ELECTRICITY PRICES PREDICTION

Feature Engineering:

Choose the most relevant features for our prediction task is called as Feature Engineering.

```
In [12]: import pandas as pd
         import numpy as np
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
         from sklearn.preprocessing import StandardScaler
```

Before Feature Engineering,We have to load the dataset

```
In [2]: data = pd.read_csv("Electricity_Prices_Prediction.csv")
```

Printing First Few rows,

```
In [3]: print(data.head())

         DateTime Holiday  HolidayFlag  DayOfWeek  WeekOfYear  Day  Month  \
0  1/11/2011 0:00    None            0          1          44    1     11
1  1/11/2011 0:30    None            0          1          44    1     11
2  1/11/2011 1:00    None            0          1          44    1     11
3  1/11/2011 1:30    None            0          1          44    1     11
4  1/11/2011 2:00    None            0          1          44    1     11

   Year  PeriodOfDay ForecastWindProduction SystemLoadEA  SMPEA  \
0  2011            0                 315.31      3388.77  49.26
1  2011            1                  321.8      3196.66  49.26
2  2011            2                 328.57      3060.71   49.1
3  2011            3                  335.6      2945.56  48.04
4  2011            4                  342.9      2849.34  33.75

  ORKTemperature ORKWindspeed CO2Intensity ActualWindProduction SystemLoadEP2  \
0              6          9.3       600.71                  356        3159.6
1              6         11.1       605.42                  317       2973.01
2              5         11.1       589.97                  311          2834
3              6          9.3       585.94                  313       2725.99
4              6         11.1       571.52                  346       2655.64

  SMPEP2
0  54.32
1  54.23
2  54.23
3  53.47
4  39.87
```

Let’s have a look at all the columns of this dataset:

In [4]:
```
print(data.info())
```

```
<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                 38014 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
```

We can see that so many features with numerical values are string values in the dataset and not integers or float values. So before moving further, we have to convert these string values to float values:

In [5]:
```
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')
```

Now let’s have a look at whether this dataset contains any null values or not:

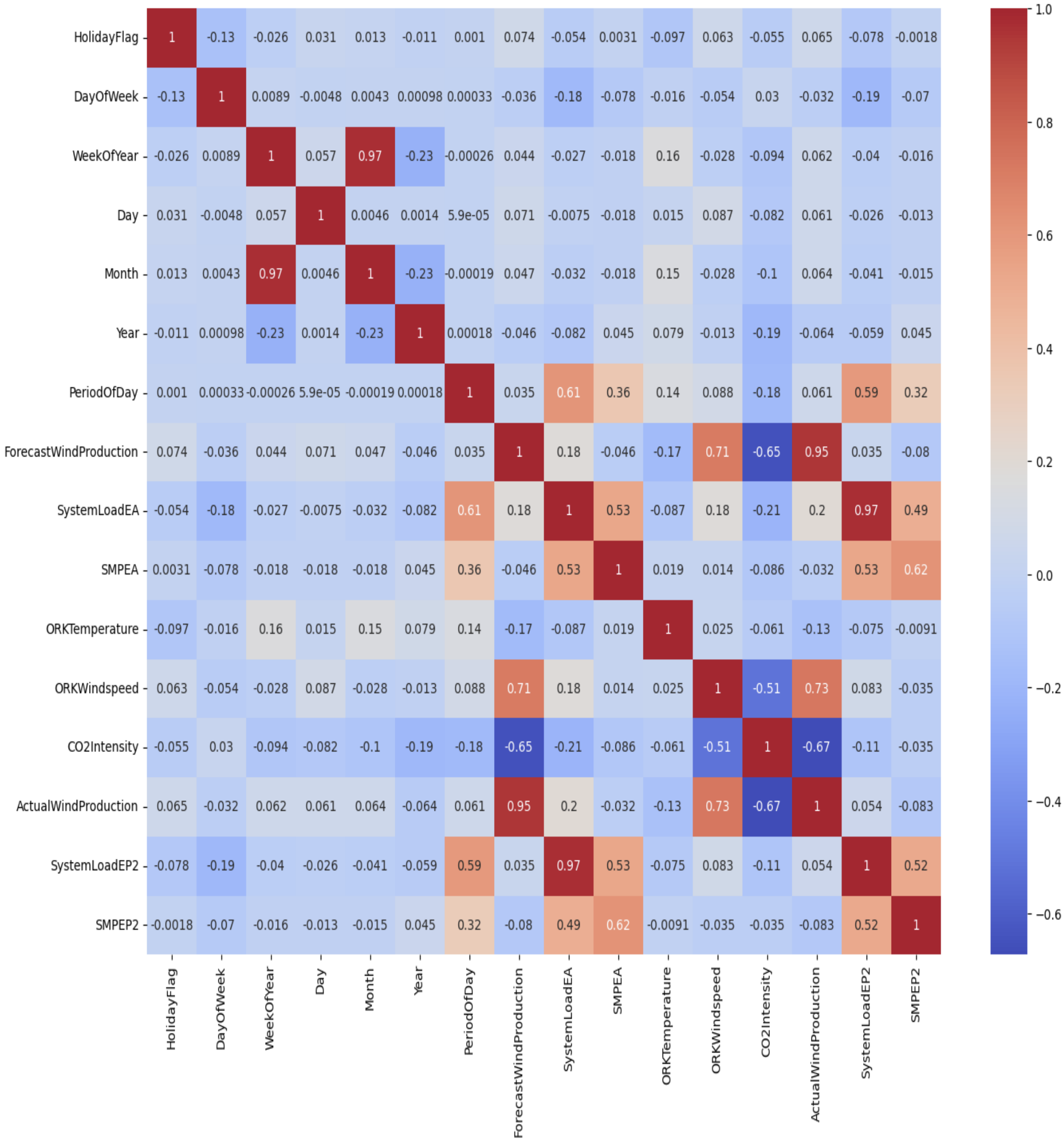
```
In [6]: data.isnull().sum()

Out[6]: DateTime                    0
        Holiday                     0
        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
```

So there are some columns with null values, We will drop all these rows containing null values from the dataset:

```
In [7]: data = data.dropna()
```

Now let’s have a look at the correlation between all the columns in the dataset:

```
In [8]: import seaborn as sns
        import matplotlib.pyplot as plt
        correlations = data.corr(method='pearson')
        plt.figure(figsize=(16, 12))
        sns.heatmap(correlations, cmap="coolwarm", annot=True)
        plt.show()
```



Model Training:

Now let's move to the task of training an electricity price prediction model. Here we will first add all the important features to x and the target column to y, and then we will split the data into training and test sets:

```
In [9]: x = data[["Day", "Month", "ForecastWindProduction", "SystemLoadEA",
                  "SMPEA", "ORKTemperature", "ORKWindspeed", "CO2Intensity",
                  "ActualWindProduction", "SystemLoadEP2"]]
        y = data["SMPEP2"]
        from sklearn.model_selection import train_test_split
        xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                         test_size=0.2,
                                                         random_state=42)
```

As this is the problem of regression, so here I will choose the Random Forest regression algorithm to train the electricity price prediction model:

```
In [10]: from sklearn.ensemble import RandomForestRegressor
         model = RandomForestRegressor()
         model.fit(xtrain, ytrain)

Out[10]: RandomForestRegressor()
```

```
In [13]: features = np.array([[10, 12, 54.10, 4241.05, 49.56, 9.0, 14.8, 491.32, 54.0, 4426.84]])
         model.predict(features)

Out[13]: array([68.3696])
```

Model Evaluation:

Evaluate the model's performance on the testing dataset using appropriate metrics for regression tasks. Common evaluation metrics include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2) for regression models.

In [14]:
```
ypred = model.predict(xtest)
```

In [15]:
```
mse = mean_squared_error(ytest, ypred)
mae = mean_absolute_error(ytest, ypred)
r2 = r2_score(ytest, ypred)
```

In [16]:
```
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 595.0806655157822
Mean Absolute Error: 9.351011025607004
R-squared: 0.54185050626134
```