eledem-final

November 27, 2024

[4]:
```
#!pip install -U tensorflow
```

[5]:
```
import numpy as np
import pandas as pd
import math
import seaborn as sns
import xgboost as xgb
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import tensorflow as tf


from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVR
from plotly.subplots import make_subplots
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import acf, pacf
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error
from keras import Sequential
from keras import layers
from keras.models import Model
from keras.layers import LSTM, BatchNormalization, Dropout, Dense, Flatten,␣
 ↪Conv1D
from keras.layers import MaxPooling1D, GRU, Input,Masking, Concatenate, dot
from keras.optimizers import Adam, SGD
from keras.losses import MeanAbsoluteError
from keras.metrics import RootMeanSquaredError
from keras.callbacks import EarlyStopping
from keras.callbacks import LearningRateScheduler
```

[6]:
```
file_path = 'energy_dataset.csv'
df_energy = pd.read_csv(file_path)
```

[7]:
```
df_energy.head()
```

[7]:

	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
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
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
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
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
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

```
   forecast wind onshore day ahead  total load forecast  total load actual  \
0                           6436.0              26118.0            25385.0
1                           5856.0              24934.0            24382.0
2                           5454.0              23515.0            22734.0
3                           5151.0              22642.0            21286.0
4                           4861.0              21785.0            20264.0

   price day ahead  price actual
0            50.10         65.41
1            48.10         64.92
2            47.33         64.48
3            42.27         59.32
4            38.41         56.04

[5 rows x 29 columns]
```

[8]:
```
df_energy.describe().T
```

[8]:

	count	mean	\
generation biomass	35045.0	383.513540	
generation fossil brown coal/lignite	35046.0	448.059208	
generation fossil coal-derived gas	35046.0	0.000000	
generation fossil gas	35046.0	5622.737488	
generation fossil hard coal	35046.0	4256.065742	
generation fossil oil	35045.0	298.319789	
generation fossil oil shale	35046.0	0.000000	
generation fossil peat	35046.0	0.000000	
generation geothermal	35046.0	0.000000	
generation hydro pumped storage aggregated	0.0	NaN	
generation hydro pumped storage consumption	35045.0	475.577343	
generation hydro run-of-river and poundage	35045.0	972.116108	
generation hydro water reservoir	35046.0	2605.114735	
generation marine	35045.0	0.000000	
generation nuclear	35047.0	6263.907039	
generation other	35046.0	60.228585	
generation other renewable	35046.0	85.639702	
generation solar	35046.0	1432.665925	
generation waste	35045.0	269.452133	
generation wind offshore	35046.0	0.000000	
generation wind onshore	35046.0	5464.479769	
forecast solar day ahead	35064.0	1439.066735	
forecast wind offshore eday ahead	0.0	NaN	
forecast wind onshore day ahead	35064.0	5471.216689	
total load forecast	35064.0	28712.129962	
total load actual	35028.0	28696.939905	
price day ahead	35064.0	49.874341	
price actual	35064.0	57.884023	

	std	min	\
generation biomass	85.353943	0.00	
generation fossil brown coal/lignite	354.568590	0.00	
generation fossil coal-derived gas	0.000000	0.00	
generation fossil gas	2201.830478	0.00	
generation fossil hard coal	1961.601013	0.00	
generation fossil oil	52.520673	0.00	
generation fossil oil shale	0.000000	0.00	
generation fossil peat	0.000000	0.00	
generation geothermal	0.000000	0.00	
generation hydro pumped storage aggregated	NaN	NaN	
generation hydro pumped storage consumption	792.406614	0.00	
generation hydro run-of-river and poundage	400.777536	0.00	
generation hydro water reservoir	1835.199745	0.00	
generation marine	0.000000	0.00	
generation nuclear	839.667958	0.00	
generation other	20.238381	0.00	
generation other renewable	14.077554	0.00	
generation solar	1680.119887	0.00	
generation waste	50.195536	0.00	
generation wind offshore	0.000000	0.00	
generation wind onshore	3213.691587	0.00	
forecast solar day ahead	1677.703355	0.00	
forecast wind offshore eday ahead	NaN	NaN	
forecast wind onshore day ahead	3176.312853	237.00	
total load forecast	4594.100854	18105.00	
total load actual	4574.987950	18041.00	
price day ahead	14.618900	2.06	
price actual	14.204083	9.33	

	25%	50%	75%	\
generation biomass	333.0000	367.00	433.00	
generation fossil brown coal/lignite	0.0000	509.00	757.00	
generation fossil coal-derived gas	0.0000	0.00	0.00	
generation fossil gas	4126.0000	4969.00	6429.00	
generation fossil hard coal	2527.0000	4474.00	5838.75	
generation fossil oil	263.0000	300.00	330.00	
generation fossil oil shale	0.0000	0.00	0.00	
generation fossil peat	0.0000	0.00	0.00	
generation geothermal	0.0000	0.00	0.00	
generation hydro pumped storage aggregated	NaN	NaN	NaN	
generation hydro pumped storage consumption	0.0000	68.00	616.00	
generation hydro run-of-river and poundage	637.0000	906.00	1250.00	
generation hydro water reservoir	1077.2500	2164.00	3757.00	
generation marine	0.0000	0.00	0.00	
generation nuclear	5760.0000	6566.00	7025.00	

```
generation other                                53.0000     57.00     80.00
generation other renewable                      73.0000     88.00     97.00
generation solar                                71.0000    616.00   2578.00
generation waste                               240.0000    279.00    310.00
generation wind offshore                         0.0000      0.00      0.00
generation wind onshore                       2933.0000   4849.00   7398.00
forecast solar day ahead                        69.0000    576.00   2636.00
forecast wind offshore eday ahead                   NaN       NaN       NaN
forecast wind onshore day ahead               2979.0000   4855.00   7353.00
total load forecast                          24793.7500  28906.00  32263.25
total load actual                            24807.7500  28901.00  32192.00
price day ahead                                 41.4900     50.52     60.53
price actual                                    49.3475     58.02     68.01

                                                    max
generation biomass                               592.00
generation fossil brown coal/lignite             999.00
generation fossil coal-derived gas                 0.00
generation fossil gas                          20034.00
generation fossil hard coal                     8359.00
generation fossil oil                            449.00
generation fossil oil shale                        0.00
generation fossil peat                             0.00
generation geothermal                              0.00
generation hydro pumped storage aggregated          NaN
generation hydro pumped storage consumption     4523.00
generation hydro run-of-river and poundage      2000.00
generation hydro water reservoir                9728.00
generation marine                                  0.00
generation nuclear                              7117.00
generation other                                 106.00
generation other renewable                       119.00
generation solar                                5792.00
generation waste                                 357.00
generation wind offshore                           0.00
generation wind onshore                        17436.00
forecast solar day ahead                        5836.00
forecast wind offshore eday ahead                   NaN
forecast wind onshore day ahead                17430.00
total load forecast                            41390.00
total load actual                              41015.00
price day ahead                                  101.99
price actual                                     116.80
```

[9]:
```
df_energy.info()
```

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

```
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
```

[10]:
```
# columns to be removed due to all 0 or Nan values
col_names = ['generation fossil coal-derived gas', 'generation fossil oil␣
 ↪shale','generation fossil peat',
           'generation geothermal', 'generation hydro pumped storage␣
 ↪aggregated', 'generation marine',
           'generation wind offshore', 'forecast wind offshore eday ahead',␣
 ↪'forecast solar day ahead',
            'forecast wind onshore day ahead']
```

[11]:
```
df_energy = df_energy.drop(col_names, axis = 1)
```

[12]:
```python
def check_Nans_Dups(df):
    """
    This function checks for NaNs and duplicate rows in the dataframe.

    Args:
        df: The input dataframe.

    Returns:
        None. Prints information about NaNs and duplicates.
    """
    # Check for NaNs
    num_nans = df.isna().sum().sum()
    if num_nans > 0:
        print(f"The dataframe has {num_nans} NaN values.")
        print(df.isna().sum())  # Print NaNs per column
    else:
        print("The dataframe has no NaN values.")

    # Check for duplicates
    num_dups = df.duplicated().sum()
    if num_dups > 0:
        print(f"The dataframe has {num_dups} duplicate rows.")
    else:
        print("The dataframe has no duplicate rows.")

# Now you can call the function
check_Nans_Dups(df_energy)
```

```
The dataframe has 292 NaN values.
time                                            0
generation biomass                             19
generation fossil brown coal/lignite           18
generation fossil gas                          18
generation fossil hard coal                    18
generation fossil oil                          19
generation hydro pumped storage consumption    19
generation hydro run-of-river and poundage     19
generation hydro water reservoir               18
generation nuclear                             17
generation other                               18
generation other renewable                     18
generation solar                               18
generation waste                               19
generation wind onshore                        18
total load forecast                             0
total load actual                              36
price day ahead                                 0
```

```
price actual                                    0
dtype: int64
The dataframe has no duplicate rows.
```

[13]:
```
df_energy['time'] = pd.to_datetime(df_energy['time'])
df_energy = df_energy.set_index('time')
df_energy
```

```
<ipython-input-13-c0cd3855ec4e>:1: FutureWarning: In a future version of pandas,
parsing datetimes with mixed time zones will raise an error unless `utc=True`.
Please specify `utc=True` to opt in to the new behaviour and silence this
warning. To create a `Series` with mixed offsets and `object` dtype, please use
`apply` and `datetime.datetime.strptime`
  df_energy['time'] = pd.to_datetime(df_energy['time'])
```

[13]:

time	generation biomass	generation fossil brown coal/lignite	generation fossil gas	generation fossil hard coal
2015-01-01 00:00:00+01:00	447.0	329.0	4844.0	4821.0
2015-01-01 01:00:00+01:00	449.0	328.0	5196.0	4755.0
2015-01-01 02:00:00+01:00	448.0	323.0		
2015-01-01 03:00:00+01:00	438.0	254.0		
2015-01-01 04:00:00+01:00	428.0	187.0		
…	…	…		
2018-12-31 19:00:00+01:00	297.0	0.0		
2018-12-31 20:00:00+01:00	296.0	0.0		
2018-12-31 21:00:00+01:00	292.0	0.0		
2018-12-31 22:00:00+01:00	293.0	0.0		
2018-12-31 23:00:00+01:00	290.0	0.0		

```
2015-01-01 02:00:00+01:00                4857.0                        4581.0
2015-01-01 03:00:00+01:00                4314.0                        4131.0
2015-01-01 04:00:00+01:00                4130.0                        3840.0
…                                         …                            …
2018-12-31 19:00:00+01:00                7634.0                        2628.0
2018-12-31 20:00:00+01:00                7241.0                        2566.0
2018-12-31 21:00:00+01:00                7025.0                        2422.0
2018-12-31 22:00:00+01:00                6562.0                        2293.0
2018-12-31 23:00:00+01:00                6926.0                        2166.0

                          generation fossil oil  \
time
2015-01-01 00:00:00+01:00                 162.0
2015-01-01 01:00:00+01:00                 158.0
2015-01-01 02:00:00+01:00                 157.0
2015-01-01 03:00:00+01:00                 160.0
2015-01-01 04:00:00+01:00                 156.0
…                                          …
2018-12-31 19:00:00+01:00                 178.0
2018-12-31 20:00:00+01:00                 174.0
2018-12-31 21:00:00+01:00                 168.0
2018-12-31 22:00:00+01:00                 163.0
2018-12-31 23:00:00+01:00                 163.0

                          generation hydro pumped storage consumption  \
time
2015-01-01 00:00:00+01:00                                       863.0
2015-01-01 01:00:00+01:00                                       920.0
2015-01-01 02:00:00+01:00                                      1164.0
2015-01-01 03:00:00+01:00                                      1503.0
2015-01-01 04:00:00+01:00                                      1826.0
…                                                               …
2018-12-31 19:00:00+01:00                                         1.0
2018-12-31 20:00:00+01:00                                         1.0
2018-12-31 21:00:00+01:00                                        50.0
2018-12-31 22:00:00+01:00                                       108.0
2018-12-31 23:00:00+01:00                                       108.0

                          generation hydro run-of-river and poundage  \
time
2015-01-01 00:00:00+01:00                                     1051.0
2015-01-01 01:00:00+01:00                                     1009.0
2015-01-01 02:00:00+01:00                                      973.0
2015-01-01 03:00:00+01:00                                      949.0
2015-01-01 04:00:00+01:00                                      953.0
…                                                               …
2018-12-31 19:00:00+01:00                                     1135.0
```

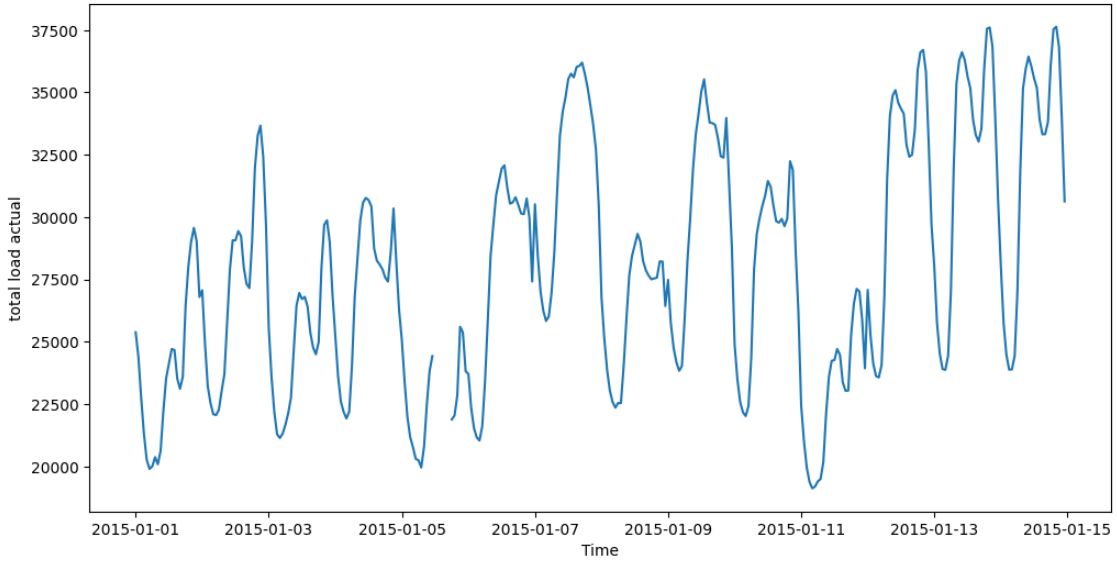
```
2018-12-31 20:00:00+01:00                                    1172.0
2018-12-31 21:00:00+01:00                                    1148.0
2018-12-31 22:00:00+01:00                                    1128.0
2018-12-31 23:00:00+01:00                                    1069.0

                           generation hydro water reservoir  \
time
2015-01-01 00:00:00+01:00                            1899.0
2015-01-01 01:00:00+01:00                            1658.0
2015-01-01 02:00:00+01:00                            1371.0
2015-01-01 03:00:00+01:00                             779.0
2015-01-01 04:00:00+01:00                             720.0
…                                                      …
2018-12-31 19:00:00+01:00                            4836.0
2018-12-31 20:00:00+01:00                            3931.0
2018-12-31 21:00:00+01:00                            2831.0
2018-12-31 22:00:00+01:00                            2068.0
2018-12-31 23:00:00+01:00                            1686.0

                           generation nuclear  generation other  \
time
2015-01-01 00:00:00+01:00              7096.0              43.0
2015-01-01 01:00:00+01:00              7096.0              43.0
2015-01-01 02:00:00+01:00              7099.0              43.0
2015-01-01 03:00:00+01:00              7098.0              43.0
2015-01-01 04:00:00+01:00              7097.0              43.0
…                                       …                 …
2018-12-31 19:00:00+01:00              6073.0              63.0
2018-12-31 20:00:00+01:00              6074.0              62.0
2018-12-31 21:00:00+01:00              6076.0              61.0
2018-12-31 22:00:00+01:00              6075.0              61.0
2018-12-31 23:00:00+01:00              6075.0              61.0

                           generation other renewable  generation solar  \
time
2015-01-01 00:00:00+01:00                        73.0              49.0
2015-01-01 01:00:00+01:00                        71.0              50.0
2015-01-01 02:00:00+01:00                        73.0              50.0
2015-01-01 03:00:00+01:00                        75.0              50.0
2015-01-01 04:00:00+01:00                        74.0              42.0
…                                                 …                 …
2018-12-31 19:00:00+01:00                        95.0              85.0
2018-12-31 20:00:00+01:00                        95.0              33.0
2018-12-31 21:00:00+01:00                        94.0              31.0
2018-12-31 22:00:00+01:00                        93.0              31.0
2018-12-31 23:00:00+01:00                        92.0              31.0
```

```
                           generation waste  generation wind onshore  \
time
2015-01-01 00:00:00+01:00             196.0                   6378.0
2015-01-01 01:00:00+01:00             195.0                   5890.0
2015-01-01 02:00:00+01:00             196.0                   5461.0
2015-01-01 03:00:00+01:00             191.0                   5238.0
2015-01-01 04:00:00+01:00             189.0                   4935.0
…                                         …                       …
2018-12-31 19:00:00+01:00             277.0                   3113.0
2018-12-31 20:00:00+01:00             280.0                   3288.0
2018-12-31 21:00:00+01:00             286.0                   3503.0
2018-12-31 22:00:00+01:00             287.0                   3586.0
2018-12-31 23:00:00+01:00             287.0                   3651.0

                           total load forecast  total load actual  \
time
2015-01-01 00:00:00+01:00              26118.0            25385.0
2015-01-01 01:00:00+01:00              24934.0            24382.0
2015-01-01 02:00:00+01:00              23515.0            22734.0
2015-01-01 03:00:00+01:00              22642.0            21286.0
2015-01-01 04:00:00+01:00              21785.0            20264.0
…                                        …                …
2018-12-31 19:00:00+01:00              30619.0            30653.0
2018-12-31 20:00:00+01:00              29932.0            29735.0
2018-12-31 21:00:00+01:00              27903.0            28071.0
2018-12-31 22:00:00+01:00              25450.0            25801.0
2018-12-31 23:00:00+01:00              24424.0            24455.0

                           price day ahead  price actual
time
2015-01-01 00:00:00+01:00            50.10         65.41
2015-01-01 01:00:00+01:00            48.10         64.92
2015-01-01 02:00:00+01:00            47.33         64.48
2015-01-01 03:00:00+01:00            42.27         59.32
2015-01-01 04:00:00+01:00            38.41         56.04
…                                      …             …
2018-12-31 19:00:00+01:00            68.85         77.02
2018-12-31 20:00:00+01:00            68.40         76.16
2018-12-31 21:00:00+01:00            66.88         74.30
2018-12-31 22:00:00+01:00            63.93         69.89
2018-12-31 23:00:00+01:00            64.27         69.88

[35064 rows x 18 columns]
```

[14]:
```
plt.figure(figsize=(12,6))
# plot total load actual for two weeks duration
plt.plot(df_energy['total load actual'][:24*7*2])
```

```
plt.xlabel('Time')
plt.ylabel('total load actual')
```

[14]: Text(0, 0.5, 'total load actual')



[15]:
```
df_energy[df_energy.isna().any(axis = 1)]
```

[15]:

time	generation biomass
2015-01-05 03:00:00+01:00	NaN
2015-01-05 12:00:00+01:00	NaN
2015-01-05 13:00:00+01:00	NaN
2015-01-05 14:00:00+01:00	NaN
2015-01-05 15:00:00+01:00	NaN
2015-01-05 16:00:00+01:00	NaN
2015-01-05 17:00:00+01:00	NaN
2015-01-19 19:00:00+01:00	NaN
2015-01-19 20:00:00+01:00	NaN
2015-01-27 19:00:00+01:00	NaN
2015-01-28 13:00:00+01:00	NaN
2015-02-01 07:00:00+01:00	449.0
2015-02-01 08:00:00+01:00	453.0
2015-02-01 09:00:00+01:00	452.0
2015-02-01 12:00:00+01:00	405.0
2015-02-01 13:00:00+01:00	402.0
2015-02-01 14:00:00+01:00	400.0
2015-02-01 15:00:00+01:00	393.0
2015-02-01 16:00:00+01:00	413.0

```
2015-02-01 17:00:00+01:00                465.0
2015-02-01 18:00:00+01:00                482.0
2015-02-01 19:00:00+01:00                474.0
2015-04-05 03:00:00+02:00                371.0
2015-04-16 09:00:00+02:00                  NaN
2015-04-20 08:00:00+02:00                424.0
2015-04-23 21:00:00+02:00                  NaN
2015-05-02 10:00:00+02:00                497.0
2015-05-29 03:00:00+02:00                569.0
2015-06-15 09:00:00+02:00                  NaN
2015-10-02 08:00:00+02:00                483.0
2015-10-02 11:00:00+02:00                  NaN
2015-12-02 09:00:00+01:00                  NaN
2016-04-13 05:00:00+02:00                220.0
2016-04-25 05:00:00+02:00                190.0
2016-04-25 07:00:00+02:00                206.0
2016-05-10 23:00:00+02:00                348.0
2016-06-12 01:00:00+02:00                356.0
2016-07-09 22:00:00+02:00                  NaN
2016-07-12 00:00:00+02:00                346.0
2016-09-28 09:00:00+02:00                347.0
2016-10-27 23:00:00+02:00                351.0
2016-11-23 04:00:00+01:00                  NaN
2017-11-14 12:00:00+01:00                  0.0
2017-11-14 19:00:00+01:00                  0.0
2018-06-11 18:00:00+02:00                331.0
2018-07-11 09:00:00+02:00                  NaN

                          generation fossil brown coal/lignite  \
time
2015-01-05 03:00:00+01:00                                   NaN
2015-01-05 12:00:00+01:00                                   NaN
2015-01-05 13:00:00+01:00                                   NaN
2015-01-05 14:00:00+01:00                                   NaN
2015-01-05 15:00:00+01:00                                   NaN
2015-01-05 16:00:00+01:00                                   NaN
2015-01-05 17:00:00+01:00                                   NaN
2015-01-19 19:00:00+01:00                                   NaN
2015-01-19 20:00:00+01:00                                   NaN
2015-01-27 19:00:00+01:00                                   NaN
2015-01-28 13:00:00+01:00                                   NaN
2015-02-01 07:00:00+01:00                                 312.0
2015-02-01 08:00:00+01:00                                 312.0
2015-02-01 09:00:00+01:00                                 302.0
2015-02-01 12:00:00+01:00                                 317.0
2015-02-01 13:00:00+01:00                                 317.0
2015-02-01 14:00:00+01:00                                 317.0
```

2015-02-01 15:00:00+01:00	321.0
2015-02-01 16:00:00+01:00	325.0
2015-02-01 17:00:00+01:00	321.0
2015-02-01 18:00:00+01:00	326.0
2015-02-01 19:00:00+01:00	326.0
2015-04-05 03:00:00+02:00	0.0
2015-04-16 09:00:00+02:00	NaN
2015-04-20 08:00:00+02:00	642.0
2015-04-23 21:00:00+02:00	NaN
2015-05-02 10:00:00+02:00	0.0
2015-05-29 03:00:00+02:00	756.0
2015-06-15 09:00:00+02:00	NaN
2015-10-02 08:00:00+02:00	961.0
2015-10-02 11:00:00+02:00	NaN
2015-12-02 09:00:00+01:00	NaN
2016-04-13 05:00:00+02:00	0.0
2016-04-25 05:00:00+02:00	0.0
2016-04-25 07:00:00+02:00	0.0
2016-05-10 23:00:00+02:00	960.0
2016-06-12 01:00:00+02:00	595.0
2016-07-09 22:00:00+02:00	NaN
2016-07-12 00:00:00+02:00	595.0
2016-09-28 09:00:00+02:00	594.0
2016-10-27 23:00:00+02:00	554.0
2016-11-23 04:00:00+01:00	900.0
2017-11-14 12:00:00+01:00	0.0
2017-11-14 19:00:00+01:00	0.0
2018-06-11 18:00:00+02:00	506.0
2018-07-11 09:00:00+02:00	NaN

time	generation fossil gas	generation fossil hard coal	\
2015-01-05 03:00:00+01:00	NaN	NaN	
2015-01-05 12:00:00+01:00	NaN	NaN	
2015-01-05 13:00:00+01:00	NaN	NaN	
2015-01-05 14:00:00+01:00	NaN	NaN	
2015-01-05 15:00:00+01:00	NaN	NaN	
2015-01-05 16:00:00+01:00	NaN	NaN	
2015-01-05 17:00:00+01:00	NaN	NaN	
2015-01-19 19:00:00+01:00	NaN	NaN	
2015-01-19 20:00:00+01:00	NaN	NaN	
2015-01-27 19:00:00+01:00	NaN	NaN	
2015-01-28 13:00:00+01:00	NaN	NaN	
2015-02-01 07:00:00+01:00	4765.0	5269.0	
2015-02-01 08:00:00+01:00	4938.0	5652.0	
2015-02-01 09:00:00+01:00	4997.0	5770.0	
2015-02-01 12:00:00+01:00	5247.0	6008.0	

2015-02-01 13:00:00+01:00	5449.0	6005.0
2015-02-01 14:00:00+01:00	5266.0	5995.0
2015-02-01 15:00:00+01:00	5209.0	5939.0
2015-02-01 16:00:00+01:00	5642.0	6000.0
2015-02-01 17:00:00+01:00	6127.0	5912.0
2015-02-01 18:00:00+01:00	7386.0	6002.0
2015-02-01 19:00:00+01:00	7963.0	6026.0
2015-04-05 03:00:00+02:00	5015.0	3248.0
2015-04-16 09:00:00+02:00	NaN	NaN
2015-04-20 08:00:00+02:00	5614.0	5784.0
2015-04-23 21:00:00+02:00	NaN	NaN
2015-05-02 10:00:00+02:00	5502.0	5677.0
2015-05-29 03:00:00+02:00	4239.0	4635.0
2015-06-15 09:00:00+02:00	NaN	NaN
2015-10-02 08:00:00+02:00	6545.0	8250.0
2015-10-02 11:00:00+02:00	NaN	NaN
2015-12-02 09:00:00+01:00	NaN	NaN
2016-04-13 05:00:00+02:00	3390.0	1242.0
2016-04-25 05:00:00+02:00	2969.0	886.0
2016-04-25 07:00:00+02:00	3673.0	1143.0
2016-05-10 23:00:00+02:00	6800.0	5219.0
2016-06-12 01:00:00+02:00	5719.0	6165.0
2016-07-09 22:00:00+02:00	NaN	NaN
2016-07-12 00:00:00+02:00	5951.0	6131.0
2016-09-28 09:00:00+02:00	5522.0	6272.0
2016-10-27 23:00:00+02:00	7176.0	5690.0
2016-11-23 04:00:00+01:00	4838.0	4547.0
2017-11-14 12:00:00+01:00	10064.0	0.0
2017-11-14 19:00:00+01:00	12336.0	0.0
2018-06-11 18:00:00+02:00	7538.0	5360.0
2018-07-11 09:00:00+02:00	NaN	NaN

time	generation fossil oil \
2015-01-05 03:00:00+01:00	NaN
2015-01-05 12:00:00+01:00	NaN
2015-01-05 13:00:00+01:00	NaN
2015-01-05 14:00:00+01:00	NaN
2015-01-05 15:00:00+01:00	NaN
2015-01-05 16:00:00+01:00	NaN
2015-01-05 17:00:00+01:00	NaN
2015-01-19 19:00:00+01:00	NaN
2015-01-19 20:00:00+01:00	NaN
2015-01-27 19:00:00+01:00	NaN
2015-01-28 13:00:00+01:00	NaN
2015-02-01 07:00:00+01:00	222.0
2015-02-01 08:00:00+01:00	288.0

```
2015-02-01 09:00:00+01:00                 296.0
2015-02-01 12:00:00+01:00                 333.0
2015-02-01 13:00:00+01:00                 318.0
2015-02-01 14:00:00+01:00                 327.0
2015-02-01 15:00:00+01:00                 345.0
2015-02-01 16:00:00+01:00                 345.0
2015-02-01 17:00:00+01:00                 346.0
2015-02-01 18:00:00+01:00                 340.0
2015-02-01 19:00:00+01:00                 343.0
2015-04-05 03:00:00+02:00                 257.0
2015-04-16 09:00:00+02:00                   NaN
2015-04-20 08:00:00+02:00                 369.0
2015-04-23 21:00:00+02:00                   NaN
2015-05-02 10:00:00+02:00                 375.0
2015-05-29 03:00:00+02:00                 365.0
2015-06-15 09:00:00+02:00                   NaN
2015-10-02 08:00:00+02:00                 385.0
2015-10-02 11:00:00+02:00                   NaN
2015-12-02 09:00:00+01:00                   NaN
2016-04-13 05:00:00+02:00                 243.0
2016-04-25 05:00:00+02:00                 151.0
2016-04-25 07:00:00+02:00                 185.0
2016-05-10 23:00:00+02:00                 299.0
2016-06-12 01:00:00+02:00                 274.0
2016-07-09 22:00:00+02:00                   NaN
2016-07-12 00:00:00+02:00                   NaN
2016-09-28 09:00:00+02:00                 292.0
2016-10-27 23:00:00+02:00                 321.0
2016-11-23 04:00:00+01:00                 269.0
2017-11-14 12:00:00+01:00                   0.0
2017-11-14 19:00:00+01:00                   0.0
2018-06-11 18:00:00+02:00                 300.0
2018-07-11 09:00:00+02:00                   NaN

                          generation hydro pumped storage consumption  \
time
2015-01-05 03:00:00+01:00                                          NaN
2015-01-05 12:00:00+01:00                                          NaN
2015-01-05 13:00:00+01:00                                          NaN
2015-01-05 14:00:00+01:00                                          NaN
2015-01-05 15:00:00+01:00                                          NaN
2015-01-05 16:00:00+01:00                                          NaN
2015-01-05 17:00:00+01:00                                          NaN
2015-01-19 19:00:00+01:00                                          NaN
2015-01-19 20:00:00+01:00                                          NaN
2015-01-27 19:00:00+01:00                                          NaN
2015-01-28 13:00:00+01:00                                          NaN
```

```
2015-02-01 07:00:00+01:00                                    480.0
2015-02-01 08:00:00+01:00                                      0.0
2015-02-01 09:00:00+01:00                                      0.0
2015-02-01 12:00:00+01:00                                      0.0
2015-02-01 13:00:00+01:00                                      0.0
2015-02-01 14:00:00+01:00                                      0.0
2015-02-01 15:00:00+01:00                                      0.0
2015-02-01 16:00:00+01:00                                      0.0
2015-02-01 17:00:00+01:00                                      0.0
2015-02-01 18:00:00+01:00                                      0.0
2015-02-01 19:00:00+01:00                                      0.0
2015-04-05 03:00:00+02:00                                    799.0
2015-04-16 09:00:00+02:00                                      NaN
2015-04-20 08:00:00+02:00                                      0.0
2015-04-23 21:00:00+02:00                                      NaN
2015-05-02 10:00:00+02:00                                      0.0
2015-05-29 03:00:00+02:00                                    755.0
2015-06-15 09:00:00+02:00                                      NaN
2015-10-02 08:00:00+02:00                                      0.0
2015-10-02 11:00:00+02:00                                      NaN
2015-12-02 09:00:00+01:00                                      NaN
2016-04-13 05:00:00+02:00                                   2270.0
2016-04-25 05:00:00+02:00                                   1340.0
2016-04-25 07:00:00+02:00                                    162.0
2016-05-10 23:00:00+02:00                                      0.0
2016-06-12 01:00:00+02:00                                    382.0
2016-07-09 22:00:00+02:00                                      NaN
2016-07-12 00:00:00+02:00                                    494.0
2016-09-28 09:00:00+02:00                                      0.0
2016-10-27 23:00:00+02:00                                      NaN
2016-11-23 04:00:00+01:00                                   1413.0
2017-11-14 12:00:00+01:00                                      0.0
2017-11-14 19:00:00+01:00                                      0.0
2018-06-11 18:00:00+02:00                                      1.0
2018-07-11 09:00:00+02:00                                      NaN

                          generation hydro run-of-river and poundage  \
time
2015-01-05 03:00:00+01:00                                        NaN
2015-01-05 12:00:00+01:00                                        NaN
2015-01-05 13:00:00+01:00                                        NaN
2015-01-05 14:00:00+01:00                                        NaN
2015-01-05 15:00:00+01:00                                        NaN
2015-01-05 16:00:00+01:00                                        NaN
2015-01-05 17:00:00+01:00                                        NaN
2015-01-19 19:00:00+01:00                                        NaN
2015-01-19 20:00:00+01:00                                        NaN
```

```
2015-01-27 19:00:00+01:00                                    NaN
2015-01-28 13:00:00+01:00                                    NaN
2015-02-01 07:00:00+01:00                                  980.0
2015-02-01 08:00:00+01:00                                 1031.0
2015-02-01 09:00:00+01:00                                 1083.0
2015-02-01 12:00:00+01:00                                 1119.0
2015-02-01 13:00:00+01:00                                 1171.0
2015-02-01 14:00:00+01:00                                 1216.0
2015-02-01 15:00:00+01:00                                 1204.0
2015-02-01 16:00:00+01:00                                 1193.0
2015-02-01 17:00:00+01:00                                 1214.0
2015-02-01 18:00:00+01:00                                 1299.0
2015-02-01 19:00:00+01:00                                 1313.0
2015-04-05 03:00:00+02:00                                 1233.0
2015-04-16 09:00:00+02:00                                    NaN
2015-04-20 08:00:00+02:00                                 1122.0
2015-04-23 21:00:00+02:00                                    NaN
2015-05-02 10:00:00+02:00                                 1425.0
2015-05-29 03:00:00+02:00                                  667.0
2015-06-15 09:00:00+02:00                                    NaN
2015-10-02 08:00:00+02:00                                 1323.0
2015-10-02 11:00:00+02:00                                    NaN
2015-12-02 09:00:00+01:00                                    NaN
2016-04-13 05:00:00+02:00                                 1622.0
2016-04-25 05:00:00+02:00                                 1564.0
2016-04-25 07:00:00+02:00                                 1648.0
2016-05-10 23:00:00+02:00                                  443.0
2016-06-12 01:00:00+02:00                                    NaN
2016-07-09 22:00:00+02:00                                    NaN
2016-07-12 00:00:00+02:00                                  709.0
2016-09-28 09:00:00+02:00                                  524.0
2016-10-27 23:00:00+02:00                                  417.0
2016-11-23 04:00:00+01:00                                  795.0
2017-11-14 12:00:00+01:00                                    0.0
2017-11-14 19:00:00+01:00                                    0.0
2018-06-11 18:00:00+02:00                                 1134.0
2018-07-11 09:00:00+02:00                                    NaN

                          generation hydro water reservoir  \
time
2015-01-05 03:00:00+01:00                                NaN
2015-01-05 12:00:00+01:00                                NaN
2015-01-05 13:00:00+01:00                                NaN
2015-01-05 14:00:00+01:00                                NaN
2015-01-05 15:00:00+01:00                                NaN
2015-01-05 16:00:00+01:00                                NaN
2015-01-05 17:00:00+01:00                                NaN
```

```
2015-01-19 19:00:00+01:00                              NaN
2015-01-19 20:00:00+01:00                              NaN
2015-01-27 19:00:00+01:00                              NaN
2015-01-28 13:00:00+01:00                              NaN
2015-02-01 07:00:00+01:00                           1174.0
2015-02-01 08:00:00+01:00                           3229.0
2015-02-01 09:00:00+01:00                           4574.0
2015-02-01 12:00:00+01:00                           4416.0
2015-02-01 13:00:00+01:00                           4475.0
2015-02-01 14:00:00+01:00                           4412.0
2015-02-01 15:00:00+01:00                           3403.0
2015-02-01 16:00:00+01:00                           3333.0
2015-02-01 17:00:00+01:00                           4684.0
2015-02-01 18:00:00+01:00                           6187.0
2015-02-01 19:00:00+01:00                           6895.0
2015-04-05 03:00:00+02:00                           2531.0
2015-04-16 09:00:00+02:00                              NaN
2015-04-20 08:00:00+02:00                           4050.0
2015-04-23 21:00:00+02:00                              NaN
2015-05-02 10:00:00+02:00                           5289.0
2015-05-29 03:00:00+02:00                           1277.0
2015-06-15 09:00:00+02:00                              NaN
2015-10-02 08:00:00+02:00                           5378.0
2015-10-02 11:00:00+02:00                              NaN
2015-12-02 09:00:00+01:00                              NaN
2016-04-13 05:00:00+02:00                           4515.0
2016-04-25 05:00:00+02:00                           5389.0
2016-04-25 07:00:00+02:00                           6807.0
2016-05-10 23:00:00+02:00                           1750.0
2016-06-12 01:00:00+02:00                           1325.0
2016-07-09 22:00:00+02:00                              NaN
2016-07-12 00:00:00+02:00                           1215.0
2016-09-28 09:00:00+02:00                           2494.0
2016-10-27 23:00:00+02:00                           1295.0
2016-11-23 04:00:00+01:00                            435.0
2017-11-14 12:00:00+01:00                              0.0
2017-11-14 19:00:00+01:00                              0.0
2018-06-11 18:00:00+02:00                           4258.0
2018-07-11 09:00:00+02:00                              NaN

                          generation nuclear  generation other  \
time
2015-01-05 03:00:00+01:00                 NaN               NaN
2015-01-05 12:00:00+01:00                 NaN               NaN
2015-01-05 13:00:00+01:00                 NaN               NaN
2015-01-05 14:00:00+01:00                 NaN               NaN
2015-01-05 15:00:00+01:00                 NaN               NaN
```

```
2015-01-05 16:00:00+01:00                NaN               NaN
2015-01-05 17:00:00+01:00                NaN               NaN
2015-01-19 19:00:00+01:00                NaN               NaN
2015-01-19 20:00:00+01:00                NaN               NaN
2015-01-27 19:00:00+01:00                NaN               NaN
2015-01-28 13:00:00+01:00                NaN               NaN
2015-02-01 07:00:00+01:00             7101.0              44.0
2015-02-01 08:00:00+01:00             7099.0              44.0
2015-02-01 09:00:00+01:00             7097.0              43.0
2015-02-01 12:00:00+01:00             7095.0              42.0
2015-02-01 13:00:00+01:00             7096.0              41.0
2015-02-01 14:00:00+01:00             7098.0              42.0
2015-02-01 15:00:00+01:00             7097.0              41.0
2015-02-01 16:00:00+01:00             7097.0              40.0
2015-02-01 17:00:00+01:00             7096.0              41.0
2015-02-01 18:00:00+01:00             7095.0              41.0
2015-02-01 19:00:00+01:00             7096.0              42.0
2015-04-05 03:00:00+02:00             4027.0              81.0
2015-04-16 09:00:00+02:00                NaN               NaN
2015-04-20 08:00:00+02:00             6954.0              41.0
2015-04-23 21:00:00+02:00                NaN               NaN
2015-05-02 10:00:00+02:00             6353.0              93.0
2015-05-29 03:00:00+02:00             5035.0              85.0
2015-06-15 09:00:00+02:00                NaN               NaN
2015-10-02 08:00:00+02:00             7013.0              87.0
2015-10-02 11:00:00+02:00                NaN               NaN
2015-12-02 09:00:00+01:00                NaN               NaN
2016-04-13 05:00:00+02:00             7097.0              53.0
2016-04-25 05:00:00+02:00             7094.0              50.0
2016-04-25 07:00:00+02:00             7095.0              51.0
2016-05-10 23:00:00+02:00             7002.0              50.0
2016-06-12 01:00:00+02:00             5056.0              56.0
2016-07-09 22:00:00+02:00             6923.0               NaN
2016-07-12 00:00:00+02:00             5058.0              49.0
2016-09-28 09:00:00+02:00             6997.0              61.0
2016-10-27 23:00:00+02:00             6967.0              58.0
2016-11-23 04:00:00+01:00             5040.0              60.0
2017-11-14 12:00:00+01:00                0.0               0.0
2017-11-14 19:00:00+01:00                0.0               0.0
2018-06-11 18:00:00+02:00             5856.0              52.0
2018-07-11 09:00:00+02:00                NaN               NaN

                          generation other renewable  generation solar  \
time
2015-01-05 03:00:00+01:00                        NaN               NaN
2015-01-05 12:00:00+01:00                        NaN               NaN
2015-01-05 13:00:00+01:00                        NaN               NaN
```

```
2015-01-05 14:00:00+01:00                        NaN                NaN
2015-01-05 15:00:00+01:00                        NaN                NaN
2015-01-05 16:00:00+01:00                        NaN                NaN
2015-01-05 17:00:00+01:00                        NaN                NaN
2015-01-19 19:00:00+01:00                        NaN                NaN
2015-01-19 20:00:00+01:00                        NaN                NaN
2015-01-27 19:00:00+01:00                        NaN                NaN
2015-01-28 13:00:00+01:00                        NaN                NaN
2015-02-01 07:00:00+01:00                       75.0               48.0
2015-02-01 08:00:00+01:00                       75.0               73.0
2015-02-01 09:00:00+01:00                       71.0              809.0
2015-02-01 12:00:00+01:00                       72.0             3817.0
2015-02-01 13:00:00+01:00                       73.0             3836.0
2015-02-01 14:00:00+01:00                       79.0             3701.0
2015-02-01 15:00:00+01:00                       79.0             3475.0
2015-02-01 16:00:00+01:00                       77.0             2742.0
2015-02-01 17:00:00+01:00                       77.0             1281.0
2015-02-01 18:00:00+01:00                       79.0              328.0
2015-02-01 19:00:00+01:00                       82.0              161.0
2015-04-05 03:00:00+02:00                       67.0               31.0
2015-04-16 09:00:00+02:00                        NaN                NaN
2015-04-20 08:00:00+02:00                       62.0              636.0
2015-04-23 21:00:00+02:00                        NaN                NaN
2015-05-02 10:00:00+02:00                       72.0             2535.0
2015-05-29 03:00:00+02:00                       69.0              662.0
2015-06-15 09:00:00+02:00                        NaN                NaN
2015-10-02 08:00:00+02:00                       70.0              140.0
2015-10-02 11:00:00+02:00                        NaN                NaN
2015-12-02 09:00:00+01:00                        NaN                NaN
2016-04-13 05:00:00+02:00                       69.0              150.0
2016-04-25 05:00:00+02:00                       59.0              454.0
2016-04-25 07:00:00+02:00                       62.0              283.0
2016-05-10 23:00:00+02:00                       91.0               58.0
2016-06-12 01:00:00+02:00                       86.0               30.0
2016-07-09 22:00:00+02:00                        NaN                NaN
2016-07-12 00:00:00+02:00                       83.0               31.0
2016-09-28 09:00:00+02:00                       86.0              982.0
2016-10-27 23:00:00+02:00                       91.0               70.0
2016-11-23 04:00:00+01:00                       85.0               15.0
2017-11-14 12:00:00+01:00                        0.0                0.0
2017-11-14 19:00:00+01:00                        0.0                0.0
2018-06-11 18:00:00+02:00                       96.0              170.0
2018-07-11 09:00:00+02:00                        NaN                NaN

                          generation waste  generation wind onshore  \
time
2015-01-05 03:00:00+01:00              NaN                      NaN
```

```
2015-01-05 12:00:00+01:00                NaN                      NaN
2015-01-05 13:00:00+01:00                NaN                      NaN
2015-01-05 14:00:00+01:00                NaN                      NaN
2015-01-05 15:00:00+01:00                NaN                      NaN
2015-01-05 16:00:00+01:00                NaN                      NaN
2015-01-05 17:00:00+01:00                NaN                      NaN
2015-01-19 19:00:00+01:00                NaN                      NaN
2015-01-19 20:00:00+01:00                NaN                      NaN
2015-01-27 19:00:00+01:00                NaN                      NaN
2015-01-28 13:00:00+01:00                NaN                      NaN
2015-02-01 07:00:00+01:00              208.0                   3289.0
2015-02-01 08:00:00+01:00              207.0                   3102.0
2015-02-01 09:00:00+01:00              204.0                   2838.0
2015-02-01 12:00:00+01:00              200.0                   1413.0
2015-02-01 13:00:00+01:00              193.0                   1347.0
2015-02-01 14:00:00+01:00              199.0                   1345.0
2015-02-01 15:00:00+01:00              204.0                   1487.0
2015-02-01 16:00:00+01:00              203.0                   1648.0
2015-02-01 17:00:00+01:00              207.0                   1857.0
2015-02-01 18:00:00+01:00              208.0                   1864.0
2015-02-01 19:00:00+01:00              207.0                   1813.0
2015-04-05 03:00:00+02:00              142.0                   3153.0
2015-04-16 09:00:00+02:00                NaN                      NaN
2015-04-20 08:00:00+02:00              147.0                    797.0
2015-04-23 21:00:00+02:00                NaN                      NaN
2015-05-02 10:00:00+02:00              205.0                  10903.0
2015-05-29 03:00:00+02:00              201.0                   6503.0
2015-06-15 09:00:00+02:00                NaN                      NaN
2015-10-02 08:00:00+02:00              205.0                   4362.0
2015-10-02 11:00:00+02:00                NaN                      NaN
2015-12-02 09:00:00+01:00                NaN                      NaN
2016-04-13 05:00:00+02:00                NaN                   8596.0
2016-04-25 05:00:00+02:00              195.0                   5989.0
2016-04-25 07:00:00+02:00              214.0                   5682.0
2016-05-10 23:00:00+02:00              280.0                   3311.0
2016-06-12 01:00:00+02:00              291.0                   2019.0
2016-07-09 22:00:00+02:00                NaN                      NaN
2016-07-12 00:00:00+02:00              309.0                   2031.0
2016-09-28 09:00:00+02:00              300.0                   5478.0
2016-10-27 23:00:00+02:00              299.0                   3193.0
2016-11-23 04:00:00+01:00              227.0                   4598.0
2017-11-14 12:00:00+01:00                0.0                      0.0
2017-11-14 19:00:00+01:00                0.0                      0.0
2018-06-11 18:00:00+02:00              269.0                   9165.0
2018-07-11 09:00:00+02:00                NaN                      NaN

                           total load forecast  total load actual  \
```

time		
2015-01-05 03:00:00+01:00	21912.0	21182.0
2015-01-05 12:00:00+01:00	23209.0	NaN
2015-01-05 13:00:00+01:00	23725.0	NaN
2015-01-05 14:00:00+01:00	23614.0	NaN
2015-01-05 15:00:00+01:00	22381.0	NaN
2015-01-05 16:00:00+01:00	21371.0	NaN
2015-01-05 17:00:00+01:00	20760.0	NaN
2015-01-19 19:00:00+01:00	38642.0	39304.0
2015-01-19 20:00:00+01:00	38758.0	39262.0
2015-01-27 19:00:00+01:00	38968.0	38335.0
2015-01-28 13:00:00+01:00	36239.0	NaN
2015-02-01 07:00:00+01:00	24379.0	NaN
2015-02-01 08:00:00+01:00	27389.0	NaN
2015-02-01 09:00:00+01:00	30619.0	NaN
2015-02-01 12:00:00+01:00	31357.0	NaN
2015-02-01 13:00:00+01:00	31338.0	NaN
2015-02-01 14:00:00+01:00	30874.0	NaN
2015-02-01 15:00:00+01:00	30124.0	NaN
2015-02-01 16:00:00+01:00	29714.0	NaN
2015-02-01 17:00:00+01:00	29801.0	NaN
2015-02-01 18:00:00+01:00	32257.0	NaN
2015-02-01 19:00:00+01:00	33183.0	NaN
2015-04-05 03:00:00+02:00	20016.0	NaN
2015-04-16 09:00:00+02:00	31001.0	NaN
2015-04-20 08:00:00+02:00	29287.0	NaN
2015-04-23 21:00:00+02:00	31421.0	NaN
2015-05-02 10:00:00+02:00	39644.0	NaN
2015-05-29 03:00:00+02:00	23132.0	NaN
2015-06-15 09:00:00+02:00	29899.0	30047.0
2015-10-02 08:00:00+02:00	36798.0	NaN
2015-10-02 11:00:00+02:00	38921.0	NaN
2015-12-02 09:00:00+01:00	37413.0	NaN
2016-04-13 05:00:00+02:00	23514.0	23614.0
2016-04-25 05:00:00+02:00	21471.0	NaN
2016-04-25 07:00:00+02:00	27635.0	NaN
2016-05-10 23:00:00+02:00	26641.0	NaN
2016-06-12 01:00:00+02:00	24715.0	24155.0
2016-07-09 22:00:00+02:00	34985.0	NaN
2016-07-12 00:00:00+02:00	25313.0	25103.0
2016-09-28 09:00:00+02:00	31072.0	NaN
2016-10-27 23:00:00+02:00	26423.0	26583.0
2016-11-23 04:00:00+01:00	23469.0	23112.0
2017-11-14 12:00:00+01:00	33805.0	NaN
2017-11-14 19:00:00+01:00	35592.0	NaN
2018-06-11 18:00:00+02:00	34752.0	NaN
2018-07-11 09:00:00+02:00	33938.0	NaN

time	price day ahead	price actual
2015-01-05 03:00:00+01:00	35.20	59.68
2015-01-05 12:00:00+01:00	35.50	79.14
2015-01-05 13:00:00+01:00	36.80	73.95
2015-01-05 14:00:00+01:00	32.50	71.93
2015-01-05 15:00:00+01:00	30.00	71.50
2015-01-05 16:00:00+01:00	30.00	71.85
2015-01-05 17:00:00+01:00	30.60	80.53
2015-01-19 19:00:00+01:00	70.01	88.95
2015-01-19 20:00:00+01:00	69.00	87.94
2015-01-27 19:00:00+01:00	66.00	83.97
2015-01-28 13:00:00+01:00	65.00	77.62
2015-02-01 07:00:00+01:00	56.10	16.98
2015-02-01 08:00:00+01:00	57.69	19.56
2015-02-01 09:00:00+01:00	60.01	23.13
2015-02-01 12:00:00+01:00	59.97	22.51
2015-02-01 13:00:00+01:00	59.69	23.44
2015-02-01 14:00:00+01:00	58.69	24.10
2015-02-01 15:00:00+01:00	58.13	21.12
2015-02-01 16:00:00+01:00	59.00	21.73
2015-02-01 17:00:00+01:00	59.69	25.93
2015-02-01 18:00:00+01:00	63.76	54.13
2015-02-01 19:00:00+01:00	65.01	68.53
2015-04-05 03:00:00+02:00	42.55	29.04
2015-04-16 09:00:00+02:00	56.54	67.55
2015-04-20 08:00:00+02:00	62.00	72.92
2015-04-23 21:00:00+02:00	69.49	82.57
2015-05-02 10:00:00+02:00	58.49	59.09
2015-05-29 03:00:00+02:00	45.93	55.07
2015-06-15 09:00:00+02:00	62.48	73.82
2015-10-02 08:00:00+02:00	66.19	70.13
2015-10-02 11:00:00+02:00	70.09	70.49
2015-12-02 09:00:00+01:00	75.71	80.44
2016-04-13 05:00:00+02:00	18.69	25.14
2016-04-25 05:00:00+02:00	15.00	22.65
2016-04-25 07:00:00+02:00	32.97	40.18
2016-05-10 23:00:00+02:00	51.57	39.11
2016-06-12 01:00:00+02:00	60.23	48.72
2016-07-09 22:00:00+02:00	45.72	51.72
2016-07-12 00:00:00+02:00	64.99	47.49
2016-09-28 09:00:00+02:00	49.72	56.40
2016-10-27 23:00:00+02:00	55.70	62.84
2016-11-23 04:00:00+01:00	43.19	49.11
2017-11-14 12:00:00+01:00	60.53	66.17
2017-11-14 19:00:00+01:00	68.05	75.45

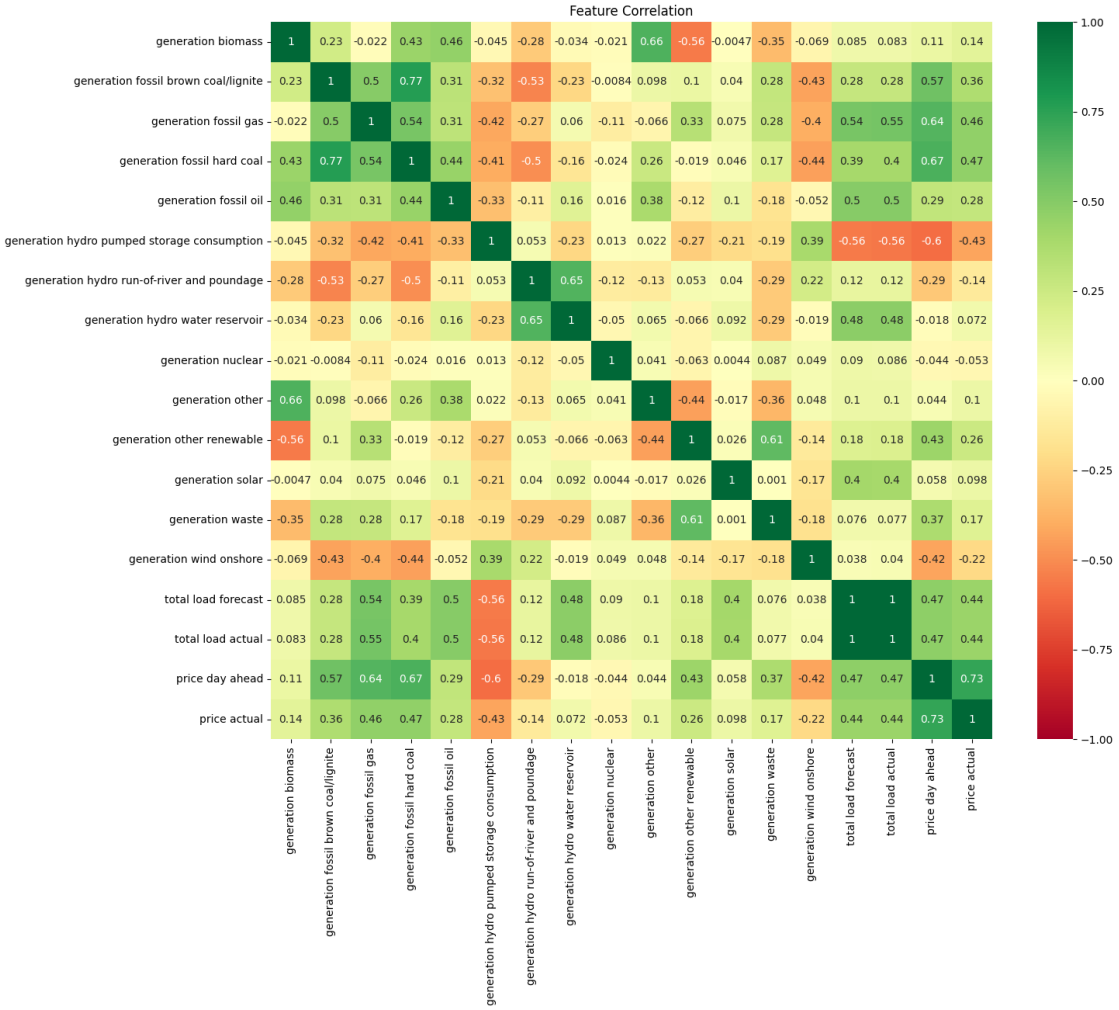
```
2018-06-11 18:00:00+02:00            69.87         64.93
2018-07-11 09:00:00+02:00            63.01         69.79
```

[16]:
```
df_energy.isnull().sum()
```

[16]:
```
generation biomass                             19
generation fossil brown coal/lignite           18
generation fossil gas                          18
generation fossil hard coal                    18
generation fossil oil                          19
generation hydro pumped storage consumption    19
generation hydro run-of-river and poundage     19
generation hydro water reservoir               18
generation nuclear                             17
generation other                               18
generation other renewable                     18
generation solar                               18
generation waste                               19
generation wind onshore                        18
total load forecast                             0
total load actual                              36
price day ahead                                 0
price actual                                    0
dtype: int64
```

[17]:
```
def feat_corr(input_df):
    corr = input_df.corr()
    plt.figure(figsize=(15,12))
    #plot heat map
    g=sns.heatmap(corr,annot=True,cmap="RdYlGn", vmin=-1, vmax=1)
    plt.title('Feature Correlation')

    return plt.show()
```

[18]:
```
feat_corr(df_energy)
```



[19]:
```
file_path = 'weather_features.csv'
df_weather = pd.read_csv(file_path)
```

[20]:
```
df_weather.head()
```

[20]:

	dt_iso	city_name	temp	temp_min	temp_max	pressure	\
0	2015-01-01 00:00:00+01:00	Valencia	270.475	270.475	270.475	1001	
1	2015-01-01 01:00:00+01:00	Valencia	270.475	270.475	270.475	1001	
2	2015-01-01 02:00:00+01:00	Valencia	269.686	269.686	269.686	1002	
3	2015-01-01 03:00:00+01:00	Valencia	269.686	269.686	269.686	1002	
4	2015-01-01 04:00:00+01:00	Valencia	269.686	269.686	269.686	1002	

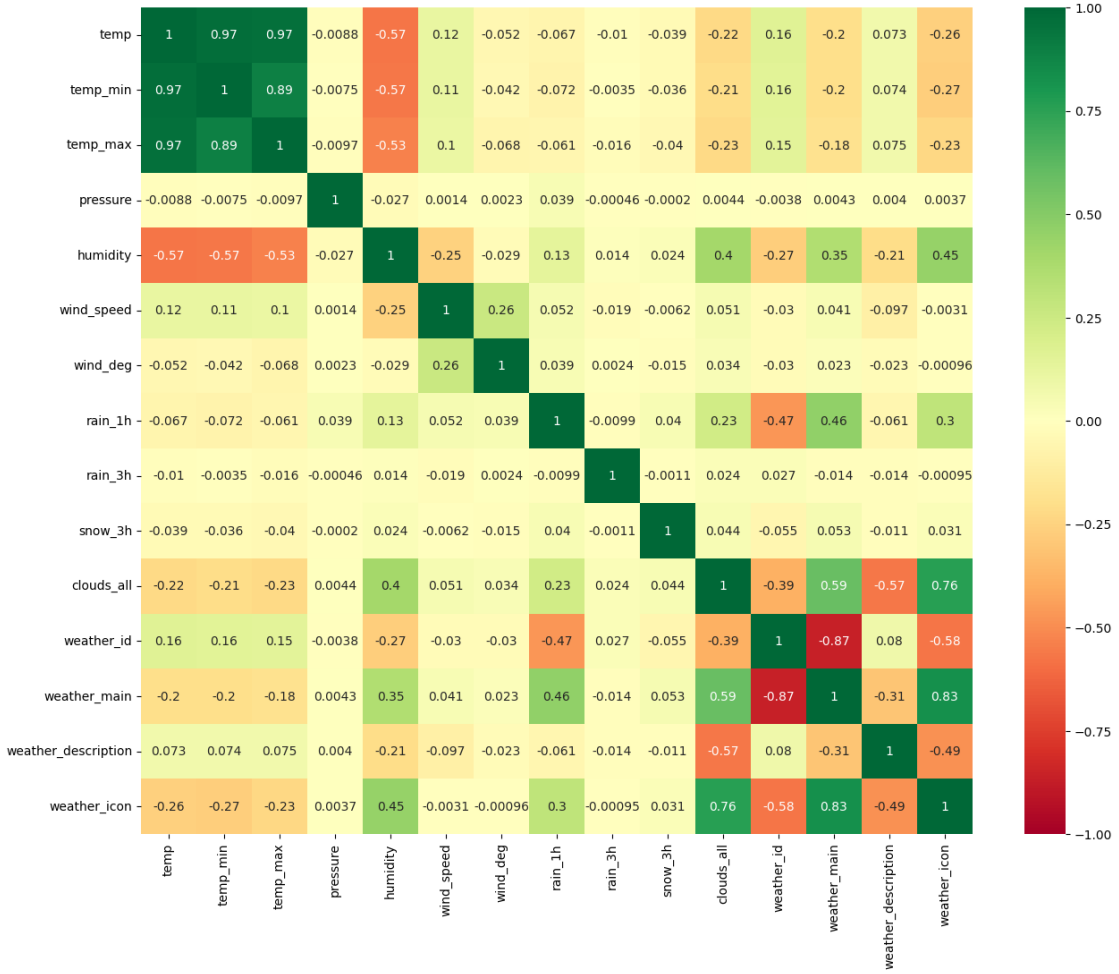
	humidity	wind_speed	wind_deg	rain_1h	rain_3h	snow_3h	clouds_all	\
0	77	1	62	0.0	0.0	0.0	0	
1	77	1	62	0.0	0.0	0.0	0	
2	78	0	23	0.0	0.0	0.0	0	

```
3        78           0        23      0.0      0.0      0.0           0
4        78           0        23      0.0      0.0      0.0           0

   weather_id weather_main weather_description weather_icon
0         800        clear        sky is clear          01n
1         800        clear        sky is clear          01n
2         800        clear        sky is clear          01n
3         800        clear        sky is clear          01n
4         800        clear        sky is clear          01n
```

[21]:
```
df_temp = df_weather.copy(deep = True)
labels = ['weather_id', 'weather_main','weather_description','weather_icon']
for col in labels:
    df_temp[col] = LabelEncoder().fit_transform(df_weather[col])
```

[22]:
```
def feat_corr(input_df):
    # Exclude non-numeric columns before calculating correlation
    numeric_df = input_df.select_dtypes(include=np.number)

    corr = numeric_df.corr()
    plt.figure(figsize=(15, 12))
    # plot heat map
    g = sns.heatmap(corr, annot=True, cmap="RdYlGn", vmin=-1, vmax=1)
    plt.show() # This line is added to display the plot
```

[23]:
```
feat_corr(df_temp)
```



[24]:
```
df_weather['weather_id'].unique()
```

[24]:
```
array([800, 801, 802, 803, 804, 500, 501, 502, 701, 522, 521, 503, 202,
       200, 201, 211, 520, 300, 741, 301, 711, 302, 721, 310, 600, 616,
       615, 601, 210, 602, 611, 311, 612, 620, 531, 731, 761, 771])
```

[25]:
```
col_drop_name = ['weather_id',␣
 ↪'weather_main','weather_description','weather_icon', 'temp_min', 'temp_max']
# col_drop_name = ['weather_id',␣
 ↪'weather_main','weather_description','weather_icon']
df_weather.drop(col_drop_name, axis = 1 , inplace = True)
```

[26]:
```
check_Nans_Dups(df_weather)
```

```
The dataframe has no NaN values.
The dataframe has 3076 duplicate rows.
```

[27]:
```
df_weather = df_weather.reset_index().drop_duplicates()
```

[28]:
```
df_weather['time'] = pd.to_datetime(df_weather['dt_iso'])
df_weather.drop(["dt_iso"] , axis = 1, inplace = True)
df_weather = df_weather.set_index('time')
df_weather.drop(["index"] , axis = 1, inplace = True)
```

```
<ipython-input-28-f2d592ef1b17>:1: FutureWarning: In a future version of pandas,
parsing datetimes with mixed time zones will raise an error unless `utc=True`.
Please specify `utc=True` to opt in to the new behaviour and silence this
warning. To create a `Series` with mixed offsets and `object` dtype, please use
`apply` and `datetime.datetime.strptime`
  df_weather['time'] = pd.to_datetime(df_weather['dt_iso'])
```

[29]:
```
df_weather
```

[29]:

time	city_name	temp	pressure	humidity	wind_speed	wind_deg	rain_1h	rain_3h	snow_3h	clouds_all
2015-01-01 00:00:00+01:00	Valencia	270.475	1001	77	1	62	0.0	0.0	0.0	0
2015-01-01 01:00:00+01:00	Valencia	270.475	1001	77	1	62	0.0	0.0	0.0	0
2015-01-01 02:00:00+01:00	Valencia	269.686	1002	78	0	23	0.0	0.0	0.0	0
2015-01-01 03:00:00+01:00	Valencia	269.686	1002	78	0	23	0.0	0.0	0.0	0
2015-01-01 04:00:00+01:00	Valencia	269.686	1002	78	0	23	0.0	0.0	0.0	0
…	…	…	…	…	…	…	…	…	…	…
2018-12-31 19:00:00+01:00	Seville	287.760	1028	54	3	30	0.0	0.0	0.0	0
2018-12-31 20:00:00+01:00	Seville	285.760	1029	62	3	30	0.0	0.0	0.0	0
2018-12-31 21:00:00+01:00	Seville	285.150	1028	58	4	50	0.0	0.0	0.0	0
2018-12-31 22:00:00+01:00	Seville	284.150	1029	57	4	60	0.0	0.0	0.0	0
2018-12-31 23:00:00+01:00	Seville	283.970	1029	70	3	50	0.0	0.0	0.0	0

[178396 rows x 10 columns]

[30]:
```
df_energy
```

[30]:
```
                           generation biomass  \
time
2015-01-01 00:00:00+01:00              447.0
2015-01-01 01:00:00+01:00              449.0
2015-01-01 02:00:00+01:00              448.0
2015-01-01 03:00:00+01:00              438.0
2015-01-01 04:00:00+01:00              428.0
…                                      …
2018-12-31 19:00:00+01:00              297.0
2018-12-31 20:00:00+01:00              296.0
2018-12-31 21:00:00+01:00              292.0
2018-12-31 22:00:00+01:00              293.0
2018-12-31 23:00:00+01:00              290.0

                           generation fossil brown coal/lignite  \
time
2015-01-01 00:00:00+01:00                                 329.0
2015-01-01 01:00:00+01:00                                 328.0
2015-01-01 02:00:00+01:00                                 323.0
2015-01-01 03:00:00+01:00                                 254.0
2015-01-01 04:00:00+01:00                                 187.0
…                                                         …
2018-12-31 19:00:00+01:00                                   0.0
2018-12-31 20:00:00+01:00                                   0.0
2018-12-31 21:00:00+01:00                                   0.0
2018-12-31 22:00:00+01:00                                   0.0
2018-12-31 23:00:00+01:00                                   0.0

                           generation fossil gas  generation fossil hard coal  \
time
2015-01-01 00:00:00+01:00                 4844.0                        4821.0
2015-01-01 01:00:00+01:00                 5196.0                        4755.0
2015-01-01 02:00:00+01:00                 4857.0                        4581.0
2015-01-01 03:00:00+01:00                 4314.0                        4131.0
2015-01-01 04:00:00+01:00                 4130.0                        3840.0
…                                          …                            …
2018-12-31 19:00:00+01:00                 7634.0                        2628.0
2018-12-31 20:00:00+01:00                 7241.0                        2566.0
2018-12-31 21:00:00+01:00                 7025.0                        2422.0
2018-12-31 22:00:00+01:00                 6562.0                        2293.0
2018-12-31 23:00:00+01:00                 6926.0                        2166.0

                           generation fossil oil  \
time
2015-01-01 00:00:00+01:00                  162.0
2015-01-01 01:00:00+01:00                  158.0
2015-01-01 02:00:00+01:00                  157.0
```

```
2015-01-01 03:00:00+01:00                  160.0
2015-01-01 04:00:00+01:00                  156.0
…                                          …
2018-12-31 19:00:00+01:00                  178.0
2018-12-31 20:00:00+01:00                  174.0
2018-12-31 21:00:00+01:00                  168.0
2018-12-31 22:00:00+01:00                  163.0
2018-12-31 23:00:00+01:00                  163.0

                           generation hydro pumped storage consumption  \
time
2015-01-01 00:00:00+01:00                                          863.0
2015-01-01 01:00:00+01:00                                          920.0
2015-01-01 02:00:00+01:00                                         1164.0
2015-01-01 03:00:00+01:00                                         1503.0
2015-01-01 04:00:00+01:00                                         1826.0
…                                                                  …
2018-12-31 19:00:00+01:00                                            1.0
2018-12-31 20:00:00+01:00                                            1.0
2018-12-31 21:00:00+01:00                                           50.0
2018-12-31 22:00:00+01:00                                          108.0
2018-12-31 23:00:00+01:00                                          108.0

                           generation hydro run-of-river and poundage  \
time
2015-01-01 00:00:00+01:00                                       1051.0
2015-01-01 01:00:00+01:00                                       1009.0
2015-01-01 02:00:00+01:00                                        973.0
2015-01-01 03:00:00+01:00                                        949.0
2015-01-01 04:00:00+01:00                                        953.0
…                                                                 …
2018-12-31 19:00:00+01:00                                       1135.0
2018-12-31 20:00:00+01:00                                       1172.0
2018-12-31 21:00:00+01:00                                       1148.0
2018-12-31 22:00:00+01:00                                       1128.0
2018-12-31 23:00:00+01:00                                       1069.0

                           generation hydro water reservoir  \
time
2015-01-01 00:00:00+01:00                             1899.0
2015-01-01 01:00:00+01:00                             1658.0
2015-01-01 02:00:00+01:00                             1371.0
2015-01-01 03:00:00+01:00                              779.0
2015-01-01 04:00:00+01:00                              720.0
…                                                      …
2018-12-31 19:00:00+01:00                             4836.0
2018-12-31 20:00:00+01:00                             3931.0
```

```
2018-12-31 21:00:00+01:00                          2831.0
2018-12-31 22:00:00+01:00                          2068.0
2018-12-31 23:00:00+01:00                          1686.0

                           generation nuclear  generation other  \
time
2015-01-01 00:00:00+01:00              7096.0              43.0
2015-01-01 01:00:00+01:00              7096.0              43.0
2015-01-01 02:00:00+01:00              7099.0              43.0
2015-01-01 03:00:00+01:00              7098.0              43.0
2015-01-01 04:00:00+01:00              7097.0              43.0
…                                       …                 …
2018-12-31 19:00:00+01:00              6073.0              63.0
2018-12-31 20:00:00+01:00              6074.0              62.0
2018-12-31 21:00:00+01:00              6076.0              61.0
2018-12-31 22:00:00+01:00              6075.0              61.0
2018-12-31 23:00:00+01:00              6075.0              61.0

                           generation other renewable  generation solar  \
time
2015-01-01 00:00:00+01:00                        73.0              49.0
2015-01-01 01:00:00+01:00                        71.0              50.0
2015-01-01 02:00:00+01:00                        73.0              50.0
2015-01-01 03:00:00+01:00                        75.0              50.0
2015-01-01 04:00:00+01:00                        74.0              42.0
…                                                 …                 …
2018-12-31 19:00:00+01:00                        95.0              85.0
2018-12-31 20:00:00+01:00                        95.0              33.0
2018-12-31 21:00:00+01:00                        94.0              31.0
2018-12-31 22:00:00+01:00                        93.0              31.0
2018-12-31 23:00:00+01:00                        92.0              31.0

                           generation waste  generation wind onshore  \
time
2015-01-01 00:00:00+01:00             196.0                   6378.0
2015-01-01 01:00:00+01:00             195.0                   5890.0
2015-01-01 02:00:00+01:00             196.0                   5461.0
2015-01-01 03:00:00+01:00             191.0                   5238.0
2015-01-01 04:00:00+01:00             189.0                   4935.0
…                                      …                      …
2018-12-31 19:00:00+01:00             277.0                   3113.0
2018-12-31 20:00:00+01:00             280.0                   3288.0
2018-12-31 21:00:00+01:00             286.0                   3503.0
2018-12-31 22:00:00+01:00             287.0                   3586.0
2018-12-31 23:00:00+01:00             287.0                   3651.0

                           total load forecast  total load actual  \
```

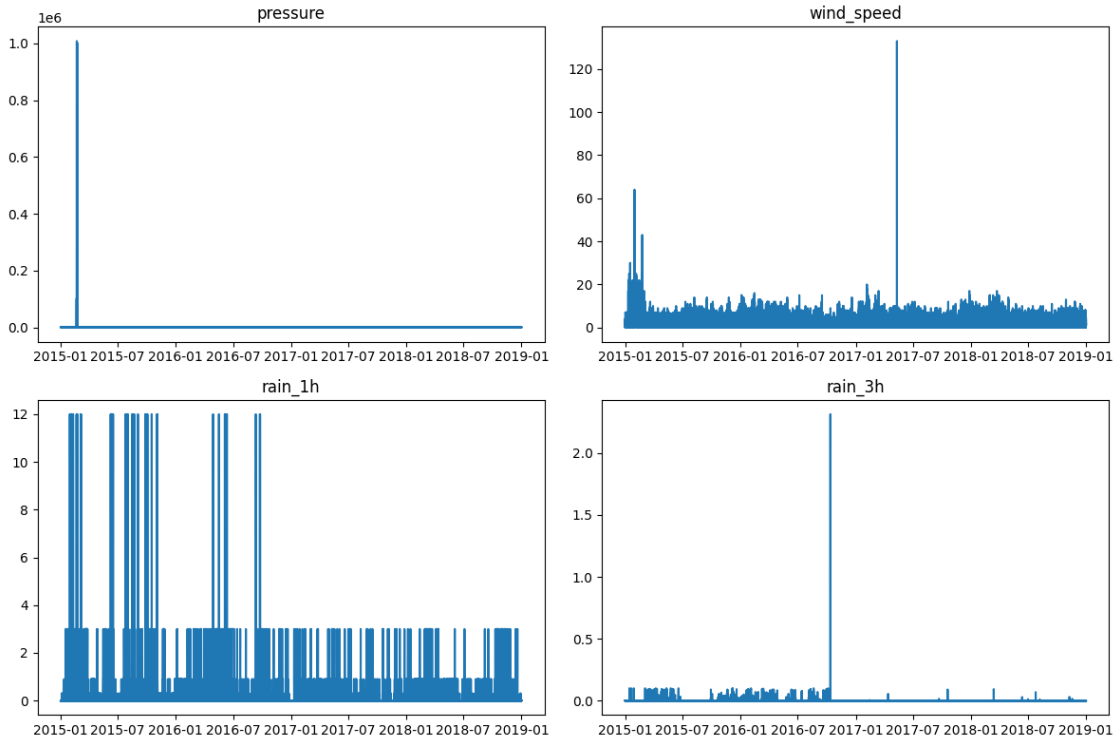
```
time
2015-01-01 00:00:00+01:00              26118.0            25385.0
2015-01-01 01:00:00+01:00              24934.0            24382.0
2015-01-01 02:00:00+01:00              23515.0            22734.0
2015-01-01 03:00:00+01:00              22642.0            21286.0
2015-01-01 04:00:00+01:00              21785.0            20264.0
…                                        …                …
2018-12-31 19:00:00+01:00              30619.0            30653.0
2018-12-31 20:00:00+01:00              29932.0            29735.0
2018-12-31 21:00:00+01:00              27903.0            28071.0
2018-12-31 22:00:00+01:00              25450.0            25801.0
2018-12-31 23:00:00+01:00              24424.0            24455.0

                           price day ahead  price actual
time
2015-01-01 00:00:00+01:00            50.10         65.41
2015-01-01 01:00:00+01:00            48.10         64.92
2015-01-01 02:00:00+01:00            47.33         64.48
2015-01-01 03:00:00+01:00            42.27         59.32
2015-01-01 04:00:00+01:00            38.41         56.04
…                                      …             …
2018-12-31 19:00:00+01:00            68.85         77.02
2018-12-31 20:00:00+01:00            68.40         76.16
2018-12-31 21:00:00+01:00            66.88         74.30
2018-12-31 22:00:00+01:00            63.93         69.89
2018-12-31 23:00:00+01:00            64.27         69.88

[35064 rows x 18 columns]
```

[31]:
```
df_weather.describe().round(2)
```

[31]:
```
              temp    pressure   humidity  wind_speed   wind_deg    rain_1h  \
count  178396.00   178396.00  178396.00   178396.00  178396.00  178396.00
mean      289.62     1069.26      68.42        2.47     166.59       0.08
std         8.03     5969.63      21.90        2.10     116.61       0.40
min       262.24        0.00       0.00        0.00       0.00       0.00
25%       283.67     1013.00      53.00        1.00      55.00       0.00
50%       289.15     1018.00      72.00        2.00     177.00       0.00
75%       295.15     1022.00      87.00        4.00     270.00       0.00
max       315.60  1008371.00     100.00      133.00     360.00      12.00

         rain_3h    snow_3h  clouds_all
count  178396.00  178396.00   178396.00
mean        0.00       0.00       25.07
std         0.01       0.22       30.77
min         0.00       0.00        0.00
25%         0.00       0.00        0.00
```

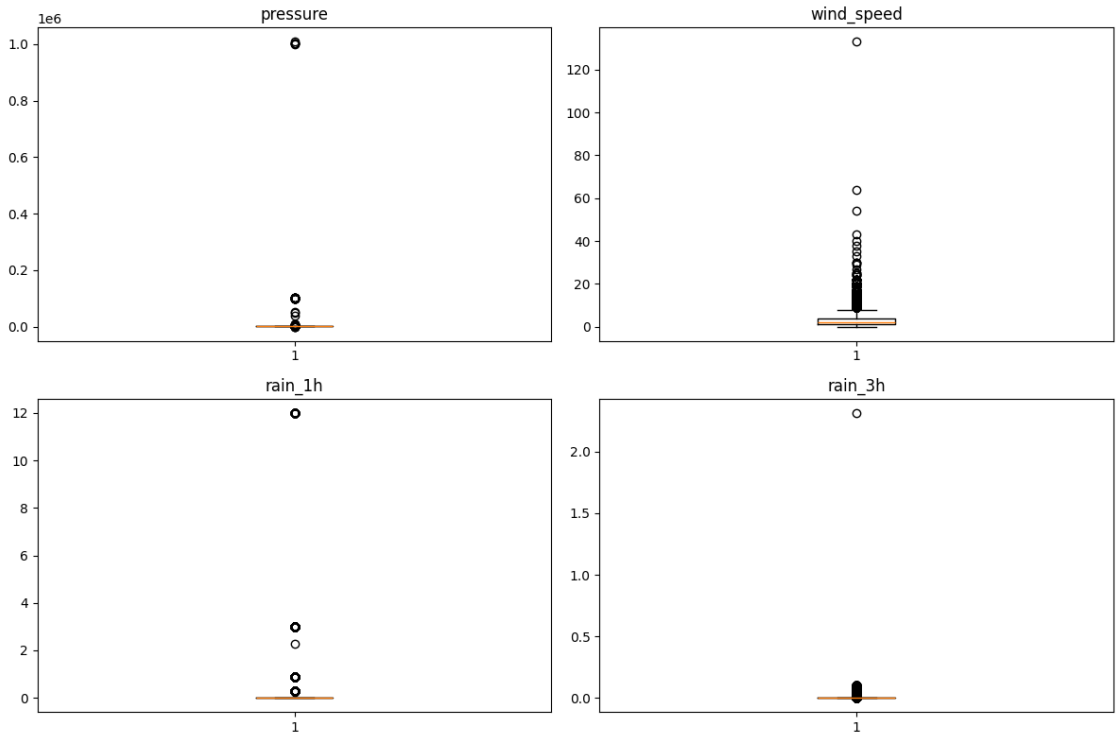
```
50%          0.00        0.00        20.00
75%          0.00        0.00        40.00
max          2.32       21.50       100.00
```

[32]:
```python
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12,8))

# select the columns to plot
columns_to_plot = ['pressure', 'wind_speed', 'rain_1h', 'rain_3h']

# loop through the subplots and plot each column
for i, ax in enumerate(axes.flat):
    if i < len(columns_to_plot):
        ax.plot(df_weather.index, df_weather[columns_to_plot[i]])
        ax.set_title(columns_to_plot[i])
    else:
        ax.set_visible(False)

plt.tight_layout()  # adjust the spacing between subplots
plt.show()  # display the plot
```



[33]:
```python
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12,8))

# select the columns to plot
```

```
columns_to_plot = ['pressure', 'wind_speed', 'rain_1h', 'rain_3h']

# loop through the subplots and plot each column
for i, ax in enumerate(axes.flat):
    if i < len(columns_to_plot):
        ax.boxplot(x = df_weather[columns_to_plot[i]])
        ax.set_title(columns_to_plot[i])
    else:
        ax.set_visible(False)

plt.tight_layout()  # adjust the spacing between subplots
plt.show()  # display the plot
```

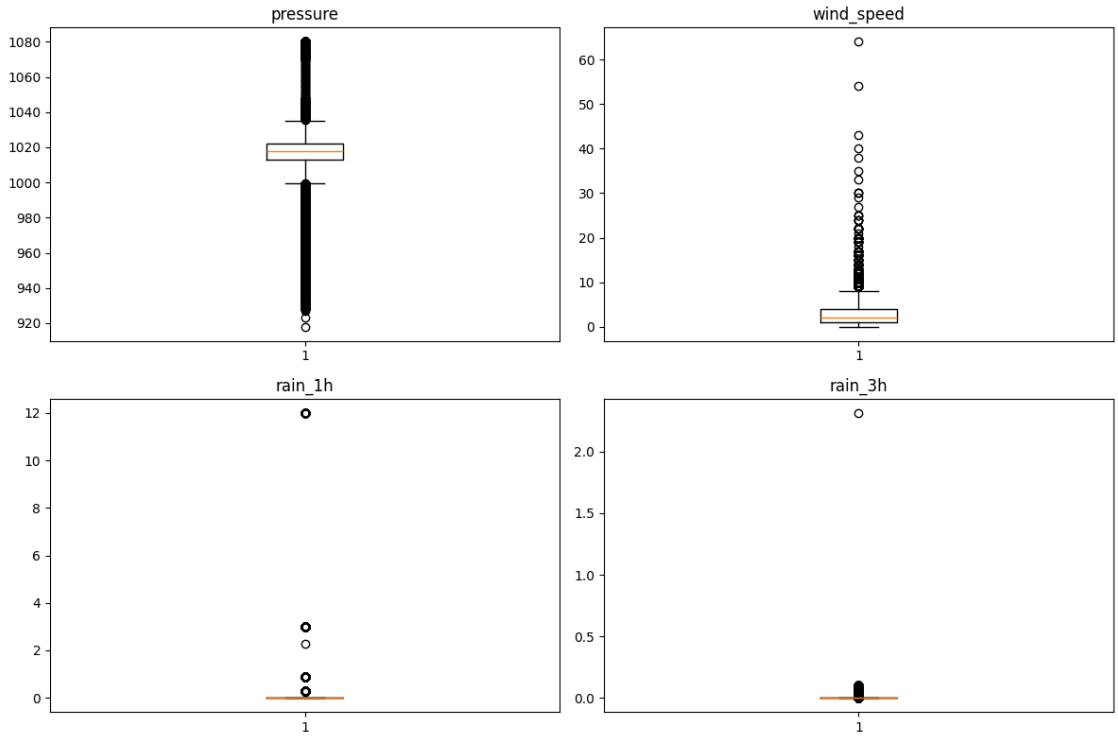


[34]:
```
df_weather.loc[df_weather['pressure']  > 1080,  'pressure'] = np.nan
df_weather.loc[df_weather['pressure']   < 870,  'pressure'] = np.nan
df_weather.loc[df_weather['wind_speed'] > 113, 'wind_speed'] = np.nan

df_weather.interpolate(method ='linear', limit_direction ='forward', inplace =␣
↪True)
```

```
<ipython-input-34-e09d3d1c477d>:5: FutureWarning: DataFrame.interpolate with
object dtype is deprecated and will raise in a future version. Call
obj.infer_objects(copy=False) before interpolating instead.
  df_weather.interpolate(method ='linear', limit_direction ='forward', inplace =
```

True)

[35]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12,8))

# select the columns to plot
columns_to_plot = ['pressure', 'wind_speed', 'rain_1h', 'rain_3h']

# loop through the subplots and plot each column
for i, ax in enumerate(axes.flat):
    if i < len(columns_to_plot):
        ax.boxplot(x = df_weather[columns_to_plot[i]])
        ax.set_title(columns_to_plot[i])
    else:
        ax.set_visible(False)

plt.tight_layout()  # adjust the spacing between subplots
plt.show()  # display the plot
```



[36]:
```
# be sure to drop rain_h3
df_weather.drop(['rain_3h'], axis = 1 , inplace = True)
```

[37]:
```
print(f'Number of samples in df_energy is {df_energy.shape[0]}')

city_list = df_weather['city_name'].unique()
```

```
grouped_weather = df_weather.groupby('city_name')

for city in city_list:
    print(f'Number of samples in df_weather in {city} is {grouped_weather.
 ↪get_group(city).shape[0]}')
```

```
Number of samples in df_energy is 35064
Number of samples in df_weather in Valencia is 35145
Number of samples in df_weather in Madrid is 36267
Number of samples in df_weather in Bilbao is 35951
Number of samples in df_weather in  Barcelona is 35476
Number of samples in df_weather in Seville is 35557
```

[38]:
```
df_weather_cleaned = df_weather.reset_index().drop_duplicates(subset=['time',␣
 ↪'city_name'], keep='first').set_index('time')
```

[39]:
```
df_weather_cleaned
```

[39]:

time	city_name	temp	pressure	humidity	wind_speed	wind_deg	rain_1h	snow_3h	clouds_all
2015-01-01 00:00:00+01:00	Valencia	270.475	1001.0	77	1.0	62	0.0	0.0	0
2015-01-01 01:00:00+01:00	Valencia	270.475	1001.0	77	1.0	62	0.0	0.0	0
2015-01-01 02:00:00+01:00	Valencia	269.686	1002.0	78	0.0	23	0.0	0.0	0
2015-01-01 03:00:00+01:00	Valencia	269.686	1002.0	78	0.0	23	0.0	0.0	0
2015-01-01 04:00:00+01:00	Valencia	269.686	1002.0	78	0.0	23	0.0	0.0	0
…	…	…	…	…	…	…	…	…	…
2018-12-31 19:00:00+01:00	Seville	287.760	1028.0	54	3.0	30	0.0	0.0	0
2018-12-31 20:00:00+01:00	Seville	285.760	1029.0	62	3.0	30	0.0	0.0	0
2018-12-31 21:00:00+01:00	Seville	285.150	1028.0	58	4.0	50	0.0	0.0	0
2018-12-31 22:00:00+01:00	Seville	284.150	1029.0	57	4.0	60	0.0	0.0	0
2018-12-31 23:00:00+01:00	Seville	283.970	1029.0	70	3.0	50	0.0	0.0	0

[175320 rows x 9 columns]

[40]:
```
print(f'Number of samples in df_energy is {df_energy.shape[0]}')

city_list = df_weather['city_name'].unique()
grouped_weather = df_weather_cleaned.groupby('city_name')

for city in city_list:
    print(f'Number of samples in df_weather in {city} is {grouped_weather.
 ↪get_group(city).shape[0]}')
```

```
Number of samples in df_energy is 35064
Number of samples in df_weather in Valencia is 35064
Number of samples in df_weather in Madrid is 35064
Number of samples in df_weather in Bilbao is 35064
Number of samples in df_weather in  Barcelona is 35064
Number of samples in df_weather in Seville is 35064
```

[41]:
```
df_weather_all_cities = [grouped_weather.get_group(x) for x in grouped_weather.
 ↪groups]
```

[42]:
```
df_weather_all_cities[0]
```

[42]:

time	city_name	temp	pressure	humidity	wind_speed	wind_deg	rain_1h	snow_3h	clouds_all
2015-01-01 00:00:00+01:00	Barcelona	281.625	1035.0	100	7.0	58	0.0	0.0	0
2015-01-01 01:00:00+01:00	Barcelona	281.625	1035.0	100	7.0	58	0.0	0.0	0
2015-01-01 02:00:00+01:00	Barcelona	281.286	1036.0	100	7.0	48	0.0	0.0	0
2015-01-01 03:00:00+01:00	Barcelona	281.286	1036.0	100	7.0	48	0.0	0.0	0
2015-01-01 04:00:00+01:00	Barcelona	281.286	1036.0	100	7.0	48	0.0	0.0	0
…	…	…	…	…	…	…	…	…	…
2018-12-31 19:00:00+01:00	Barcelona	284.130	1027.0	71	1.0	250	0.0	0.0	0
2018-12-31 20:00:00+01:00	Barcelona	282.640	1027.0	62	3.0	270	0.0	0.0	0
2018-12-31 21:00:00+01:00	Barcelona	282.140	1028.0	53	4.0	300	0.0	0.0	0
2018-12-31 22:00:00+01:00	Barcelona	281.130	1028.0	50	5.0	320	0.0	0.0	0
2018-12-31 23:00:00+01:00	Barcelona	280.130	1028.0	100					

```
2018-12-31 23:00:00+01:00         5.0       310       0.0       0.0            0

[35064 rows x 9 columns]
```

[43]:
```
df_weather_energy = df_energy

for df_city in df_weather_all_cities:
    city_name = df_city.iloc[0]['city_name'].replace(' ', '')
    df_temp_city = df_city.add_suffix(f'_{city_name}')
    df_weather_energy = pd.concat([df_weather_energy, df_temp_city], axis=1)
    df_weather_energy = df_weather_energy.drop(f'city_name_{city_name}' ,␣
 ↪axis=1)
```

[44]:
```
df_weather_energy.columns
```

[44]:
```
Index(['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', 'total load forecast',
       'total load actual', 'price day ahead', 'price actual',
       'temp_Barcelona', 'pressure_Barcelona', 'humidity_Barcelona',
       'wind_speed_Barcelona', 'wind_deg_Barcelona', 'rain_1h_Barcelona',
       'snow_3h_Barcelona', 'clouds_all_Barcelona', 'temp_Bilbao',
       'pressure_Bilbao', 'humidity_Bilbao', 'wind_speed_Bilbao',
       'wind_deg_Bilbao', 'rain_1h_Bilbao', 'snow_3h_Bilbao',
       'clouds_all_Bilbao', 'temp_Madrid', 'pressure_Madrid',
       'humidity_Madrid', 'wind_speed_Madrid', 'wind_deg_Madrid',
       'rain_1h_Madrid', 'snow_3h_Madrid', 'clouds_all_Madrid', 'temp_Seville',
       'pressure_Seville', 'humidity_Seville', 'wind_speed_Seville',
       'wind_deg_Seville', 'rain_1h_Seville', 'snow_3h_Seville',
       'clouds_all_Seville', 'temp_Valencia', 'pressure_Valencia',
       'humidity_Valencia', 'wind_speed_Valencia', 'wind_deg_Valencia',
       'rain_1h_Valencia', 'snow_3h_Valencia', 'clouds_all_Valencia'],
      dtype='object')
```

[45]:
```
check_Nans_Dups(df_weather_energy)
```

```
The dataframe has 292 NaN values.
generation biomass                            19
generation fossil brown coal/lignite          18
generation fossil gas                         18
generation fossil hard coal                   18
generation fossil oil                         19
generation hydro pumped storage consumption   19
```

```
generation hydro run-of-river and poundage     19
generation hydro water reservoir               18
generation nuclear                             17
generation other                               18
generation other renewable                     18
generation solar                               18
generation waste                               19
generation wind onshore                        18
total load forecast                             0
total load actual                              36
price day ahead                                 0
price actual                                    0
temp_Barcelona                                  0
pressure_Barcelona                              0
humidity_Barcelona                              0
wind_speed_Barcelona                            0
wind_deg_Barcelona                              0
rain_1h_Barcelona                               0
snow_3h_Barcelona                               0
clouds_all_Barcelona                            0
temp_Bilbao                                     0
pressure_Bilbao                                 0
humidity_Bilbao                                 0
wind_speed_Bilbao                               0
wind_deg_Bilbao                                 0
rain_1h_Bilbao                                  0
snow_3h_Bilbao                                  0
clouds_all_Bilbao                               0
temp_Madrid                                     0
pressure_Madrid                                 0
humidity_Madrid                                 0
wind_speed_Madrid                               0
wind_deg_Madrid                                 0
rain_1h_Madrid                                  0
snow_3h_Madrid                                  0
clouds_all_Madrid                               0
temp_Seville                                    0
pressure_Seville                                0
humidity_Seville                                0
wind_speed_Seville                              0
wind_deg_Seville                                0
rain_1h_Seville                                 0
snow_3h_Seville                                 0
clouds_all_Seville                              0
temp_Valencia                                   0
pressure_Valencia                               0
humidity_Valencia                               0
wind_speed_Valencia                             0
```

```
wind_deg_Valencia                            0
rain_1h_Valencia                             0
snow_3h_Valencia                             0
clouds_all_Valencia                          0
dtype: int64
The dataframe has no duplicate rows.
```

[46]:
```
df_weather_energy['hour'] = df_weather_energy.index.map(lambda x : x.hour)
df_weather_energy['weekday'] = df_weather_energy.index.map(lambda x : x.
 ↪weekday())
df_weather_energy['month'] = df_weather_energy.index.map(lambda x : x.month)
df_weather_energy['year'] = df_weather_energy.index.map(lambda x: x.year)
```

[47]:
```
df_weather_energy.columns
```

[47]:
```
Index(['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', 'total load forecast',
       'total load actual', 'price day ahead', 'price actual',
       'temp_Barcelona', 'pressure_Barcelona', 'humidity_Barcelona',
       'wind_speed_Barcelona', 'wind_deg_Barcelona', 'rain_1h_Barcelona',
       'snow_3h_Barcelona', 'clouds_all_Barcelona', 'temp_Bilbao',
       'pressure_Bilbao', 'humidity_Bilbao', 'wind_speed_Bilbao',
       'wind_deg_Bilbao', 'rain_1h_Bilbao', 'snow_3h_Bilbao',
       'clouds_all_Bilbao', 'temp_Madrid', 'pressure_Madrid',
       'humidity_Madrid', 'wind_speed_Madrid', 'wind_deg_Madrid',
       'rain_1h_Madrid', 'snow_3h_Madrid', 'clouds_all_Madrid', 'temp_Seville',
       'pressure_Seville', 'humidity_Seville', 'wind_speed_Seville',
       'wind_deg_Seville', 'rain_1h_Seville', 'snow_3h_Seville',
       'clouds_all_Seville', 'temp_Valencia', 'pressure_Valencia',
       'humidity_Valencia', 'wind_speed_Valencia', 'wind_deg_Valencia',
       'rain_1h_Valencia', 'snow_3h_Valencia', 'clouds_all_Valencia', 'hour',
       'weekday', 'month', 'year'],
      dtype='object')
```

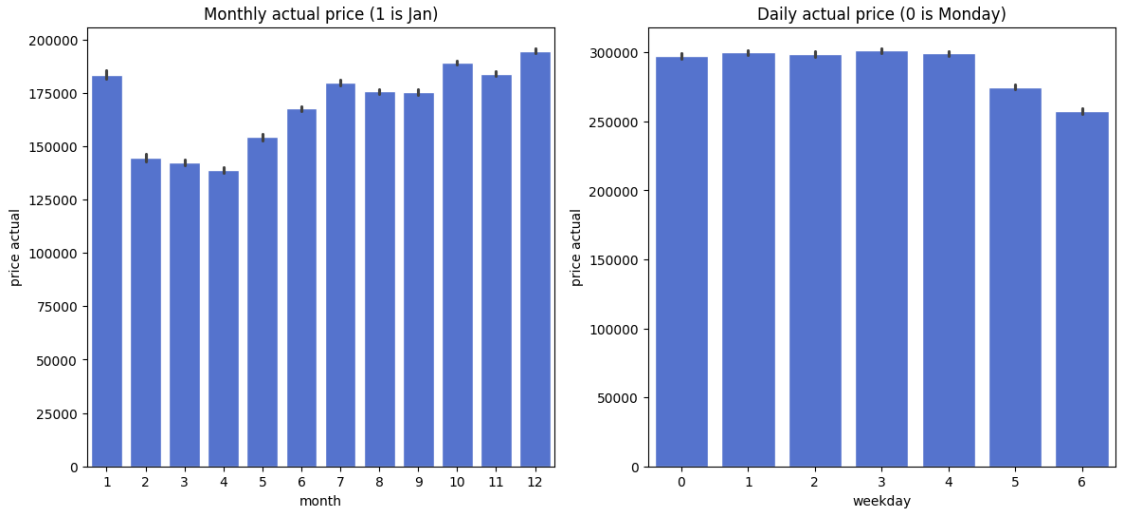
[48]:
```
fig = make_subplots()

fig.add_trace(
    go.Line(x=df_weather_energy.index, y=df_weather_energy["price actual"],
                                name="price actual"))
fig.add_trace(
    go.Line(x=df_weather_energy.index,y=df_weather_energy.rolling(window=24).
 ↪mean()["price actual"],
```

```
                                         name="rolling window = daily ave"))
fig.add_trace(
    go.Line(x=df_weather_energy.index,y=df_weather_energy.rolling(window=24*7).
 ↪mean()["price actual"],
                                         name="rolling window = weekly ave"))
# fig.update_xaxes(rangeslider_visible=True)
fig.show()
```

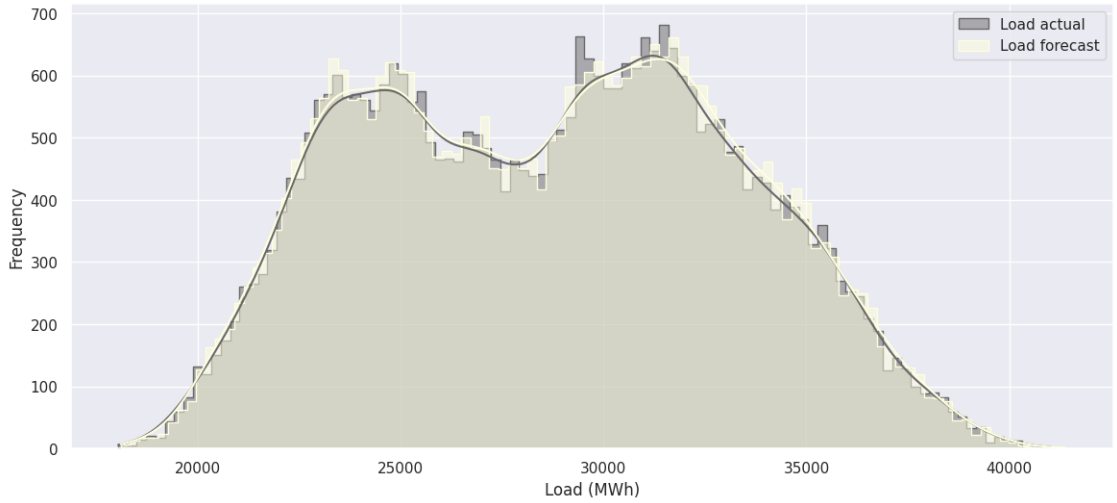
```
/usr/local/lib/python3.10/dist-packages/plotly/graph_objs/_deprecations.py:378:
DeprecationWarning:

plotly.graph_objs.Line is deprecated.
Please replace it with one of the following more specific types
  - plotly.graph_objs.scatter.Line
  - plotly.graph_objs.layout.shape.Line
  - etc.
```

[49]:
```
fig, axes = plt.subplots(ncols=2, figsize=(14, 6))
sns.set(style="darkgrid")

sns.barplot(
    x="month",
    y="price actual",
    data=df_weather_energy,
    estimator=sum,
    color='royalblue',
    ax=axes[0]);
axes[0].set_title('Monthly actual price (1 is Jan)')

sns.barplot(
    x="weekday",
    y="price actual",
    data=df_weather_energy,
    estimator=sum,
    color='royalblue',
    ax=axes[1]);
axes[1].set_title('Daily actual price (0 is Monday)')
```

[49]:
```
Text(0.5, 1.0, 'Daily actual price (0 is Monday)')
```



[50]:
```
plt.figure(figsize=(14,6))

gr = sns.histplot(df_weather_energy['price day ahead'], bins=100, label='TSO␣
 ↪Prediction', element="step", color='lightcoral', kde = True)
gr = sns.histplot(df_weather_energy['price actual'], bins=100, label='Actual␣
 ↪Price', element="step", color='royalblue', kde = True)


gr.set(xlabel="Price of Electricity (€/MWh)", ylabel="Frequency")
plt.legend()
plt.show()

plt.figure(figsize=(14,6))
gr = sns.histplot(df_weather_energy['total load actual'], bins=100, label='Load␣
 ↪actual', element="step", color='dimgrey', kde = True)
gr = sns.histplot(df_weather_energy['total load forecast'], bins=100,␣
 ↪label='Load forecast', element="step", color='lightyellow', kde = True)

gr.set(xlabel="Load (MWh)", ylabel="Frequency")
plt.legend()
plt.show()
```



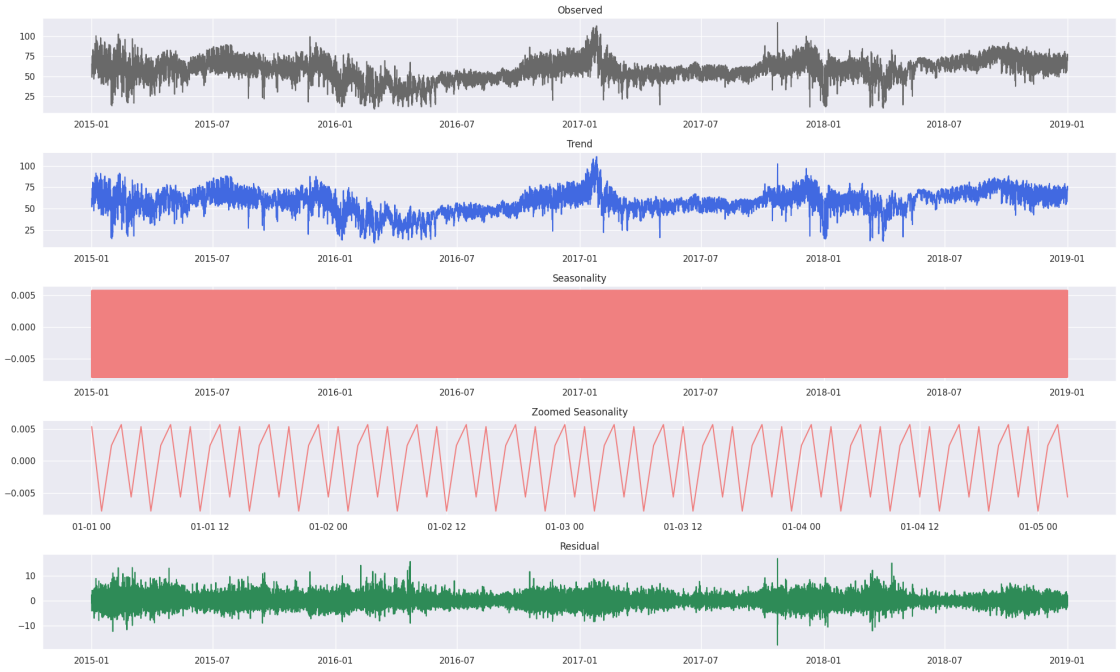
[51]:
```
y_scaler_actual = MinMaxScaler()
y_scaler_dayahead = MinMaxScaler()

train_cutoff = int(0.8*df_weather_energy.shape[0])
val_cutoff = int(0.9*df_weather_energy.shape[0])

y_price_actual   = df_weather_energy[['price actual']]
y_price_dayahead = df_weather_energy[['price day ahead']]

y_scaler_actual.fit(y_price_actual[:train_cutoff])
actual_norm = y_scaler_actual.transform(y_price_actual)

y_scaler_dayahead.fit(df_weather_energy[['price day ahead']][:train_cutoff])
```

```
dayahead_norm = y_scaler_dayahead.transform(y_price_dayahead)

print(f' mean absolute error for normalized acutal price and TSO predcition is :
  ↪ {round(mean_absolute_error(actual_norm, dayahead_norm),3)}')
```

```
 mean absolute error for normalized acutal price and TSO predcition is : 0.071
```

[52]:
```
df_weather_energy.drop(['total load forecast'], axis = 1, inplace = True)
```

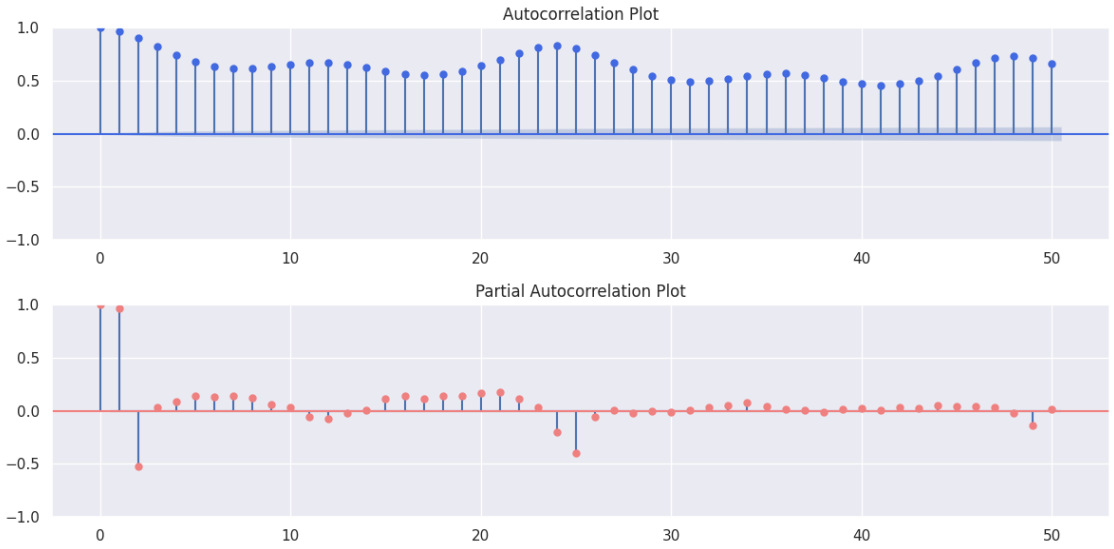
[53]:
```
fig, axes = plt.subplots(5, 1, figsize=(20, 12))
decom_data = df_weather_energy[['price actual']].copy()

decompose_result = seasonal_decompose(decom_data, period =5, model='additive')

observed    = decompose_result.observed
trend    = decompose_result.trend
seasonal = decompose_result.seasonal
residual = decompose_result.resid

axes[0].plot(observed, color='dimgrey')
axes[0].set_title('Observed')
axes[1].plot(trend, color='royalblue')
axes[1].set_title('Trend')
axes[2].plot(seasonal, color='lightcoral')
axes[2].set_title('Seasonality')
axes[3].plot(seasonal[:100], color='lightcoral')
axes[3].set_title('Zoomed Seasonality')
axes[4].plot(residual, color='seagreen')
axes[4].set_title('Residual')

fig.tight_layout()
plt.show()
```



[54]:
```
result = adfuller(df_weather_energy[['price actual']])
print('ADF Statistic:', result[0])
print('p-value:', result[1])
print('Critical Values:', result[4])
```

```
ADF Statistic: -9.147016232851248
p-value: 2.750493484933306e-15
Critical Values: {'1%': -3.4305367814665044, '5%': -2.8616225527935106, '10%':
-2.566813940257257}
```

[55]:
```
fig, ax = plt.subplots(2, 1, figsize=(12, 6))
plot_acf(df_weather_energy['price actual'], lags=50, ax=ax[0],␣
 ↪color='royalblue')
ax[0].set_title('Autocorrelation Plot')
plot_pacf(df_weather_energy['price actual'], lags=50, ax=ax[1],␣
 ↪color='lightcoral')
ax[1].set_title('Partial Autocorrelation Plot')
plt.tight_layout()
plt.show()
```



[56]:
```
#DIMENSIONALITY REDUCTION

X = df_weather_energy.drop(['price actual'], axis = 1)
y= df_weather_energy[['price actual']]
```

[57]:
```
def apply_PCA(X_input, cum_variance, if_apply):

    if if_apply:

        pca = PCA(n_components = cum_variance)
        # make pipeline to first standardize then apply PCA on data
        scaler_pca = make_pipeline(MinMaxScaler(), pca)
        X_pca = scaler_pca.fit(X_input).transform(X_input)

        return X_pca

    else:

        return np.array(X_input)
```

[58]:
```
!pip install scikit-learn
import numpy as np
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.impute import SimpleImputer  # Import SimpleImputer

def apply_PCA(X_input, cum_variance, if_apply):
```

```
    if if_apply:
        # Create an imputer to replace NaN with the mean of the column
        imputer = SimpleImputer(strategy='mean')

        # Apply imputation to the input data
        X_input_imputed = imputer.fit_transform(X_input)

        pca = PCA(n_components=cum_variance)
        # make pipeline to first standardize then apply PCA on data
        scaler_pca = make_pipeline(MinMaxScaler(), pca)

        # Apply the pipeline to the imputed data
        X_pca = scaler_pca.fit(X_input_imputed).transform(X_input_imputed)

        return X_pca

    else:

        return np.array(X_input)
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (1.5.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
```

[59]:
```
params_pca = {'cum_variance' : 0.8, 'if_apply' : True }
X_pca = apply_PCA(X, **params_pca)
X_pca.shape
```

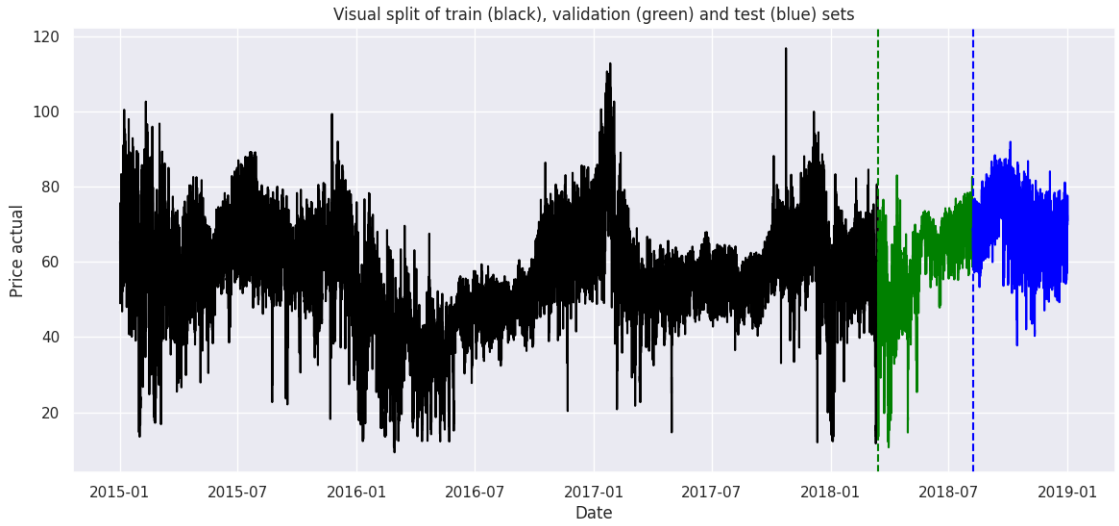
[59]:
```
(35064, 15)
```

[60]:
```
def windowing(X_input,y_input, history_size):

    data = []
    labels = []
    for i in range(history_size, len(y_input)):
        data.append(X_input[i - history_size : i, :])
        labels.append(y_input[i])

    return np.array(data), np.array(labels).reshape(-1,1)
```

[61]:
```
train_cutoff = int(0.8*X_pca.shape[0])
val_cutoff   = int(0.9*X_pca.shape[0])

scaler_y = MinMaxScaler()
scaler_y.fit(y[:train_cutoff])
y_norm = scaler_y.transform(y)
```

[62]:
```
hist_size= 24
data_norm = np.concatenate((X_pca,y_norm), axis = 1)

X_train, y_train = windowing(data_norm[:train_cutoff,:],data_norm[:
 ↪train_cutoff,-1], hist_size)
X_val, y_val     = windowing(data_norm[train_cutoff :val_cutoff,:
 ↪],data_norm[train_cutoff:val_cutoff,-1], hist_size)
X_test, y_test   = windowing(data_norm[val_cutoff :,:],data_norm[val_cutoff:
 ↪,-1], hist_size)
```

[63]:
```
fig, axes = plt.subplots(figsize = (14,6))
axes.plot(df_weather_energy['price actual'].iloc[:train_cutoff], color =␣
 ↪'black')
axes.plot(df_weather_energy['price actual'].iloc[train_cutoff + 1 :␣
 ↪val_cutoff], color = 'green')
axes.plot(df_weather_energy['price actual'].iloc[val_cutoff + 1 :], color =␣
 ↪'blue')
axes.axvline(x=df_weather_energy.index[train_cutoff], color='green',␣
 ↪linestyle='--')
axes.axvline(x=df_weather_energy.index[val_cutoff], color='blue',␣
 ↪linestyle='--')
axes.set_title('Visual split of train (black), validation (green) and test␣
 ↪(blue) sets')
axes.set_xlabel('Date')
axes.set_ylabel('Price actual')
plt.show()
```



[64]:
```
from tensorflow.keras.optimizers import Adam # Importing the Adam optimizer

def base_model_cnn():

    model = Sequential()
    model.add(Conv1D(filters=32, kernel_size=3, activation='relu',␣
 ↪input_shape=X_train.shape[-2:]))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.1))
    model.add(Dense(1))

    return model

cnn_model = base_model_cnn()
optimizer = Adam() # Assigning Adam optimizer to the variable 'optimizer'
cnn_model.compile(optimizer = optimizer , loss = 'mean_absolute_error')
cnn_model.summary()
```

```
/usr/local/lib/python3.10/dist-
packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning:

Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in
the model instead.


Model: "sequential"
```

```
Layer (type)                         Output Shape                ␣
↪Param #

conv1d (Conv1D)                      (None, 22, 32)               ␣
↪1,568

flatten (Flatten)                    (None, 704)                  ␣
↪  0

dense (Dense)                        (None, 128)                  ␣
↪90,240

dropout (Dropout)                    (None, 128)                  ␣
↪  0

dense_1 (Dense)                      (None, 1)                    ␣
↪129


Total params: 91,937 (359.13 KB)

Trainable params: 91,937 (359.13 KB)

Non-trainable params: 0 (0.00 B)
```

[65]:
```
# Define the 'epoch' and 'batch_size' variables
epoch = 40  # Example: Set the number of epochs to 10
batch_size = 128  # Example: Set the batch size to 32
# Define the callback, or remove it if not needed
from tensorflow.keras.callbacks import EarlyStopping
callback = EarlyStopping(monitor='val_loss', patience=40) # Example early␣
 ↪stopping callback

# Now run the fit method
history = cnn_model.fit(X_train, y_train, validation_data=(X_val, y_val),
                    epochs=epoch, batch_size=batch_size, callbacks=[callback])
```

```
Epoch 1/40
219/219             4s 8ms/step -
loss: 0.0974 - val_loss: 0.0532
Epoch 2/40
219/219             2s 7ms/step -
loss: 0.0482 - val_loss: 0.0495
Epoch 3/40
```

```
219/219            1s 7ms/step -
loss: 0.0403 - val_loss: 0.0383
Epoch 4/40
219/219            3s 7ms/step -
loss: 0.0369 - val_loss: 0.0381
Epoch 5/40
219/219            1s 6ms/step -
loss: 0.0335 - val_loss: 0.0315
Epoch 6/40
219/219            4s 12ms/step -
loss: 0.0319 - val_loss: 0.0390
Epoch 7/40
219/219            4s 6ms/step -
loss: 0.0300 - val_loss: 0.0390
Epoch 8/40
219/219            3s 7ms/step -
loss: 0.0285 - val_loss: 0.0364
Epoch 9/40
219/219            2s 6ms/step -
loss: 0.0277 - val_loss: 0.0425
Epoch 10/40
219/219            1s 3ms/step -
loss: 0.0265 - val_loss: 0.0390
Epoch 11/40
219/219            2s 6ms/step -
loss: 0.0258 - val_loss: 0.0321
Epoch 12/40
219/219            1s 4ms/step -
loss: 0.0246 - val_loss: 0.0342
Epoch 13/40
219/219            1s 3ms/step -
loss: 0.0235 - val_loss: 0.0306
Epoch 14/40
219/219            1s 4ms/step -
loss: 0.0230 - val_loss: 0.0253
Epoch 15/40
219/219            1s 4ms/step -
loss: 0.0231 - val_loss: 0.0275
Epoch 16/40
219/219            1s 4ms/step -
loss: 0.0221 - val_loss: 0.0323
Epoch 17/40
219/219            1s 4ms/step -
loss: 0.0216 - val_loss: 0.0326
Epoch 18/40
219/219            1s 4ms/step -
loss: 0.0211 - val_loss: 0.0248
Epoch 19/40
```

```
219/219            1s 4ms/step -
loss: 0.0205 - val_loss: 0.0250
Epoch 20/40
219/219            1s 4ms/step -
loss: 0.0199 - val_loss: 0.0258
Epoch 21/40
219/219            1s 4ms/step -
loss: 0.0195 - val_loss: 0.0270
Epoch 22/40
219/219            2s 6ms/step -
loss: 0.0195 - val_loss: 0.0247
Epoch 23/40
219/219            2s 4ms/step -
loss: 0.0191 - val_loss: 0.0252
Epoch 24/40
219/219            1s 4ms/step -
loss: 0.0190 - val_loss: 0.0274
Epoch 25/40
219/219            1s 4ms/step -
loss: 0.0185 - val_loss: 0.0291
Epoch 26/40
219/219            1s 4ms/step -
loss: 0.0182 - val_loss: 0.0252
Epoch 27/40
219/219            1s 3ms/step -
loss: 0.0180 - val_loss: 0.0244
Epoch 28/40
219/219            1s 4ms/step -
loss: 0.0180 - val_loss: 0.0238
Epoch 29/40
219/219            1s 4ms/step -
loss: 0.0177 - val_loss: 0.0248
Epoch 30/40
219/219            1s 4ms/step -
loss: 0.0172 - val_loss: 0.0218
Epoch 31/40
219/219            1s 3ms/step -
loss: 0.0176 - val_loss: 0.0235
Epoch 32/40
219/219            2s 6ms/step -
loss: 0.0171 - val_loss: 0.0227
Epoch 33/40
219/219            2s 3ms/step -
loss: 0.0170 - val_loss: 0.0246
Epoch 34/40
219/219            1s 4ms/step -
loss: 0.0172 - val_loss: 0.0215
Epoch 35/40
```

```
219/219             1s 4ms/step -
loss: 0.0169 - val_loss: 0.0211
Epoch 36/40
219/219             1s 4ms/step -
loss: 0.0172 - val_loss: 0.0219
Epoch 37/40
219/219             1s 4ms/step -
loss: 0.0168 - val_loss: 0.0205
Epoch 38/40
219/219             1s 4ms/step -
loss: 0.0164 - val_loss: 0.0205
Epoch 39/40
219/219             1s 4ms/step -
loss: 0.0161 - val_loss: 0.0185
Epoch 40/40
219/219             1s 4ms/step -
loss: 0.0162 - val_loss: 0.0217
```

[68]:
```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import mean_absolute_error

def plot_results(y_pred, y_true, history, model_name):
    fig, axes = plt.subplots(figsize=(12, 8), nrows=2, ncols=1)

    # Predictions vs Actual Values
    axes[0].plot(y_true, label='Actual')
    axes[0].plot(y_pred, label='Predictions')
    axes[0].set_title(f'{model_name} Predictions vs Actual Values')
    axes[0].set_xlabel('Time')
    axes[0].set_ylabel('Value')
    axes[0].legend()

    # Training History
    if history is not None:  # Check if history is provided
        axes[1].plot(history.history['loss'], label='Training Loss')
        axes[1].plot(history.history['val_loss'], label='Validation Loss')
        axes[1].set_title(f'{model_name} Training History')
        axes[1].set_xlabel('Epoch')
        axes[1].set_ylabel('Loss')
        axes[1].legend()
    plt.tight_layout()

# Predict on the test set and inverse transform to get actual values
y_pred = cnn_model.predict(X_test)
y_pred_actual = scaler_y.inverse_transform(y_pred)  # Inverse transform␣
 ↪predictions
```

```
y_test_inv = scaler_y.inverse_transform(y_test)       # Inverse transform actual␣
 ↪values

print('')
print('')
print('-----------------------------------------------------')
print(f'CNN MAE for test set :␣
 ↪{round(mean_absolute_error(y_pred_actual,y_test_inv),3)}') # Use␣
 ↪y_pred_actual and y_test_inv
print('-----------------------------------------------------')
print('')
plot_results(y_pred_actual, y_test_inv, history,'CNN') # Use y_pred_actual and␣
 ↪y_test_inv
```

```
109/109             0s 1ms/step


-----------------------------------------------------
CNN MAE for test set : 2.643
-----------------------------------------------------
```