Electricity Price Prediction using RandomForestRegressor

In [2]:
```
import pandas as pd
import numpy as np
data = pd.read_csv(r"C:\Users\Admin\Downloads\electricity.csv")
data.head()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_11804\1934404862.py:3: DtypeWarning: Colu
mns (9,10,11,14,15,16,17) have mixed types. Specify dtype option on import or set low
_memory=False.
  data = pd.read_csv(r"C:\Users\Admin\Downloads\electricity.csv")
```

Out[2]:

	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	For
0	01/11/2011 00:00	NaN	0	1	44	1	11	2011	0	
1	01/11/2011 00:30	NaN	0	1	44	1	11	2011	1	
2	01/11/2011 01:00	NaN	0	1	44	1	11	2011	2	
3	01/11/2011 01:30	NaN	0	1	44	1	11	2011	3	
4	01/11/2011 02:00	NaN	0	1	44	1	11	2011	4	

In [3]:
```
# show all the columns of this dataset
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                  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
```

In [4]:
```
# to check the null values
data.isnull().sum()
```

Out[4]:
```
DateTime                    0
Holiday                 36478
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
```

In [5]:
```
# fill the missing values
data['Holiday'].fillna(data['Holiday'].mode()[0],inplace=True)
```

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    0
SystemLoadEA              0
SMPEA                     0
ORKTemperature            0
ORKWindspeed              0
CO2Intensity              0
ActualWindProduction      0
SystemLoadEP2             0
SMPEP2                    0
dtype: int64
```

In [7]:
```
# to drop all these rows containing null values from the dataset
# data = data.dropna()
```

In [8]:
```
# # converting categories it number --- Label Encoder
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
```

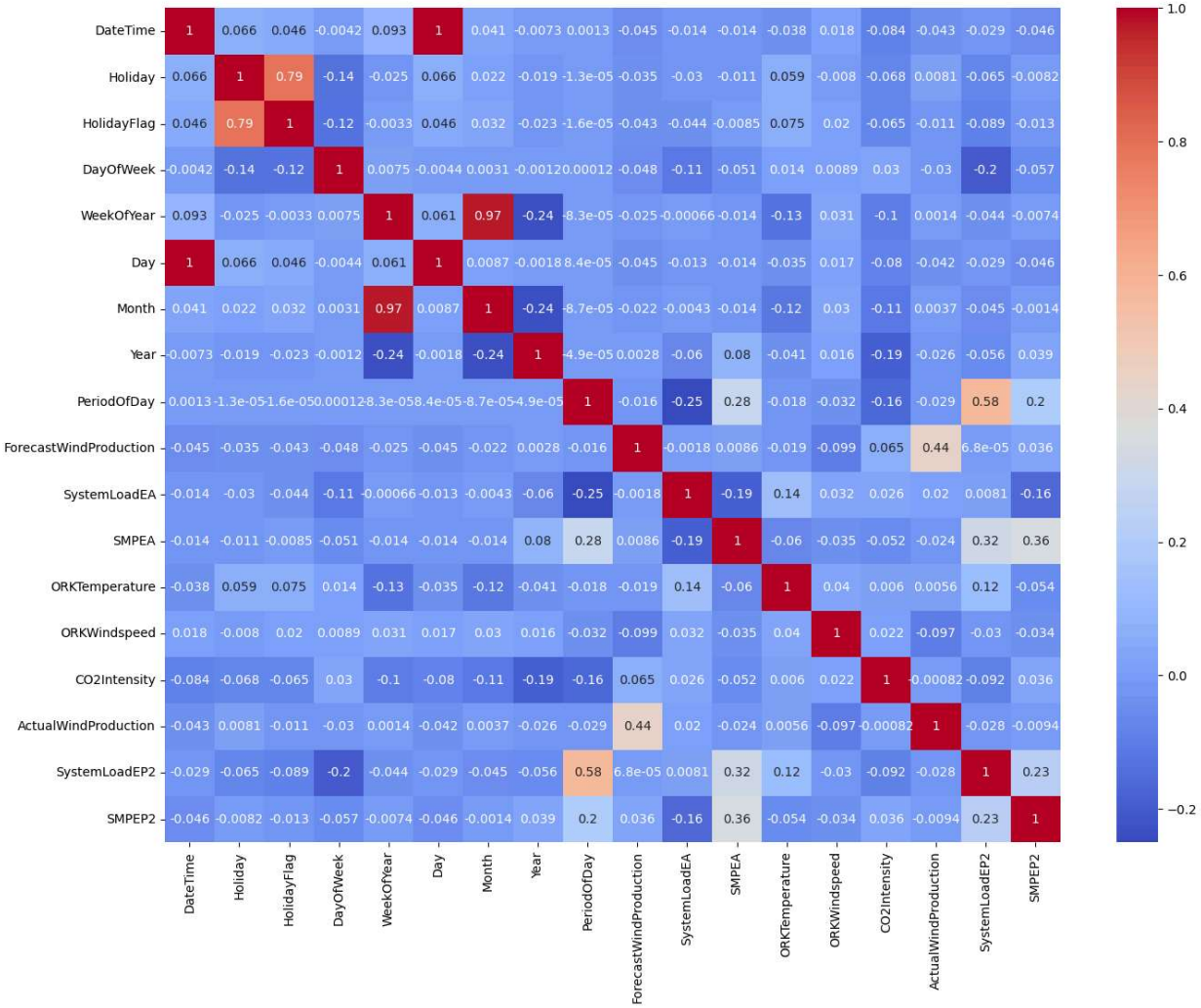
In [10]:
```
data['DateTime'] = label_encoder.fit_transform(data['DateTime'])
data['Holiday'] = label_encoder.fit_transform(data['Holiday'])

# Convert all values in 'ForecastWindProduction' column to strings
data['ForecastWindProduction'] = data['ForecastWindProduction'].astype(str)

data['ForecastWindProduction'] = label_encoder.fit_transform(data['ForecastWindProduct
```

In [13]:
```
data['SystemLoadEA'] = data['SystemLoadEA'].astype(str)
data['SystemLoadEA'] = label_encoder.fit_transform(data['SystemLoadEA'])

data['SMPEA'] = data['SMPEA'].astype(str)
data['SMPEA'] = label_encoder.fit_transform(data['SMPEA'])

data['ORKTemperature'] = data['ORKTemperature'].astype(str)
data['ORKTemperature'] = label_encoder.fit_transform(data['ORKTemperature'])
```

In [14]:
```
data['ORKWindspeed'] = data['ORKWindspeed'].astype(str)
data['ORKWindspeed'] = label_encoder.fit_transform(data['ORKWindspeed'])

data['CO2Intensity'] = data['CO2Intensity'].astype(str)
data['CO2Intensity'] = label_encoder.fit_transform(data['CO2Intensity'])

data['ActualWindProduction'] = data['ActualWindProduction'].astype(str)
data['ActualWindProduction'] = label_encoder.fit_transform(data['ActualWindProduction'
```

In [15]:
```
data['SystemLoadEP2'] = data['SystemLoadEP2'].astype(str)
data['SystemLoadEP2'] = label_encoder.fit_transform(data['SystemLoadEP2'])

data['SMPEP2'] = data['SMPEP2'].astype(str)
data['SMPEP2'] = label_encoder.fit_transform(data['SMPEP2'])
```

In [16]:
```
# to know the correlation between all the columns in the dataset
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 [17]:
```
# Assign the values of X and y
x = data[["Day", "Month", "ForecastWindProduction", "SystemLoadEA",
          "SMPEA", "ORKTemperature", "ORKWindspeed", "CO2Intensity",
          "ActualWindProduction", "SystemLoadEP2"]]
y = data["SMPEP2"]
```

In [18]:
```
# split the data set to test set and train set
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 [19]:
```
# model selection
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
model.fit(xtrain, ytrain)
```

Out[19]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [22]:
```
y_pred = model.predict(xtest)
y_pred
```

Out[22]:
```
array([5221.63, 3960.59, 4406.46, ..., 1720.7 , 4768.07, 3495.71])
```

In [25]:
```
import pandas as pd

# Assuming ytest and y_pred are your lists or arrays of actual and predicted values
df = pd.DataFrame({'Actual': ytest, 'Predicted': y_pred})
df
```

Out[25]:

	Actual	Predicted
35833	5109	5221.63
198	3752	3960.59
36547	4911	4406.46
26373	3728	3715.99
21156	5050	4394.12
...	...	...
13927	3708	3941.14
16926	5883	5729.91
24520	471	1720.70
9059	7420	4768.07
9786	3541	3495.71

7603 rows × 2 columns

In [35]:
```
# Calculate evaluation metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
mae = mean_absolute_error(ytest, y_pred)
mse = mean_squared_error(ytest, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(ytest, y_pred)

print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Absolute Error: 752.619826384322
Mean Squared Error: 1417762.0598960149
Root Mean Squared Error: 1190.6981397046084
R-squared: 0.5660115459731367
```

In [ ]: