import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from warnings import filterwarnings

filterwarnings('ignore')

energy\_dataset = pd.read\_csv("energy\_dataset.csv")

weather\_features = pd.read\_csv("weather\_features.csv")

energy\_dataset.head(10)

time generation biomass generation fossil brown coal/lignite generation fossil coal-derived gas generation fossil gas generation fossil hard coal generation fossil oil generation fossil oil shale generation fossil peat generation geothermal ... generation waste generation wind offshore generation wind onshore forecast solar day ahead forecast wind offshore eday ahead forecast wind onshore day ahead total load forecast total load actual price day ahead price actual

0 2015-01-01 00:00:00+01:00 447.0 329.0 0.0 4844.0 4821.0 162.0 0.0 0.0 0.0 ... 196.0 0.0 6378.0 17.0 NaN 6436.0 26118.0 25385.0 50.10 65.41

1 2015-01-01 01:00:00+01:00 449.0 328.0 0.0 5196.0 4755.0 158.0 0.0 0.0 0.0 ... 195.0 0.0 5890.0 16.0 NaN 5856.0 24934.0 24382.0 48.10 64.92

2 2015-01-01 02:00:00+01:00 448.0 323.0 0.0 4857.0 4581.0 157.0 0.0 0.0 0.0 ... 196.0 0.0 5461.0 8.0 NaN 5454.0 23515.0 22734.0 47.33 64.48

3 2015-01-01 03:00:00+01:00 438.0 254.0 0.0 4314.0 4131.0 160.0 0.0 0.0 0.0 ... 191.0 0.0 5238.0 2.0 NaN 5151.0 22642.0 21286.0 42.27 59.32

4 2015-01-01 04:00:00+01:00 428.0 187.0 0.0 4130.0 3840.0 156.0 0.0 0.0 0.0 ... 189.0 0.0 4935.0 9.0 NaN 4861.0 21785.0 20264.0 38.41 56.04

5 2015-01-01 05:00:00+01:00 410.0 178.0 0.0 4038.0 3590.0 156.0 0.0 0.0 0.0 ... 188.0 0.0 4618.0 4.0 NaN 4617.0 21441.0 19905.0 35.72 53.63

6 2015-01-01 06:00:00+01:00 401.0 172.0 0.0 4040.0 3368.0 158.0 0.0 0.0 0.0 ... 186.0 0.0 4397.0 3.0 NaN 4276.0 21285.0 20010.0 35.13 51.73

7 2015-01-01 07:00:00+01:00 408.0 172.0 0.0 4030.0 3208.0 160.0 0.0 0.0 0.0 ... 189.0 0.0 3992.0 12.0 NaN 3994.0 21545.0 20377.0 36.22 51.43

8 2015-01-01 08:00:00+01:00 413.0 177.0 0.0 4052.0 3335.0 161.0 0.0 0.0 0.0 ... 198.0 0.0 3629.0 39.0 NaN 3602.0 21443.0 20094.0 32.40 48.98

9 2015-01-01 09:00:00+01:00 419.0 177.0 0.0 4137.0 3437.0 163.0 0.0 0.0 0.0 ... 198.0 0.0 3073.0 784.0 NaN 3212.0 21560.0 20637.0 36.60 54.20

10 rows × 29 columns

weather\_features.head(10)

dt\_iso city\_name temp temp\_min temp\_max pressure humidity wind\_speed wind\_deg rain\_1h rain\_3h snow\_3h clouds\_all weather\_id weather\_main weather\_description weather\_icon

0 2015-01-01 00:00:00+01:00 Valencia 270.475 270.475 270.475 1001 77 1 62 0.0 0.0 0.0 0 800 clear sky is clear 01n

1 2015-01-01 01:00:00+01:00 Valencia 270.475 270.475 270.475 1001 77 1 62 0.0 0.0 0.0 0 800 clear sky is clear 01n

2 2015-01-01 02:00:00+01:00 Valencia 269.686 269.686 269.686 1002 78 0 23 0.0 0.0 0.0 0 800 clear sky is clear 01n

3 2015-01-01 03:00:00+01:00 Valencia 269.686 269.686 269.686 1002 78 0 23 0.0 0.0 0.0 0 800 clear sky is clear 01n

4 2015-01-01 04:00:00+01:00 Valencia 269.686 269.686 269.686 1002 78 0 23 0.0 0.0 0.0 0 800 clear sky is clear 01n

5 2015-01-01 05:00:00+01:00 Valencia 270.292 270.292 270.292 1004 71 2 321 0.0 0.0 0.0 0 800 clear sky is clear 01n

6 2015-01-01 06:00:00+01:00 Valencia 270.292 270.292 270.292 1004 71 2 321 0.0 0.0 0.0 0 800 clear sky is clear 01n

7 2015-01-01 07:00:00+01:00 Valencia 270.292 270.292 270.292 1004 71 2 321 0.0 0.0 0.0 0 800 clear sky is clear 01n

8 2015-01-01 08:00:00+01:00 Valencia 274.601 274.601 274.601 1005 71 1 307 0.0 0.0 0.0 0 800 clear sky is clear 01d

9 2015-01-01 09:00:00+01:00 Valencia 274.601 274.601 274.601 1005 71 1 307 0.0 0.0 0.0 0 800 clear sky is clear 01d

df\_energy = energy\_dataset.copy()

df\_weather = weather\_features.copy()

print(f" Veri setinin boyut sayısı: {df\_energy.ndim}\n",

f"Veri setinin boyut bilgisi: {df\_energy.shape}\n",

f"Veri setindeki toplam eleman sayısı: {df\_energy.size}\n")

Veri setinin boyut sayısı: 2

Veri setinin boyut bilgisi: (35064, 29)

Veri setindeki toplam eleman sayısı: 1016856

print(f" Veri setinin boyut sayısı: {df\_weather.ndim}\n",

f"Veri setinin boyut bilgisi: {df\_weather.shape}\n",

f"Veri setindeki toplam eleman sayısı: {df\_weather.size}\n")

Veri setinin boyut sayısı: 2

Veri setinin boyut bilgisi: (178396, 17)

Veri setindeki toplam eleman sayısı: 3032732

df\_energy.info()

RangeIndex: 35064 entries, 0 to 35063

Data columns (total 29 columns):

# Column Non-Null Count Dtype

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

0 time 35064 non-null object

1 generation biomass 35045 non-null float64

2 generation fossil brown coal/lignite 35046 non-null float64

3 generation fossil coal-derived gas 35046 non-null float64

4 generation fossil gas 35046 non-null float64

5 generation fossil hard coal 35046 non-null float64

6 generation fossil oil 35045 non-null float64

7 generation fossil oil shale 35046 non-null float64

8 generation fossil peat 35046 non-null float64

9 generation geothermal 35046 non-null float64

10 generation hydro pumped storage aggregated 0 non-null float64

11 generation hydro pumped storage consumption 35045 non-null float64

12 generation hydro run-of-river and poundage 35045 non-null float64

13 generation hydro water reservoir 35046 non-null float64

14 generation marine 35045 non-null float64

15 generation nuclear 35047 non-null float64

16 generation other 35046 non-null float64

17 generation other renewable 35046 non-null float64

18 generation solar 35046 non-null float64

19 generation waste 35045 non-null float64

20 generation wind offshore 35046 non-null float64

21 generation wind onshore 35046 non-null float64

22 forecast solar day ahead 35064 non-null float64

23 forecast wind offshore eday ahead 0 non-null float64

24 forecast wind onshore day ahead 35064 non-null float64

25 total load forecast 35064 non-null float64

26 total load actual 35028 non-null float64

27 price day ahead 35064 non-null float64

28 price actual 35064 non-null float64

dtypes: float64(28), object(1)

memory usage: 7.8+ MB

df\_weather.info()

RangeIndex: 178396 entries, 0 to 178395

Data columns (total 17 columns):

# Column Non-Null Count Dtype

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

0 dt\_iso 178396 non-null object

1 city\_name 178396 non-null object

2 temp 178396 non-null float64

3 temp\_min 178396 non-null float64

4 temp\_max 178396 non-null float64

5 pressure 178396 non-null int64

6 humidity 178396 non-null int64

7 wind\_speed 178396 non-null int64

8 wind\_deg 178396 non-null int64

9 rain\_1h 178396 non-null float64

10 rain\_3h 178396 non-null float64

11 snow\_3h 178396 non-null float64

12 clouds\_all 178396 non-null int64

13 weather\_id 178396 non-null int64

14 weather\_main 178396 non-null object

15 weather\_description 178396 non-null object

16 weather\_icon 178396 non-null object

dtypes: float64(6), int64(6), object(5)

memory usage: 23.1+ MB

df\_energy.isnull().sum()

time 0

generation biomass 19

generation fossil brown coal/lignite 18

generation fossil coal-derived gas 18

generation fossil gas 18

generation fossil hard coal 18

generation fossil oil 19

generation fossil oil shale 18

generation fossil peat 18

generation geothermal 18

generation hydro pumped storage aggregated 35064

generation hydro pumped storage consumption 19

generation hydro run-of-river and poundage 19

generation hydro water reservoir 18

generation marine 19

generation nuclear 17

generation other 18

generation other renewable 18

generation solar 18

generation waste 19

generation wind offshore 18

generation wind onshore 18

forecast solar day ahead 0

forecast wind offshore eday ahead 35064

forecast wind onshore day ahead 0

total load forecast 0

total load actual 36

price day ahead 0

price actual 0

dtype: int64

df\_energy.fillna(df\_energy.median()[:], inplace = True)

df\_energy.isnull().sum()

time 0

generation biomass 0

generation fossil brown coal/lignite 0

generation fossil coal-derived gas 0

generation fossil gas 0

generation fossil hard coal 0

generation fossil oil 0

generation fossil oil shale 0

generation fossil peat 0

generation geothermal 0

generation hydro pumped storage aggregated 35064

generation hydro pumped storage consumption 0

generation hydro run-of-river and poundage 0

generation hydro water reservoir 0

generation marine 0

generation nuclear 0

generation other 0

generation other renewable 0

generation solar 0

generation waste 0

generation wind offshore 0

generation wind onshore 0

forecast solar day ahead 0

forecast wind offshore eday ahead 35064

forecast wind onshore day ahead 0

total load forecast 0

total load actual 0

price day ahead 0

price actual 0

dtype: int64

df\_energy.describe().T

count mean std min 25% 50% 75% max

generation biomass 35064.0 383.504592 85.331679 0.00 333.0000 367.00 433.00 592.00

generation fossil brown coal/lignite 35064.0 448.090492 354.480256 0.00 0.0000 509.00 757.00 999.00

generation fossil coal-derived gas 35064.0 0.000000 0.000000 0.00 0.0000 0.00 0.00 0.00

generation fossil gas 35064.0 5622.401894 2201.315046 0.00 4127.0000 4969.00 6428.00 20034.00

generation fossil hard coal 35064.0 4256.177618 1961.103657 0.00 2528.7500 4474.00 5837.00 8359.00

generation fossil oil 35064.0 298.320699 52.506455 0.00 263.0000 300.00 330.00 449.00

generation fossil oil shale 35064.0 0.000000 0.000000 0.00 0.0000 0.00 0.00 0.00

generation fossil peat 35064.0 0.000000 0.000000 0.00 0.0000 0.00 0.00 0.00

generation geothermal 35064.0 0.000000 0.000000 0.00 0.0000 0.00 0.00 0.00

generation hydro pumped storage aggregated 0.0 NaN NaN NaN NaN NaN NaN NaN

generation hydro pumped storage consumption 35064.0 475.356491 792.248672 0.00 0.0000 68.00 615.00 4523.00

generation hydro run-of-river and poundage 35064.0 972.080282 400.671889 0.00 637.0000 906.00 1250.00 2000.00

generation hydro water reservoir 35064.0 2604.888290 1834.755832 0.00 1078.0000 2164.00 3756.25 9728.00

generation marine 35064.0 0.000000 0.000000 0.00 0.0000 0.00 0.00 0.00

generation nuclear 35064.0 6264.053502 839.490721 0.00 5761.0000 6566.00 7024.00 7117.00

generation other 35064.0 60.226928 20.233318 0.00 53.0000 57.00 80.00 106.00

generation other renewable 35064.0 85.640914 14.074042 0.00 74.0000 88.00 97.00 119.00

generation solar 35064.0 1432.246692 1679.790440 0.00 71.0000 616.00 2575.25 5792.00

generation waste 35064.0 269.457307 50.182426 0.00 240.0000 279.00 310.00 357.00

generation wind offshore 35064.0 0.000000 0.000000 0.00 0.0000 0.00 0.00 0.00

generation wind onshore 35064.0 5464.163815 3212.896837 0.00 2933.7500 4849.00 7397.00 17436.00

forecast solar day ahead 35064.0 1439.066735 1677.703355 0.00 69.0000 576.00 2636.00 5836.00

forecast wind offshore eday ahead 0.0 NaN NaN NaN NaN NaN NaN NaN

forecast wind onshore day ahead 35064.0 5471.216689 3176.312853 237.00 2979.0000 4855.00 7353.00 17430.00

total load forecast 35064.0 28712.129962 4594.100854 18105.00 24793.7500 28906.00 32263.25 41390.00

total load actual 35064.0 28697.149413 4572.643394 18041.00 24810.0000 28901.00 32186.25 41015.00

price day ahead 35064.0 49.874341 14.618900 2.06 41.4900 50.52 60.53 101.99

price actual 35064.0 57.884023 14.204083 9.33 49.3475 58.02 68.01 116.80

columns\_to\_drop = ["generation fossil coal-derived gas", "generation fossil oil shale", "generation fossil peat",

"generation geothermal", "generation marine", "generation wind offshore","generation hydro pumped storage aggregated", "forecast wind offshore eday ahead"]

df\_energy = df\_energy.drop(columns = columns\_to\_drop)

df\_energy.describe().T

count mean std min 25% 50% 75% max

generation biomass 35064.0 383.504592 85.331679 0.00 333.0000 367.00 433.00 592.00

generation fossil brown coal/lignite 35064.0 448.090492 354.480256 0.00 0.0000 509.00 757.00 999.00

generation fossil gas 35064.0 5622.401894 2201.315046 0.00 4127.0000 4969.00 6428.00 20034.00

generation fossil hard coal 35064.0 4256.177618 1961.103657 0.00 2528.7500 4474.00 5837.00 8359.00

generation fossil oil 35064.0 298.320699 52.506455 0.00 263.0000 300.00 330.00 449.00

generation hydro pumped storage consumption 35064.0 475.356491 792.248672 0.00 0.0000 68.00 615.00 4523.00

generation hydro run-of-river and poundage 35064.0 972.080282 400.671889 0.00 637.0000 906.00 1250.00 2000.00

generation hydro water reservoir 35064.0 2604.888290 1834.755832 0.00 1078.0000 2164.00 3756.25 9728.00

generation nuclear 35064.0 6264.053502 839.490721 0.00 5761.0000 6566.00 7024.00 7117.00

generation other 35064.0 60.226928 20.233318 0.00 53.0000 57.00 80.00 106.00

generation other renewable 35064.0 85.640914 14.074042 0.00 74.0000 88.00 97.00 119.00

generation solar 35064.0 1432.246692 1679.790440 0.00 71.0000 616.00 2575.25 5792.00

generation waste 35064.0 269.457307 50.182426 0.00 240.0000 279.00 310.00 357.00

generation wind onshore 35064.0 5464.163815 3212.896837 0.00 2933.7500 4849.00 7397.00 17436.00

forecast solar day ahead 35064.0 1439.066735 1677.703355 0.00 69.0000 576.00 2636.00 5836.00

forecast wind onshore day ahead 35064.0 5471.216689 3176.312853 237.00 2979.0000 4855.00 7353.00 17430.00

total load forecast 35064.0 28712.129962 4594.100854 18105.00 24793.7500 28906.00 32263.25 41390.00

total load actual 35064.0 28697.149413 4572.643394 18041.00 24810.0000 28901.00 32186.25 41015.00

price day ahead 35064.0 49.874341 14.618900 2.06 41.4900 50.52 60.53 101.99

price actual 35064.0 57.884023 14.204083 9.33 49.3475 58.02 68.01 116.80

list(df\_energy.columns)[15:21]

['forecast solar day ahead',

'forecast wind onshore day ahead',

'total load forecast',

'total load actual',

'price day ahead',

'price actual']

print(df\_energy["generation biomass"].value\_counts())

print("-" \* 20)

print(df\_energy["generation fossil brown coal/lignite"].value\_counts())

print("-" \* 20)

print(df\_energy["generation fossil gas"].value\_counts())

print("-" \* 20)

print(df\_energy["generation fossil hard coal"].value\_counts())

print("-" \* 20)

print(df\_energy["generation fossil oil"].value\_counts())

print("-" \* 20)

print(df\_energy["generation hydro pumped storage consumption"].value\_counts())

print("-" \* 20)

print(df\_energy["generation hydro run-of-river and poundage"].value\_counts())

print("-" \* 20)

print(df\_energy["generation hydro water reservoir"].value\_counts())

print("-" \* 20)

print(df\_energy["generation nuclear"].value\_counts())

print("-" \* 20)

print(df\_energy["generation other"].value\_counts())

print("-" \* 20)

print(df\_energy["generation other renewable"].value\_counts())

print("-" \* 20)

print(df\_energy["generation waste"].value\_counts())

print("-" \* 20)

print(df\_energy["generation wind onshore"].value\_counts())

print("-" \* 20)

print(df\_energy["forecast solar day ahead"].value\_counts())

print("-" \* 20)

print(df\_energy["forecast wind onshore day ahead"].value\_counts())

print("-" \* 20)

print(df\_energy["total load forecast"].value\_counts())

print("-" \* 20)

print(df\_energy["total load actual"].value\_counts())

print("-" \* 20)

print(df\_energy["price day ahead"].value\_counts())

print("-" \* 20)

print(df\_energy["price actual"].value\_counts())

print("-" \* 20)

361.0 321

362.0 318

367.0 315

351.0 310

359.0 305

...

101.0 1

589.0 1

174.0 1

175.0 1

168.0 1

Name: generation biomass, Length: 423, dtype: int64

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

0.0 10517

663.0 165

664.0 124

595.0 108

657.0 103

...

144.0 1

39.0 1

87.0 1

41.0 1

35.0 1

Name: generation fossil brown coal/lignite, Length: 956, dtype: int64

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

4969.0 25

4180.0 24

3993.0 24

4227.0 21

3856.0 21

..

3284.0 1

3323.0 1

8903.0 1

9341.0 1

8024.0 1

Name: generation fossil gas, Length: 8297, dtype: int64

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

4474.0 23

5266.0 16

6176.0 15

4747.0 15

5324.0 15

..

666.0 1

671.0 1

654.0 1

663.0 1

3443.0 1

Name: generation fossil hard coal, Length: 7266, dtype: int64

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

300.0 342

303.0 335

309.0 328

304.0 326

308.0 320

...

122.0 1

145.0 1

131.0 1

129.0 1

44.0 1

Name: generation fossil oil, Length: 321, dtype: int64

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

0.0 12607

1.0 1641

2.0 300

3.0 184

4.0 130

...

2743.0 1

2505.0 1

3068.0 1

1913.0 1

2217.0 1

Name: generation hydro pumped storage consumption, Length: 3311, dtype: int64

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

600.0 59

632.0 58

615.0 56

552.0 56

553.0 56

..

1989.0 1

2000.0 1

301.0 1

287.0 1

1918.0 1

Name: generation hydro run-of-river and poundage, Length: 1684, dtype: int64

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

2164.0 26

801.0 26

621.0 22

559.0 21

1311.0 21

..

7550.0 1

5507.0 1

5986.0 1

6295.0 1

3144.0 1

Name: generation hydro water reservoir, Length: 7029, dtype: int64

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

7102.0 376

7103.0 367

7104.0 366

7101.0 364

7098.0 344

...

5135.0 1

5705.0 1

4367.0 1

5642.0 1

5812.0 1

Name: generation nuclear, Length: 2388, dtype: int64

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

57.0 2306

56.0 2271

55.0 2183

58.0 2093

54.0 1775

...

31.0 2

102.0 2

103.0 1

106.0 1

3.0 1

Name: generation other, Length: 103, dtype: int64

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

94.0 990

92.0 981

99.0 978

93.0 961

96.0 954

...

43.0 2

119.0 2

14.0 1

4.0 1

45.0 1

Name: generation other renewable, Length: 78, dtype: int64

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

317.0 405

312.0 398

316.0 397

319.0 391

314.0 387

...

85.0 1

84.0 1

39.0 1

100.0 1

356.0 1

Name: generation waste, Length: 262, dtype: int64

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

4849.0 23

2845.0 15

3932.0 15

2422.0 14

2590.0 14

..

9707.0 1

7882.0 1

9427.0 1

9447.0 1

6133.0 1

Name: generation wind onshore, Length: 11465, dtype: int64

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

10.0 555

0.0 539

11.0 508

1.0 373

12.0 358

...

3641.0 1

2075.0 1

3521.0 1

3862.0 1

1896.0 1

Name: forecast solar day ahead, Length: 5356, dtype: int64

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

2802.0 14

3488.0 13

3575.0 13

3416.0 12

3196.0 12

..

15654.0 1

4223.0 1

10555.0 1

10233.0 1

5904.0 1

Name: forecast wind onshore day ahead, Length: 11332, dtype: int64

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

31051.0 13

29932.0 12

31596.0 11

35277.0 11

23863.0 11

..

21240.0 1

21751.0 1

22263.0 1

35432.0 1

28517.0 1

Name: total load forecast, Length: 14790, dtype: int64

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

28901.0 41

23665.0 12

30960.0 10

33176.0 10

32023.0 10

..

36765.0 1

37498.0 1

37058.0 1

35979.0 1

21723.0 1

Name: total load actual, Length: 15127, dtype: int64

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

40.00 179

50.00 178

45.00 151

60.00 126

55.00 116

...

73.61 1

70.06 1

12.70 1

73.53 1

71.97 1

Name: price day ahead, Length: 5747, dtype: int64

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

56.85 24

55.91 22

56.71 22

52.35 21

51.33 21

..

17.96 1

17.47 1

16.34 1

18.41 1

80.10 1

Name: price actual, Length: 6653, dtype: int64

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

sns.boxplot(df\_energy["generation biomass"])

sns.boxplot(df\_energy["generation fossil brown coal/lignite"])

from sklearn.neighbors import LocalOutlierFactor

df\_num = df\_energy.select\_dtypes(include=["float64", "int64"])

clf = LocalOutlierFactor(n\_neighbors=20, contamination=0.1)

fit = clf.fit\_predict(df\_num)

df\_scores = clf.negative\_outlier\_factor\_

print(df\_scores[0:10])

[-1.07287361 -1.15535587 -1.16073867 -1.01477163 -1.08179224 -1.08870622

-1.04675871 -1.04660999 -1.09663193 -1.17142102]

np.sort(df\_scores)[0:50]

array([-4.38955627, -4.32712305, -4.17489329, -4.0029864 , -3.80567405,

-3.43961544, -3.28859679, -2.95463012, -2.85648109, -2.74139603,

-2.60991698, -2.52277086, -2.52260459, -2.50933089, -2.50577667,

-2.43650375, -2.42474736, -2.42289071, -2.40095399, -2.30173643,

-2.26795101, -2.25326253, -2.21993406, -2.21962716, -2.19360932,

-2.12122224, -2.11877181, -2.07655147, -2.00345074, -1.92934095,

-1.68438241, -1.65925543, -1.65079179, -1.63311836, -1.62633599,

-1.60638708, -1.60231089, -1.56810008, -1.55470291, -1.54241435,

-1.54033786, -1.53267001, -1.53141665, -1.52858214, -1.51040426,

-1.50994577, -1.49884335, -1.49310154, -1.49152365, -1.49022744])

esik\_deger = np.sort(df\_scores)[4:5]

aykiri\_tf = df\_scores > esik\_deger

new\_df = df\_num[df\_scores > esik\_deger]

df\_num[df\_scores < esik\_deger]

generation biomass generation fossil brown coal/lignite generation fossil gas generation fossil hard coal generation fossil oil generation hydro pumped storage consumption generation hydro run-of-river and poundage generation hydro water reservoir generation nuclear generation other generation other renewable generation solar generation waste generation wind onshore forecast solar day ahead forecast wind onshore day ahead total load forecast total load actual price day ahead price actual

2025 101.0 0.0 1854.0 1762.0 87.0 0.0 283.0 4070.0 6314.0 17.0 14.0 401.0 39.0 2387.0 1949.0 10036.0 34724.0 21218.0 51.50 62.79

5706 324.0 481.0 4219.0 4044.0 122.0 15.0 362.0 716.0 3939.0 48.0 43.0 1567.0 156.0 3806.0 2615.0 6380.0 31545.0 18616.0 62.23 70.71

25164 0.0 0.0 10064.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 4325.0 7561.0 33805.0 28901.0 60.53 66.17

25171 0.0 0.0 12336.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 128.0 5679.0 35592.0 28901.0 68.05 75.45

new\_df

generation biomass generation fossil brown coal/lignite generation fossil gas generation fossil hard coal generation fossil oil generation hydro pumped storage consumption generation hydro run-of-river and poundage generation hydro water reservoir generation nuclear generation other generation other renewable generation solar generation waste generation wind onshore forecast solar day ahead forecast wind onshore day ahead total load forecast total load actual price day ahead price actual

0 447.0 329.0 4844.0 4821.0 162.0 863.0 1051.0 1899.0 7096.0 43.0 73.0 49.0 196.0 6378.0 17.0 6436.0 26118.0 25385.0 50.10 65.41

1 449.0 328.0 5196.0 4755.0 158.0 920.0 1009.0 1658.0 7096.0 43.0 71.0 50.0 195.0 5890.0 16.0 5856.0 24934.0 24382.0 48.10 64.92

2 448.0 323.0 4857.0 4581.0 157.0 1164.0 973.0 1371.0 7099.0 43.0 73.0 50.0 196.0 5461.0 8.0 5454.0 23515.0 22734.0 47.33 64.48

3 438.0 254.0 4314.0 4131.0 160.0 1503.0 949.0 779.0 7098.0 43.0 75.0 50.0 191.0 5238.0 2.0 5151.0 22642.0 21286.0 42.27 59.32

4 428.0 187.0 4130.0 3840.0 156.0 1826.0 953.0 720.0 7097.0 43.0 74.0 42.0 189.0 4935.0 9.0 4861.0 21785.0 20264.0 38.41 56.04

... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...

35059 297.0 0.0 7634.0 2628.0 178.0 1.0 1135.0 4836.0 6073.0 63.0 95.0 85.0 277.0 3113.0 96.0 3253.0 30619.0 30653.0 68.85 77.02

35060 296.0 0.0 7241.0 2566.0 174.0 1.0 1172.0 3931.0 6074.0 62.0 95.0 33.0 280.0 3288.0 51.0 3353.0 29932.0 29735.0 68.40 76.16

35061 292.0 0.0 7025.0 2422.0 168.0 50.0 1148.0 2831.0 6076.0 61.0 94.0 31.0 286.0 3503.0 36.0 3404.0 27903.0 28071.0 66.88 74.30

35062 293.0 0.0 6562.0 2293.0 163.0 108.0 1128.0 2068.0 6075.0 61.0 93.0 31.0 287.0 3586.0 29.0 3273.0 25450.0 25801.0 63.93 69.89

35063 290.0 0.0 6926.0 2166.0 163.0 108.0 1069.0 1686.0 6075.0 61.0 92.0 31.0 287.0 3651.0 26.0 3117.0 24424.0 24455.0 64.27 69.88

35059 rows × 20 columns

list(new\_df.columns)

['generation biomass',

'generation fossil brown coal/lignite',

'generation fossil gas',

'generation fossil hard coal',

'generation fossil oil',

'generation hydro pumped storage consumption',

'generation hydro run-of-river and poundage',

'generation hydro water reservoir',

'generation nuclear',

'generation other',

'generation other renewable',

'generation solar',

'generation waste',

'generation wind onshore',

'forecast solar day ahead',

'forecast wind onshore day ahead',

'total load forecast',

'total load actual',

'price day ahead',

'price actual']

new\_df.rename(columns={'generation biomass': 'biomass',

'generation fossil brown coal/lignite': 'brown\_coal',

'generation fossil gas': 'gas',

'generation fossil hard coal': 'hard\_coal',

'generation fossil oil': 'oil',

'generation hydro pumped storage consumption': 'hydro\_pumped\_storage',

'generation hydro run-of-river and poundage': 'hydro\_ror\_poundage',

'generation hydro water reservoir': 'hydro\_w\_reservoir',

'generation nuclear': 'nuclear',

'generation other': 'other',

'generation other renewable': 'other\_renewable',

'generation solar': 'solar',

'generation waste': 'waste',

'generation wind onshore': 'wind\_onshore',

'forecast solar day ahead': 'solar\_day',

'forecast wind onshore day ahead': 'wind\_onshore',

'total load forecast': 't\_forecast',

'total load actual': 't\_actual',

'price day ahead': 'p\_ahead',

'price actual': 'p\_actual'}, inplace=True)

new\_df.head()

biomass brown\_coal gas hard\_coal oil hydro\_pumped\_storage hydro\_ror\_poundage hydro\_w\_reservoir nuclear other other\_renewable solar waste wind\_onshore solar\_day wind\_onshore t\_forecast t\_actual p\_ahead p\_actual

0 447.0 329.0 4844.0 4821.0 162.0 863.0 1051.0 1899.0 7096.0 43.0 73.0 49.0 196.0 6378.0 17.0 6436.0 26118.0 25385.0 50.10 65.41

1 449.0 328.0 5196.0 4755.0 158.0 920.0 1009.0 1658.0 7096.0 43.0 71.0 50.0 195.0 5890.0 16.0 5856.0 24934.0 24382.0 48.10 64.92

2 448.0 323.0 4857.0 4581.0 157.0 1164.0 973.0 1371.0 7099.0 43.0 73.0 50.0 196.0 5461.0 8.0 5454.0 23515.0 22734.0 47.33 64.48

3 438.0 254.0 4314.0 4131.0 160.0 1503.0 949.0 779.0 7098.0 43.0 75.0 50.0 191.0 5238.0 2.0 5151.0 22642.0 21286.0 42.27 59.32

4 428.0 187.0 4130.0 3840.0 156.0 1826.0 953.0 720.0 7097.0 43.0 74.0 42.0 189.0 4935.0 9.0 4861.0 21785.0 20264.0 38.41 56.04

new\_df['time'] = df\_energy['time']

new\_df.tail()

biomass brown\_coal gas hard\_coal oil hydro\_pumped\_storage hydro\_ror\_poundage hydro\_w\_reservoir nuclear other ... solar waste wind\_onshore solar\_day wind\_onshore t\_forecast t\_actual p\_ahead p\_actual time

35059 297.0 0.0 7634.0 2628.0 178.0 1.0 1135.0 4836.0 6073.0 63.0 ... 85.0 277.0 3113.0 96.0 3253.0 30619.0 30653.0 68.85 77.02 2018-12-31 19:00:00+01:00

35060 296.0 0.0 7241.0 2566.0 174.0 1.0 1172.0 3931.0 6074.0 62.0 ... 33.0 280.0 3288.0 51.0 3353.0 29932.0 29735.0 68.40 76.16 2018-12-31 20:00:00+01:00

35061 292.0 0.0 7025.0 2422.0 168.0 50.0 1148.0 2831.0 6076.0 61.0 ... 31.0 286.0 3503.0 36.0 3404.0 27903.0 28071.0 66.88 74.30 2018-12-31 21:00:00+01:00

35062 293.0 0.0 6562.0 2293.0 163.0 108.0 1128.0 2068.0 6075.0 61.0 ... 31.0 287.0 3586.0 29.0 3273.0 25450.0 25801.0 63.93 69.89 2018-12-31 22:00:00+01:00

35063 290.0 0.0 6926.0 2166.0 163.0 108.0 1069.0 1686.0 6075.0 61.0 ... 31.0 287.0 3651.0 26.0 3117.0 24424.0 24455.0 64.27 69.88 2018-12-31 23:00:00+01:00

5 rows × 21 columns

new\_df\_energy = new\_df.copy()

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

df\_weather.head(10)

dt\_iso city\_name temp temp\_min temp\_max pressure humidity wind\_speed wind\_deg rain\_1h rain\_3h snow\_3h clouds\_all weather\_id weather\_main weather\_description weather\_icon

0 2015-01-01 00:00:00+01:00 Valencia 270.475 270.475 270.475 1001 77 1 62 0.0 0.0 0.0 0 800 clear sky is clear 01n

1 2015-01-01 01:00:00+01:00 Valencia 270.475 270.475 270.475 1001 77 1 62 0.0 0.0 0.0 0 800 clear sky is clear 01n

2 2015-01-01 02:00:00+01:00 Valencia 269.686 269.686 269.686 1002 78 0 23 0.0 0.0 0.0 0 800 clear sky is clear 01n

3 2015-01-01 03:00:00+01:00 Valencia 269.686 269.686 269.686 1002 78 0 23 0.0 0.0 0.0 0 800 clear sky is clear 01n

4 2015-01-01 04:00:00+01:00 Valencia 269.686 269.686 269.686 1002 78 0 23 0.0 0.0 0.0 0 800 clear sky is clear 01n

5 2015-01-01 05:00:00+01:00 Valencia 270.292 270.292 270.292 1004 71 2 321 0.0 0.0 0.0 0 800 clear sky is clear 01n

6 2015-01-01 06:00:00+01:00 Valencia 270.292 270.292 270.292 1004 71 2 321 0.0 0.0 0.0 0 800 clear sky is clear 01n

7 2015-01-01 07:00:00+01:00 Valencia 270.292 270.292 270.292 1004 71 2 321 0.0 0.0 0.0 0 800 clear sky is clear 01n

8 2015-01-01 08:00:00+01:00 Valencia 274.601 274.601 274.601 1005 71 1 307 0.0 0.0 0.0 0 800 clear sky is clear 01d

9 2015-01-01 09:00:00+01:00 Valencia 274.601 274.601 274.601 1005 71 1 307 0.0 0.0 0.0 0 800 clear sky is clear 01d

df\_weather.describe().T

count mean std min 25% 50% 75% max

temp 178396.0 289.618605 8.026199 262.24 283.670000 289.15 295.150000 315.600

temp\_min 178396.0 288.330442 7.955491 262.24 282.483602 288.15 293.730125 315.150

temp\_max 178396.0 291.091267 8.612454 262.24 284.650000 290.15 297.150000 321.150

pressure 178396.0 1069.260740 5969.631893 0.00 1013.000000 1018.00 1022.000000 1008371.000

humidity 178396.0 68.423457 21.902888 0.00 53.000000 72.00 87.000000 100.000

wind\_speed 178396.0 2.470560 2.095910 0.00 1.000000 2.00 4.000000 133.000

wind\_deg 178396.0 166.591190 116.611927 0.00 55.000000 177.00 270.000000 360.000

rain\_1h 178396.0 0.075492 0.398847 0.00 0.000000 0.00 0.000000 12.000

rain\_3h 178396.0 0.000380 0.007288 0.00 0.000000 0.00 0.000000 2.315

snow\_3h 178396.0 0.004763 0.222604 0.00 0.000000 0.00 0.000000 21.500

clouds\_all 178396.0 25.073292 30.774129 0.00 0.000000 20.00 40.000000 100.000

weather\_id 178396.0 759.831902 108.733223 200.00 800.000000 800.00 801.000000 804.000

df\_weather.info()

RangeIndex: 178396 entries, 0 to 178395

Data columns (total 17 columns):

# Column Non-Null Count Dtype

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

0 dt\_iso 178396 non-null object

1 city\_name 178396 non-null object

2 temp 178396 non-null float64

3 temp\_min 178396 non-null float64

4 temp\_max 178396 non-null float64

5 pressure 178396 non-null int64

6 humidity 178396 non-null int64

7 wind\_speed 178396 non-null int64

8 wind\_deg 178396 non-null int64

9 rain\_1h 178396 non-null float64

10 rain\_3h 178396 non-null float64

11 snow\_3h 178396 non-null float64

12 clouds\_all 178396 non-null int64

13 weather\_id 178396 non-null int64

14 weather\_main 178396 non-null object

15 weather\_description 178396 non-null object

16 weather\_icon 178396 non-null object

dtypes: float64(6), int64(6), object(5)

memory usage: 23.1+ MB

df\_weather['weather\_main'].unique()

array(['clear', 'clouds', 'rain', 'mist', 'thunderstorm', 'drizzle',

'fog', 'smoke', 'haze', 'snow', 'dust', 'squall'], dtype=object)

df\_weather['weather\_description'].unique()

array(['sky is clear', 'few clouds', 'scattered clouds', 'broken clouds',

'overcast clouds', 'light rain', 'moderate rain',

'heavy intensity rain', 'mist', 'heavy intensity shower rain',

'shower rain', 'very heavy rain', 'thunderstorm with heavy rain',

'thunderstorm with light rain', 'thunderstorm with rain',

'proximity thunderstorm', 'thunderstorm',

'light intensity shower rain', 'light intensity drizzle', 'fog',

'drizzle', 'smoke', 'heavy intensity drizzle', 'haze',

'proximity shower rain', 'light intensity drizzle rain',

'light snow', 'rain and snow', 'light rain and snow', 'snow',

'light thunderstorm', 'heavy snow', 'sleet', 'rain and drizzle',

'shower sleet', 'light shower sleet', 'light shower snow',

'proximity moderate rain', 'ragged shower rain',

'sand dust whirls', 'proximity drizzle', 'dust', 'squalls'],

dtype=object)

df\_weather['city\_name'].unique()

array(['Valencia', 'Madrid', 'Bilbao', ' Barcelona', 'Seville'],

dtype=object)

df\_weather['wind\_speed'].unique()

array([ 1, 0, 2, 3, 6, 4, 8, 5, 12, 17, 9, 7, 22,

11, 25, 20, 27, 30, 14, 19, 16, 10, 35, 33, 64, 24,

54, 40, 38, 21, 15, 29, 43, 13, 133, 18], dtype=int64)

df\_weather['rain\_1h'].unique()

array([ 0. , 0.3 , 0.9 , 3. , 12. , 2.29, 0.25])

columns\_to\_drop = ['city\_name', 'weather\_description', 'weather\_icon', 'weather\_id', 'temp\_min', 'temp\_max',

'pressure', 'rain\_1h', 'clouds\_all']

df\_weather = df\_weather.drop(columns = columns\_to\_drop)

df\_weather.info()

RangeIndex: 178396 entries, 0 to 178395

Data columns (total 8 columns):

# Column Non-Null Count Dtype

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

0 dt\_iso 178396 non-null object

1 temp 178396 non-null float64

2 humidity 178396 non-null int64

3 wind\_speed 178396 non-null int64

4 wind\_deg 178396 non-null int64

5 rain\_3h 178396 non-null float64

6 snow\_3h 178396 non-null float64

7 weather\_main 178396 non-null object

dtypes: float64(3), int64(3), object(2)

memory usage: 10.9+ MB

df\_weather = df\_weather.drop(columns = "wind\_deg")

df\_weather.info()

RangeIndex: 178396 entries, 0 to 178395

Data columns (total 7 columns):

# Column Non-Null Count Dtype

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

0 dt\_iso 178396 non-null object

1 temp 178396 non-null float64

2 humidity 178396 non-null int64

3 wind\_speed 178396 non-null int64

4 rain\_3h 178396 non-null float64

5 snow\_3h 178396 non-null float64

6 weather\_main 178396 non-null object

dtypes: float64(3), int64(2), object(2)

memory usage: 9.5+ MB

from sklearn.preprocessing import LabelEncoder

lbe = LabelEncoder()

lbe.fit\_transform(df\_weather["weather\_main"])

new\_df\_weather = pd.get\_dummies(df\_weather, columns=["weather\_main"], prefix = ["weather\_main"])

new\_df\_weather.head()

dt\_iso temp humidity wind\_speed rain\_3h snow\_3h weather\_main\_clear weather\_main\_clouds weather\_main\_drizzle weather\_main\_dust weather\_main\_fog weather\_main\_haze weather\_main\_mist weather\_main\_rain weather\_main\_smoke weather\_main\_snow weather\_main\_squall weather\_main\_thunderstorm

0 2015-01-01 00:00:00+01:00 270.475 77 1 0.0 0.0 1 0 0 0 0 0 0 0 0 0 0 0

1 2015-01-01 01:00:00+01:00 270.475 77 1 0.0 0.0 1 0 0 0 0 0 0 0 0 0 0 0

2 2015-01-01 02:00:00+01:00 269.686 78 0 0.0 0.0 1 0 0 0 0 0 0 0 0 0 0 0

3 2015-01-01 03:00:00+01:00 269.686 78 0 0.0 0.0 1 0 0 0 0 0 0 0 0 0 0 0

4 2015-01-01 04:00:00+01:00 269.686 78 0 0.0 0.0 1 0 0 0 0 0 0 0 0 0 0 0

list(new\_df\_weather.columns)

['dt\_iso',

'temp',

'humidity',

'wind\_speed',

'rain\_3h',

'snow\_3h',

'weather\_main\_clear',

'weather\_main\_clouds',

'weather\_main\_drizzle',

'weather\_main\_dust',

'weather\_main\_fog',

'weather\_main\_haze',

'weather\_main\_mist',

'weather\_main\_rain',

'weather\_main\_smoke',

'weather\_main\_snow',

'weather\_main\_squall',

'weather\_main\_thunderstorm']

new\_df\_weather.rename(columns={ 'weather\_main\_clear': 'clear',

'weather\_main\_clouds': 'clouds',

'weather\_main\_drizzle': 'drizzle',

'weather\_main\_dust': 'dust',

'weather\_main\_fog': 'fog',

'weather\_main\_haze': 'haze',

'weather\_main\_mist': 'mist',

'weather\_main\_rain': 'rain',

'weather\_main\_smoke': 'smoke',

'weather\_main\_snow': 'snow',

'weather\_main\_squall': 'squall',

'weather\_main\_thunderstorm': 'thunderstorm'}, inplace= True)

new\_df\_weather.head()

dt\_iso temp humidity wind\_speed rain\_3h snow\_3h clear clouds drizzle dust fog haze mist rain smoke snow squall thunderstorm

0 2015-01-01 00:00:00+01:00 270.475 77 1 0.0 0.0 1 0 0 0 0 0 0 0 0 0 0 0

1 2015-01-01 01:00:00+01:00 270.475 77 1 0.0 0.0 1 0 0 0 0 0 0 0 0 0 0 0

2 2015-01-01 02:00:00+01:00 269.686 78 0 0.0 0.0 1 0 0 0 0 0 0 0 0 0 0 0

3 2015-01-01 03:00:00+01:00 269.686 78 0 0.0 0.0 1 0 0 0 0 0 0 0 0 0 0 0

4 2015-01-01 04:00:00+01:00 269.686 78 0 0.0 0.0 1 0 0 0 0 0 0 0 0 0 0 0

new\_df\_weather.describe().T

count mean std min 25% 50% 75% max

temp 178396.0 289.618605 8.026199 262.24 283.67 289.15 295.15 315.600

humidity 178396.0 68.423457 21.902888 0.00 53.00 72.00 87.00 100.000

wind\_speed 178396.0 2.470560 2.095910 0.00 1.00 2.00 4.00 133.000

rain\_3h 178396.0 0.000380 0.007288 0.00 0.00 0.00 0.00 2.315

snow\_3h 178396.0 0.004763 0.222604 0.00 0.00 0.00 0.00 21.500

clear 178396.0 0.463491 0.498667 0.00 0.00 0.00 1.00 1.000

clouds 178396.0 0.381483 0.485752 0.00 0.00 0.00 1.00 1.000

drizzle 178396.0 0.009664 0.097829 0.00 0.00 0.00 0.00 1.000

dust 178396.0 0.001945 0.044061 0.00 0.00 0.00 0.00 1.000

fog 178396.0 0.014047 0.117687 0.00 0.00 0.00 0.00 1.000

haze 178396.0 0.002438 0.049320 0.00 0.00 0.00 0.00 1.000

mist 178396.0 0.021906 0.146378 0.00 0.00 0.00 0.00 1.000

rain 178396.0 0.097485 0.296618 0.00 0.00 0.00 0.00 1.000

smoke 178396.0 0.000185 0.013600 0.00 0.00 0.00 0.00 1.000

snow 178396.0 0.001513 0.038874 0.00 0.00 0.00 0.00 1.000

squall 178396.0 0.000006 0.002368 0.00 0.00 0.00 0.00 1.000

thunderstorm 178396.0 0.005835 0.076166 0.00 0.00 0.00 0.00 1.000

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

Birleştirme işlemleri: Sadece int ve float türlerinden oluşan veri setlerimizin bir tek zaman dilimleri object olacak şekilde bıraktım. Ardından her iki veri seti için ortak olan bu object veriler üzerinden merge ile birleştirme işlemi sağladım ve ardından bu veri setinden object türlerini çıkararak makine modellemesine uygun hale getirdim.

df\_enegry\_weather= pd.merge(new\_df\_energy, new\_df\_weather, left\_on='time', right\_on='dt\_iso')

drop\_columns = df\_enegry\_weather.nunique()[df\_enegry\_weather.nunique() < 2].index.to\_list()

df\_enegry\_weather = df\_enegry\_weather.drop(drop\_columns,axis=1)

df\_enegry\_weather.info()

Int64Index: 178371 entries, 0 to 178370

Data columns (total 39 columns):

# Column Non-Null Count Dtype

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

0 biomass 178371 non-null float64

1 brown\_coal 178371 non-null float64

2 gas 178371 non-null float64

3 hard\_coal 178371 non-null float64

4 oil 178371 non-null float64

5 hydro\_pumped\_storage 178371 non-null float64

6 hydro\_ror\_poundage 178371 non-null float64

7 hydro\_w\_reservoir 178371 non-null float64

8 nuclear 178371 non-null float64

9 other 178371 non-null float64

10 other\_renewable 178371 non-null float64

11 solar 178371 non-null float64

12 waste 178371 non-null float64

13 wind\_onshore 178371 non-null float64

14 solar\_day 178371 non-null float64

15 wind\_onshore 178371 non-null float64

16 t\_forecast 178371 non-null float64

17 t\_actual 178371 non-null float64

18 p\_ahead 178371 non-null float64

19 p\_actual 178371 non-null float64

20 time 178371 non-null object

21 dt\_iso 178371 non-null object

22 temp 178371 non-null float64

23 humidity 178371 non-null int64

24 wind\_speed 178371 non-null int64

25 rain\_3h 178371 non-null float64

26 snow\_3h 178371 non-null float64

27 clear 178371 non-null uint8

28 clouds 178371 non-null uint8

29 drizzle 178371 non-null uint8

30 dust 178371 non-null uint8

31 fog 178371 non-null uint8

32 haze 178371 non-null uint8

33 mist 178371 non-null uint8

34 rain 178371 non-null uint8

35 smoke 178371 non-null uint8

36 snow 178371 non-null uint8

37 squall 178371 non-null uint8

38 thunderstorm 178371 non-null uint8

dtypes: float64(23), int64(2), object(2), uint8(12)

memory usage: 40.1+ MB

df\_enegry\_weather.isnull().sum()

biomass 0

brown\_coal 0

gas 0

hard\_coal 0

oil 0

hydro\_pumped\_storage 0

hydro\_ror\_poundage 0

hydro\_w\_reservoir 0

nuclear 0

other 0

other\_renewable 0

solar 0

waste 0

wind\_onshore 0

solar\_day 0

wind\_onshore 0

t\_forecast 0

t\_actual 0

p\_ahead 0

p\_actual 0

time 0

dt\_iso 0

temp 0

humidity 0

wind\_speed 0

rain\_3h 0

snow\_3h 0

clear 0

clouds 0

drizzle 0

dust 0

fog 0

haze 0

mist 0

rain 0

smoke 0

snow 0

squall 0

thunderstorm 0

dtype: int64

column\_to\_drop = ['time', 'dt\_iso']

df\_enegry\_weather = df\_enegry\_weather.drop(columns=column\_to\_drop)

df\_enegry\_weather.info()

Int64Index: 178371 entries, 0 to 178370

Data columns (total 37 columns):

# Column Non-Null Count Dtype

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

0 biomass 178371 non-null float64

1 brown\_coal 178371 non-null float64

2 gas 178371 non-null float64

3 hard\_coal 178371 non-null float64

4 oil 178371 non-null float64

5 hydro\_pumped\_storage 178371 non-null float64

6 hydro\_ror\_poundage 178371 non-null float64

7 hydro\_w\_reservoir 178371 non-null float64

8 nuclear 178371 non-null float64

9 other 178371 non-null float64

10 other\_renewable 178371 non-null float64

11 solar 178371 non-null float64

12 waste 178371 non-null float64

13 wind\_onshore 178371 non-null float64

14 solar\_day 178371 non-null float64

15 wind\_onshore 178371 non-null float64

16 t\_forecast 178371 non-null float64

17 t\_actual 178371 non-null float64

18 p\_ahead 178371 non-null float64

19 p\_actual 178371 non-null float64

20 temp 178371 non-null float64

21 humidity 178371 non-null int64

22 wind\_speed 178371 non-null int64

23 rain\_3h 178371 non-null float64

24 snow\_3h 178371 non-null float64

25 clear 178371 non-null uint8

26 clouds 178371 non-null uint8

27 drizzle 178371 non-null uint8

28 dust 178371 non-null uint8

29 fog 178371 non-null uint8

30 haze 178371 non-null uint8

31 mist 178371 non-null uint8

32 rain 178371 non-null uint8

33 smoke 178371 non-null uint8

34 snow 178371 non-null uint8

35 squall 178371 non-null uint8

36 thunderstorm 178371 non-null uint8

dtypes: float64(23), int64(2), uint8(12)

memory usage: 37.4 MB

df\_enegry\_weather.tail()

biomass brown\_coal gas hard\_coal oil hydro\_pumped\_storage hydro\_ror\_poundage hydro\_w\_reservoir nuclear other ... drizzle dust fog haze mist rain smoke snow squall thunderstorm

178366 290.0 0.0 6926.0 2166.0 163.0 108.0 1069.0 1686.0 6075.0 61.0 ... 0 0 0 0 0 0 0 0 0 0

178367 290.0 0.0 6926.0 2166.0 163.0 108.0 1069.0 1686.0 6075.0 61.0 ... 0 0 0 0 0 0 0 0 0 0

178368 290.0 0.0 6926.0 2166.0 163.0 108.0 1069.0 1686.0 6075.0 61.0 ... 0 0 0 0 0 0 0 0 0 0

178369 290.0 0.0 6926.0 2166.0 163.0 108.0 1069.0 1686.0 6075.0 61.0 ... 0 0 0 0 0 0 0 0 0 0

178370 290.0 0.0 6926.0 2166.0 163.0 108.0 1069.0 1686.0 6075.0 61.0 ... 0 0 0 0 0 0 0 0 0 0

5 rows × 37 columns

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

Makine Öğrenmesi ve Modelleme Bölümü: Temel işlemler

print(df\_enegry\_weather[['biomass', 'p\_ahead']].corr())

print(df\_enegry\_weather[['gas', 'p\_ahead']].corr())

print(df\_enegry\_weather[['solar\_day', 'p\_ahead']].corr())

print("-" \* 30)

print(df\_enegry\_weather[['temp', 'p\_ahead']].corr())

print(df\_enegry\_weather[['humidity', 'p\_ahead']].corr())

print(df\_enegry\_weather[['wind\_speed', 'p\_ahead']].corr())

biomass p\_ahead

biomass 1.000000 0.109059

p\_ahead 0.109059 1.000000

gas p\_ahead

gas 1.000000 0.640937

p\_ahead 0.640937 1.000000

solar\_day p\_ahead

solar\_day 1.000000 0.061743

p\_ahead 0.061743 1.000000

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

temp p\_ahead

temp 1.000000 0.061608

p\_ahead 0.061608 1.000000

humidity p\_ahead

humidity 1.000000 -0.025738

p\_ahead -0.025738 1.000000

wind\_speed p\_ahead

wind\_speed 1.000000 -0.079953

p\_ahead -0.079953 1.000000

y\_df = df\_enegry\_weather[["p\_ahead"]]

df\_enegry\_weather.drop(["p\_ahead"], axis = 1, inplace = True)

X\_df = df\_enegry\_weather

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

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_df, y\_df, test\_size = 0.30, random\_state = 42)

Normalizasyon işlemleri:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.fit\_transform(X\_test)

Random Forests Regresyon: RF - Modeli

Model ve Tahmin bölümü:

from sklearn.ensemble import RandomForestRegressor

rf\_model = RandomForestRegressor(random\_state=42)

rf\_model.fit(X\_train, np.ravel(y\_train))

y\_pred = rf\_model.predict(X\_test)

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

0.8751900023824659

Model Tunning bölümü:

# ?rf\_model

rf\_params = {'max\_depth': list(range(1,10)),

'max\_features': [5,10,20],

'n\_estimators': [100,200,500]}

rf\_model\_cv = GridSearchCV(rf\_model, rf\_params, cv=10, n\_jobs=-1, verbose=2).fit(X\_train, y\_train)

print(f"En iyi parametreler: {str(rf\_model\_cv.best\_params\_)}")

Gradient Boosting Regresyon - GBM Modeli

Model ve Tahmin bölümü:

from sklearn.ensemble import GradientBoostingRegressor

gbm\_model = GradientBoostingRegressor().fit(X\_train, y\_train)

y\_pred = gbm\_model.predict(X\_test)

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

4.645516720056741

gbm\_params = {"learning\_rate": [0.001,0.01,0.1,0.2],

"max\_depth": [3,5,10,50,100],

"n\_estimators": [200,500,1000,2000],

"subsample": [1,0.5,0.75]}

gbm\_cv\_model = GridSearchCV(gbm\_model, gbm\_params, cv = 10, n\_jobs=-1, verbose=2).fit(X\_train, y\_train)

print(f"En iyi parametrelerimiz: {str(gbm\_cv\_model.best\_params\_)}")

Fitting 10 folds for each of 240 candidates, totalling 2400 fits

XGBoost Regresyon - XGB Modeli

Model ve Tahmin bölümü:

# pip install xgboost

Requirement already satisfied: xgboost in c:\users\husey\anaconda3\lib\site-packages (1.7.0)

Requirement already satisfied: scipy in c:\users\husey\anaconda3\lib\site-packages (from xgboost) (1.7.3)

Requirement already satisfied: numpy in c:\users\husey\anaconda3\lib\site-packages (from xgboost) (1.21.5)

Note: you may need to restart the kernel to use updated packages.

from xgboost import XGBRegressor

xgb\_model = XGBRegressor().fit(X\_train, y\_train)

y\_pred = xgb\_model.predict(X\_test)

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

2.5828733137499373

xgb\_params = {"colsample\_bytree": [0.4, 0.5, 0.6, 0.9, 1],

"n\_estimators": [100,200,500,1000],

"max\_depth": [2,3,4,5,6],

"learning\_rate": [0.1, 0.01, 0.5]}

xgb\_cv\_model = GridSearchCV(xgb\_model, xgb\_params, cv=10).fit(X\_train, y\_train)

print(f"En iyi parametrelerimiz: {str(xgb\_cv\_model.best\_params\_)}")

Light GBM - LGM Modeli

Model ve Tahmin bölümü:

# pip install lightgbm

Collecting lightgbm

Downloading lightgbm-3.3.3-py3-none-win\_amd64.whl (1.0 MB)

Requirement already satisfied: scikit-learn!=0.22.0 in c:\users\husey\anaconda3\lib\site-packages (from lightgbm) (1.0.2)

Requirement already satisfied: scipy in c:\users\husey\anaconda3\lib\site-packages (from lightgbm) (1.7.3)

Requirement already satisfied: numpy in c:\users\husey\anaconda3\lib\site-packages (from lightgbm) (1.21.5)

Requirement already satisfied: wheel in c:\users\husey\anaconda3\lib\site-packages (from lightgbm) (0.37.1)

Requirement already satisfied: joblib>=0.11 in c:\users\husey\anaconda3\lib\site-packages (from scikit-learn!=0.22.0->lightgbm) (1.1.0)

Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\husey\anaconda3\lib\site-packages (from scikit-learn!=0.22.0->lightgbm) (2.2.0)

Installing collected packages: lightgbm

Successfully installed lightgbm-3.3.3

Note: you may need to restart the kernel to use updated packages.

from lightgbm import LGBMRegressor

lgbm\_model = LGBMRegressor().fit(X\_train, y\_train)

y\_pred = lgbm\_model.predict(X\_test)

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

3.2562665143588228

Tüm modellerin birden fazla metrikle sınanması:

rf\_predict = rf\_model.predict(X\_test)

gbm\_predict = gbm\_model.predict(X\_test)

xgb\_predict = xgb\_model.predict(X\_test)

lgbm\_predict = lgbm\_model.predict(X\_test)

predict = [rf\_predict, gbm\_predict, xgb\_predict, lgbm\_predict]

algoritma\_adlari = ["Random Forests Regresyon", "Gradient Boosting Regresyon", "Extreme Gradient Boosting", "Light GBM"]

def metrics(y\_pred):

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

data = [mae, mse, rmse, r2]

return data

seriler = []

metric\_s = ["Mean Absolute Error", "Mean squared Error", "Root Mean Squared Error", "R2"]

for i in predict:

data = metrics(i)

seriler.append(data)

df\_df = pd.DataFrame(data=seriler, index=algoritma\_adlari, columns = metric\_s)

pd.set\_option('display.colheader\_justify', 'center')

print(df\_df.to\_string())

Mean Absolute Error Mean squared Error Root Mean Squared Error R2

Random Forests Regresyon 0.422527 0.765958 0.875190 0.996398

Gradient Boosting Regresyon 3.465306 21.580826 4.645517 0.898503

Extreme Gradient Boosting 1.886129 6.671235 2.582873 0.968624

Light GBM 2.387166 10.603272 3.256267 0.950132