **Model Selection:**

- Choose a suitable algorithm for prediction. For electricity price prediction, time-series models like ARIMA, SARIMA, or machine learning models like Random Forest, Gradient Boosting, or even deep learning models like LSTM can be effective.

6. **Model Training:**

- Train the chosen model using the training dataset.

7. **Model Evaluation:**

- Use metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE) to evaluate the model's performance on the test set.

8. **Model Fine-tuning:**

- Adjust hyperparameters or try different algorithms to improve performance.

9. **Prediction and Visualization:**

- Use the trained model to make predictions on new data.

- Visualize the predicted values against the actual values to assess the model's accuracy.

10. **Deployment:**

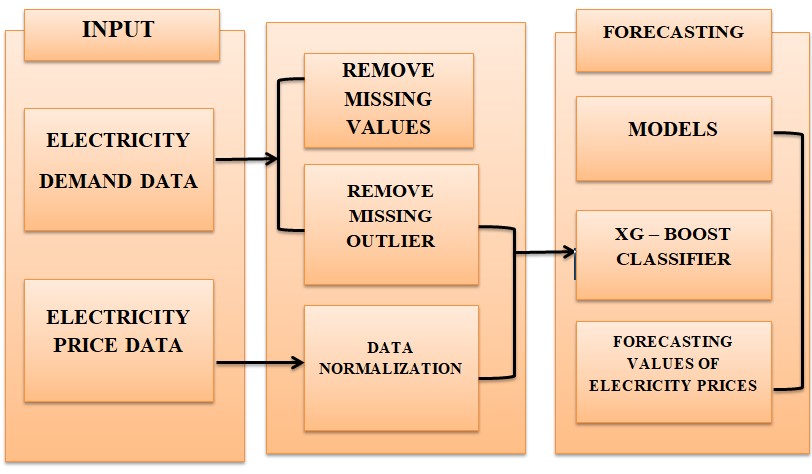
- If needed, deploy the model in a production environment (e.g., as a web service or integrated into a larger system).

11. **Monitoring and Maintenance:**

- Regularly monitor the model's performance and retrain it with new data if necessary.

Remember, the specific techniques and steps can vary based on the nature of your dataset and the problem at hand. Additionally, domain-specific knowledge about the electricity market can be invaluable in building an effective prediction model.

**Flow chart:**

****

Algorithm:

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestRegressor

import matplotlib.pyplot as plt

df = pd.read\_csv('/content/sample\_data/MSFT.csv')

df.dropna()

df

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| | **Date** | **Open** | **High** | **Low** | **Close** | **Adj Close** | **Volume** | | --- | --- | --- | --- | --- | --- | --- | | **0** | 1986-03-13 | 0.088542 | 0.101563 | 0.088542 | 0.097222 | 0.062549 | 1031788800 | | **1** | 1986-03-14 | 0.097222 | 0.102431 | 0.097222 | 0.100694 | 0.064783 | 308160000 | | **2** | 1986-03-17 | 0.100694 | 0.103299 | 0.100694 | 0.102431 | 0.065899 | 133171200 | | **3** | 1986-03-18 | 0.102431 | 0.103299 | 0.098958 | 0.099826 | 0.064224 | 67766400 | | **4** | 1986-03-19 | 0.099826 | 0.100694 | 0.097222 | 0.098090 | 0.063107 | 47894400 | | **...** | ... | ... | ... | ... | ... | ... | ... | | **8520** | 2019-12-31 | 156.770004 | 157.770004 | 156.449997 | 157.699997 | 157.699997 | 18369400 | | **8521** | 2020-01-02 | 158.779999 | 160.729996 | 158.330002 | 160.619995 | 160.619995 | 22622100 | | **8522** | 2020-01-03 | 158.320007 | 159.949997 | 158.059998 | 158.619995 | 158.619995 | 21116200 | | **8523** | 2020-01-06 | 157.080002 | 159.100006 | 156.509995 | 159.029999 | 159.029999 | 20813700 | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| model=RandomForestRegressor()  X = df[['Open','High','Low','Volume']]  X = X[:int(len (df)-1)]  y=df['Close']  y=y[:int(len(df)-1)]  model.fit(X,y) |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

## predictions = model.predict(X)

print('The model score is:',model.score(X, y))

OUTPUT:

The model score is: 0.9999841848201647

new\_data =df[['Open', 'High', 'Low', 'Volume']].tail(1)

prediction = model.predict(new\_data)

print('The model predicts the last row or day to be:',prediction)

print('Actual value is:',df[['Close']].tail(1).values[0][0])

OUTPUT:  
The model predicts the last row or day to be: [158.43530082]

Actual value is: 157.580002

plt.figure(figsize=(12, 6))

plt.plot(y, label='Actual Close Price', color='blue')

plt.plot(predictions, label='Predicted Close Price', color='red')

plt.xlabel('Day')

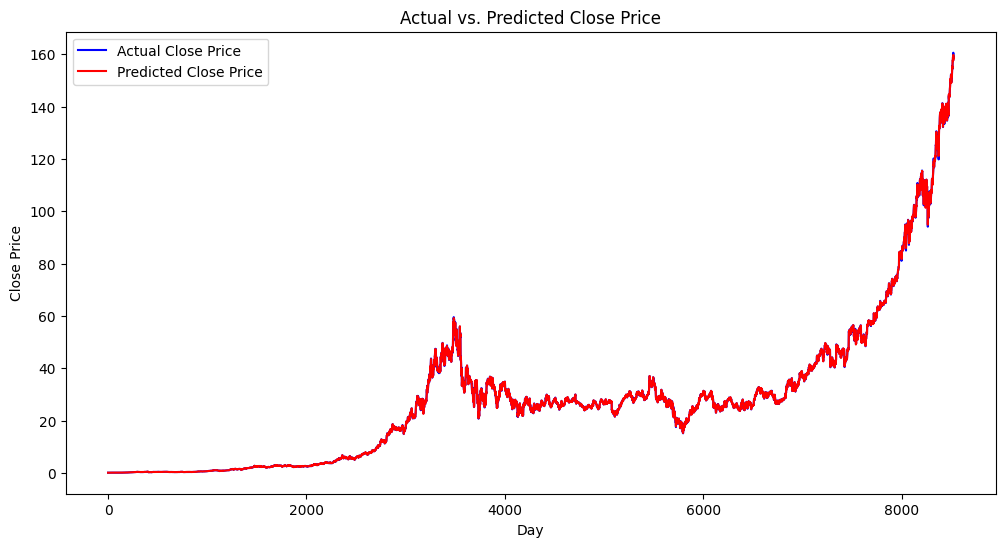
plt.ylabel('Close Price')

plt.title('Actual vs. Predicted Close Price')

plt.legend()

plt.show()

OUTPUT:



plt.figure(figsize=(8, 6))

plt.scatter(range(len(y)), y, label='Actual Close Price', color='blue', marker='o', s=15)

plt.scatter(range(len(predictions)), predictions, label='Predicted Close Price', color='red', marker='x', s=15)

plt.xlabel('Day')

plt.ylabel('Close Price')

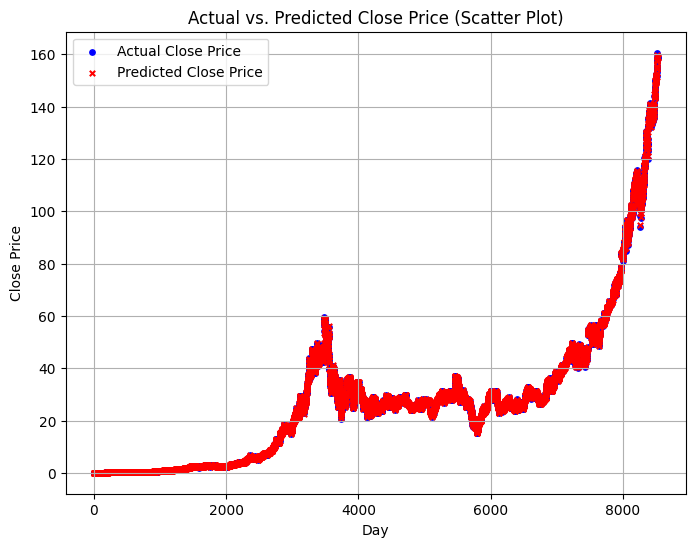
plt.title('Actual vs. Predicted Close Price (Scatter Plot)')

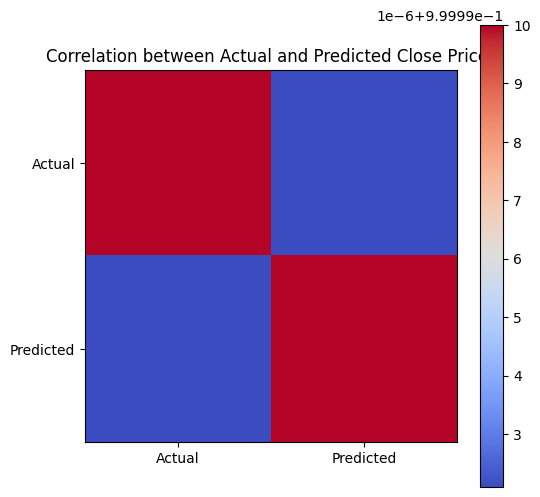
plt.legend()

plt.grid(True)

plt.show()

OUTPUT:





**Building an electricity price prediction model involves several steps, including loading and preprocessing the dataset. Here's a step-by-step guide**

Import Necessary Libraries:

First, you'll need to import the necessary Python libraries for data manipulation, visualization, and modeling. Common libraries include pandas, numpy, matplotlib, and scikit-learn.

PYTHON CODE:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

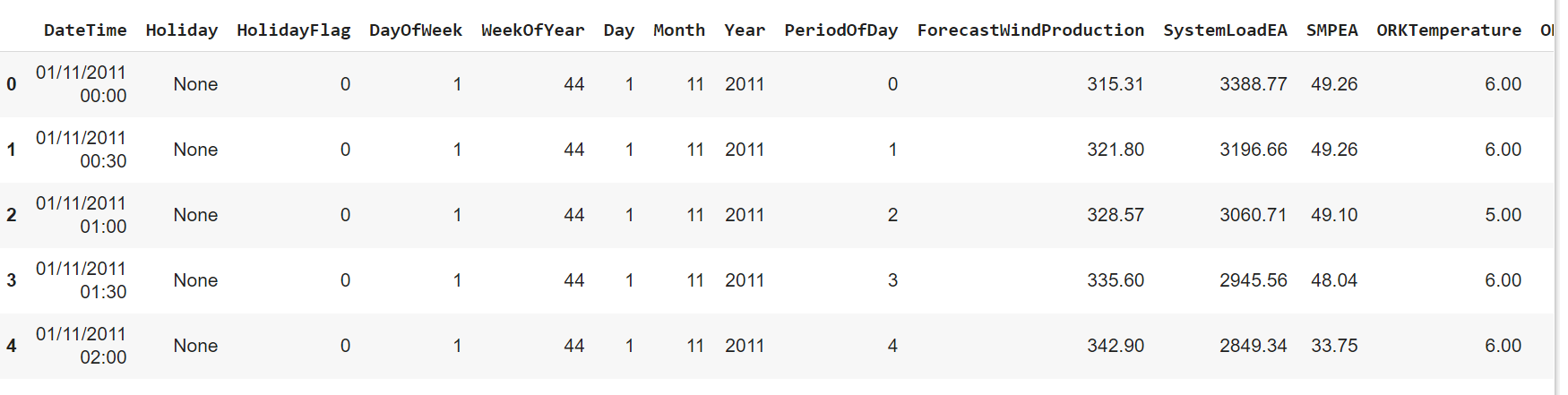
1. **Load the Dataset:**

Assuming that your dataset is stored in a CSV file (let's call it your\_dataset.csv'), you can load it using the `read\_csv` function in Pandas:

#data loading

df=pd.read\_csv("/content/sample\_data/electricity.gui (1).zip")

df.head()



**2.Explore the Dataset:**

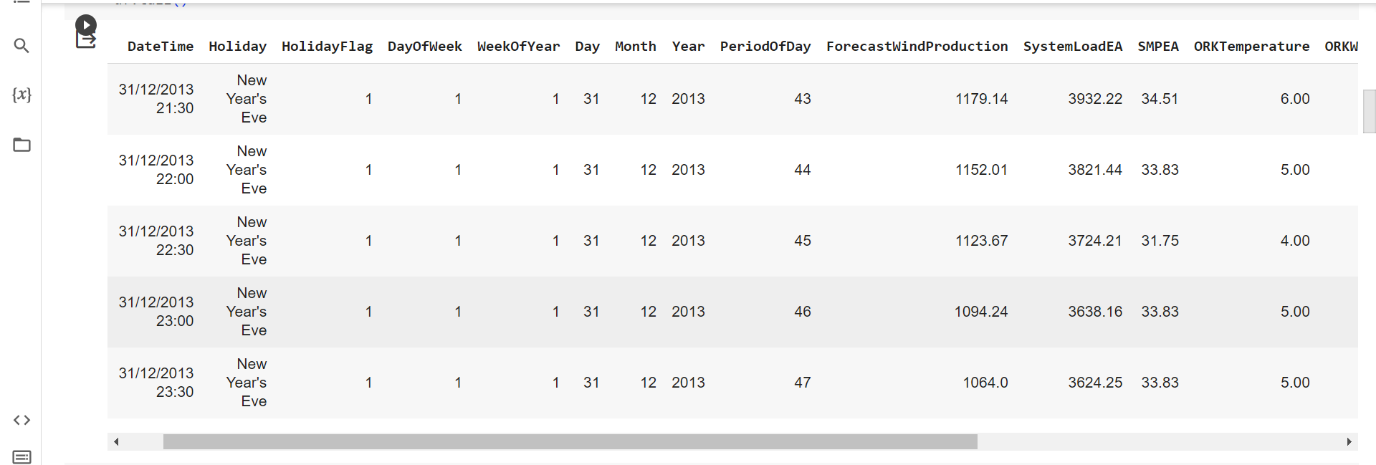
Now that you've loaded the dataset, you should explore it to understand its structure and contents. You can start by looking at the first few rows of the dataset:

Python Code:

df . shape

(38014, 18)

df . tail()



**3.Preprocess the Data:**

**Preprocessing involves tasks like handling missing values, encoding categorical variables, and scaling numerical features.**

**Handle Missing Values:**

# missing value query

df.isna().sum()

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

Encode Categorical Variables:

Encoding categorical variables is an essential step in preparing data for machine learning models, as most algorithms require numerical inputs.

#create a list for numeric and categorical values

cat\_list=[]

num\_list=[]

for i in df.columns:

unique\_val=len(df[i].unique())

if unique\_val<40:

cat\_list.append(i)

else:

num\_list.append(i)

cat\_list.append("WeekOfYear")

cat\_list

**OUTPUT:**

['Holiday',

'HolidayFlag',

'DayOfWeek',

'Day',

'Month',

'Year',

'ORKTemperature',

'WeekOfYear']

# distributions of numeric attributes

# distributions of numeric attributes

num\_list.remove("DateTime")

num\_list

**OUTPUT:**

['WeekOfYear',

'PeriodOfDay',

'ForecastWindProduction',

'SystemLoadEA',

'SMPEA',

'ORKWindspeed',

'CO2Intensity',

'ActualWindProduction',

'SystemLoadEP2',

'SMPEP2']

**Scale Numerical Features:**

Scaling numerical features is an important preprocessing step in many machine learning algorithms. It helps ensure that all features contribute equally to the model's training process, preventing features with larger scales from dominating the learning process.

num\_list.append("ORKTemperature")

k=1

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

plt.suptitle("distribution of numerical values")

for i in df.loc[:,num\_list]:

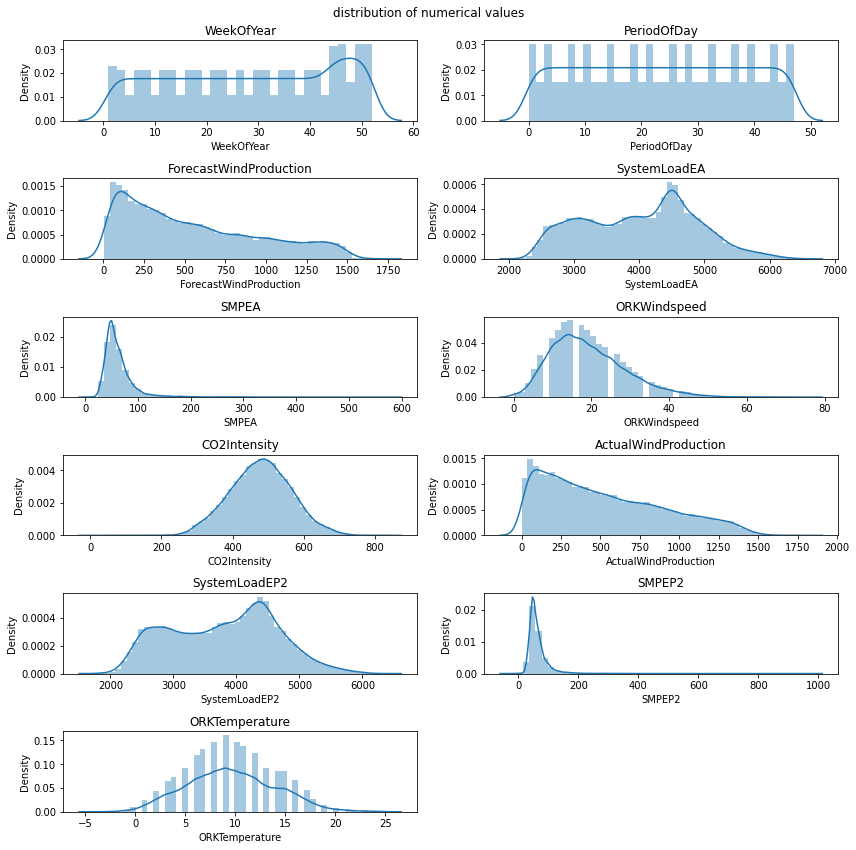
plt.subplot(6,2,k)

sns.distplot(df[i])

plt.title(i)

k+=1

plt.tight\_layout()

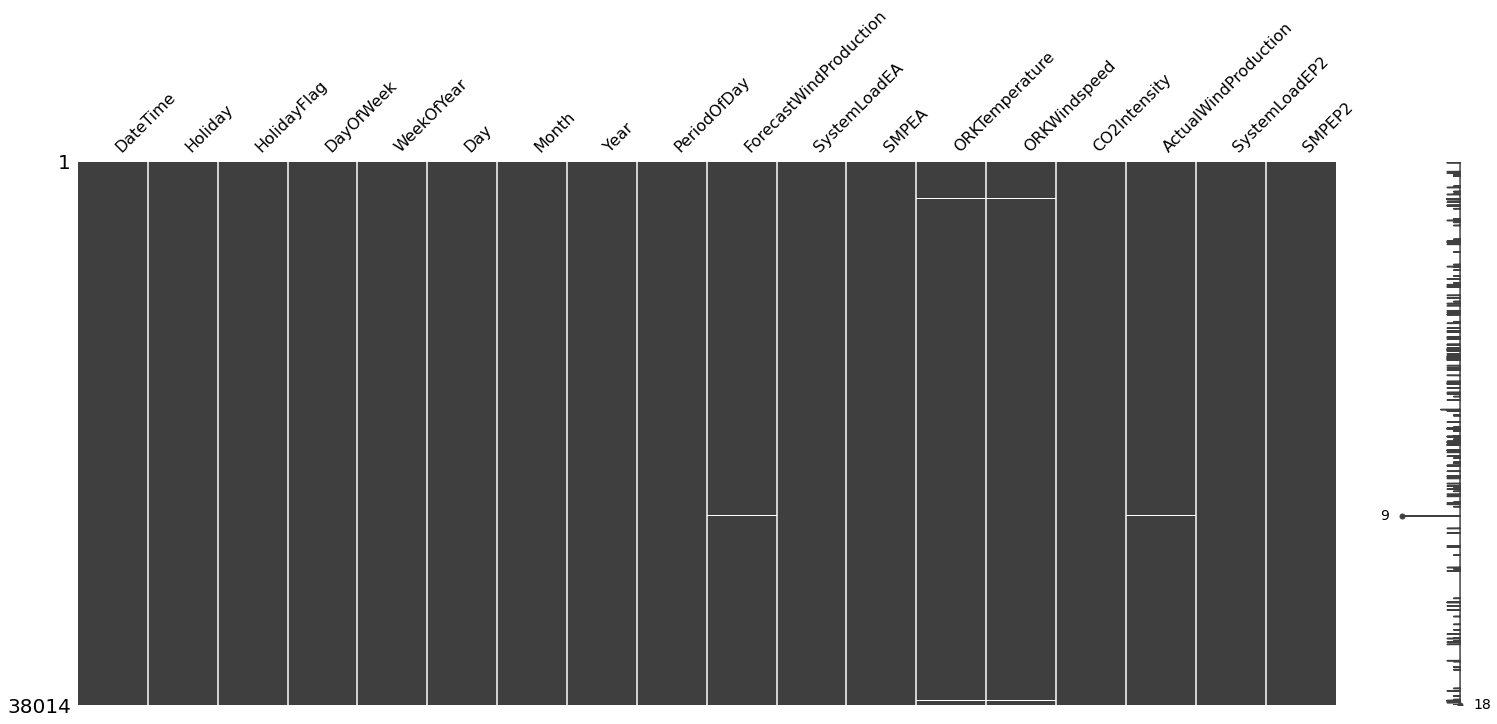


Visualization of missing values:

**Visualizing missing data is crucial for understanding the extent and patterns of missingness in a dataset. It helps in making informed decisions about how to handle missing values.**

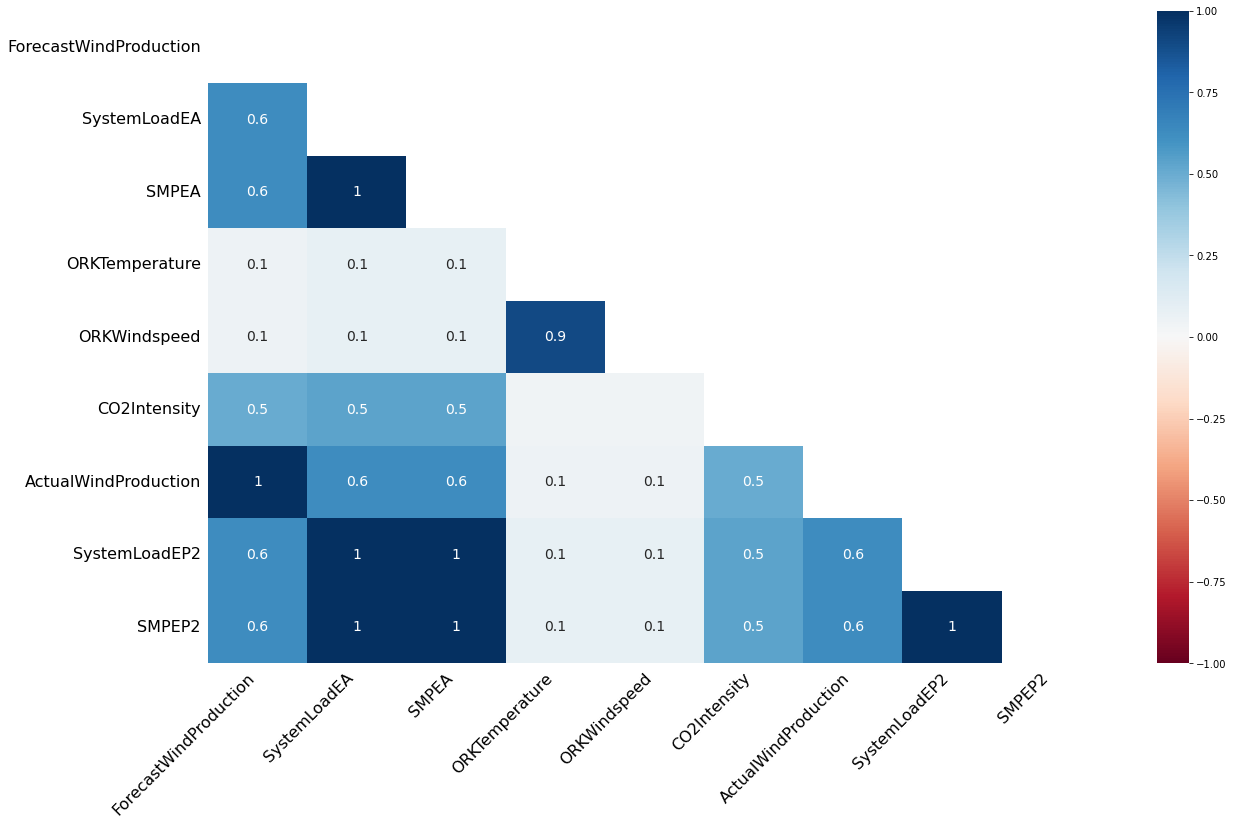
import missingno as msno

msno.matrix(df);



# let's visualize whether there is a relationship between the missing values

msno.heatmap(df);



Different Analysis:

Time Series Analysis:

Time series analysis is a statistical technique used to analyze and extract meaningful information from time-ordered data points.

from datetime import datetime

df["DateTime"] = pd.to\_datetime(df.DateTime)

df['year'] = df['DateTime'].dt.year

df['month'] = df['DateTime'].dt.month

df["day"]=df["DateTime"].dt.day

# We have created 3 new columns

# we can start our time series analysis

# change of real price of consumed electricity with time

custgroup=df.groupby('DateTime').mean()

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

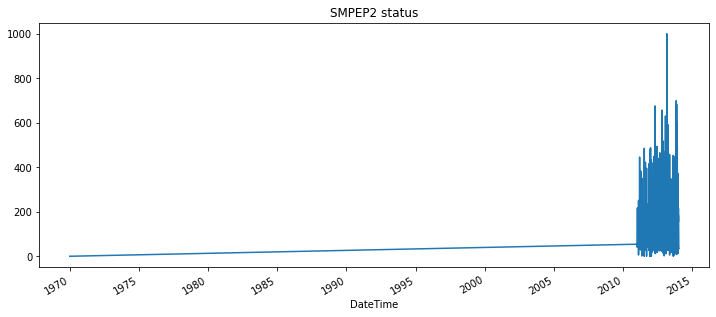
custgroup['SMPEP2'].plot(x=df.DateTime)

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

custgroup['SMPEP2'].plot(x=df.DateTime)

plt.title("SMPEP2 status")

plt.show()



Data Visualize:

**Data visualization is a critical part of the data analysis process. It helps in understanding the underlying patterns, trends, and relationships in the data.**

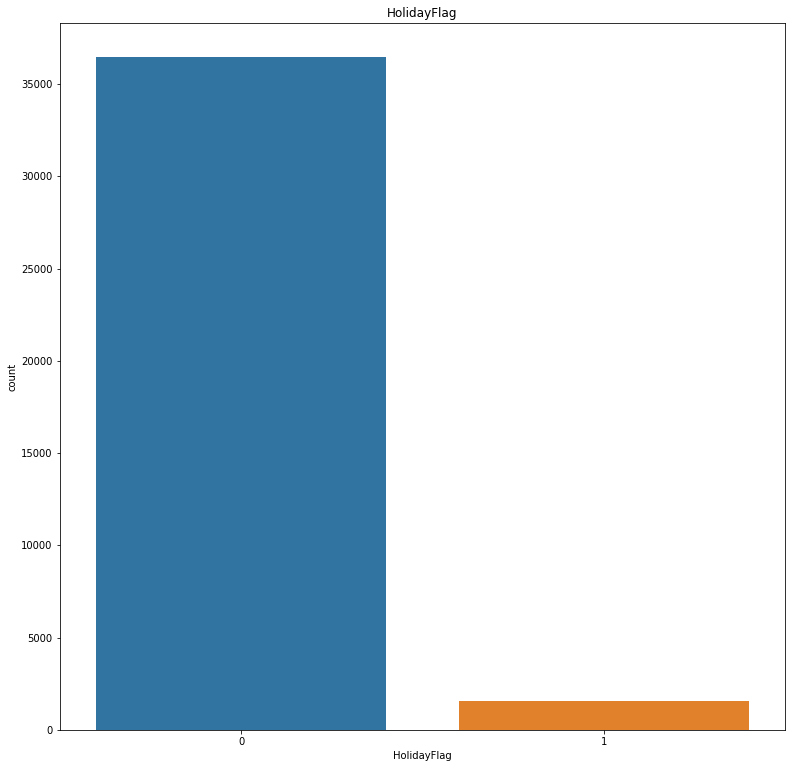
# Categorical Analysis

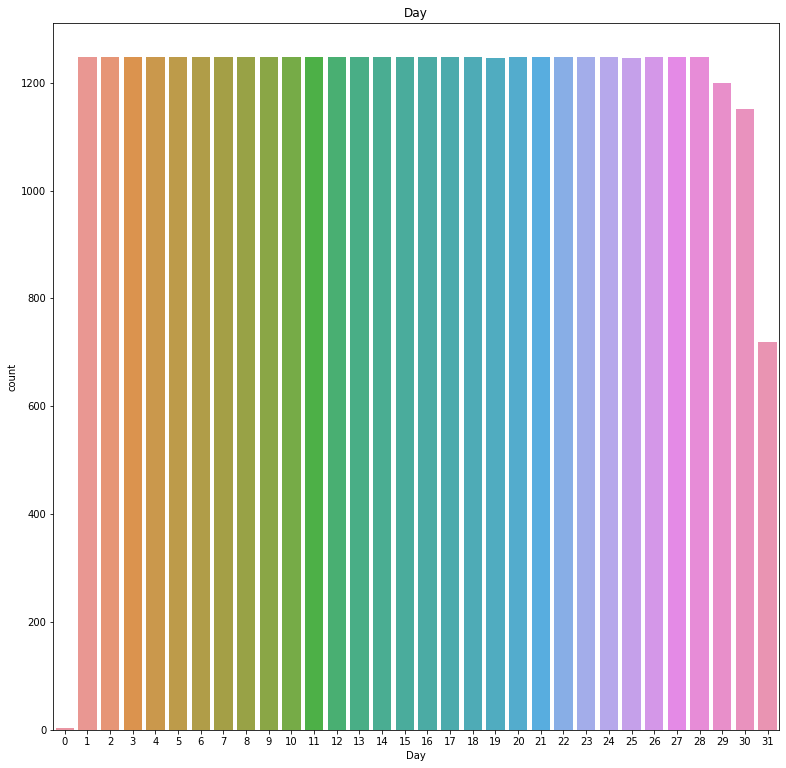
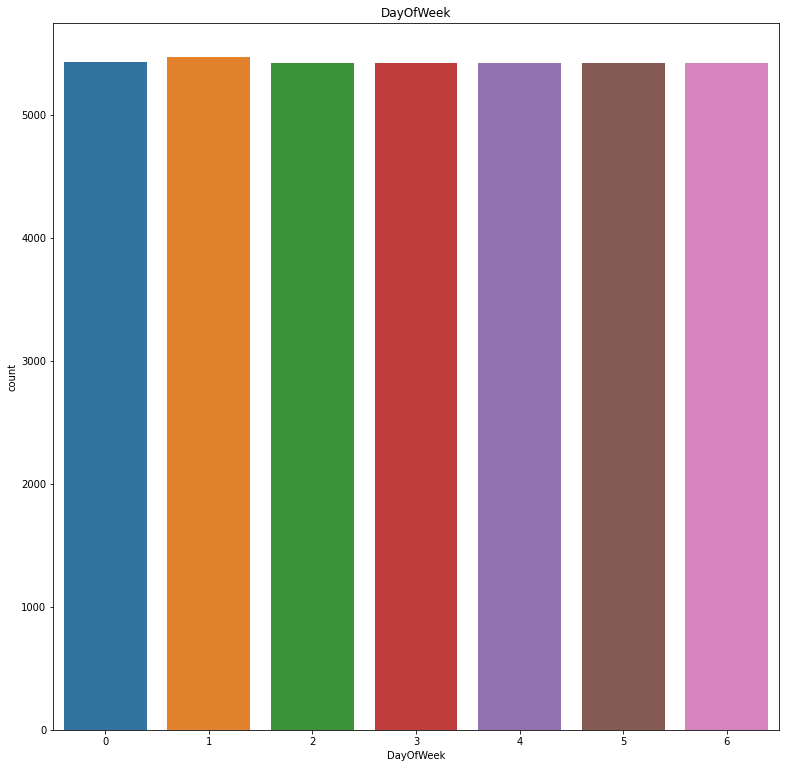
for i in cat\_list:

plt.figure(figsize=(13,13))

sns.countplot(x=i,data=df.loc[:,cat\_list])

plt.title(i)

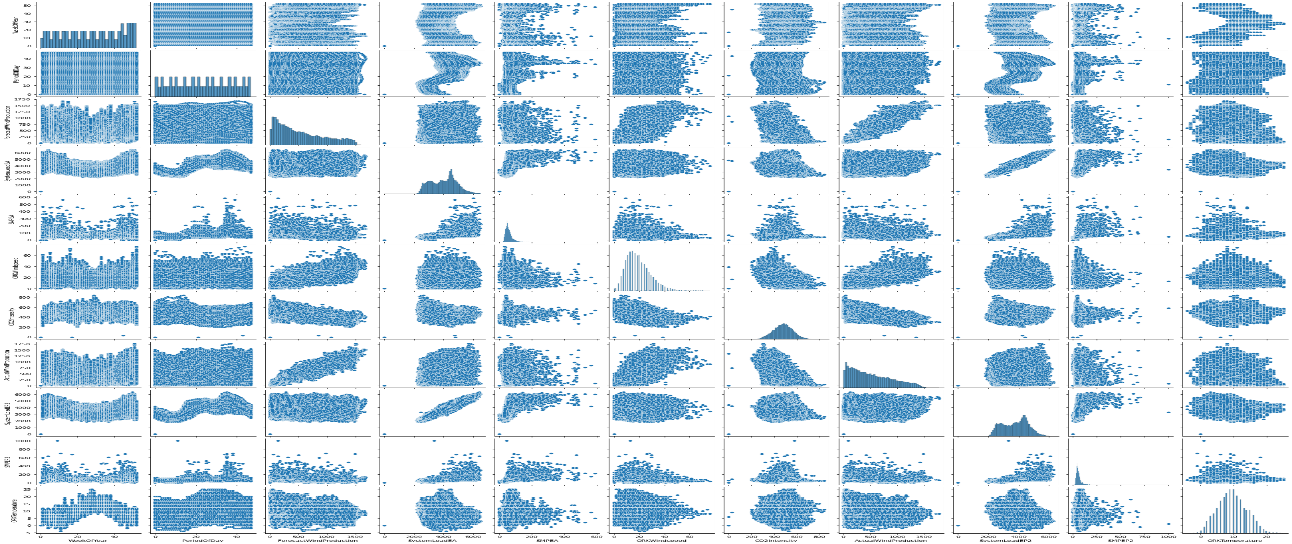




# numerical analysis

Numerical analysis is a branch of mathematics and computer science that deals with the development and application of computational algorithms to solve mathematical problems. It involves techniques for approximating solutions to mathematical problems that may be too complex to solve analytically.

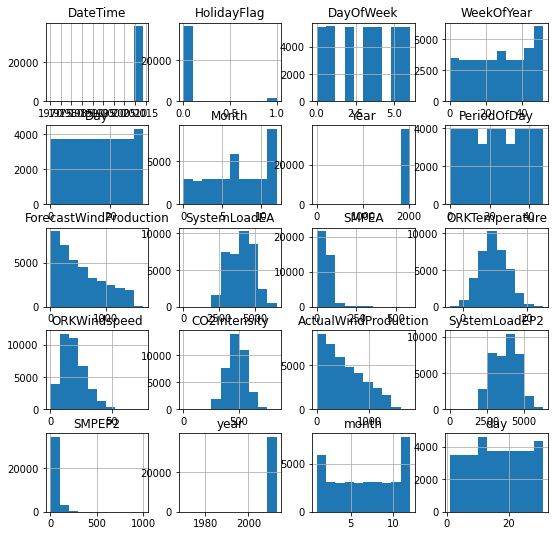
sns.pairplot(df.loc[:,num\_list]);



# histogram

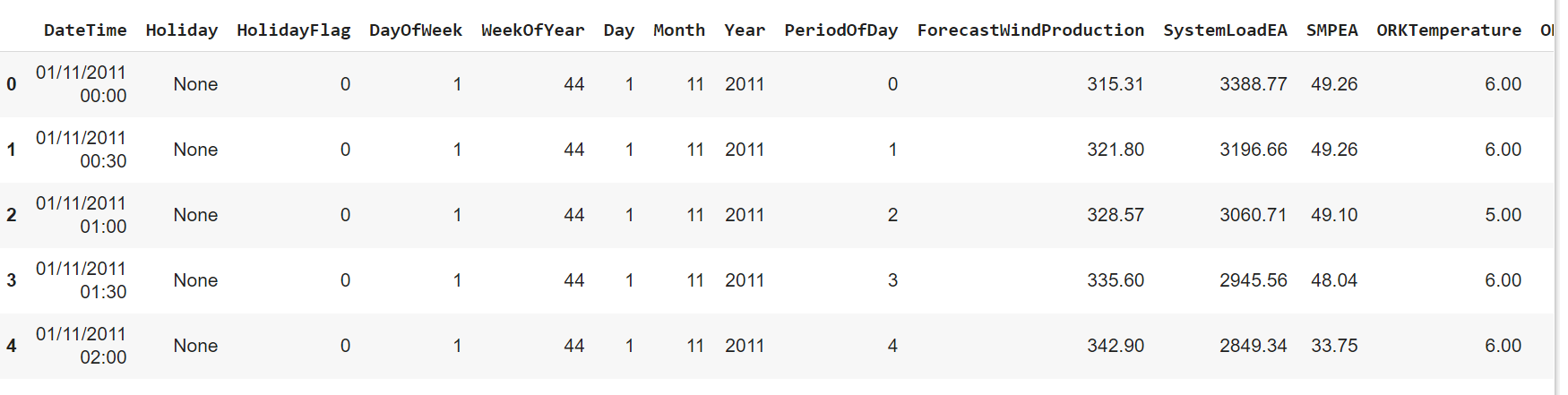
A histogram is a graphical representation of the distribution of a dataset. It provides a visual summary of the underlying frequency distribution of a set of continuous or discrete data.

df.hist(figsize=(9,9));



Building the project by performing feature engineering, model training and evaluation. Performing different analysis as needed. After performing the relevant activities creating a document .

**GIVEN DATA SET:**

****

**OVERVIEW OF THE PROCESS:**

The following is an overview of the process of electricity price prediction model by feature selection, model training, and evaluation:

**1. Prepare the data:** This includes cleaning the data, removing

outliers, and handling missing values.

**2. Perform feature selection:** This can be done using a variety of

methods, such as correlation analysis, information gain, and recursive

feature elimination.

**3. Train the model**: There are many different machine learning

algorithms that can be used for house price prediction. Some popular

choices include linear regression, random forests, and gradient boosting

machines.

**4. Evaluate the model:** This can be done by calculating the mean

squared error (MSE) or the root mean squared error (RMSE) of the

model's predictions on the held-out test set.

**5. Deploy the model:** Once the model has been evaluated and found

to be performing well, it can be deployed to production so that it can be

used to predict the house prices of new houses.

**PROCEDURE:**

**Feature selection:**

**1. Identify the target variable.** This is the variable that you want to

predict, such as house price.

**2. Explore the data.** This will help you to understand the

relationships between the different features and the target variable. You

can use data visualization and correlation analysis to identify features

that are highly correlated with the target variable.

**3. Remove redundant features.** If two features are highly correlated

with each other, then you can remove one of the features, as they are

likely to contain redundant information.

**4. Remove irrelevant features.** If a feature is not correlated with the

target variable, then you can remove it, as it is unlikely to be useful for

prediction

**FEATURE SELECTION:**

**Checking for missing value**

In[2]:# missing value query

df.isna().sum()

**Out[2]:**

DayOfWeek 0

WeekOfYear 0

Day 0

Month 0

Year 0

PeriodOfDay 0

ForecastWindProduction 5

SystemLoadEA 2

SMPEA 2

ORKTemperature 295

ORKWindspeed 299

CO2Intensity 7

ActualWindProduction 5

SystemLoadEP2 2

SMPEP2 2

dtype: int64

**Model training:**

1. Choose a machine learning algorithm.

There are a number of different machine learning algorithms that can be used for house price prediction, such as linear regression, ridge regression, lasso regression ,decision trees, and random forests are Covered above.

**Machine learning Models:**

In[44]:X=df.drop("SMPEP2",axis=1)

y=df["SMPEP2"] X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3,random\_state=0)

In[45]:from xgboost import XGBRegressor

from catboost import CatBoostRegressor

from lightgbm import LGBMRegressor

In[46]:ridge=Ridge().fit(X\_train,y\_train)

lasso=Lasso().fit(X\_train,y\_train)

enet=ElasticNet().fit(X\_train,y\_train)

knn=KNeighborsRegressor().fit(X\_train,y\_train)

ada=AdaBoostRegressor().fit(X\_train,y\_train)

In[47]:svm=SVR().fit(X\_train,y\_train)

mlpc=MLPRegressor().fit(X\_train,y\_train)

dtc=DecisionTreeRegressor().fit(X\_train,y\_train)

rf=RandomForestRegressor().fit(X\_train,y\_train)

xgb=XGBRegressor().fit(X\_train,y\_train)

gbm=GradientBoostingRegressor().fit(X\_train,y\_train)

lgb=LGBMRegressor().fit(X\_train,y\_train)

catbost=CatBoostRegressor().fit(X\_train,y\_train)

In[48]:models=[ridge,lasso,dtc,rf,xgb,gbm,lgb,catbost,enet,knn,ada, mlpc,svm]

In[49]:def ML(y,models):

accuary=models.score(X\_train,y\_train)

return accuary

In[50]:for i in models:

print(i,"Algorithm succed rate :",ML("SMPEP2",i))

Out[50]: Ridge() Algorithm succed rate : 0.43121105926644243

Lasso() Algorithm succed rate : 0.42883198265818245

DecisionTreeRegressor() Algorithm succed rate : 1.0

RandomForestRegressor() Algorithm succed rate : 0.9424727172628374

XGBRegressor(base\_score=0.5, booster='gbtree', callbacks=None,

colsample\_bylevel=1, colsample\_bynode=1, colsample\_bytree=1,

early\_stopping\_rounds=None, enable\_categorical=False,

eval\_metric=None, gamma=0, gpu\_id=-1, grow\_policy='depthwise',

importance\_type=None, interaction\_constraints='',

learning\_rate=0.300000012, max\_bin=256, max\_cat\_to\_onehot=4,

max\_delta\_step=0, max\_depth=6, max\_leaves=0, min\_child\_weight=1,

missing=nan, monotone\_constraints='()', n\_estimators=100, n\_jobs=0,

num\_parallel\_tree=1, predictor='auto', random\_state=0,

reg\_alpha=0,

reg\_lambda=1, ...) Algorithm succed rate 0.8732530340524252

GradientBoostingRegressor() Algorithm succed rate : 0.5739399134995518

LGBMRegressor() Algorithm succed rate : 0.6953551703738294

<catboost.core.CatBoostRegressor object at 0x7f29e8bcc0d0> Algorithm succed rate : 0.7878389350009978

ElasticNet() Algorithm succed rate : 0.4290970871174

KNeighborsRegressor() Algorithm succed rate : 0.5964451293083145

AdaBoostRegressor() Algorithm succed rate : 0.26365965295085403

MLPRegressor() Algorithm succed rate : 0.15355550237922466

SVR() Algorithm succed rate : 0.23458514968207922

**RANDOM FOREST REGRESSOR:**

In[51]:X2=df[["SMPEA","SystemLoadEP2","SystemLoadEA","PeriodOfDay","year","ActualWindProduction"]]

y2=df["SMPEP2"]

In[52]:X\_train2,X\_test2,y\_train2,y\_test2=train\_test\_split(X2,y2,test\_size=0.3,random\_state=0)

In[53]:rf2=RandomForestRegressor().fit(X\_train2,y\_train2)

In[54]:rf2.score(X\_train2,y\_train2)

Out[55]:

**0.9345853165856398**

In[56]:X3=df\_remove\_out.drop("SMPEP2",axis=1)

y3=df\_remove\_out["SMPEP2"]

In[57]:X\_train3,X\_test3,y\_train3,y\_test3=train\_test\_split(X3,y3,test\_size=0.3,random\_state=0)

In[58]:rf3=RandomForestRegressor().fit(X\_train3,y\_train3)

In[59]:rf3.score(X\_train,y\_train)

Out[59]:

0.8965242074080007

**DECISION TREE REGRESSOR:**

In[60]:dtc3=DecisionTreeRegressor().fit(X\_train3,y\_train3)

In[61]:rf3.score(X\_train3,y\_train3)

Out[62]:

0.9525199962605772

**To train a model for electricity price prediction, follow these steps:**

**Data Preparation:**

**Choose a Model:**

Based on analysis and requirements, selecting a suitable model for the task. Common choices include regression models, time series models, machine learning algorithms, or deep learning architectures.

**Model Training:**

* **Initialize the Model**: Create an instance of the chosen model.
* **Parameter Tuning (if applicable):** Set hyperparameters based on domain knowledge or use techniques like grid search or random search for optimization.
* **Train the Model:** Fit the model to the training data. This involves adjusting the model's parameters to minimize the prediction error.

**Validation:**

Use the validation set to fine-tune hyperparameters and evaluate the model's performance.

**Hyperparameter Tuning:**

Experiment with different hyperparameter values to find the optimal configuration that minimizes the prediction error on the validation set.

**Evaluate Performance:**

Use appropriate evaluation metrics (e.g., Mean Absolute Error, Root Mean Squared Error) to assess the model's performance on both the validation and test sets.

**Model Selection:**

Based on the evaluation results, select the best-performing model.

**Final Model Training:**

Train the selected model on the combined training and validation sets to utilize the maximum amount of data.

**Testing:**

Use the test set to evaluate the final model's performance. This set serves as an unbiased evaluation of the model's generalization ability

**MODEL EVALUATION:**

To evaluate a model for electricity price prediction, you would typically use relevant metrics and techniques to assess its performance. Here are some steps:

**1.Split Data:** Divide your dataset into training and testing sets. This allows you to train the model on one portion and evaluate its performance on another.

**2.Select Evaluation Metrics:**

* **Mean Absolute Error (MAE):** Measures the average absolute difference between predicted and actual values.
* **Mean Squared Error (MSE):** Measures the average of the squared differences between predicted and actual values. It penalizes larger errors more heavily.
* **Root Mean Squared Error (RMSE):** The square root of MSE, providing an interpretable metric in the same unit as the target variable.
* **R-squared (R2):** Measures the proportion of the variance in the dependent variable that is predictable from the independent variables.

**3.Baseline Model**:

Start with a simple baseline model (like mean prediction) to compare the performance of your chosen model.

**4.Train and Validate Model:**

* Train your chosen model on the training set.
* Validate it on a validation set (if applicable) to fine-tune hyperparameters and avoid overfitting.

**5.Evaluate Performance:**

Use the selected evaluation metrics to assess how well the model performs on the test set.

**6.Compare with Baseline:**

Compare the model's performance with the baseline model. A good model should outperform the baseline.

**7.Visualize Results:**

* Plot actual vs. predicted values to visually inspect the model's performance.
* Create time-series plots if applicable to see how well the model captures trends and patterns.

**8.Consider Domain Knowledge:** Take into account any domain-specific insights that might affect the evaluation. For example, consider if there are certain periods or events that significantly impact electricity prices.

**9.Adjust and Iterate:**

* Based on the evaluation results, you may need to adjust hyperparameters, try different algorithms, or engineer features to improve performance.

**10.Finalize and Deploy:**

* Once satisfied with the model's performance, can finalize it and deploy it for practical use.

In this phase, we tested various models for electricity price estimation and achieved high success rates...

**Feature engineering:**

It is crucial for accurate electricity price prediction. Here are some steps you can take:

**Time-based Features:** Extract features like hour of the day, day of the week, month, holidays, and seasons. These can capture periodic patterns in electricity demand.

**Lagged Variables:** Include lagged versions of the target variable (previous prices) and other relevant features. This can help capture temporal dependencies.

**Weather Data**: Integrate weather-related features like temperature, humidity, wind speed, and sunlight hours. Weather strongly influences electricity demand.

**Calendar Events:** Incorporate special events or holidays that might affect electricity consumption patterns.

**Market Data:** Include features related to market conditions, such as supply-demand balance, fuel prices, and generation mix.

**Load Forecasting:** Use load forecasting models to generate predictions of electricity demand, which can be used as features.

**Temporal Aggregates:** Create summary statistics like rolling means, medians, and standard deviations over specific time periods.

**Categorical Variables:** Encode categorical variables like day of the week, month, and any other relevant categories.

**Interaction Features:** Explore interactions between different features, for example, the interaction between temperature and day of the week.

**Dimensionality Reduction**: Apply techniques like PCA or feature selection to reduce the number of irrelevant or redundant features.

**Historical Averages:** Compute statistics like historical averages, minimums, and maximums for different time frames.

**Periodic Components:** Apply Fourier or wavelet transforms to capture cyclical patterns in the data.

**Geographical Features:** If applicable, include features related to the geographical location, like regional demand patterns.

**Social and Economic Indicators:** Incorporate data related to economic activity, population density, and social events that might affect electricity consumption.

**Machine Learning Models:** Consider using more advanced models like Gradient Boosted Trees or Neural Networks that can automatically learn feature interactions.

**Different Analysis:**

**Time Series Analysis:**

Time series analysis is a statistical technique used to analyze and extract meaningful information from time-ordered data points.

In[64]:from datetime import datetime

df["DateTime"] = pd.to\_datetime(df.DateTime)

df['year'] = df['DateTime'].dt.year

df['month'] = df['DateTime'].dt.month

df["day"]=df["DateTime"].dt.day

# We have created 3 new columns

# we can start our time series analysis

# change of real price of consumed electricity with time

In[65]:custgroup=df.groupby('DateTime').mean()

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

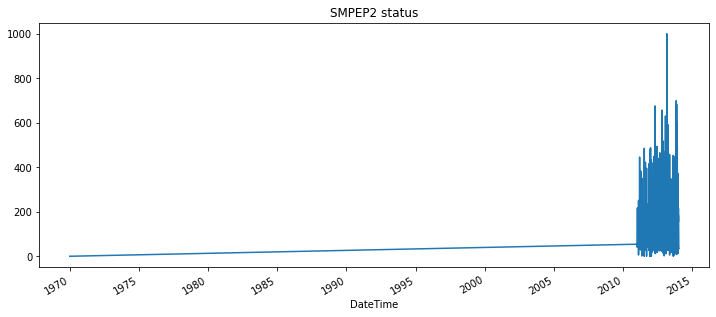
custgroup['SMPEP2'].plot(x=df.DateTime)

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

custgroup['SMPEP2'].plot(x=df.DateTime)

plt.title("SMPEP2 status")

plt.show()

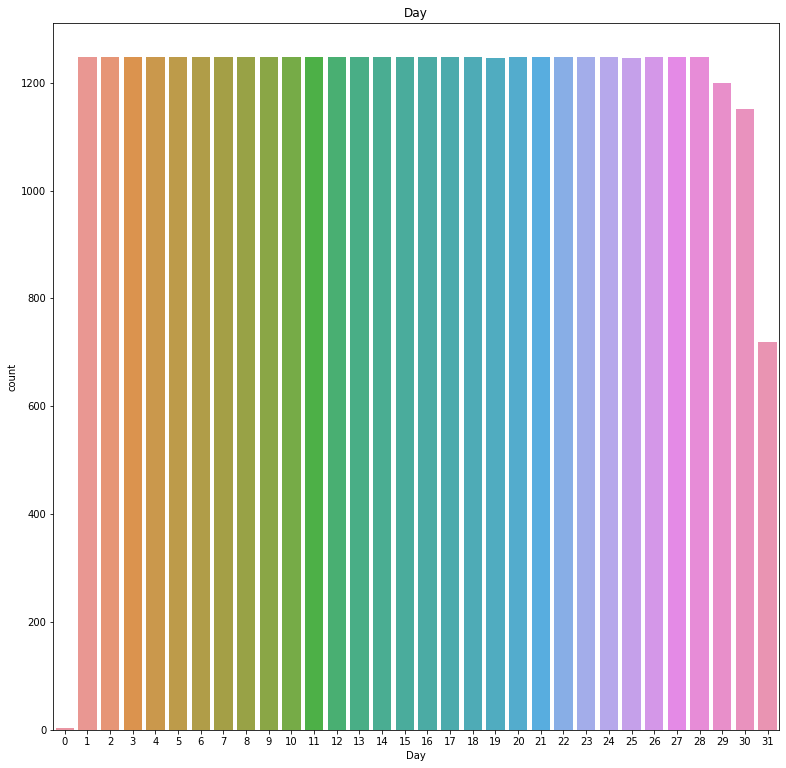
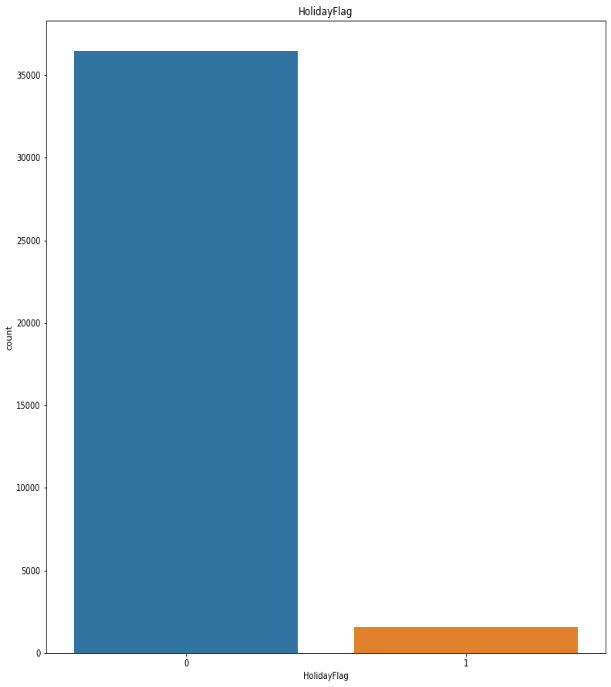
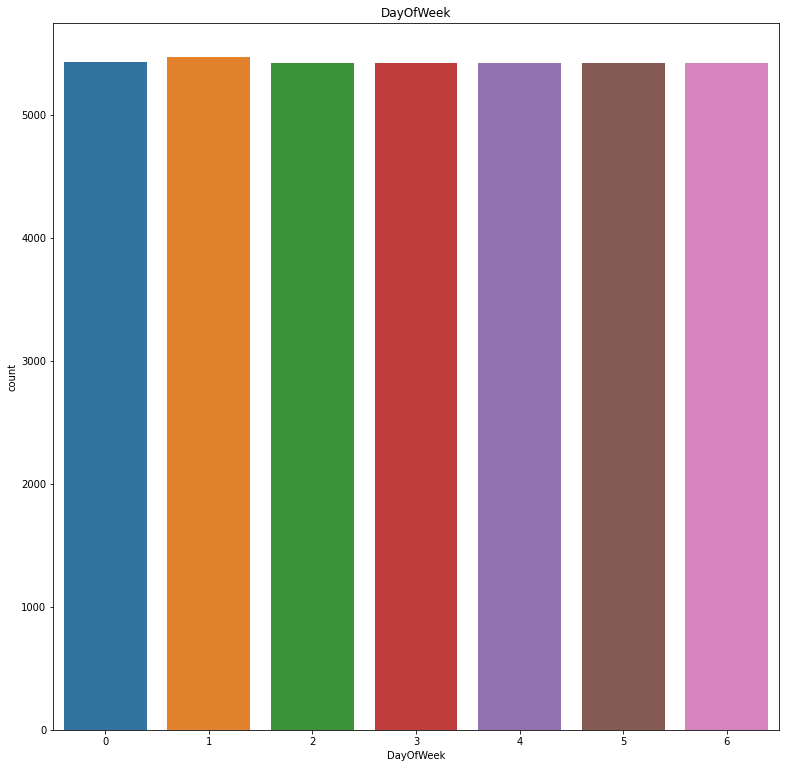
**Categorical Analysis:**

In[66]:for i in cat\_list:

plt.figure(figsize=(13,13))

countplot(x=i,data=df.loc[:,cat\_list])

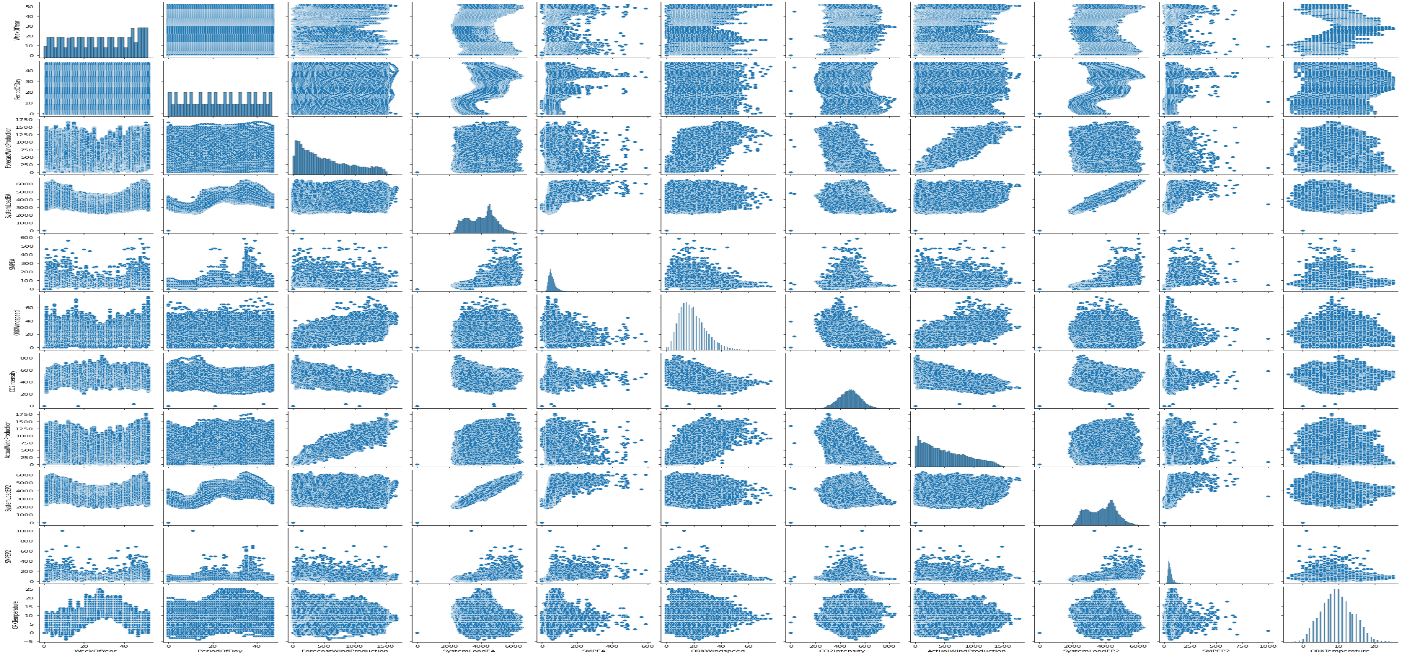
plt.title(i)



**Numerical analysis:**

Numerical analysis is a branch of mathematics and computer science that deals with the development and application of computational algorithms to solve mathematical problems. It involves techniques for approximating solutions to mathematical problems that may be too complex to solve analytically.

In[67]:sns.pairplot(df.loc[:,num\_list]);



**CONCLUSION:**

In conclusion, developing an effective electricity price prediction model is a complex undertaking, necessitating a comprehensive understanding of various factors influencing the electricity market. The challenges encountered in this endeavor range from the inherent volatility of market dynamics to the intricacies of renewable energy integration, and from policy shifts to technological advancements. Weather dependency, demand response programs, and transmission constraints further compound the difficulty in achieving accurate forecasts.