**2 Phase 4: Development Part**

**In this part you will continue building your project. Continue building the electricity prices prediction model by: Feature engineering Model training Evaluation**.

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

Importing libraries we will need in this prediction model

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

from sklearn.ensemble import RandomForestRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.linear\_model import LinearRegression

from sklearn.neighbors import KNeighborsRegressor

df=pd.read\_csv("/kaggle/input/electrity-prices/electricity\_prices.csv", low\_memory=False)

df.head()

|  | **DateTime** | **Holiday** | **HolidayFlag** | **DayOfWeek** | **WeekOfYear** | **Day** | **Month** | **Year** | **PeriodOfDay** | **ForecastWindProduction** | **SystemLoadEA** | **SMPEA** | **ORKTemperature** | **ORKWindspeed** | **CO2Intensity** | **ActualWindProduction** | **SystemLoadEP2** | **SMPEP2** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 01/11/2011 00:00 | None | 0 | 1 | 44 | 1 | 11 | 2011 | 0 | 315.31 | 3388.77 | 49.26 | 6.00 | 9.30 | 600.71 | 356.00 | 3159.60 | 54.32 |
| **1** | 01/11/2011 00:30 | None | 0 | 1 | 44 | 1 | 11 | 2011 | 1 | 321.80 | 3196.66 | 49.26 | 6.00 | 11.10 | 605.42 | 317.00 | 2973.01 | 54.23 |
| **2** | 01/11/2011 01:00 | None | 0 | 1 | 44 | 1 | 11 | 2011 | 2 | 328.57 | 3060.71 | 49.10 | 5.00 | 11.10 | 589.97 | 311.00 | 2834.00 | 54.23 |
| **3** | 01/11/2011 01:30 | None | 0 | 1 | 44 | 1 | 11 | 2011 | 3 | 335.60 | 2945.56 | 48.04 | 6.00 | 9.30 | 585.94 | 313.00 | 2725.99 | 53.47 |
| **4** | 01/11/2011 02:00 | None | 0 | 1 | 44 | 1 | 11 | 2011 | 4 | 342.90 | 2849.34 | 33.75 | 6.00 | 11.10 | 571.52 | 346.00 | 2655.64 | 39.87 |

df.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

data=df[['ForecastWindProduction',

'SystemLoadEA', 'SMPEA', 'ORKTemperature', 'ORKWindspeed',

'CO2Intensity', 'ActualWindProduction', 'SystemLoadEP2', 'SMPEP2']]

data.isin(['?']).any()

ForecastWindProduction True

SystemLoadEA True

SMPEA True

ORKTemperature True

ORKWindspeed True

CO2Intensity True

ActualWindProduction True

SystemLoadEP2 True

SMPEP2 True

dtype: bool

for col in data.columns:

data.drop(data.index[data[col] == '?'], inplace=True)

errors=errors,

data=data.apply(pd.to\_numeric)

data=data.reset\_index()

data.drop('index', axis=1, inplace=True)

data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 37682 entries, 0 to 37681

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ForecastWindProduction 37682 non-null float64

1 SystemLoadEA 37682 non-null float64

2 SMPEA 37682 non-null float64

3 ORKTemperature 37682 non-null float64

4 ORKWindspeed 37682 non-null float64

5 CO2Intensity 37682 non-null float64

6 ActualWindProduction 37682 non-null float64

7 SystemLoadEP2 37682 non-null float64

8 SMPEP2 37682 non-null float64

dtypes: float64(9)

memory usage: 2.6 MB

data.corrwith(data['SMPEP2']).abs().sort\_values(ascending=False)

SMPEP2 1.000000

SMPEA 0.618158

SystemLoadEP2 0.517081

SystemLoadEA 0.491096

ActualWindProduction 0.083434

ForecastWindProduction 0.079639

ORKWindspeed 0.035436

CO2Intensity 0.035055

ORKTemperature 0.009087

dtype: float64

X=data.drop('SMPEP2', axis=1)

y=data['SMPEP2'

x\_train, x\_test, y\_train, y\_test=train\_test\_split(X,y, test\_size=0.2, random\_state=42)

LinearRegression

linear\_model=LinearRegression()

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

linear\_predict=linear\_model.predict(x\_test)

np.sqrt(mean\_squared\_error(y\_test, linear\_predict))

27.862965246485324

RandomForestRegressor

forest\_model=RandomForestRegressor()

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

forest\_predict=forest\_model.predict(x\_test)

print(np.sqrt(mean\_squared\_error(y\_test, forest\_predict)))

25.07740121650378

DecisionTreeRegressor

tree\_model=DecisionTreeRegressor(max\_depth=50)

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

tree\_predict=tree\_model.predict(x\_test)

print(np.sqrt(mean\_squared\_error(y\_test, tree\_predict)))

34.10426264709253

KNeighborsRegressor

knn\_model=KNeighborsRegressor()

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

knn\_predict=knn\_model.predict(x\_test)

print(np.sqrt(mean\_squared\_error(y\_test, knn\_predict)))

28.533256274003907

Let's see how good the model is working

some\_data=x\_test.iloc[50:60]

some\_data\_label=y\_test.iloc[50:60]

some\_predict=forest\_model.predict(some\_data)

pd.DataFrame({'Predict':some\_predict,'Label':some\_data\_label})

|  | **Predict** | **Label** |
| --- | --- | --- |
| **4093** | 149.6479 | 188.32 |
| **22310** | 36.0076 | 33.46 |
| **8034** | 59.2229 | 62.01 |
| **35027** | 75.8247 | 49.69 |
| **23685** | 73.0210 | 69.25 |
| **268** | 57.1129 | 56.21 |
| **35261** | 46.4761 | 46.64 |
| **11905** | 71.9873 | 78.52 |
| **30903** | 75.5883 | 82.36 |
| **608** | 110.6629 | 415.99 |