



**GRT INSTITUTE OF
ENGINEERING AND
TECHNOLOGY, TIRUTTANI - 631209**
Approved by AICTE, New Delhi Affiliated to Anna University, Chennai



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PHASE 4

PROJECT TITLE

***ELECTRICITY PRICE PREDICTION USING
MACHINE LEARNING***

COLLEGE CODE : 1103

LEELAVATHI N V

3rd yr, 5th sem

Reg no. : 110321104025

nvleela2004@gmail.com

FEATURE ENGINEERING:

Feature engineering in machine learning is the art of transforming raw data into a format that's more suitable for modeling. In simpler terms, you're jazzing up your data to make your machine learning model sing in harmony.

For an electricity price prediction project, feature engineering is like composing a symphony where each instrument contributes to the overall melody. Here are some ways you can engineer features for electricity price prediction:

- ❖ **Temporal Features:** Electricity prices often have a temporal pattern. Create features like time of day, day of the week, month, or even holidays. Maybe people use more electricity during certain seasons or at specific times of the day.
- ❖ **Weather-related Features:** Weather conditions can affect electricity usage. Include features like temperature, humidity, or even special events like storms or heatwaves.
- ❖ **Historical Prices:** Past prices can be strong indicators. Create features like rolling averages or percentage changes from previous periods.
- ❖ **Market Indicators:** Consider economic indicators or market trends that might influence electricity prices. GDP, inflation rates, or even trends in the energy market could be relevant.
- ❖ **Categorical Encoding:** If you have categorical variables, like types of energy sources or regions, encode them properly. One-hot encoding or label encoding can be used.
- ❖ **Interaction Features:** Sometimes, the combination of two or more features might have a more significant impact. Experiment with creating interaction terms.
- ❖ **Outlier Handling:** Identify and handle outliers in your data. Extreme values can distort predictions.
- ❖ **Scaling:** Ensure that your features are on similar scales. Scaling can be crucial, especially for algorithms sensitive to the magnitude of variables (e.g., distance-based algorithms).
- ❖ **Feature Selection:** Not every feature may contribute equally. Use techniques like recursive feature elimination or feature importance scores to select the most relevant features.

FEATURE SELECTION:

Selecting the right features for your electricity price prediction model is crucial for its performance. From the columns you've provided, it looks like you have a mix of temporal, categorical, and numerical features. Here's a breakdown of potential features you could consider:

1. Temporal Features:

- ❖ `DateTime`: Extract components like hour, minute, day of the week, month, etc.
- ❖ `DayOfWeek`: This can be redundant if you use `DateTime`, but sometimes it's useful on its own.

2. Categorical Features:

- ❖ `Holiday`: You can encode this as binary (1 for holiday, 0 for non-holiday).
- ❖ `HolidayFlag`: If it provides additional information beyond "is holiday," include it. Otherwise, it might be redundant.
- ❖ `PeriodOfDay`: Morning, afternoon, evening, night, etc.

3. Numerical Features:

- ❖ `WeekOfYear`, `Day`, `Month`, `Year`: These could be useful, especially if there are seasonal trends.
- ❖ `ForecastWindProduction`, `ActualWindProduction`: Wind production could influence electricity prices.
- ❖ `SystemLoadEA`, `SystemLoadEP2`: Past electricity consumption.
- ❖ `SMPEA`, `SMPEP2`: Social Market Price for Electricity, if relevant.
- ❖ `ORKTemperature`, `ORKWindspeed`: Weather-related features.
- ❖ `CO2Intensity`: Environmental impact could influence electricity prices.

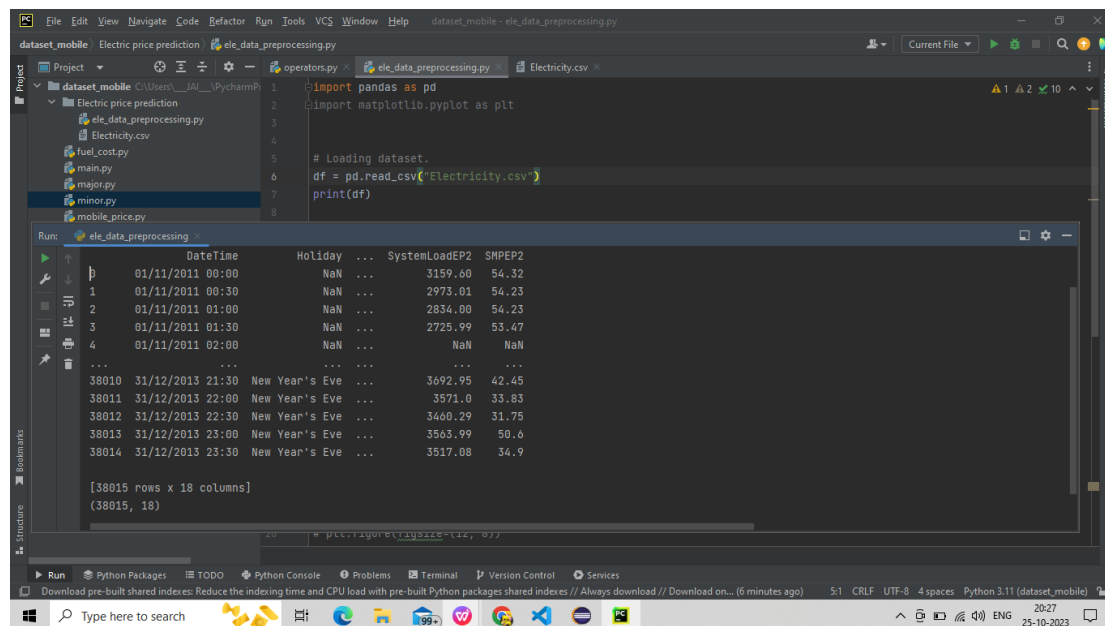
Correlation: Check for correlations between features. Highly correlated features might not add much new information.

Relevance: Ensure that each feature logically makes sense in the context of electricity price prediction. For example, if `HolidayFlag` is closely related to `Holiday`, including both might be unnecessary.

Dimensionality: Be mindful of the curse of dimensionality. Including too many irrelevant features could lead to overfitting.

Feature Importance: If you're using tree-based models like Random Forest or XGBoost, you can check feature importance to see which features contribute the most to the model's predictions.

1. Load The Dataset:



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor in the center. The code editor contains the following Python code:

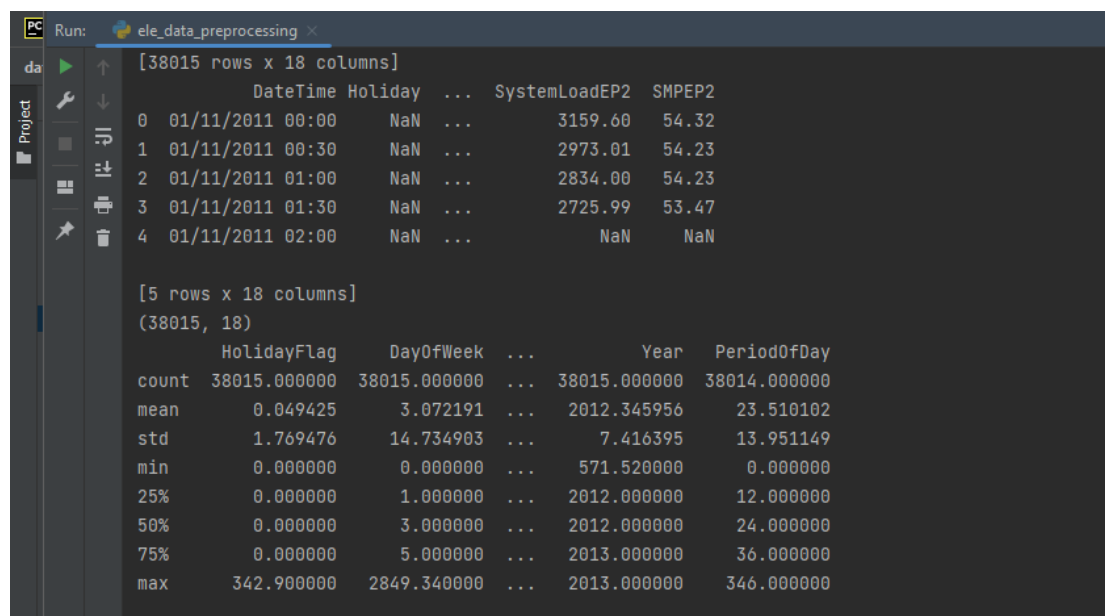
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Loading dataset.
5 df = pd.read_csv("Electricity.csv")
6 print(df)
```

The output of the code is displayed in the Run cell, showing the first few rows of the dataset and the dimensions of the DataFrame:

```
[38015 rows x 18 columns]
(38015, 18)
```

	DateTime	Holiday	...	SystemLoadEP2	SMPEP2
	01/11/2011 00:00	NaN	...	3159.60	54.32
1	01/11/2011 00:30	NaN	...	2973.01	54.23
2	01/11/2011 01:00	NaN	...	2834.00	54.23
3	01/11/2011 01:30	NaN	...	2725.99	53.47
4	01/11/2011 02:00	NaN	...	NaN	NaN
...
38010	31/12/2013 21:30	New Year's Eve	...	3692.95	42.45
38011	31/12/2013 22:00	New Year's Eve	...	3571.0	33.83
38012	31/12/2013 22:30	New Year's Eve	...	3460.29	31.75
38013	31/12/2013 23:00	New Year's Eve	...	3563.99	50.6
38014	31/12/2013 23:30	New Year's Eve	...	3517.08	34.9

2. Explore The Dataset:



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor in the center. The code editor contains the following Python code:

```
1 df.info()
2 df.describe()
```

The output of the code is displayed in the Run cell, showing the dimensions of the DataFrame and the first few rows of the dataset:

```
[38015 rows x 18 columns]
(38015, 18)
```

	DateTime	Holiday	...	SystemLoadEP2	SMPEP2
0	01/11/2011 00:00	NaN	...	3159.60	54.32
1	01/11/2011 00:30	NaN	...	2973.01	54.23
2	01/11/2011 01:00	NaN	...	2834.00	54.23
3	01/11/2011 01:30	NaN	...	2725.99	53.47
4	01/11/2011 02:00	NaN	...	NaN	NaN

The output of the code is displayed in the Run cell, showing the dimensions of the DataFrame and the first few rows of the dataset:

```
[5 rows x 18 columns]
(38015, 18)
```

	HolidayFlag	DayOfWeek	...	Year	PeriodOfDay
count	38015.000000	38015.000000	...	38015.000000	38014.000000
mean	0.049425	3.072191	...	2012.345956	23.510102
std	1.769476	14.734903	...	7.416395	13.951149
min	0.000000	0.000000	...	571.520000	0.000000
25%	0.000000	1.000000	...	2012.000000	12.000000
50%	0.000000	3.000000	...	2012.000000	24.000000
75%	0.000000	5.000000	...	2013.000000	36.000000
max	342.900000	2849.340000	...	2013.000000	346.000000

```
Run: ele_data_preprocessing x

[8 rows x 7 columns]
DateTime          1
Holiday           36478
HolidayFlag       0
DayOfWeek         0
WeekOfYear        0
Day               0
Month             0
Year              0
PeriodOfDay       1
ForecastWindProduction 1
SystemLoadEA      1
SMPEA             2
ORKTemperature    2
ORKWindspeed      2
CO2Intensity      2
ActualWindProduction 2
SystemLoadEP2     2
SMPEP2            2
dtype: int64
```

```
dtype: int64
DateTime          object
Holiday           object
HolidayFlag       float64
DayOfWeek         float64
WeekOfYear        float64
Day               float64
Month             float64
Year              float64
PeriodOfDay       float64
ForecastWindProduction object
SystemLoadEA      object
SMPEA             object
ORKTemperature    object
ORKWindspeed      object
CO2Intensity      object
ActualWindProduction object
SystemLoadEP2     object
SMPEP2            object
dtype: object
(38015, 18)

Process finished with exit code 0
```

2. Handle Missing Values:

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help dataset_mobile - ele_data_preprocessing.py
dataset_mobile Electric price prediction ele_data_preprocessing.py Electricity.csv
Project dataset_mobile C:\Users\... \PycharmP
  Electric price prediction
    ele_data_preprocessing.py
    Electricity.csv
    fuel_cost.py
    main.py
    major.py
    minor.py
    mobile_price.py
Run: ele_data_preprocessing
(38015, 18)
print(df.dtypes)
# Shape of dataset
print(df.shape)
# handling missing data
df = df.dropna()
print(df)

Run: ele_data_preprocessing
(38015, 18)
DateTime Holiday ... SystemLoadEP2 SMPEP2
2545 24/12/2011 00:00 Christmas Eve ... 3634.24 44.96
2546 24/12/2011 00:30 Christmas Eve ... 3382.16 44.96
2547 24/12/2011 01:00 Christmas Eve ... 3204.06 44.54
2548 24/12/2011 01:30 Christmas Eve ... 2997.83 43.48
2549 24/12/2011 02:00 Christmas Eve ... 2831.80 41.72
... ..
38010 31/12/2013 21:30 New Year's Eve ... 3692.95 42.45
38011 31/12/2013 22:00 New Year's Eve ... 3571.0 33.83
38012 31/12/2013 22:30 New Year's Eve ... 3460.29 31.75
38013 31/12/2013 23:00 New Year's Eve ... 3563.99 50.6
38014 31/12/2013 23:30 New Year's Eve ... 3517.08 34.9

[1536 rows x 18 columns]

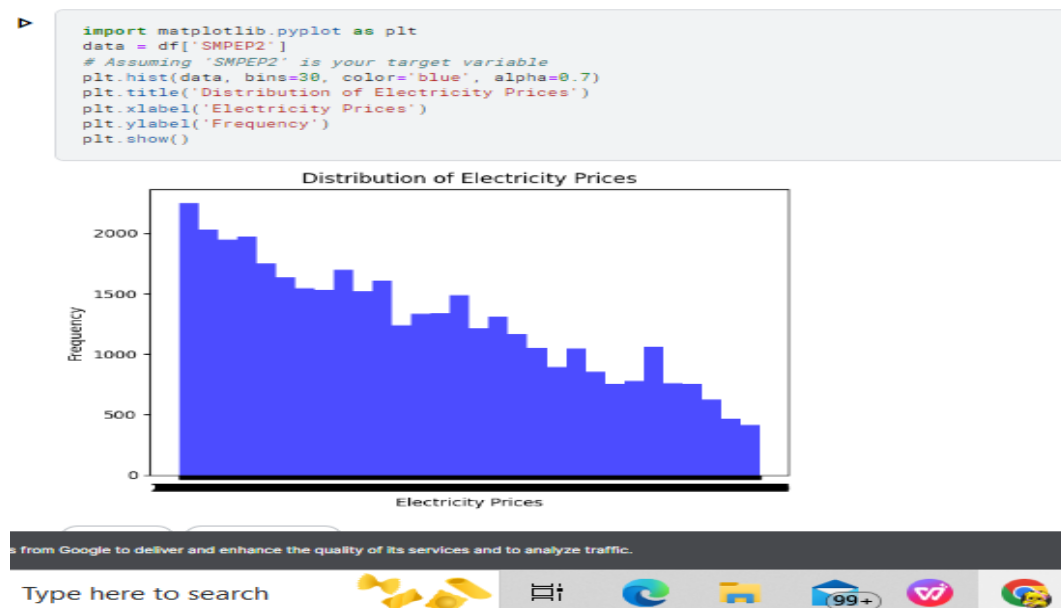
Process finished with exit code 0
```

3. Handling categorical values:

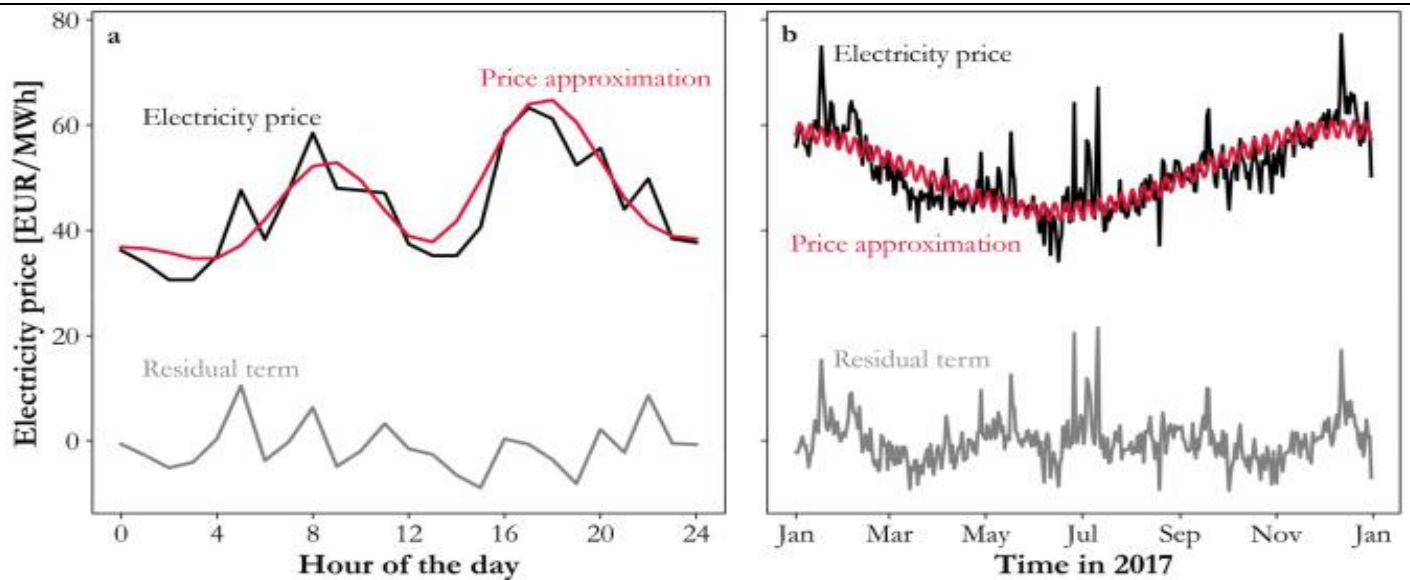
```
dataset_mobile - ele_data_preprocessing.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
dataset_mobile - ele_data_preprocessing.py
Project
dataset_mobile
  Electric price prediction
    ele_data_preprocessing.py
    Electricity.csv
    fuel_cost.py
    main.py
    major.py
    minor.py
    mobile_price.py
21 # handling missing data
22 df = df.dropna()
23 print(df)
24
25 # handling categorical data
26 df = pd.get_dummies(df, columns=['PeriodOfDay'])
27 print(df)
28

Run ele_data_preprocessing
2023-10-25 21:08:23.30 new Year's Eve ... 3317.00 34.7
[1536 rows x 18 columns]
Datetime Holiday ... PeriodOfDay_46.0 PeriodOfDay_47.0
2545 24/12/2011 00:00 Christmas Eve ... False False
2546 24/12/2011 00:30 Christmas Eve ... False False
2547 24/12/2011 01:00 Christmas Eve ... False False
2548 24/12/2011 01:30 Christmas Eve ... False False
2549 24/12/2011 02:00 Christmas Eve ... False False
... ..
38010 31/12/2013 21:30 New Year's Eve ... False False
38011 31/12/2013 22:00 New Year's Eve ... False False
38012 31/12/2013 22:30 New Year's Eve ... False False
38013 31/12/2013 23:00 New Year's Eve ... True False
38014 31/12/2013 23:30 New Year's Eve ... False True
[1536 rows x 65 columns]
Process finished with exit code 0
```

4. Exploring Data:

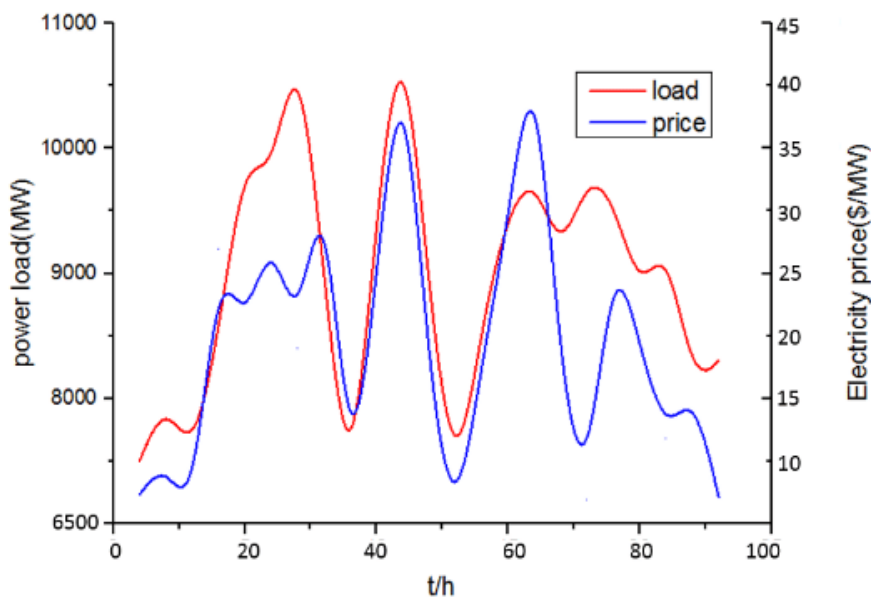


5.



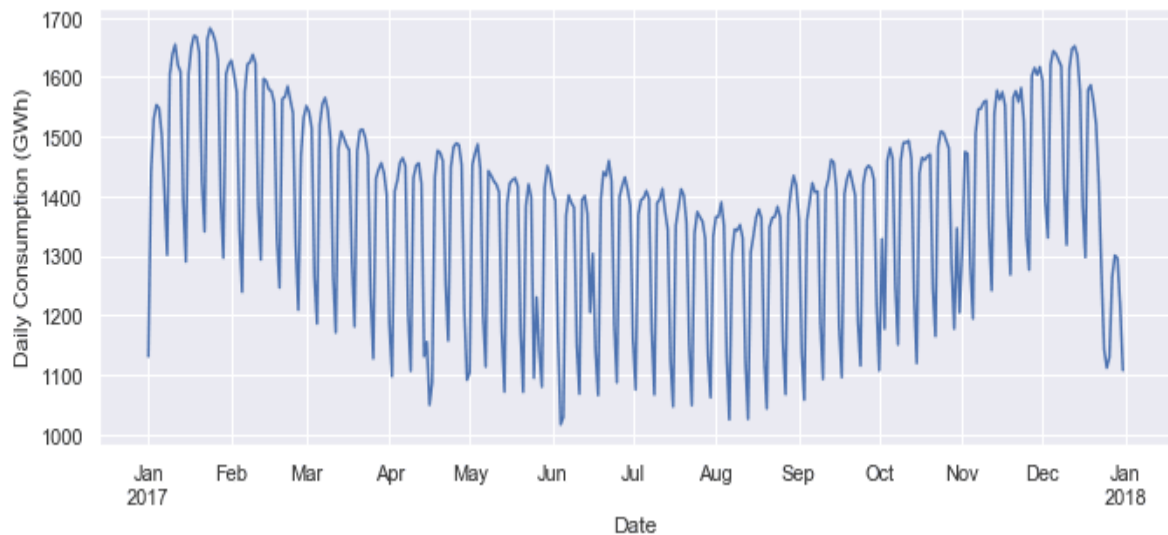
The above graph shows the relation between the Electricity price('SMPEP2') and the hours of the day, ('DateTime') and the price approximation on the year 2017 is predicted approximation electricity price prediction is visualized in the above picture.

6. Correlation Analysis

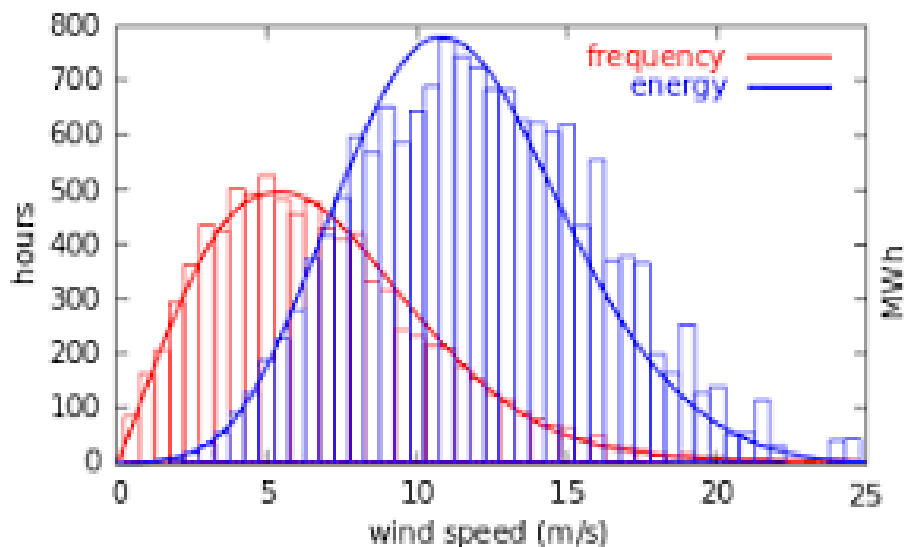


This gives an correlation between the electric prices and the power load in the given dataset.

7. Temporal analysis :



8. Wind Production Analysis:



These analysis provide a starting point of understanding your dataset and identifying potential patterns that can be useful for building an electricity price prediction model.

Summarization:

Feature Engineering:

1. Time-related features: Extracted additional features from the DateTime column, including day of the week, month, year, etc.

2. Lag features: Created lag features for the target variable ('SMPEP2') to capture temporal dependencies.

3. **Rolling statistics:** Calculated rolling mean, median, and standard deviation for relevant features to smooth out noise.
4. **Weather-related features:** Considered inclusion of weather-related features such as temperature, humidity, or wind speed if available.
5. **Special events:** Incorporated information about holidays or special events as binary features.
6. **Forecast features:** Utilized forecasted values for relevant features like 'ForecastWindProduction'.
7. **Interaction terms:** Considered creating interaction terms between features if they could have a significant impact.
8. **Cyclical features:** Encoded cyclical patterns, especially for time-related features, using sine and cosine transformations.
9. **Holiday indicators:** Created binary indicators for holidays or special events.
10. **Feature scaling:** Normalized or standardized numerical features to ensure they are on a similar scale.

Model Selection:

1. **Algorithm Selection:** Considered a variety of algorithms including time-series models (ARIMA, SARIMA) and machine learning algorithms (Random Forest, Gradient Boosting) for electricity price prediction.
2. **Data Splitting:** Split the data into training, validation, and testing sets to train and evaluate the models.
3. **Model Training:** Trained the selected models using historical data, tuning hyperparameters for optimal performance.
4. **Evaluation Metrics:** Evaluated models using appropriate metrics such as MAE, RMSE, or others.

5.**Comparison:** Compared the performance of different models to select the one that best suits the project's requirements.

6. **Deployment Considerations:** Prepared for the deployment of the chosen model for making future predictions.