**Capstone Project Report**

**Software Technology 4483**

Brynn Sloane

**Introduction**

In the dynamic landscape of energy markets, accurately predicting electricity prices is crucial for stakeholders, including utilities, regulators, and market participants. This report outlines the development and enhancement of a machine learning model designed to predict the System Marginal Price for Electricity (SMPEA). The project involved preprocessing the dataset from kaggle, addressing any issues with the data set, training and deploying a RandomForestRegressor model, and creation of a user-friendly GUI for predictions.

### **Objectives**

The primary goal of this project was to develop an accurate model to predict SMPEA. The project aimed to provide a GUI to make the model accessible, enabling users to input relevant features and receive predictions in seconds.

### **Data Preparation**

The dataset has 38014 rows of data, spanning the years 2011 to 2013, and 18 columns. To ensure data quality and performance of the model, a number of preprocessing steps were performed prior to model training:

* Managing Missing Values: NaN was used in place of "?" and any relevant columns were converted to numeric types, with errors being forced to NaN. Missing values are forward filled to preserve continuity.
* Outlier Removal: To improve the robustness of the model, outliers were identified and removed using the Interquartile Range approach.
* Correlation Analysis: To examine correlations, a heatmap was created. This allowed features that were highly associated to be removed.
* Class Imbalance: In order to ensure that the model provided equal importance to both holiday and non-holiday predictions during training, the imbalance in the HolidayFlag feature was addressed by oversampling the minority class.

### 

### 

### **Model Development**

The Linear Regression approach was first used during the early stages of development, but I discovered it to be generally incorrect. As a result, I experimented with several machine learning models that were assessed using K-fold cross-validation, and assessing their accuracy using algorithm comparison and SMAPE, such as Decision Tree, Naive Bayes, Support Vector Machines, Gradient Boosting, and Random Forest. Based on generalization and accuracy, the Random Forest model turned out to be the best-performing model.

* **Final Model: RandomForestRegressor with a random state of 42 and 100 estimators.**
* **Model Accuracy: The model's strong SMAPE-based accuracy demonstrates how well it can forecast the price of power.**

### **GUI Creation**

A GUI was created with Tkinter to improve accessibility and make the model usable. With the GUI, users can input parameters like temperature, system load, and anticipated wind production to get projections for SMPEA in AUD/MWh. In order to ensure a seamless and productive user experience, the interface has user-friendly input areas that are clearly labeled with easy instructions for each feature. It also contains extensive error handling to assist users in correcting any mistakes.

### **Conclusion**

This project was a first attempt to create and improve a machine learning model for estimating the cost of electricity and incorporating it into a graphical user interface. A large audience can use the tool thanks to its GUI, and the model's performance was carefully assessed. By helping participants in the energy market make well-informed decisions based on precise price estimates, this predictive technology can help the market run more smoothly.

**EDA**

Performing the EDA was a crucial step, and to do it, I used Google Colab, as it was an easy way to gather graphs needed, and it was one I was deeply familiar with.

**Figuring out graphs needed:**

**from google.colab import drive**

**drive.mount('/content/drive')**

**# Import libraries**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**import missingno as msno**

**import plotly.graph\_objects as go**

**import plotly.express as px**

**import warnings**

**# Configure settings**

**warnings.filterwarnings('ignore')**

**%matplotlib inline**

**# Read dataset**

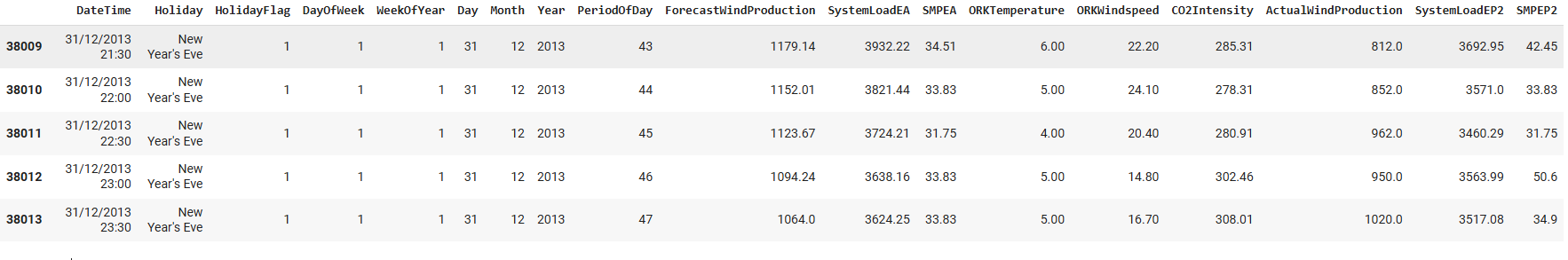
**df = pd.read\_csv("/content/drive/My Drive/thing/Electricity.csv")**

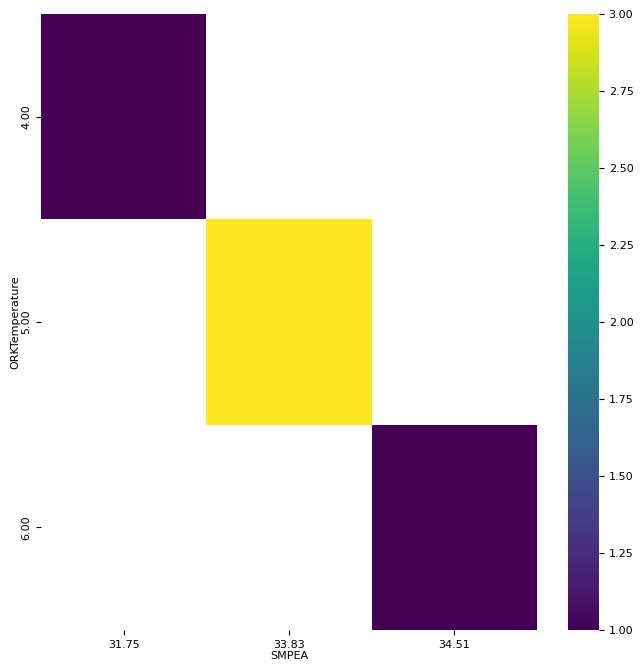
**# Checking description (first 5 rows)**

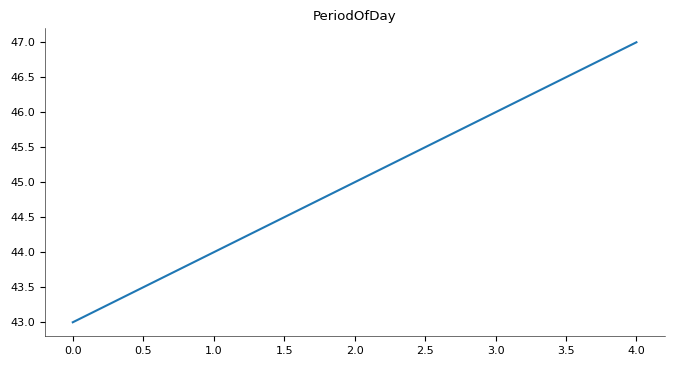
**df.head()**

**# Checking description (last 5 rows)**

**df.tail()**

****





# Checking description (last 5 rows)

df.tail()

# Rows and columns - data shape (attributes & samples)

df.shape

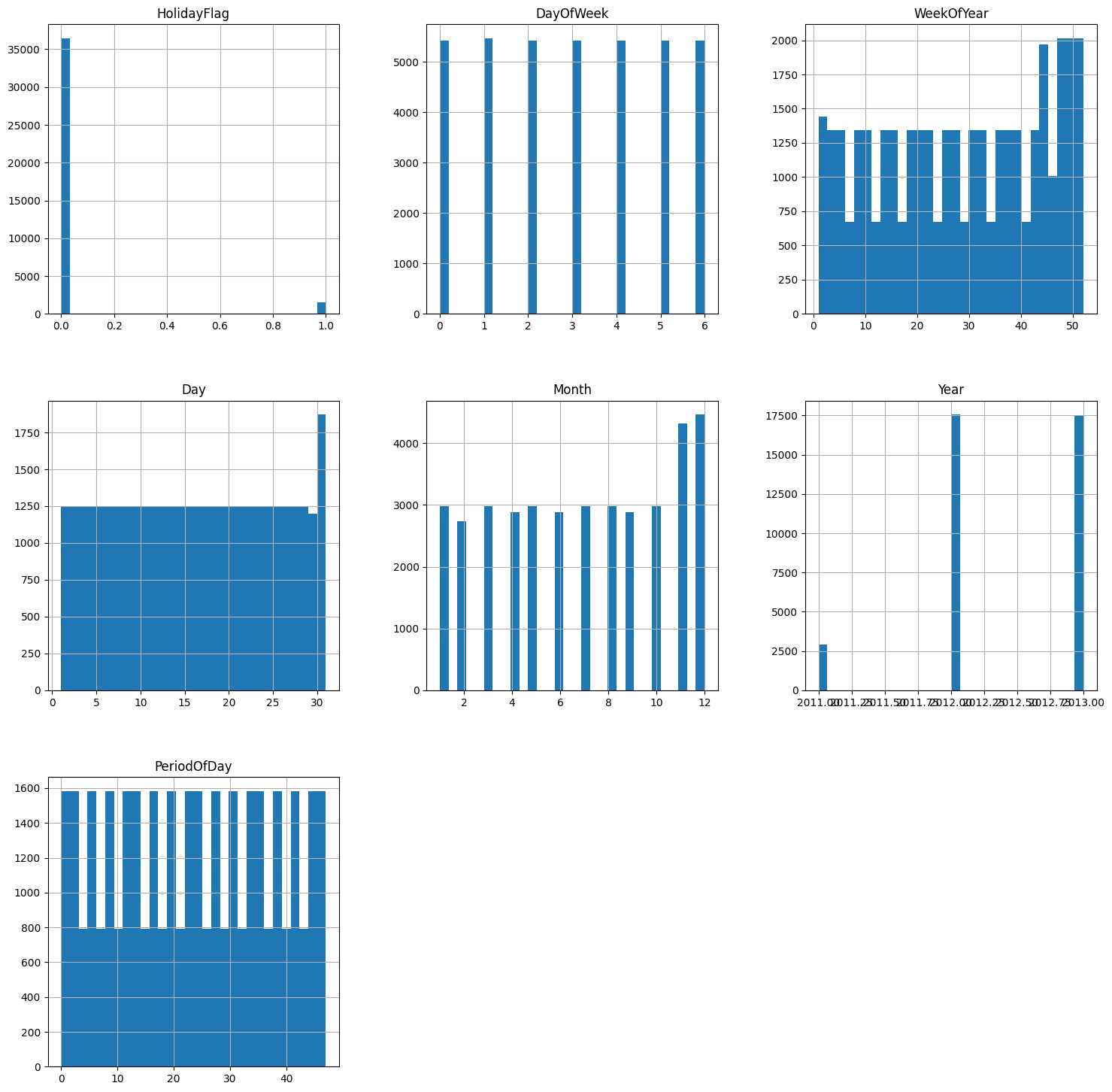
# Visualizing data distribution in detail

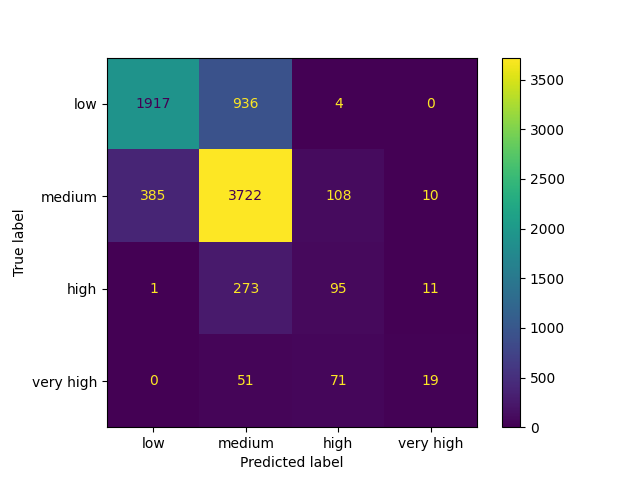
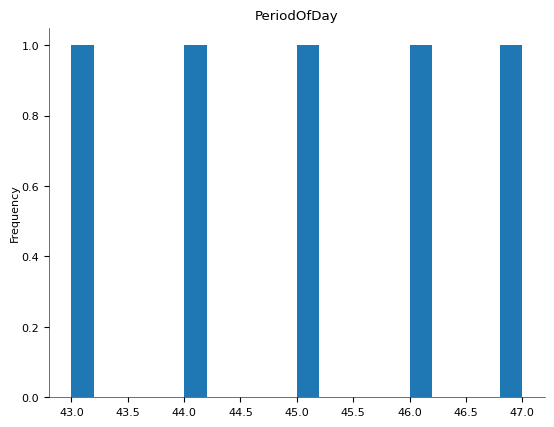
fig = plt.figure(figsize=(18,18))

ax = fig.gca()

df.hist(ax=ax, bins=30)

plt.show()





# Mount Google Drive

from google.colab import drive

drive.mount('/content/drive')

# Import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

# Configure settings

warnings.filterwarnings('ignore')

%matplotlib inline

# Read dataset

df = pd.read\_csv("/content/drive/My Drive/thing/Electricity.csv")

# Checking description (first 5 rows)

df.head()

# Checking description (last 5 rows)

df.tail()

# Drop non-numeric columns (like datetime)

df = df.select\_dtypes(include=[np.number])

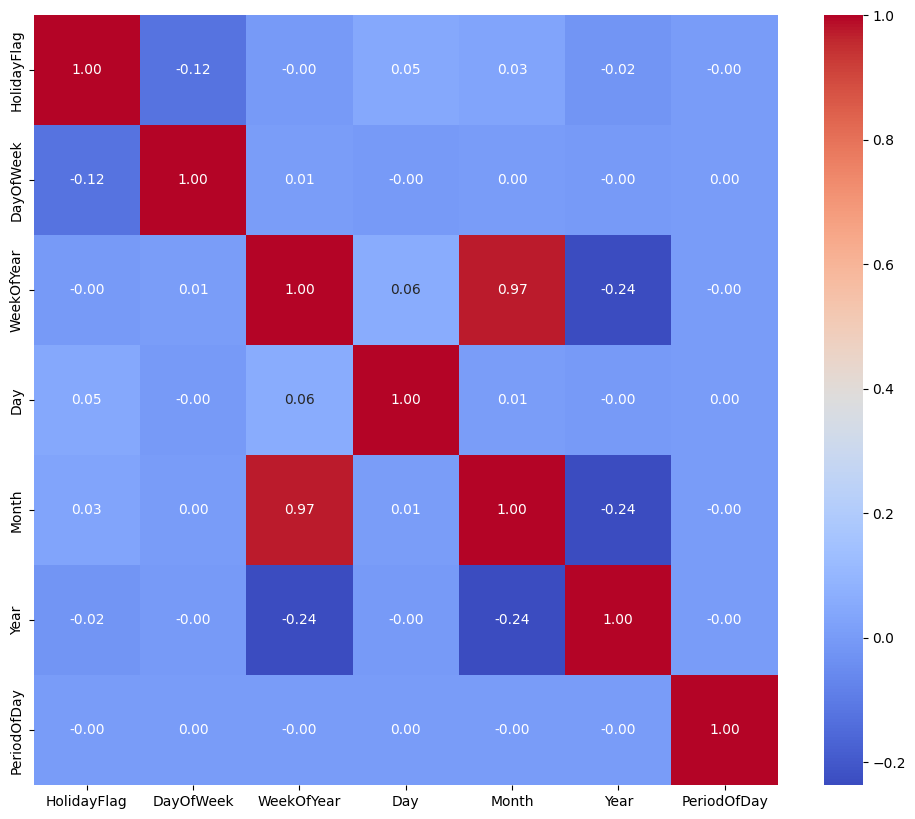
corr\_matrix = df.corr()

plt.figure(figsize=(12, 10))

sns.heatmap(corr\_matrix, annot=True, fmt=".2f", cmap="coolwarm", cbar=True)

# Display the heatmap

plt.show()



**Month and WeekOfYear:**

* Month and WeekOfYear have a very strong connection. Significant feature correlations may result in multicollinearity, which may introduce interference into the model and make it harder to interpret. In order to get around this, I eliminated the WeekOfYear feature in favor of the Month feature, which more accurately depicts time-related aspects of the data and is more closely tied to changes in the seasons.

**HolidayFlag, DayOfWeek, PeriodOfDay:**

* These features have very low correlations with other features, which means they are probably independent predictors.

**PDA**

Calculating accuracy is crucial for understanding how well a model performs, and SMAPE is a useful method for doing this. SMAPE measures the percentage difference between predicted and actual values, giving a clear picture of the model's accuracy. It's a good way to see how close your predictions are to the real values, helping to ensure that the model is reliable.

### 1. Outliers Removal:

The following columns are subjected to the outliers function: "ForecastWindProduction," "SystemLoadEA," "SMPEA," "ORKTemperature," "ORKWindspeed," "CO2Intensity," "ActualWindProduction," "SystemLoadEP2," and "SMPEP2." The Interquartile Range (IQR) approach is used to identify and eliminate outliers by removing values that exhibit a considerable deviation from the middle range of the data.

# Define models to compare

models = []

models.append(('DT', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC()))

models.append(('GBM', GradientBoostingClassifier()))

models.append(('RF', RandomForestClassifier()))

# Evaluate each model

results = []

names = []

kfold = KFold(n\_splits=10, random\_state=42, shuffle=True)

for name, model in models:

cv\_results = cross\_val\_score(model, x\_train, y\_train, cv=kfold, scoring='accuracy')

results.append(cv\_results)

names.append(name)

mean\_accuracy = cv\_results.mean()

std\_dev = cv\_results.std()

msg = f"{name}: Mean Accuracy = {mean\_accuracy:.6f}, Standard Deviation = {std\_dev:.6f}"

print(msg)

# Compare algorithm performance

fig = plt.figure()

fig.suptitle('Algorithm Comparison')

ax = fig.add\_subplot(111)

plt.boxplot(results)

ax.set\_xticklabels(names)

plt.show()

# Selecting best model

best\_model = RandomForestClassifier()

best\_model.fit(x\_train, y\_train)

y\_pred = best\_model.predict(x\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Best Model Accuracy Score on Test Set:", accuracy)

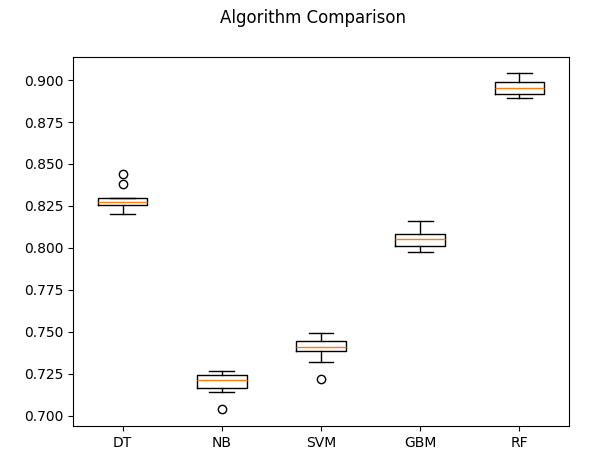
print(classification\_report(y\_test, y\_pred))

cm = confusion\_matrix(y\_test, y\_pred, labels=labels)

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=labels)

disp.plot()

plt.show()



**DT: 0.829174 (0.006578)**

**NB: 0.719641 (0.006485)**

**SVM: 0.739568 (0.007472)**

**GBM: 0.805432 (0.005250)**

**RF: 0.895597 (0.004806)**

precision recall f1-score support

high 0.87 0.48 0.62 380

low 0.91 0.91 0.91 2853

medium 0.89 0.94 0.91 4225

nan 0.00 0.00 0.00 4

very high 0.58 0.49 0.53 141

accuracy 0.90 7603

macro avg 0.65 0.56 0.60 7603

weighted avg 0.89 0.90 0.89 7603

**Brief Overview of Prediction Test**

# Prediction report

print("\nPrediction Report")

for x in range(len(y\_pred)):

print(f"Predicted: {y\_pred[x]}, Actual: {y\_test.iloc[x]}, Data: {x\_test.iloc[x].to\_dict()}")

Predicted: medium, Actual: medium, Data: {'HolidayFlag': 1.0, 'DayOfWeek': 5.0, 'WeekOfYear': 13.0, 'Day': 30.0, 'Month': 3.0, 'Year': 2013.0, 'PeriodOfDay': 39.0, 'ForecastWindProduction': 628.8, 'SystemLoadEA': 5010.42, 'ORKTemperature': 3.0, 'ORKWindspeed': 22.2, 'CO2Intensity': 437.95, 'ActualWindProduction': 646.0, 'SystemLoadEP2': 4642.17, 'SMPEP2': 187.28}

Predicted: low, Actual: low, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 6.0, 'WeekOfYear': 20.0, 'Day': 20.0, 'Month': 5.0, 'Year': 2012.0, 'PeriodOfDay': 21.0, 'ForecastWindProduction': 32.21, 'SystemLoadEA': 3755.06, 'ORKTemperature': 12.0, 'ORKWindspeed': 3.7, 'CO2Intensity': 601.33, 'ActualWindProduction': 51.0, 'SystemLoadEP2': 3698.0, 'SMPEP2': 51.65}

Predicted: medium, Actual: medium, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 3.0, 'WeekOfYear': 45.0, 'Day': 10.0, 'Month': 11.0, 'Year': 2011.0, 'PeriodOfDay': 28.0, 'ForecastWindProduction': 666.0, 'SystemLoadEA': 4972.9, 'ORKTemperature': 12.0, 'ORKWindspeed': 29.6, 'CO2Intensity': 450.34, 'ActualWindProduction': 702.0, 'SystemLoadEP2': 4476.89, 'SMPEP2': 46.75}

Predicted: medium, Actual: medium, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 2.0, 'WeekOfYear': 15.0, 'Day': 10.0, 'Month': 4.0, 'Year': 2013.0, 'PeriodOfDay': 3.0, 'ForecastWindProduction': 147.03, 'SystemLoadEA': 3280.84, 'ORKTemperature': 5.0, 'ORKWindspeed': 13.0, 'CO2Intensity': 608.7, 'ActualWindProduction': 295.0, 'SystemLoadEP2': 3240.94, 'SMPEP2': 52.34}

Predicted: medium, Actual: medium, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 4.0, 'WeekOfYear': 52.0, 'Day': 28.0, 'Month': 12.0, 'Year': 2012.0, 'PeriodOfDay': 30.0, 'ForecastWindProduction': 1544.7, 'SystemLoadEA': 4419.65, 'ORKTemperature': 11.0, 'ORKWindspeed': 40.7, 'CO2Intensity': 291.54, 'ActualWindProduction': 1407.0, 'SystemLoadEP2': 3997.38, 'SMPEP2': 49.02}

Predicted: medium, Actual: medium, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 5.0, 'WeekOfYear': 43.0, 'Day': 26.0, 'Month': 10.0, 'Year': 2013.0, 'PeriodOfDay': 26.0, 'ForecastWindProduction': 1239.0, 'SystemLoadEA': 4190.2, 'ORKTemperature': 13.0, 'ORKWindspeed': 44.4, 'CO2Intensity': 321.85, 'ActualWindProduction': 1102.0, 'SystemLoadEP2': 3852.89, 'SMPEP2': 33.14}

Predicted: medium, Actual: medium, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 5.0, 'WeekOfYear': 20.0, 'Day': 19.0, 'Month': 5.0, 'Year': 2012.0, 'PeriodOfDay': 27.0, 'ForecastWindProduction': 224.03, 'SystemLoadEA': 4220.28, 'ORKTemperature': 9.0, 'ORKWindspeed': 22.2, 'CO2Intensity': 524.96, 'ActualWindProduction': 264.0, 'SystemLoadEP2': 4047.52, 'SMPEP2': 79.9}

Predicted: high, Actual: high, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 4.0, 'WeekOfYear': 17.0, 'Day': 27.0, 'Month': 4.0, 'Year': 2012.0, 'PeriodOfDay': 40.0, 'ForecastWindProduction': 504.93, 'SystemLoadEA': 4238.41, 'ORKTemperature': 8.0, 'ORKWindspeed': 11.1, 'CO2Intensity': 563.61, 'ActualWindProduction': 435.0, 'SystemLoadEP2': 3942.49, 'SMPEP2': 73.01}

Predicted: low, Actual: low, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 1.0, 'WeekOfYear': 21.0, 'Day': 22.0, 'Month': 5.0, 'Year': 2012.0, 'PeriodOfDay': 8.0, 'ForecastWindProduction': 353.92, 'SystemLoadEA': 2737.48, 'ORKTemperature': 11.0, 'ORKWindspeed': 16.7, 'CO2Intensity': 505.44, 'ActualWindProduction': 611.0, 'SystemLoadEP2': 2468.41, 'SMPEP2': 38.0}

Predicted: medium, Actual: medium, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 0.0, 'WeekOfYear': 51.0, 'Day': 19.0, 'Month': 12.0, 'Year': 2011.0, 'PeriodOfDay': 42.0, 'ForecastWindProduction': 834.35, 'SystemLoadEA': 5586.23, 'ORKTemperature': 8.0, 'ORKWindspeed': 22.2, 'CO2Intensity': 461.68, 'ActualWindProduction': 896.0, 'SystemLoadEP2': 4996.19, 'SMPEP2': 54.48}

Predicted: low, Actual: medium, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 1.0, 'WeekOfYear': 6.0, 'Day': 7.0, 'Month': 2.0, 'Year': 2012.0, 'PeriodOfDay': 47.0, 'ForecastWindProduction': 803.69, 'SystemLoadEA': 4308.16, 'ORKTemperature': 7.0, 'ORKWindspeed': 16.7, 'CO2Intensity': 406.6, 'ActualWindProduction': 1068.0, 'SystemLoadEP2': 3783.22, 'SMPEP2': 45.91}

Predicted: medium, Actual: medium, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 5.0, 'WeekOfYear': 40.0, 'Day': 5.0, 'Month': 10.0, 'Year': 2013.0, 'PeriodOfDay': 15.0, 'ForecastWindProduction': 298.3, 'SystemLoadEA': 3064.15, 'ORKTemperature': 13.0, 'ORKWindspeed': 11.1, 'CO2Intensity': 644.39, 'ActualWindProduction': 199.0, 'SystemLoadEP2': 3031.64, 'SMPEP2': 65.72}

Predicted: medium, Actual: medium, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 2.0, 'WeekOfYear': 16.0, 'Day': 17.0, 'Month': 4.0, 'Year': 2013.0, 'PeriodOfDay': 5.0, 'ForecastWindProduction': 829.93, 'SystemLoadEA': 3208.62, 'ORKTemperature': 9.0, 'ORKWindspeed': 33.3, 'CO2Intensity': 464.93, 'ActualWindProduction': 841.0, 'SystemLoadEP2': 2814.87, 'SMPEP2': 47.46}

Predicted: low, Actual: low, Data: {'HolidayFlag': 1.0, 'DayOfWeek': 0.0, 'WeekOfYear': 44.0, 'Day': 29.0, 'Month': 10.0, 'Year': 2012.0, 'PeriodOfDay': 9.0, 'ForecastWindProduction': 174.51, 'SystemLoadEA': 2755.12, 'ORKTemperature': 6.0, 'ORKWindspeed': 18.5, 'CO2Intensity': 658.42, 'ActualWindProduction': 188.0, 'SystemLoadEP2': 2599.98, 'SMPEP2': 52.94}

Predicted: medium, Actual: medium, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 6.0, 'WeekOfYear': 7.0, 'Day': 19.0, 'Month': 2.0, 'Year': 2012.0, 'PeriodOfDay': 41.0, 'ForecastWindProduction': 214.3, 'SystemLoadEA': 4769.83, 'ORKTemperature': 4.0, 'ORKWindspeed': 13.0, 'CO2Intensity': 532.88, 'ActualWindProduction': 312.0, 'SystemLoadEP2': 4439.27, 'SMPEP2': 52.76}

Predicted: low, Actual: low, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 6.0, 'WeekOfYear': 49.0, 'Day': 8.0, 'Month': 12.0, 'Year': 2013.0, 'PeriodOfDay': 11.0, 'ForecastWindProduction': 990.87, 'SystemLoadEA': 2897.16, 'ORKTemperature': 6.0, 'ORKWindspeed': 14.8, 'CO2Intensity': 368.09, 'ActualWindProduction': 921.0, 'SystemLoadEP2': 2428.16, 'SMPEP2': 33.97}

Predicted: medium, Actual: medium, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 6.0, 'WeekOfYear': 5.0, 'Day': 5.0, 'Month': 2.0, 'Year': 2012.0, 'PeriodOfDay': 28.0, 'ForecastWindProduction': 93.11, 'SystemLoadEA': 4534.34, 'ORKTemperature': 6.0, 'ORKWindspeed': 9.3, 'CO2Intensity': 574.79, 'ActualWindProduction': 121.0, 'SystemLoadEP2': 4492.65, 'SMPEP2': 100.19}

Predicted: medium, Actual: medium, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 0.0, 'WeekOfYear': 41.0, 'Day': 8.0, 'Month': 10.0, 'Year': 2012.0, 'PeriodOfDay': 21.0, 'ForecastWindProduction': 336.3, 'SystemLoadEA': 4678.54, 'ORKTemperature': 11.0, 'ORKWindspeed': 25.9, 'CO2Intensity': 486.39, 'ActualWindProduction': 326.0, 'SystemLoadEP2': 4466.64, 'SMPEP2': 78.6}

Predicted: low, Actual: low, Data: {'HolidayFlag': 0.0, 'DayOfWeek': 6.0, 'WeekOfYear': 46.0, 'Day': 18.0, 'Month': 11.0, 'Year': 2012.0, 'PeriodOfDay': 5.0, 'ForecastWindProduction': 187.77, 'SystemLoadEA': 2983.78, 'ORKTemperature': 3.0, 'ORKWindspeed': 9.3, 'CO2Intensity': 689.54, 'ActualWindProduction': 217.0, 'SystemLoadEP2': 2912.35, 'SMPEP2': 39.41}

**References**

* Bobbitt, Z. (2021a, April 15). *How to calculate SMAPE in Python*. Statology. https://www.statology.org/smape-python/#:~:text=There%20is%20no%20built-in%20Python%20function%20to%20calculate,%2A%20np.sum%282%20%2A%20np.abs%28f-a%29%20%2F%20%28np.abs%28a%29%20%2B%20np.abs%28f%29%29%2A100%29
* Bobbitt, Z. (2021b, June 23). *How to calculate the interquartile range in Python*. Statology. https://www.statology.org/interquartile-range-python/
* Chakure, A. (2024, August 20). *How to preprocess data in Python*. Built In. https://builtin.com/machine-learning/how-to-preprocess-data-python
* *How do you calculate interquartile range (IQR) correctly using Python?* (n.d.). Stack Overflow. https://stackoverflow.com/questions/53337391/how-do-you-calculate-interquartile-range-iqr-correctly-using-python
* K, G. M. (2021, December 15). Machine Learning Basics: Random forest regression - towards Data science. *Medium*. https://towardsdatascience.com/machine-learning-basics-random-forest-regression-be3e1e3bb91a
* Koehrsen, W. (2019, December 10). Hyperparameter tuning the random forest in Python - towards data science. *Medium*. https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74