Project 4: Electricity Prices Prediction

Phase 4: Development Part - 2

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

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.

Phase 4 Task: To continue building the electricity prices prediction model by: Feature Engineering, Model Training and Evaluation

Dataset:

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.

Source Code:

```
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)
```

#Model Training

```
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}")
```

Output:

Model Training: Getting Past Values

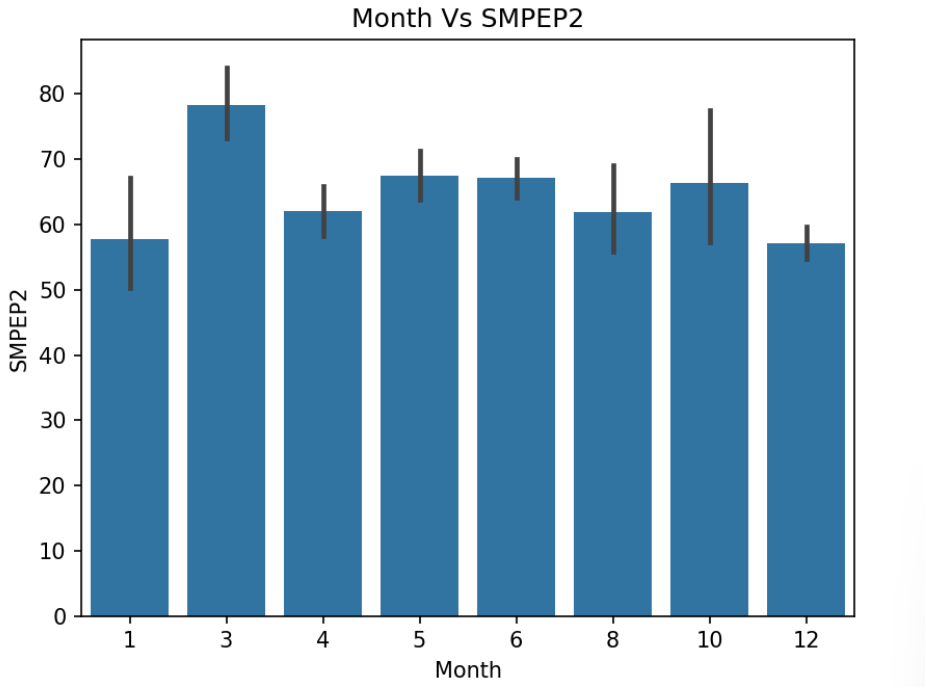
```
MODEL TRAINING: RANDOM FOREST REGRESSOR


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