Project 4: Electricity Prices Prediction

Phase 5 - Final Submission

Team Members:
Ajita Fairen J - 715521106003 - B.E. ECE - 3rd year
M. Mekanya - 715521106028 - B.E. ECE - 3rd year
Shanmugapriya M - 715521106043 - B.E. ECE - 3rd year
K Shree Harini - 715521106044 - B.E. ECE - 3rd year
Vasanth L - 715521106055 - B.E. ECE - 3rd year
G. R. Tharunika - 715521106310 - B.E. ECE - 3rd year

Phase 5 Task: To document the model and submit the code files and other related files in GitHub or personal portfolio for others to access and review.

Problem Statement:
To create a predictive model that utilizes electricity prices and relevant factors to forecast future electricity prices, assisting energy providers and consumers in making informed decisions regarding consumption and investment.

Objective: Create a tool that assists both energy providers and consumers in making informed decisions regarding consumption and investment by predicting future electricity prices.

Design Thinking Process:

Step 1: Empathize
- Understand the perspective of energy providers and consumers.
- Identify their pain points and needs related to electricity price forecasting.
- Gather feedback from stakeholders to align the project with their requirements.

Step 2: Define
- Clearly define the problem statement and objectives.
- Identify the key success criteria for the predictive model.

- Determine the relevant factors that may affect electricity prices (e.g., weather, demand, market trends).

Step 3: Ideate
- Brainstorm potential data sources for historical electricity prices and relevant factors.
- Explore different modeling approaches, algorithms, and techniques for price forecasting.
- Consider the scalability and interpretability of the chosen model.

Step 4: Implementation and Prototype
- Develop a data pipeline to collect and preprocess historical data.
- Create initial data visualizations to gain insights into the data.
- Implement a basic predictive model as a starting point for experimentation.

Step 5: Test
- Evaluate the initial model's performance using appropriate metrics (e.g., RMSE, MAE).
- Collect feedback from stakeholders and make necessary adjustments.
- Consider incorporating additional features or data sources to improve the model.

Phases of Development:

1. Data Collection:
- Collect historical electricity price data.
- Gather relevant factors data (e.g., weather, demand, market data).

2. Data Preprocessing:
- Clean any unwanted data
- Preprocess the data to handle missing values and outliers.

3. Feature Engineering:
- Create meaningful features that may impact electricity prices.
- Explore time-based features, seasonal patterns, and lag variables.

4. Data Splitting
- Split the data into two for testing and training accordingly.

5. Model Selection:

- Experiment with various predictive models (e.g., regression, time series models).
- Assess the model's performance using cross-validation techniques.

6. Model Training:

- Train the selected model on historical data.
- Fine-tune hyperparameters for optimal performance.

7. Model Evaluation:

- Evaluate the model's accuracy and reliability using appropriate metrics.
- Validate the model against out-of-sample data.

Dataset used:

Source:
https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction/

This dataset contains information related to electricity markets and factors that can influence electricity prices.

1. DateTime:
   Records the timestamp, which can be used for time-based analysis.
2. Holiday:
   Indicates if a day is a holiday (categorical variable).
3. HolidayFlag:
   Possibly a binary flag related to holidays.
4. DayOfWeek:
   Indicates the day of the week (categorical variable).
5. WeekOfYear:
   Represents the week number within the year.
6. Day:
   Captures the day of the month.
7. Month:
   Records the month of the year.
8. Year:
   Specifies the year.

9. PeriodOfDay:
    Indicates a time period within a day (morning, afternoon, evening, etc.).
10. ForecastWindProduction:
    Predicted wind power production.
11. SystemLoadEA:
    Represents the electricity system load.
12. SMPEA:
    Likely the spot market price for energy in Australia (target variable).
13. ORKTemperature:
    Records temperature data.
14. ORKWindspeed:
    Captures wind speed data.
15. CO2 Intensity:
    Represents carbon dioxide intensity, possibly related to environmental factors.
16. ActualWindProduction:
    Actual wind power production.
17. SystemLoadEP2:
    Another measurement of the electricity system load.
18. SMPEP2:
    Possibly another instance of spot market price for energy.

Columns used:

1. DateTime:
This column can be useful for capturing temporal patterns and trends in electricity prices. You may need to extract features such as the time of day or day of the week from this column to improve the model's performance.

2. SystemLoadEA:
This column represents the system load in the electricity grid, which is a critical factor affecting electricity prices. Including this as a feature is essential.

3. SMPEA:
SMPEA (Spot Market Price for Energy Australia) likely represents the target variable, i.e., the electricity prices you want to predict. You should include this column as the target variable in your model.

4. ORKTemperature:
Weather-related features, such as temperature, can have a significant impact on electricity prices. Including temperature data as a feature can help capture weather-related price fluctuations.

5. ORKWindspeed:
Similar to temperature, wind speed can also impact electricity prices, particularly in regions where wind power generation is significant. Include this column as a feature.

6. CO2Intensity:
Carbon dioxide (CO_2) intensity can be another relevant factor in electricity prices, especially in regions where carbon pricing mechanisms are in place. Including this column can help account for environmental factors affecting prices.

7. ActualWindProduction:
Wind power generation can directly influence electricity prices. Including this column as a feature can capture the impact of wind power on prices.

Data Preprocessing steps:

Data Loading:
- The code starts by loading the dataset from a CSV file using the pd.read_csv function from the Pandas library.
- The "low_memory" parameter is set to False to suppress a warning related to the data type inference.

Data Transformation:
- The code performs data type conversion for several columns to ensure they are of numerical data type. This is done using pd.to_numeric with the errors='coerce' argument, which converts non-numeric values to NaN (missing values).

- The columns that are converted include "ForecastWindProduction," "SystemLoadEA," "SMPEA," "ORKTemperature," "ORKWindspeed," "CO2Intensity," "ActualWindProduction," "SystemLoadEP2," and "SMPEP2".

Data Scaling:
- Standardization (scaling) of the numerical features is performed using the StandardScaler from Scikit-Learn.
- The selected columns are scaled to have a mean of 0 and a standard deviation of 1. The following columns are scaled: "Day," "Month," "ForecastWindProduction," "SystemLoadEA," "SMPEA," "ORKTemperature," "ORKWindspeed," "CO2Intensity," "ActualWindProduction," and "SystemLoadEP2."

Data Cleaning:
- The code prints the number of null values in the dataset using data.isnull().sum() to identify columns with missing values.
- Null values are then dropped from the dataset using data.dropna(). This step removes rows with missing values in any of the columns.
- After this step, the dataset no longer contains any null values, as confirmed by printing data.isnull().sum() again.

Data Splitting:
- The code splits the dataset into training and testing sets using train_test_split from Scikit-Learn.
- The feature columns are assigned to x, and the target column ("SMPEP2") is assigned to y.
- The data is split into an 80% training set and a 20% testing set, with a random seed for reproducibility. The resulting sets are xtrain, xtest, ytrain, and ytest.

<u>Model Training Process:</u>

A Random Forest Regressor model is created using RandomForestRegressor() and trained on the training data with model.fit(xtrain, ytrain).

Feature Input and Prediction:
- The code allows the user to input values for the features used in the model. The user is prompted to enter values for the following features: "Day," "Month,"

"ForecastWindProduction," "SystemLoadEA," "SMPEA," "ORKTemperature," "ORKWindspeed," "CO2Intensity," "ActualWindProduction," and "SystemLoadEP2."
- The user's input is transformed using the same scaler applied to the training data.
- The model is used to predict the target variable ("SMPEP2") based on the user's input features.

Evaluation Metrics

- The user is prompted to enter the actual price for comparison with the model's prediction.
- Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) are calculated to evaluate the model's performance, comparing the actual and predicted values.

Source Code:

EPP.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error

#Data Loading

print("DATA LOADING\n\n")
data = pd.read_csv("Electricity.csv",low_memory=False)
print("Head of the dataset\n")
print(data.head())
print("\nInfo of the dataset\n")
print(data.info())
print("\nDescription of the dataset\n")
```

```
print(data.describe())

#Data Preprocessing

print("\n\n\nDATA TRANSFORMATION\n\n")
#Changing the type of data in the dataset to numerical values
data["ForecastWindProduction"] = pd.to_numeric(data["ForecastWindProduction"],errors
= 'coerce')
data["SystemLoadEA"] = pd.to_numeric(data["SystemLoadEA"],errors = 'coerce')
data["SMPEA"] = pd.to_numeric(data["SMPEA"],errors = 'coerce')
data["ORKTemperature"] = pd.to_numeric(data["ORKTemperature"],errors = 'coerce')
data["ORKWindspeed"] = pd.to_numeric(data["ORKWindspeed"],errors = 'coerce')
data["CO2Intensity"] = pd.to_numeric(data["CO2Intensity"],errors = 'coerce')
data["ActualWindProduction"] = pd.to_numeric(data["ActualWindProduction"],errors =
'coerce')
data["SystemLoadEP2"] = pd.to_numeric(data["SystemLoadEP2"],errors = 'coerce')
data["SMPEP2"] = pd.to_numeric(data["SMPEP2"],errors = 'coerce')
print(data.info())

#Data Scaling
scaler = StandardScaler()
# Fit and transform the features for scaling
data[["Day", "Month", "ForecastWindProduction", "SystemLoadEA", "SMPEA",
"ORKTemperature", "ORKWindspeed", "CO2Intensity", "ActualWindProduction",
"SystemLoadEP2"]] = scaler.fit_transform(data[["Day", "Month", "ForecastWindProduction",
"SystemLoadEA", "SMPEA", "ORKTemperature", "ORKWindspeed", "CO2Intensity",
"ActualWindProduction", "SystemLoadEP2"]])

print("\n\n\nDATA CLEANING\n\n")
#Data Cleaning
#Displaying the no. of data which has null values in it
print("With Null Values\n\n")
print(data.isnull().sum())
#Dropping or cleaning the null values
data = data.dropna()
#When displayed again there are no null values
```

```
print("\n\nAfter Dropping Null Values\n\n")
print(data.isnull().sum())

#Data Splitting
#Data is split into training and test tests
x = data[["Day", "Month", "ForecastWindProduction", "SystemLoadEA", "SMPEA",
"ORKTemperature", "ORKWindspeed", "CO2Intensity", "ActualWindProduction",
"SystemLoadEP2"]]
y = data["SMPEP2"]
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)

print("\n\n\nDATA SPLITTING\n\n")
print("x train\n\n")
print(xtrain)
print("\n\nx test\n")
print(xtest)
print("\n\ny train\n")
print(ytrain)
print("\n\ny test \n")
print(ytest)

print("\n\nMODEL TRAINING: RANDOM FOREST REGRESSOR")
model = RandomForestRegressor()
model.fit(xtrain, ytrain)

print("\n\nFEATURES IN THE MODEL")
#features = [["Day", "Month", "ForecastWindProduction", "SystemLoadEA", "SMPEA",
"ORKTemperature", "ORKWindspeed", "CO2Intensity", "ActualWindProduction",
"SystemLoadEP2"]]
Day = int(input("Enter Day:"))
Month = int(input("Enter Month:"))
FWP = float(input("Enter ForecastWindProduction:"))
SLE = float(input("Enter SystemLoadEA:"))
SMP = float(input("Enter SMPEA:"))
ORKT= float(input("Enter ORKTemperature:"))
ORKW = float(input("Enter ORKWindspeed:"))
```

```
CO2 = float(input("Enter CO2Intensity:"))
Actualwind = float(input("Enter Actual Wind Production:"))
SLE2 = float(input("Enter SystemLoadEP2:"))


features = np.array([[Day, Month, FWP, SLE, SMP, ORKT, ORKW, CO2, Actualwind, SLE2]])
# Transform the features with the same scaler
features_scaled = scaler.transform(features)
predictions = model.predict(features_scaled)
print("\n\nPredicted Price:\n\n", predictions)

#Evaluating the Model
print("\n\n\nMODEL EVALUATION")
actual = float(input("\nEnter the Actual Price:"))
mae = mean_absolute_error([actual], predictions)
mse = mean_squared_error([actual], predictions)
rmse = np.sqrt(mse)

print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
```

Month_Vs_SMPEP2.py

```
import pandas as pd
import numpy as np
data = pd.read_csv("Electricity.csv")
print(data.head())
data.info()

data["ForecastWindProduction"] = pd.to_numeric(data["ForecastWindProduction"],
errors= 'coerce')
data["SystemLoadEA"] = pd.to_numeric(data["SystemLoadEA"], errors= 'coerce')
data["SMPEA"] = pd.to_numeric(data["SMPEA"], errors= 'coerce')
data["ORKTemperature"] = pd.to_numeric(data["ORKTemperature"], errors= 'coerce')
data["ORKWindspeed"] = pd.to_numeric(data["ORKWindspeed"], errors= 'coerce')
```

```
data["CO2Intensity"] = pd.to_numeric(data["CO2Intensity"], errors= 'coerce')
data["ActualWindProduction"] = pd.to_numeric(data["ActualWindProduction"], errors=
'coerce')
data["SystemLoadEP2"] = pd.to_numeric(data["SystemLoadEP2"], errors= 'coerce')
data["SMPEP2"] = pd.to_numeric(data["SMPEP2"], errors= 'coerce')
data.isnull().sum()
data = data.dropna()

x = data[["Day", "Month", "ForecastWindProduction", "SystemLoadEA", "SMPEA",
"ORKTemperature", "ORKWindspeed", "CO2Intensity", "ActualWindProduction",
"SystemLoadEP2"]]
y = data["SMPEP2"]
import seaborn as sns
import matplotlib.pyplot as plt
sns.barplot(data=data, x="Month", y="SMPEP2")

plt.xlabel("Month")
plt.ylabel("SMPEP2")
plt.title("Month Vs SMPEP2")
plt.show()
```

Correlation.py

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Load your dataset
data = pd.read_csv("Electricity.csv")
# Print the first few rows and data info
print(data.head())
data.info()
# Convert relevant columns to numeric, handling errors
numeric_columns = [
    "ForecastWindProduction", "SystemLoadEA", "SMPEA",
```

```
    "ORKTemperature", "ORKWindspeed", "CO2Intensity",
    "ActualWindProduction", "SystemLoadEP2", "SMPEP2"
]
data[numeric_columns] = data[numeric_columns].apply(pd.to_numeric, errors='coerce')

# Check for missing values and drop rows with missing values
data.isnull().sum()
data.dropna(inplace=True)

# Select features for correlation calculation
features = [
    "ForecastWindProduction", "SystemLoadEA", "SMPEA",
    "ORKTemperature", "ORKWindspeed", "CO2Intensity",
    "ActualWindProduction", "SystemLoadEP2", "SMPEP2"
]
# Calculate correlations and create a heatmap
correlations = data[features].corr(method='pearson')
plt.figure(figsize=(16, 12))
sns.heatmap(correlations, cmap="coolwarm", annot=True)
plt.show()
```

<u>Outputs:</u>

Data Loading:

```
DATA LOADING


Head of the dataset

           DateTime Holiday  HolidayFlag  DayOfWeek  WeekOfYear  Day  ... ORKTemperature  ORKWindspeed  CO2Intensity ActualWindProduction SystemLoadEP2 SMPEP2
0  01/11/2011 00:00     NaN            0          1          44    1  ...           6.00          9.30        600.71               356.00       3159.60  54.32
1  01/11/2011 00:30     NaN            0          1          44    1  ...           6.00         11.10        605.42               317.00       2973.01  54.23
2  01/11/2011 01:00     NaN            0          1          44    1  ...           5.00         11.10        589.97               311.00       2834.00  54.23
3  01/11/2011 01:30     NaN            0          1          44    1  ...           6.00          9.30        585.94               313.00       2725.99  53.47
4  01/11/2011 02:00     NaN            0          1          44    1  ...           6.00         11.10        571.52               346.00       2655.64  39.87

[5 rows x 18 columns]
```

Info of the dataset

```
Info of the dataset

<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
None
```

Description of the dataset:

```
Description of the dataset

        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
```

Data Transformation

```
DATA TRANSFORMATION


<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  38009 non-null  float64
 10  SystemLoadEA            38012 non-null  float64
 11  SMPEA                   38012 non-null  float64
 12  ORKTemperature          37719 non-null  float64
 13  ORKWindspeed            37715 non-null  float64
 14  CO2Intensity            38007 non-null  float64
 15  ActualWindProduction    38009 non-null  float64
 16  SystemLoadEP2           38012 non-null  float64
 17  SMPEP2                  38012 non-null  float64
dtypes: float64(9), int64(7), object(2)
memory usage: 5.2+ MB
None
```

Data Cleaning:

```
DATA CLEANING


With Null Values


DateTime                      0
Holiday                   36478
HolidayFlag                   0
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
```

After Dropping Null Values:

```
After Dropping Null Values


DateTime                  0
Holiday                   0
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 Splitting:

```
DATA SPLITTING


x train


       Day  Month  ForecastWindProduction  SystemLoadEA  SMPEA  ORKTemperature  ORKWindspeed  CO2Intensity  ActualWindProduction  SystemLoadEP2
20485   31     12                  937.42       4709.48  84.13             5.0          20.4        432.98                 735.0        4454.59
10376    4      6                   34.79       2359.78  49.89             7.0           9.3        641.22                  53.0        2260.84
2592    25     12                 1225.85       4266.13  44.38             9.0          25.2        367.90                1227.0        3280.81
2990     2      1                 1264.26       2980.84  25.05             2.0          20.4        402.11                 750.0        2663.97
17434   29     10                  205.36       2865.76  38.90             5.0          18.5        730.15                 185.0        2737.51
...    ...    ...                     ...           ...    ...             ...           ...           ...                   ...            ...
26529    6      5                  258.80       4073.50  90.69            11.0          16.7        438.98                 178.0        3912.03
27860    3      6                   66.10       3933.23  75.24            15.0          13.0        464.02                  30.0        3927.27
37673   24     12                  715.87       4525.94  76.91             3.0           7.4        362.45                 613.0        3918.75
24178   18      3                  297.39       4712.93  74.24             5.0          22.2        490.32                 355.0        4648.78
27856    3      6                   56.80       3545.50  70.82            14.0          13.0        533.35                  22.0        3554.09

[1132 rows x 10 columns]
```

```
x test

       Day  Month  ForecastWindProduction  SystemLoadEA  SMPEA  ORKTemperature  ORKWindspeed  CO2Intensity  ActualWindProduction  SystemLoadEP2
20530    1      1                  499.60       4893.88  98.95             6.0          13.0        410.33                 421.0        4524.85
24714   29      3                  887.40       4640.69  74.03             3.0          22.2        493.69                 576.0        4086.59
7637     8      4                  333.31       2640.94  49.60             8.0          11.1        422.36                 397.0        2453.58
6675    19      3                  245.40       3198.43  54.10             3.0           5.6        619.30                 226.0        2871.33
6686    19      3                  498.70       3376.08  51.83             5.0          11.1        530.52                 529.0        2775.95
...    ...    ...                     ...           ...    ...             ...           ...           ...                   ...            ...
38006   31     12                 1160.57       4188.85  66.08             5.0          18.5        262.97                1143.0        4207.57
3023     2      1                 1456.80       4214.78  42.57            10.0          48.2        373.80                1274.0        3493.14
7608     7      4                  290.70       4096.35  55.35            11.0          25.9        543.80                 533.0        3871.78
24746   30      3                  454.40       4370.79  66.08             3.0          25.9        503.30                 591.0        4024.49
20120   24     12                  740.20       3195.72  47.81             7.0          14.8        469.45                 532.0        2452.40

[284 rows x 10 columns]
```

```
y train

20485    87.47
10376    39.75
2592     43.96
2990     38.35
17434    51.26
         ...
26529    74.81
27860    73.33
37673    74.74
24178    98.09
27856    73.10
Name: SMPEP2, Length: 1132, dtype: float64


y test

20530    255.04
24714     99.50
7637      47.12
7637      47.12
6675      45.79
6686      45.88
          ...
38006     62.05
3023      41.38
7608      59.85
24746     66.08
20120     45.45
Name: SMPEP2, Length: 284, dtype: float64
```

Model Training: Getting Past Values

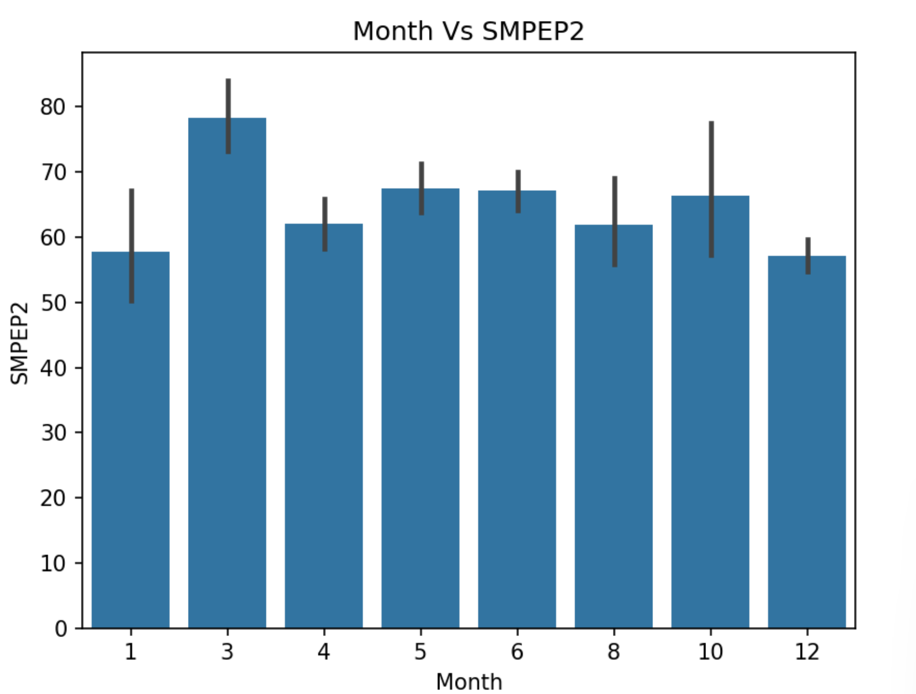
```
MODEL TRAINING: RANDOM FOREST REGRESSOR


FEATURES IN THE MODEL
Enter Day:10
Enter Month:12
Enter ForecastWindProduction:54.10
Enter SystemLoadEA:4241.05
Enter SMPEA:49.56
Enter ORKTemperature:9.0
Enter ORKWindspeed:14.8
Enter CO2Intensity:491.32
Enter Actual Wind Production:54.0
Enter SystemLoadEP2:4426.85
```

Predicted Price and Evaluation of the output with the actual price

```
Predicted Price:

 [94.8056]



MODEL EVALUATION

Enter the Actual Price:100.00
Mean Absolute Error (MAE): 5.19
Mean Squared Error (MSE): 26.98
Root Mean Squared Error (RMSE): 5.19
```

Data Visualization

Month Vs SMPEP2



Correlation Graph