Electricity price Prediction

Suppose that your business relies on computing services where the power consumed by your machines varies throughout the day. You do not know the actual cost of the electricity consumed by the machines throughout the day, but the organization has provided you with historical data of the price of the electricity consumed by the machines. Below is the information of the data we have for the task of forecasting electricity prices:

DateTime: Date and time of the record

Holiday: contains the name of the holiday if the day is a national holiday

HolidayFlag: contains 1 if it's a bank holiday otherwise 0

DayOfWeek: contains values between 0-6 where 0 is Monday

WeekOfYear: week of the year

Day: Day of the date

Month: Month of the date

Year: Year of the date

PeriodOfDay: half-hour period of the day

ForcastWindProduction: forecasted wind production

SystemLoadEA forecasted national load

SMPEA: forecasted price

ORKTemperature: actual temperature measured

ORKWindspeed: actual windspeed measured

CO2Intensity: actual C02 intensity for the electricity produced ActualWindProduction: actual wind energy production

SystemLoadEP2: actual national system load

SMPEP2: the actual price of the electricity consumed (labels or values to be predicted)

So your task here is to use this data to train a machine learning model to predict the price of electricity consumed by the machines. In the section below, I will take you through the task of electricity price prediction with machine learning using Python.

In [1]:
```python
# Importing the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
data = pd.read_csv("C:/Users/Simeon/Desktop/electricity.csv")
```

```
C:\Users\Simeon\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3444: DtypeWarning: Columns (9,10,11,1
4,15,16,17) have mixed types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

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

Out[3]:

	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA
0	01/11/2011 00:00	None	0	1	44	1	11	2011	0	315.31	3388.77	49.26
1	01/11/2011 00:30	None	0	1	44	1	11	2011	1	321.80	3196.66	49.26
2	01/11/2011 01:00	None	0	1	44	1	11	2011	2	328.57	3060.71	49.10
3	01/11/2011 01:30	None	0	1	44	1	11	2011	3	335.60	2945.56	48.04
4	01/11/2011 02:00	None	0	1	44	1	11	2011	4	342.90	2849.34	33.75

In [4]:
```
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
```

I 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:

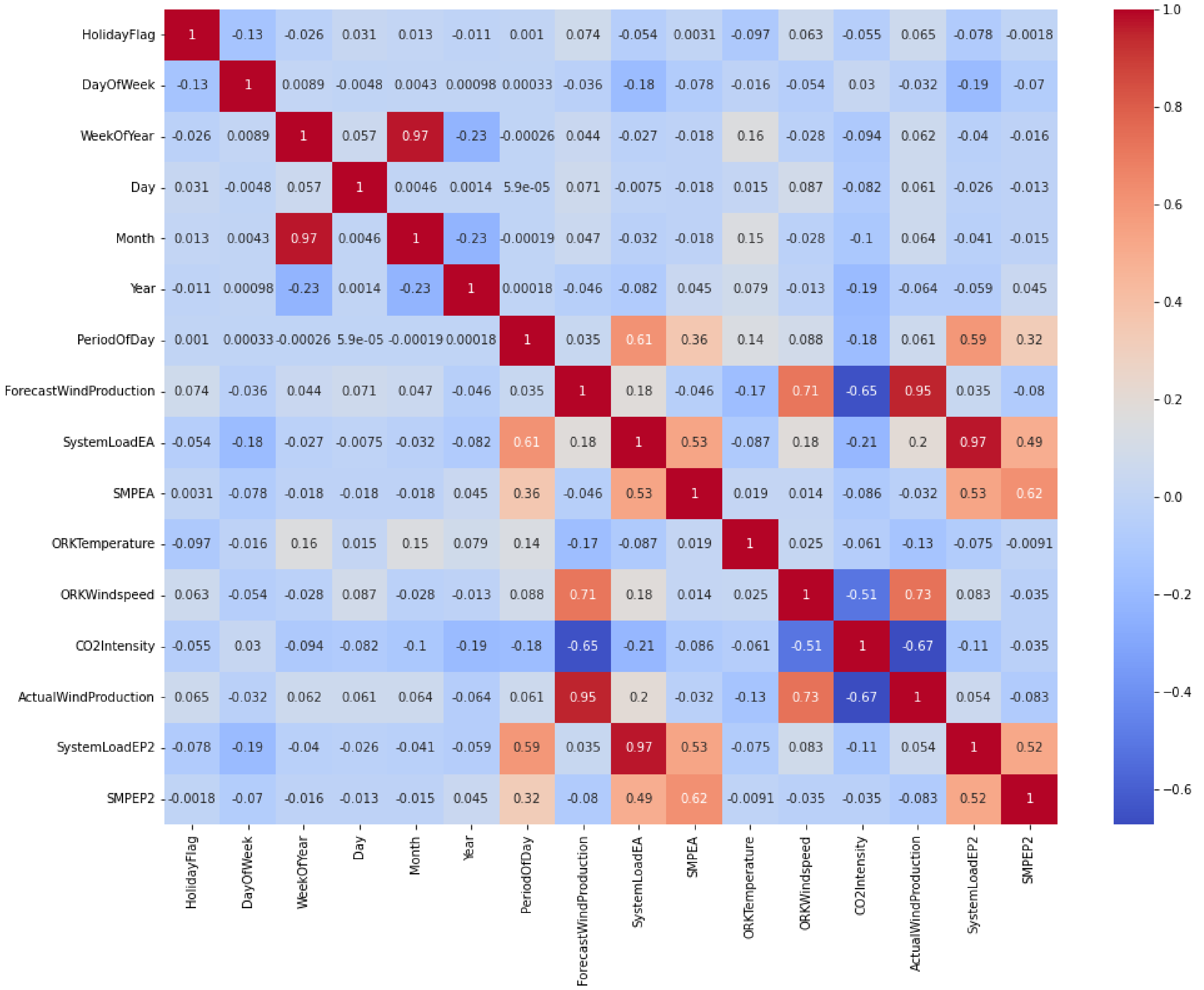
In [5]:
```
data["ForecastWindProduction"] = pd.to_numeric(data["ForecastWindProduction"], errors= 'coerce')
data["SystemLoadEA"] = pd.to_numeric(data["SystemLoadEA"], errors= 'coerce')
data["SMPEA"] = pd.to_numeric(data["SMPEA"], errors= 'coerce')
data["ORKTemperature"] = pd.to_numeric(data["ORKTemperature"], 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')
```

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
```

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

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()
```



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)
```

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

Out[10]:
```
RandomForestRegressor()
```

Now let’s input all the values of the necessary features that we used to train the model and have a look at the price of the electricity predicted by the model:

In [12]:
```
#features = [["Day", "Month", "ForecastWindProduction", "SystemLoadEA", "SMPEA", "ORKTemperature", "ORKWindspeed", "C
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[12]:
```
array([65.861])
```