Hourly energy demand generation and weather_r0

March 9, 2021

1 Forecasting The Spanish Electricity Power Generation Load and Cost of MWH USING ARIMA and DEEP Leering - TensorFlow LSTM

Analysis and Model Development by Tamer Hanna

1.1 Abstract

The objective of this project is to perform various Time Series Analysis and modeling to understand the Power generation market of Spain and to be able to predict the hourly load and use it for the prediction of the Euro per Megawatt hour.

Several open datasets from different websites are used in this analysis.

1- The Power Generation dataset that includes all the different sources of generation (hourly dataset for records between 31st Dec. 2014 to 31st Dec 2018 1,016,305 records of data)

2- Weather conditions in Spain (Temperature, Humidity, Pressure, Rain, clouds) for 5 big cities (hourly dataset for weather records between 31st Dec. 2014 to 31st Dec 2018 3,032,732 for ['Valencia' 'Madrid' 'Bilbao' ' Barcelona' 'Seville'])

3- Global price of Coal monthly average for the period between 2015 and 2020 in USD (49 records)

4- History of Henry Hub natural gas prices in USD for the last 10 years (3290 records)

5- West Texas Intermediate (WTI or NYMEX) crude oil prices per barrel from 2008 to 2020 in USD (3268 records)

6- Euro Dollar Daily Exchange Rate (EUR USD) - for the period starting from 1999 to 2020 (5825 records)

Datasets links

https://www.kaggle.com/nicholasjhana/energy-consumption-generation-prices-and-weather

https://www.macrotrends.net/1369/crude-oil-price-history-chart

https://www.macrotrends.net/2478/natural-gas-prices-historical-chart

https://www.macrotrends.net/2548/euro-dollar-exchange-rate-historical-chart

https://fred.stlouisfed.org/series/PCOALAUUSDM#0

1.2 Introduction

In this notebook I am exploring the Power Generation sector in Spain and what is affecting the price of MWH. the models built is to allow the privet power generation plants to be able to bid on the generation through forecasting the hourly load and the cost We are trying to predict the hourly load based on the hourly history of using ARIMA model and the cost of the MWH in EUROS using Tensor flow- LSTM deep learning model

The Electrical Energy Generation Price depends on several factors that affects the cost built in it, and it is divided into Fixed cost and variable costs that is based on several inputs as :

1- The Load or the required electric consumption which is affected by several factors as the weak days and hours, weather conditions, population density.

2- The Generation Facility running costs in terms of Operations and maintenance and initial capital invested in it

3- The Fuel cost that is changing from day to day.

4- The transmission cost.

The Datasets we are using should be able to help in forecasting the price range and the load with the all the variables and history of data provided. The Energy dataset is composed of energy demand, The Generation load from different sources and price of MWH and forecasted price. The dataset is unique because it contains hourly data for electrical consumption and the respective forecasts for consumption and pricing. This allows prospective forecasts to be benchmarked against the current state of the art forecasts being used in industry. Based on the various approaches to implementing a time series application. By adding the market prices change of the fuels used in generation and the exchange rate from USD to Euros this can add more forecasting power to the models for forecasting the price

1.3 Ethical ML Framework

1.3.1 Data Governance:

As the goal of this report is only to research the time series methods, many aspects of the ethical ML framework do not directly apply. The dataset used in this report are accessible to the public websites for the study and analysis of machine learning technics and not for any business use and does not require any approval or request to use it for this reason as stated by the publishers
1- ENTSOE(European Network of Transmission System Operators for Electricity), a public portal for Transmission Service Operator (TSO) data. Settlement prices were obtained from the Spanish TSO Red Electric Espana. 2- Weather data is available on Kaggle by owner and open to public study use. 3- Prices of Fuels and exchange rate are available and open for public use are from Micro trends and Fred Economic data research

This data is used as provided on the websites without any certificates or confirmation of the accuracy of and is not tested or verified by any sort test or expert or compared to any source of data through calibrated devices. However, since this data will be used for the discovery of the machine learning tools and techniques and not to drive any decision related to Energy usage or cost analysis then these verifications are not required for the purpose of this report

1.3.2 Accuracy/Trust of the model:

The output models and analysis of this report are not intended to be put in use or even to be used to give any advice related to the field or energy usage or environment affect or socioeconomic studies The data and the analysis are manly done for the exploration of the Data Science and may be some assumptions and interpolations are done that will affect the result of the analysis

1.3.3 Social Impact

Since this report is focused on exploring the Machine learning tools and technics the inputs, outputs and results are not verified and could have a negative affect or biases on the community. It also should not be used in any decisions that could have a Social Impact or through businesses operations and decisions related to sustainability in the Energy sector.

In case of applying the same techniques in life scenario, the machine learning techniques, data inputs and results should be assessed against the ML framework for the Social Impact, Accuracy/Trust and Governance

[705]:
```
# Libraries used
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.api import VAR
from statsmodels.tsa.arima_model import ARIMA
import statsmodels.api as sm
import tensorflow as tf
import xgboost as xgb
import warnings
from tensorflow.keras.layers import Dense, LSTM, Conv1D, MaxPooling1D,␣
 ↪TimeDistributed, Flatten, Dropout
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller, kpss
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
import math
from math import sqrt
from pylab import rcParams
import itertools
from statsmodels.tsa.api import VAR
from random import randrange
from pandas import Series
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

[706]:
```
# Importing the datasets
```

```
energy = pd.read_csv('C:/Users/16472/OneDrive/Documents/Data analytics/Advanced␣
↪course Big data/Course 2/Project 3/dataset/energy_dataset.csv')
```

[707]:
```
energy.head()
```

[707]:
```
                        time  generation biomass  \
0  2015-01-01 00:00:00+01:00               447.0
1  2015-01-01 01:00:00+01:00               449.0
2  2015-01-01 02:00:00+01:00               448.0
3  2015-01-01 03:00:00+01:00               438.0
4  2015-01-01 04:00:00+01:00               428.0

   generation fossil brown coal/lignite  generation fossil coal-derived gas  \
0                                 329.0                                 0.0
1                                 328.0                                 0.0
2                                 323.0                                 0.0
3                                 254.0                                 0.0
4                                 187.0                                 0.0

   generation fossil gas  generation fossil hard coal  generation fossil oil  \
0                 4844.0                       4821.0                  162.0
1                 5196.0                       4755.0                  158.0
2                 4857.0                       4581.0                  157.0
3                 4314.0                       4131.0                  160.0
4                 4130.0                       3840.0                  156.0

   generation fossil oil shale  generation fossil peat  generation geothermal  \
0                          0.0                     0.0                    0.0
1                          0.0                     0.0                    0.0
2                          0.0                     0.0                    0.0
3                          0.0                     0.0                    0.0
4                          0.0                     0.0                    0.0

   …  generation waste  generation wind offshore  generation wind onshore  \
0  …             196.0                       0.0                   6378.0
1  …             195.0                       0.0                   5890.0
2  …             196.0                       0.0                   5461.0
3  …             191.0                       0.0                   5238.0
4  …             189.0                       0.0                   4935.0

   forecast solar day ahead  forecast wind offshore eday ahead  \
0                      17.0                                NaN
1                      16.0                                NaN
2                       8.0                                NaN
3                       2.0                                NaN
4                       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]
```

[708]:
```
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
```

[709]:
```
weather = pd.read_csv('C:/Users/16472/OneDrive/Documents/Data analytics/
↪Advanced course Big data/Course 2/Project 3/dataset/weather_features.csv')
```

[710]:
```
weather.head(10)
```

[710]:
```
                      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
5  2015-01-01 05:00:00+01:00  Valencia  270.292   270.292   270.292      1004
6  2015-01-01 06:00:00+01:00  Valencia  270.292   270.292   270.292      1004
7  2015-01-01 07:00:00+01:00  Valencia  270.292   270.292   270.292      1004
8  2015-01-01 08:00:00+01:00  Valencia  274.601   274.601   274.601      1005
9  2015-01-01 09:00:00+01:00  Valencia  274.601   274.601   274.601      1005

   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
5        71           2       321      0.0      0.0      0.0           0
6        71           2       321      0.0      0.0      0.0           0
7        71           2       321      0.0      0.0      0.0           0
8        71           1       307      0.0      0.0      0.0           0
9        71           1       307      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
5         800        clear        sky is clear          01n
6         800        clear        sky is clear          01n
7         800        clear        sky is clear          01n
8         800        clear        sky is clear          01d
9         800        clear        sky is clear          01d
```

[711]:
```
weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178396 entries, 0 to 178395
Data columns (total 17 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   dt_iso               178396 non-null  object
 1   city_name            178396 non-null  object
 2   temp                 178396 non-null  float64
 3   temp_min             178396 non-null  float64
 4   temp_max             178396 non-null  float64
 5   pressure             178396 non-null  int64
 6   humidity             178396 non-null  int64
 7   wind_speed           178396 non-null  int64
 8   wind_deg             178396 non-null  int64
 9   rain_1h              178396 non-null  float64
 10  rain_3h              178396 non-null  float64
 11  snow_3h              178396 non-null  float64
 12  clouds_all           178396 non-null  int64
 13  weather_id           178396 non-null  int64
 14  weather_main         178396 non-null  object
 15  weather_description  178396 non-null  object
 16  weather_icon         178396 non-null  object
dtypes: float64(6), int64(6), object(5)
memory usage: 23.1+ MB
```

[712]:
```
oil_price = pd.read_csv('C:/Users/16472/OneDrive/Documents/Data analytics/
↪Advanced course Big data/Course 2/Project 3/dataset/Crude oil price.csv')
```

[713]:
```
oil_price.head()
```

[713]:

	date	value
0	2008-03-07	105.12
1	2008-03-10	107.90
2	2008-03-11	108.73
3	2008-03-12	109.86
4	2008-03-13	110.21

[714]:
```
oil_price.describe()
```

[714]:

	value
count	3268.000000
mean	70.731513
std	24.436998
min	11.258000
25%	49.970000
50%	67.245000

```
75%      92.845000
max     145.310000
```

[715]:
```
ntrl_gas_price = pd.read_csv('C:/Users/16472/OneDrive/Documents/Data analytics/
↪Advanced course Big data/Course 2/Project 3/dataset/natural gas prices.csv')
```

[716]:
```
ntrl_gas_price.head()
```

[716]:

	date	value
0	2008-03-07	9.82
1	2008-03-10	9.59
2	2008-03-11	9.85
3	2008-03-12	9.69
4	2008-03-13	9.74

[717]:
```
coal_price = pd.read_csv('C:/Users/16472/OneDrive/Documents/Data analytics/
↪Advanced course Big data/Course 2/Project 3/dataset/Coal_prices.csv')
```

[718]:
```
coal_price.head()
```

[718]:

	time	coal_price
0	2015-01-01	64.716327
1	2015-02-01	70.659107
2	2015-03-01	68.344968
3	2015-04-01	61.197857
4	2015-05-01	65.671241

[719]:
```
xchg_rate = pd.read_csv('C:/Users/16472/OneDrive/Documents/Data analytics/
↪Advanced course Big data/Course 2/Project 3/dataset/euro_dollar_xch.csv')
```

[720]:
```
xchg_rate.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5825 entries, 0 to 5824
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    5825 non-null   object
 1    value  5825 non-null   float64
dtypes: float64(1), object(1)
memory usage: 91.1+ KB
```

[721]:
```
parse_dates=[energy.time]
parse_dates=[weather.dt_iso]
parse_dates=[oil_price.date]
parse_dates=[ntrl_gas_price.date]
parse_dates=[xchg_rate.date]
```

[722]:
```
energy.describe()
```

[722]:
```
       generation biomass  generation fossil brown coal/lignite  \
count        35045.000000                          35046.000000
mean           383.513540                            448.059208
std             85.353943                            354.568590
min              0.000000                              0.000000
25%            333.000000                              0.000000
50%            367.000000                            509.000000
75%            433.000000                            757.000000
max            592.000000                            999.000000

       generation fossil coal-derived gas  generation fossil gas  \
count                             35046.0           35046.000000
mean                                  0.0            5622.737488
std                                   0.0            2201.830478
min                                   0.0               0.000000
25%                                   0.0            4126.000000
50%                                   0.0            4969.000000
75%                                   0.0            6429.000000
max                                   0.0           20034.000000

       generation fossil hard coal  generation fossil oil  \
count                 35046.000000           35045.000000
mean                   4256.065742             298.319789
std                    1961.601013              52.520673
min                       0.000000               0.000000
25%                    2527.000000             263.000000
50%                    4474.000000             300.000000
75%                    5838.750000             330.000000
max                    8359.000000             449.000000

       generation fossil oil shale  generation fossil peat  \
count                      35046.0                 35046.0
mean                           0.0                     0.0
std                            0.0                     0.0
min                            0.0                     0.0
25%                            0.0                     0.0
50%                            0.0                     0.0
75%                            0.0                     0.0
max                            0.0                     0.0

       generation geothermal  generation hydro pumped storage aggregated  …  \
count                35046.0                                          0.0  …
mean                     0.0                                          NaN  …
std                      0.0                                          NaN  …
min                      0.0                                          NaN  …
```

```
25%                  0.0                                        NaN  ...
50%                  0.0                                        NaN  ...
75%                  0.0                                        NaN  ...
max                  0.0                                        NaN  ...

       generation waste  generation wind offshore  generation wind onshore  \
count      35045.000000                   35046.0             35046.000000
mean         269.452133                       0.0              5464.479769
std           50.195536                       0.0              3213.691587
min            0.000000                       0.0                 0.000000
25%          240.000000                       0.0              2933.000000
50%          279.000000                       0.0              4849.000000
75%          310.000000                       0.0              7398.000000
max          357.000000                       0.0             17436.000000

       forecast solar day ahead  forecast wind offshore eday ahead  \
count              35064.000000                                0.0
mean                1439.066735                                NaN
std                 1677.703355                                NaN
min                    0.000000                                NaN
25%                   69.000000                                NaN
50%                  576.000000                                NaN
75%                 2636.000000                                NaN
max                 5836.000000                                NaN

       forecast wind onshore day ahead  total load forecast  \
count                     35064.000000         35064.000000
mean                       5471.216689         28712.129962
std                        3176.312853          4594.100854
min                         237.000000         18105.000000
25%                        2979.000000         24793.750000
50%                        4855.000000         28906.000000
75%                        7353.000000         32263.250000
max                       17430.000000         41390.000000

       total load actual  price day ahead  price actual
count       35028.000000     35064.000000  35064.000000
mean        28696.939905        49.874341     57.884023
std          4574.987950        14.618900     14.204083
min         18041.000000         2.060000      9.330000
25%         24807.750000        41.490000     49.347500
50%         28901.000000        50.520000     58.020000
75%         32192.000000        60.530000     68.010000
max         41015.000000       101.990000    116.800000

[8 rows x 28 columns]
```

[723]:
```
weather.describe()
```

[723]:
```
                temp       temp_min       temp_max      pressure  \
count  178396.000000  178396.000000  178396.000000  1.783960e+05
mean      289.618605     288.330442     291.091267  1.069261e+03
std         8.026199       7.955491       8.612454  5.969632e+03
min       262.240000     262.240000     262.240000  0.000000e+00
25%       283.670000     282.483602     284.650000  1.013000e+03
50%       289.150000     288.150000     290.150000  1.018000e+03
75%       295.150000     293.730125     297.150000  1.022000e+03
max       315.600000     315.150000     321.150000  1.008371e+06

            humidity    wind_speed       wind_deg        rain_1h  \
count  178396.000000  178396.00000  178396.000000  178396.000000
mean       68.423457       2.47056     166.591190       0.075492
std        21.902888       2.09591     116.611927       0.398847
min         0.000000       0.00000       0.000000       0.000000
25%        53.000000       1.00000      55.000000       0.000000
50%        72.000000       2.00000     177.000000       0.000000
75%        87.000000       4.00000     270.000000       0.000000
max       100.000000     133.00000     360.000000      12.000000

             rain_3h        snow_3h     clouds_all     weather_id
count  178396.000000  178396.000000  178396.000000  178396.000000
mean        0.000380       0.004763      25.073292     759.831902
std         0.007288       0.222604      30.774129     108.733223
min         0.000000       0.000000       0.000000     200.000000
25%         0.000000       0.000000       0.000000     800.000000
50%         0.000000       0.000000      20.000000     800.000000
75%         0.000000       0.000000      40.000000     801.000000
max         2.315000      21.500000     100.000000     804.000000
```

1.4 Data Cleaning

1.4.1 Energy Genration dataset Cleaning

[586]:
```
#Data Cleaning : Checking for null values
energy.isnull().sum()
```

[586]:
```
time                                            0
generation biomass                             19
generation fossil brown coal/lignite           18
generation fossil coal-derived gas             18
generation fossil gas                          18
generation fossil hard coal                    18
generation fossil oil                          19
generation fossil oil shale                    18
generation fossil peat                         18
```

```
generation geothermal                           18
generation hydro pumped storage aggregated   35064
generation hydro pumped storage consumption     19
generation hydro run-of-river and poundage      19
generation hydro water reservoir                18
generation marine                               19
generation nuclear                              17
generation other                                18
generation other renewable                      18
generation solar                                18
generation waste                                19
generation wind offshore                        18
generation wind onshore                         18
forecast solar day ahead                         0
forecast wind offshore eday ahead            35064
forecast wind onshore day ahead                  0
total load forecast                              0
total load actual                               36
price day ahead                                  0
price actual                                     0
dtype: int64
```

We found 2 Important findings from the above 2 steps

1- The following columns has 0 values for all cells 'generation fossil coal-derived gas' , 'generation fossil oil shale', generation fossil peat', 'generation geothermal','generation wind offshore','

2- ' generation hydro pumped storage aggregated' and 'forecast wind offshore eday ahead' are all null value so we will drop these 2 coulmns and the rest of null values we willl will look at them later

We will have to drop all these columns from the dataset

[587]:
```
# Drop empty columns
energy.drop(columns=['generation hydro pumped storage aggregated',
                     'forecast wind offshore eday ahead','generation fossil␣
 ↪coal-derived gas',
                     'generation fossil oil shale',
                     'generation fossil peat',
                     'generation geothermal',
                     'generation marine',
                     'generation wind offshore'], inplace =True)
```

[588]:
```
#Checking datatypes
energy.describe()
```

[588]:
```
       generation biomass  generation fossil brown coal/lignite  \
count        35045.000000                           35046.000000
mean           383.513540                             448.059208
```

```
std          85.353943                                354.568590
min           0.000000                                  0.000000
25%         333.000000                                  0.000000
50%         367.000000                                509.000000
75%         433.000000                                757.000000
max         592.000000                                999.000000

       generation fossil gas  generation fossil hard coal  \
count           35046.000000                 35046.000000
mean             5622.737488                  4256.065742
std              2201.830478                  1961.601013
min                 0.000000                     0.000000
25%              4126.000000                  2527.000000
50%              4969.000000                  4474.000000
75%              6429.000000                  5838.750000
max             20034.000000                  8359.000000

       generation fossil oil  generation hydro pumped storage consumption  \
count           35045.000000                                 35045.000000
mean              298.319789                                   475.577343
std                52.520673                                   792.406614
min                 0.000000                                     0.000000
25%               263.000000                                     0.000000
50%               300.000000                                    68.000000
75%               330.000000                                   616.000000
max               449.000000                                  4523.000000

       generation hydro run-of-river and poundage  \
count                                35045.000000
mean                                   972.116108
std                                    400.777536
min                                      0.000000
25%                                    637.000000
50%                                    906.000000
75%                                   1250.000000
max                                   2000.000000

       generation hydro water reservoir  generation nuclear  generation other  \
count                      35046.000000        35047.000000      35046.000000
mean                        2605.114735         6263.907039         60.228585
std                         1835.199745          839.667958         20.238381
min                            0.000000            0.000000          0.000000
25%                         1077.250000         5760.000000         53.000000
50%                         2164.000000         6566.000000         57.000000
75%                         3757.000000         7025.000000         80.000000
max                         9728.000000         7117.000000        106.000000
```

```
       generation other renewable  generation solar  generation waste  \
count                35046.000000      35046.000000      35045.000000
mean                    85.639702       1432.665925        269.452133
std                     14.077554       1680.119887         50.195536
min                      0.000000          0.000000          0.000000
25%                     73.000000         71.000000        240.000000
50%                     88.000000        616.000000        279.000000
75%                     97.000000       2578.000000        310.000000
max                    119.000000       5792.000000        357.000000

       generation wind onshore  forecast solar day ahead  \
count             35046.000000              35064.000000
mean               5464.479769               1439.066735
std                3213.691587               1677.703355
min                   0.000000                  0.000000
25%                2933.000000                 69.000000
50%                4849.000000                576.000000
75%                7398.000000               2636.000000
max               17436.000000               5836.000000

       forecast wind onshore day ahead  total load forecast  \
count                     35064.000000         35064.000000
mean                       5471.216689         28712.129962
std                        3176.312853          4594.100854
min                         237.000000         18105.000000
25%                        2979.000000         24793.750000
50%                        4855.000000         28906.000000
75%                        7353.000000         32263.250000
max                       17430.000000         41390.000000

       total load actual  price day ahead  price actual
count       35028.000000     35064.000000  35064.000000
mean        28696.939905        49.874341     57.884023
std          4574.987950        14.618900     14.204083
min         18041.000000         2.060000      9.330000
25%         24807.750000        41.490000     49.347500
50%         28901.000000        50.520000     58.020000
75%         32192.000000        60.530000     68.010000
max         41015.000000       101.990000    116.800000
```

[589]:
```
#converting time format to index

energy['time'] = pd.to_datetime(energy['time'], utc=True,␣
 ↪infer_datetime_format=True)
energy = energy.set_index('time')
```

[590]:
```
energy.head()
```

[590]:

```
                           generation biomass  \
time
2014-12-31 23:00:00+00:00               447.0
2015-01-01 00:00:00+00:00               449.0
2015-01-01 01:00:00+00:00               448.0
2015-01-01 02:00:00+00:00               438.0
2015-01-01 03:00:00+00:00               428.0

                           generation fossil brown coal/lignite  \
time
2014-12-31 23:00:00+00:00                                 329.0
2015-01-01 00:00:00+00:00                                 328.0
2015-01-01 01:00:00+00:00                                 323.0
2015-01-01 02:00:00+00:00                                 254.0
2015-01-01 03:00:00+00:00                                 187.0

                           generation fossil gas  generation fossil hard coal  \
time
2014-12-31 23:00:00+00:00                 4844.0                       4821.0
2015-01-01 00:00:00+00:00                 5196.0                       4755.0
2015-01-01 01:00:00+00:00                 4857.0                       4581.0
2015-01-01 02:00:00+00:00                 4314.0                       4131.0
2015-01-01 03:00:00+00:00                 4130.0                       3840.0

                           generation fossil oil  \
time
2014-12-31 23:00:00+00:00                  162.0
2015-01-01 00:00:00+00:00                  158.0
2015-01-01 01:00:00+00:00                  157.0
2015-01-01 02:00:00+00:00                  160.0
2015-01-01 03:00:00+00:00                  156.0

                           generation hydro pumped storage consumption  \
time
2014-12-31 23:00:00+00:00                                        863.0
2015-01-01 00:00:00+00:00                                        920.0
2015-01-01 01:00:00+00:00                                       1164.0
2015-01-01 02:00:00+00:00                                       1503.0
2015-01-01 03:00:00+00:00                                       1826.0

                           generation hydro run-of-river and poundage  \
time
2014-12-31 23:00:00+00:00                                      1051.0
2015-01-01 00:00:00+00:00                                      1009.0
2015-01-01 01:00:00+00:00                                       973.0
2015-01-01 02:00:00+00:00                                       949.0
2015-01-01 03:00:00+00:00                                       953.0
```

```
                           generation hydro water reservoir  \
time
2014-12-31 23:00:00+00:00                            1899.0
2015-01-01 00:00:00+00:00                            1658.0
2015-01-01 01:00:00+00:00                            1371.0
2015-01-01 02:00:00+00:00                             779.0
2015-01-01 03:00:00+00:00                             720.0

                           generation nuclear  generation other  \
time
2014-12-31 23:00:00+00:00              7096.0              43.0
2015-01-01 00:00:00+00:00              7096.0              43.0
2015-01-01 01:00:00+00:00              7099.0              43.0
2015-01-01 02:00:00+00:00              7098.0              43.0
2015-01-01 03:00:00+00:00              7097.0              43.0

                           generation other renewable  generation solar  \
time
2014-12-31 23:00:00+00:00                        73.0              49.0
2015-01-01 00:00:00+00:00                        71.0              50.0
2015-01-01 01:00:00+00:00                        73.0              50.0
2015-01-01 02:00:00+00:00                        75.0              50.0
2015-01-01 03:00:00+00:00                        74.0              42.0

                           generation waste  generation wind onshore  \
time
2014-12-31 23:00:00+00:00             196.0                   6378.0
2015-01-01 00:00:00+00:00             195.0                   5890.0
2015-01-01 01:00:00+00:00             196.0                   5461.0
2015-01-01 02:00:00+00:00             191.0                   5238.0
2015-01-01 03:00:00+00:00             189.0                   4935.0

                           forecast solar day ahead  \
time
2014-12-31 23:00:00+00:00                      17.0
2015-01-01 00:00:00+00:00                      16.0
2015-01-01 01:00:00+00:00                       8.0
2015-01-01 02:00:00+00:00                       2.0
2015-01-01 03:00:00+00:00                       9.0

                           forecast wind onshore day ahead  \
time
2014-12-31 23:00:00+00:00                           6436.0
2015-01-01 00:00:00+00:00                           5856.0
2015-01-01 01:00:00+00:00                           5454.0
2015-01-01 02:00:00+00:00                           5151.0
```

```
2015-01-01 03:00:00+00:00                              4861.0

                           total load forecast  total load actual  \
time
2014-12-31 23:00:00+00:00              26118.0            25385.0
2015-01-01 00:00:00+00:00              24934.0            24382.0
2015-01-01 01:00:00+00:00              23515.0            22734.0
2015-01-01 02:00:00+00:00              22642.0            21286.0
2015-01-01 03:00:00+00:00              21785.0            20264.0

                           price day ahead  price actual
time
2014-12-31 23:00:00+00:00            50.10         65.41
2015-01-01 00:00:00+00:00            48.10         64.92
2015-01-01 01:00:00+00:00            47.33         64.48
2015-01-01 02:00:00+00:00            42.27         59.32
2015-01-01 03:00:00+00:00            38.41         56.04
```

[591]:
```
# checking the nan values within the
# Gather generation columns
gen_cols = [col for col in energy.columns if col.startswith('generation')]
# gen cols + total generation
total_gen_cols = gen_cols.copy()
total_gen_cols.append('total load actual')
```

[592]:
```
energy.loc[energy['generation biomass'].isnull()][total_gen_cols]
```

[592]:
```
                           generation biomass  \
time
2015-01-05 02:00:00+00:00                 NaN
2015-01-05 11:00:00+00:00                 NaN
2015-01-05 12:00:00+00:00                 NaN
2015-01-05 13:00:00+00:00                 NaN
2015-01-05 14:00:00+00:00                 NaN
2015-01-05 15:00:00+00:00                 NaN
2015-01-05 16:00:00+00:00                 NaN
2015-01-19 18:00:00+00:00                 NaN
2015-01-19 19:00:00+00:00                 NaN
2015-01-27 18:00:00+00:00                 NaN
2015-01-28 12:00:00+00:00                 NaN
2015-04-16 07:00:00+00:00                 NaN
2015-04-23 19:00:00+00:00                 NaN
2015-06-15 07:00:00+00:00                 NaN
2015-10-02 09:00:00+00:00                 NaN
2015-12-02 08:00:00+00:00                 NaN
2016-07-09 20:00:00+00:00                 NaN
2016-11-23 03:00:00+00:00                 NaN
```

```
2018-07-11 07:00:00+00:00                NaN

                           generation fossil brown coal/lignite  \
time
2015-01-05 02:00:00+00:00                                   NaN
2015-01-05 11:00:00+00:00                                   NaN
2015-01-05 12:00:00+00:00                                   NaN
2015-01-05 13:00:00+00:00                                   NaN
2015-01-05 14:00:00+00:00                                   NaN
2015-01-05 15:00:00+00:00                                   NaN
2015-01-05 16:00:00+00:00                                   NaN
2015-01-19 18:00:00+00:00                                   NaN
2015-01-19 19:00:00+00:00                                   NaN
2015-01-27 18:00:00+00:00                                   NaN
2015-01-28 12:00:00+00:00                                   NaN
2015-04-16 07:00:00+00:00                                   NaN
2015-04-23 19:00:00+00:00                                   NaN
2015-06-15 07:00:00+00:00                                   NaN
2015-10-02 09:00:00+00:00                                   NaN
2015-12-02 08:00:00+00:00                                   NaN
2016-07-09 20:00:00+00:00                                   NaN
2016-11-23 03:00:00+00:00                                 900.0
2018-07-11 07:00:00+00:00                                   NaN

                           generation fossil gas  generation fossil hard coal  \
time
2015-01-05 02:00:00+00:00                    NaN                          NaN
2015-01-05 11:00:00+00:00                    NaN                          NaN
2015-01-05 12:00:00+00:00                    NaN                          NaN
2015-01-05 13:00:00+00:00                    NaN                          NaN
2015-01-05 14:00:00+00:00                    NaN                          NaN
2015-01-05 15:00:00+00:00                    NaN                          NaN
2015-01-05 16:00:00+00:00                    NaN                          NaN
2015-01-19 18:00:00+00:00                    NaN                          NaN
2015-01-19 19:00:00+00:00                    NaN                          NaN
2015-01-27 18:00:00+00:00                    NaN                          NaN
2015-01-28 12:00:00+00:00                    NaN                          NaN
2015-04-16 07:00:00+00:00                    NaN                          NaN
2015-04-23 19:00:00+00:00                    NaN                          NaN
2015-06-15 07:00:00+00:00                    NaN                          NaN
2015-10-02 09:00:00+00:00                    NaN                          NaN
2015-12-02 08:00:00+00:00                    NaN                          NaN
2016-07-09 20:00:00+00:00                    NaN                          NaN
2016-11-23 03:00:00+00:00                 4838.0                       4547.0
2018-07-11 07:00:00+00:00                    NaN                          NaN

                           generation fossil oil  \
```

```
time
2015-01-05 02:00:00+00:00                    NaN
2015-01-05 11:00:00+00:00                    NaN
2015-01-05 12:00:00+00:00                    NaN
2015-01-05 13:00:00+00:00                    NaN
2015-01-05 14:00:00+00:00                    NaN
2015-01-05 15:00:00+00:00                    NaN
2015-01-05 16:00:00+00:00                    NaN
2015-01-19 18:00:00+00:00                    NaN
2015-01-19 19:00:00+00:00                    NaN
2015-01-27 18:00:00+00:00                    NaN
2015-01-28 12:00:00+00:00                    NaN
2015-04-16 07:00:00+00:00                    NaN
2015-04-23 19:00:00+00:00                    NaN
2015-06-15 07:00:00+00:00                    NaN
2015-10-02 09:00:00+00:00                    NaN
2015-12-02 08:00:00+00:00                    NaN
2016-07-09 20:00:00+00:00                    NaN
2016-11-23 03:00:00+00:00                  269.0
2018-07-11 07:00:00+00:00                    NaN

                           generation hydro pumped storage consumption  \
time
2015-01-05 02:00:00+00:00                                          NaN
2015-01-05 11:00:00+00:00                                          NaN
2015-01-05 12:00:00+00:00                                          NaN
2015-01-05 13:00:00+00:00                                          NaN
2015-01-05 14:00:00+00:00                                          NaN
2015-01-05 15:00:00+00:00                                          NaN
2015-01-05 16:00:00+00:00                                          NaN
2015-01-19 18:00:00+00:00                                          NaN
2015-01-19 19:00:00+00:00                                          NaN
2015-01-27 18:00:00+00:00                                          NaN
2015-01-28 12:00:00+00:00                                          NaN
2015-04-16 07:00:00+00:00                                          NaN
2015-04-23 19:00:00+00:00                                          NaN
2015-06-15 07:00:00+00:00                                          NaN
2015-10-02 09:00:00+00:00                                          NaN
2015-12-02 08:00:00+00:00                                          NaN
2016-07-09 20:00:00+00:00                                          NaN
2016-11-23 03:00:00+00:00                                       1413.0
2018-07-11 07:00:00+00:00                                          NaN

                           generation hydro run-of-river and poundage  \
time
2015-01-05 02:00:00+00:00                                         NaN
2015-01-05 11:00:00+00:00                                         NaN
```

```
2015-01-05 12:00:00+00:00                                                NaN
2015-01-05 13:00:00+00:00                                                NaN
2015-01-05 14:00:00+00:00                                                NaN
2015-01-05 15:00:00+00:00                                                NaN
2015-01-05 16:00:00+00:00                                                NaN
2015-01-19 18:00:00+00:00                                                NaN
2015-01-19 19:00:00+00:00                                                NaN
2015-01-27 18:00:00+00:00                                                NaN
2015-01-28 12:00:00+00:00                                                NaN
2015-04-16 07:00:00+00:00                                                NaN
2015-04-23 19:00:00+00:00                                                NaN
2015-06-15 07:00:00+00:00                                                NaN
2015-10-02 09:00:00+00:00                                                NaN
2015-12-02 08:00:00+00:00                                                NaN
2016-07-09 20:00:00+00:00                                                NaN
2016-11-23 03:00:00+00:00                                              795.0
2018-07-11 07:00:00+00:00                                                NaN

                           generation hydro water reservoir  \
time
2015-01-05 02:00:00+00:00                               NaN
2015-01-05 11:00:00+00:00                               NaN
2015-01-05 12:00:00+00:00                               NaN
2015-01-05 13:00:00+00:00                               NaN
2015-01-05 14:00:00+00:00                               NaN
2015-01-05 15:00:00+00:00                               NaN
2015-01-05 16:00:00+00:00                               NaN
2015-01-19 18:00:00+00:00                               NaN
2015-01-19 19:00:00+00:00                               NaN
2015-01-27 18:00:00+00:00                               NaN
2015-01-28 12:00:00+00:00                               NaN
2015-04-16 07:00:00+00:00                               NaN
2015-04-23 19:00:00+00:00                               NaN
2015-06-15 07:00:00+00:00                               NaN
2015-10-02 09:00:00+00:00                               NaN
2015-12-02 08:00:00+00:00                               NaN
2016-07-09 20:00:00+00:00                               NaN
2016-11-23 03:00:00+00:00                             435.0
2018-07-11 07:00:00+00:00                               NaN

                           generation nuclear  generation other  \
time
2015-01-05 02:00:00+00:00                 NaN               NaN
2015-01-05 11:00:00+00:00                 NaN               NaN
2015-01-05 12:00:00+00:00                 NaN               NaN
2015-01-05 13:00:00+00:00                 NaN               NaN
2015-01-05 14:00:00+00:00                 NaN               NaN
```

```
2015-01-05 15:00:00+00:00                   NaN               NaN
2015-01-05 16:00:00+00:00                   NaN               NaN
2015-01-19 18:00:00+00:00                   NaN               NaN
2015-01-19 19:00:00+00:00                   NaN               NaN
2015-01-27 18:00:00+00:00                   NaN               NaN
2015-01-28 12:00:00+00:00                   NaN               NaN
2015-04-16 07:00:00+00:00                   NaN               NaN
2015-04-23 19:00:00+00:00                   NaN               NaN
2015-06-15 07:00:00+00:00                   NaN               NaN
2015-10-02 09:00:00+00:00                   NaN               NaN
2015-12-02 08:00:00+00:00                   NaN               NaN
2016-07-09 20:00:00+00:00                6923.0               NaN
2016-11-23 03:00:00+00:00                5040.0              60.0
2018-07-11 07:00:00+00:00                   NaN               NaN

                           generation other renewable  generation solar  \
time
2015-01-05 02:00:00+00:00                         NaN               NaN
2015-01-05 11:00:00+00:00                         NaN               NaN
2015-01-05 12:00:00+00:00                         NaN               NaN
2015-01-05 13:00:00+00:00                         NaN               NaN
2015-01-05 14:00:00+00:00                         NaN               NaN
2015-01-05 15:00:00+00:00                         NaN               NaN
2015-01-05 16:00:00+00:00                         NaN               NaN
2015-01-19 18:00:00+00:00                         NaN               NaN
2015-01-19 19:00:00+00:00                         NaN               NaN
2015-01-27 18:00:00+00:00                         NaN               NaN
2015-01-28 12:00:00+00:00                         NaN               NaN
2015-04-16 07:00:00+00:00                         NaN               NaN
2015-04-23 19:00:00+00:00                         NaN               NaN
2015-06-15 07:00:00+00:00                         NaN               NaN
2015-10-02 09:00:00+00:00                         NaN               NaN
2015-12-02 08:00:00+00:00                         NaN               NaN
2016-07-09 20:00:00+00:00                         NaN               NaN
2016-11-23 03:00:00+00:00                        85.0              15.0
2018-07-11 07:00:00+00:00                         NaN               NaN

                           generation waste  generation wind onshore  \
time
2015-01-05 02:00:00+00:00               NaN                      NaN
2015-01-05 11:00:00+00:00               NaN                      NaN
2015-01-05 12:00:00+00:00               NaN                      NaN
2015-01-05 13:00:00+00:00               NaN                      NaN
2015-01-05 14:00:00+00:00               NaN                      NaN
2015-01-05 15:00:00+00:00               NaN                      NaN
2015-01-05 16:00:00+00:00               NaN                      NaN
2015-01-19 18:00:00+00:00               NaN                      NaN
```

```
2015-01-19 19:00:00+00:00               NaN                       NaN
2015-01-27 18:00:00+00:00               NaN                       NaN
2015-01-28 12:00:00+00:00               NaN                       NaN
2015-04-16 07:00:00+00:00               NaN                       NaN
2015-04-23 19:00:00+00:00               NaN                       NaN
2015-06-15 07:00:00+00:00               NaN                       NaN
2015-10-02 09:00:00+00:00               NaN                       NaN
2015-12-02 08:00:00+00:00               NaN                       NaN
2016-07-09 20:00:00+00:00               NaN                       NaN
2016-11-23 03:00:00+00:00             227.0                    4598.0
2018-07-11 07:00:00+00:00               NaN                       NaN

                           total load actual
time
2015-01-05 02:00:00+00:00            21182.0
2015-01-05 11:00:00+00:00                NaN
2015-01-05 12:00:00+00:00                NaN
2015-01-05 13:00:00+00:00                NaN
2015-01-05 14:00:00+00:00                NaN
2015-01-05 15:00:00+00:00                NaN
2015-01-05 16:00:00+00:00                NaN
2015-01-19 18:00:00+00:00            39304.0
2015-01-19 19:00:00+00:00            39262.0
2015-01-27 18:00:00+00:00            38335.0
2015-01-28 12:00:00+00:00                NaN
2015-04-16 07:00:00+00:00                NaN
2015-04-23 19:00:00+00:00                NaN
2015-06-15 07:00:00+00:00            30047.0
2015-10-02 09:00:00+00:00                NaN
2015-12-02 08:00:00+00:00                NaN
2016-07-09 20:00:00+00:00                NaN
2016-11-23 03:00:00+00:00            23112.0
2018-07-11 07:00:00+00:00                NaN
```

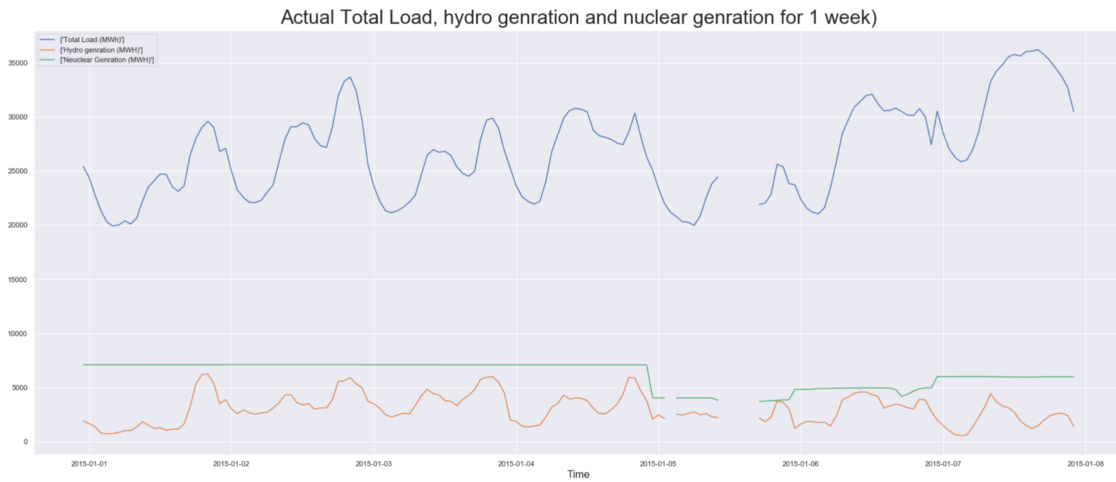
We will try to plot a zoom on the a period which is missing some values

[593]:
```python
# Define a function to plot different types of time-series fro the analysis

def plot_series(df=None, column=None, series=pd.Series([]), label=None,␣
↪ylabel=None, title=None, start=0, end=None):
    """
    Plots a certain time-series which has either been loaded in a dataframe and␣
↪which
    constitutes one of its columns or it a custom pandas series created by the␣
↪user.
    The user can define either the 'df' and the 'column' or the 'series' and␣
↪additionally,
```

```
    can also define the 'label', the 'ylabel', the 'title', the 'start' and the 'end' of the plot.
    """
    sns.set()
    fig, ax = plt.subplots(figsize=(30, 12))
    ax.set_xlabel('Time', fontsize=16)
    if column:
        ax.plot(df[column][start:end], label=label)
        ax.set_ylabel(ylabel, fontsize=22)
    if series.any():
        ax.plot(series, label=label)
        ax.set_ylabel(ylabel, fontsize=22)
    if label:
        ax.legend(fontsize=22)
    if title:
        ax.set_title(title, fontsize=30)
    ax.grid(True)
    return ax
```

[594]:

```
# Zoom into the plot of the hourly (actual) total load

ax = plot_series(df=energy, column=['total load actual','generation hydro water reservoir','generation nuclear'],
                 title='Actual Total Load, hydro genration and nuclear genration for 1 week)', end=24*7*1)
ax.legend((['Total Load (MWh)'], ['Hydro genration (MWH)'],['Neuclear Genration (MWH)']))
plt.show()
```



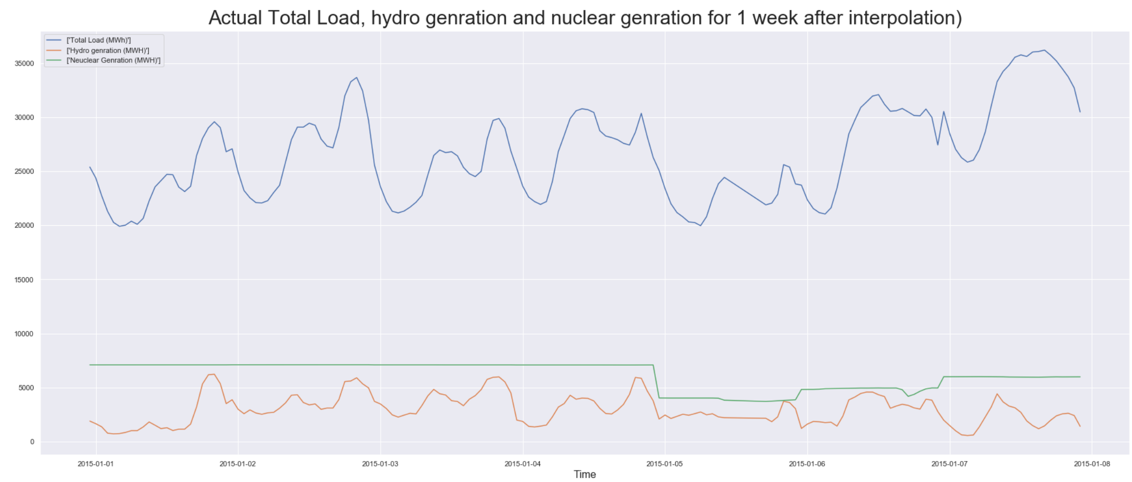
From the above curve and table:

1- It appears that te null values are common across all the genration plant types and the total so this means we will not be abel to calculate the values and we will have to interpolate this values using linear interpolation

2- As we know from the power genrtaion concepts Neuclear power genration load and demand is alwyas required to be stable for longer periods and the flutiuations in the load are manged through the other types of genration

[595]:
```
# Fill null values using interpolation with both directions forword and backword

energy.interpolate(method='linear', limit_direction='both', inplace=True,␣
↪axis=0)
```

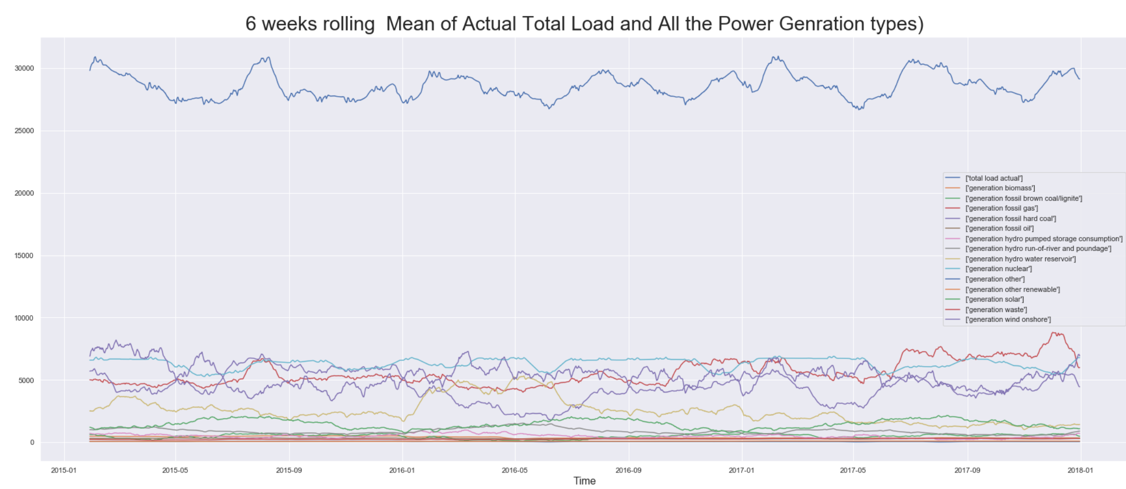
[601]:
```
# Zoom into the plot of the hourly (actual) total load

ax = plot_series(df=energy, column=['total load actual','generation hydro water␣
↪reservoir','generation nuclear'],
                 title='Actual Total Load, hydro genration and nuclear␣
↪genration for 1 week after interpolation)', end=24*7*1)
ax.legend((['Total Load (MWh)'], ['Hydro genration (MWH)'],['Neuclear Genration␣
↪(MWH)']))
plt.show()
```



[ ]:
```

```

[615]:
```
# Zoom into the plot of the hourly (actual) total load
```

```python
ax = plot_series(df=energy.rolling(window=24*7*4).mean(), column=['total load␣
↪actual','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'],
title='6 weeks rolling  Mean of Actual Total Load and All the Power Genration␣
↪types)' , end=24*365*3)
ax.legend((['total load actual'],
        ['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']))
plt.show()
```



As We can see and understand from the above graph the destribution of the genration across resources varies but some resource are dominant and stable as the neuclear plants as fluctiuation is not easy to do durig the opertaions of such plants.

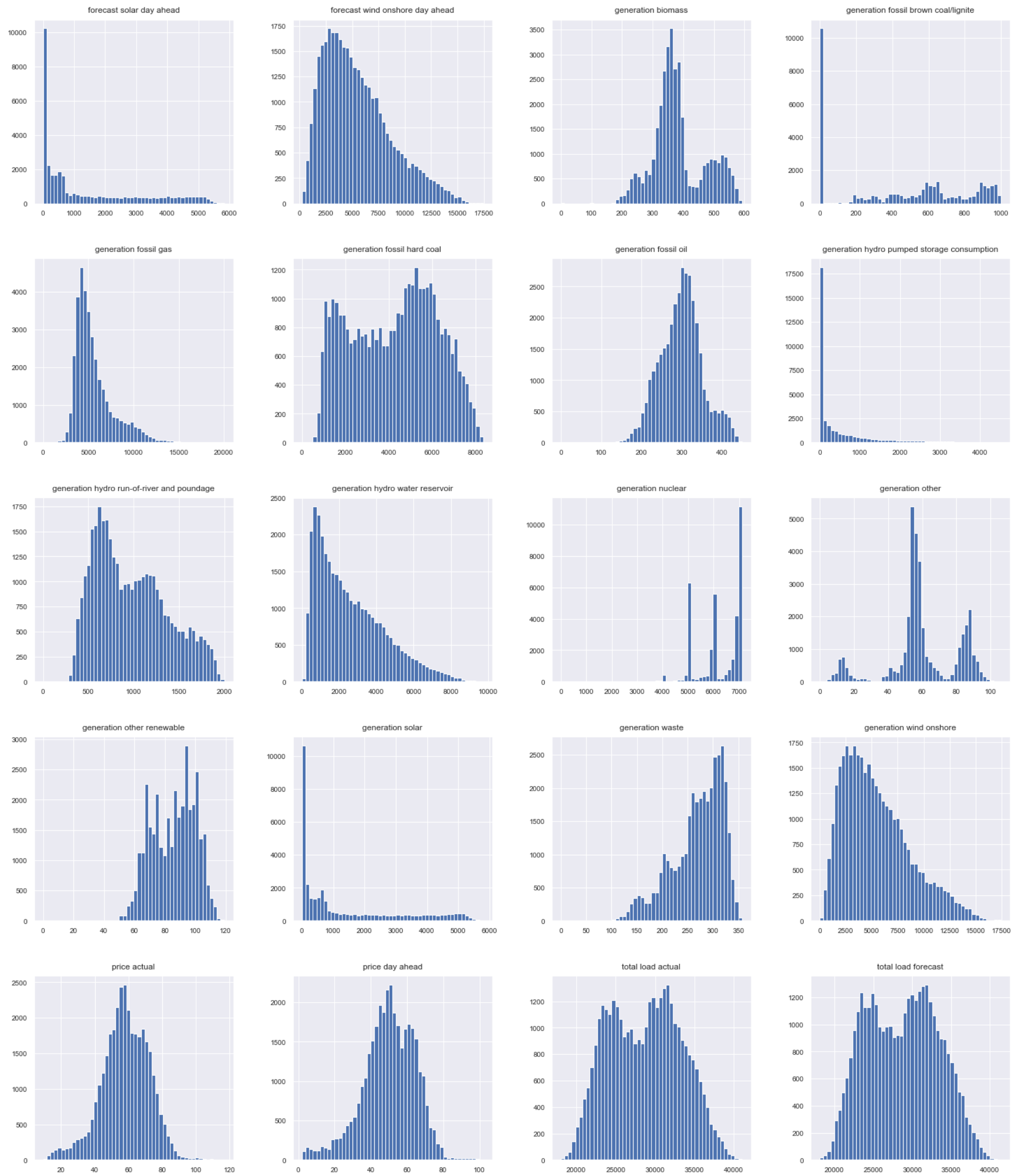
We can see alot of fluctiation between fossil fuels manly the coal and gas plants and clean energy resources and hydor flutiuates depending on the weather and season of the year.

It seems also that the genration from waste is does not have a big persentage of the spanish grid genration

[28]:
```
#Data Cleaning : Checking for null values after cleaning
print(energy.isnull().sum())
#checking any duplicates
print(energy.duplicated(keep='first').sum())
```

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

[620]:
```
energy.hist(figsize=(25, 30), bins=50, xlabelsize=10, ylabelsize=10)
plt.show()
```



The Histogram show an anlysis for all the sources of power genration and the limit of each source of genration and we can see the normal operating range of genration for this source

[30]:
```
energy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 35064 entries, 2014-12-31 23:00:00+00:00 to 2018-12-31
22:00:00+00:00
Data columns (total 20 columns):
```

```
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   generation biomass                        35064 non-null  float64
 1   generation fossil brown coal/lignite      35064 non-null  float64
 2   generation fossil gas                     35064 non-null  float64
 3   generation fossil hard coal               35064 non-null  float64
 4   generation fossil oil                     35064 non-null  float64
 5   generation hydro pumped storage consumption  35064 non-null  float64
 6   generation hydro run-of-river and poundage  35064 non-null  float64
 7   generation hydro water reservoir          35064 non-null  float64
 8   generation nuclear                        35064 non-null  float64
 9   generation other                          35064 non-null  float64
 10  generation other renewable                35064 non-null  float64
 11  generation solar                          35064 non-null  float64
 12  generation waste                          35064 non-null  float64
 13  generation wind onshore                   35064 non-null  float64
 14  forecast solar day ahead                  35064 non-null  float64
 15  forecast wind onshore day ahead           35064 non-null  float64
 16  total load forecast                       35064 non-null  float64
 17  total load actual                         35064 non-null  float64
 18  price day ahead                           35064 non-null  float64
 19  price actual                              35064 non-null  float64
dtypes: float64(20)
memory usage: 5.6 MB
```

1.4.2 Weather dataset cleaning

[31]:
```
weather.head()
```

[31]:
```
                      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
```

[32]:
```
weather.describe().round(1)
```

[32]:
```
          temp  temp_min  temp_max   pressure  humidity  wind_speed  \
count 178396.0  178396.0  178396.0   178396.0  178396.0    178396.0
mean     289.6     288.3     291.1     1069.3      68.4         2.5
std        8.0       8.0       8.6     5969.6      21.9         2.1
min      262.2     262.2     262.2        0.0       0.0         0.0
25%      283.7     282.5     284.6     1013.0      53.0         1.0
50%      289.2     288.2     290.2     1018.0      72.0         2.0
75%      295.2     293.7     297.2     1022.0      87.0         4.0
max      315.6     315.2     321.2  1008371.0     100.0       133.0

      wind_deg   rain_1h   rain_3h   snow_3h  clouds_all  weather_id
count 178396.0  178396.0  178396.0  178396.0    178396.0    178396.0
mean     166.6       0.1       0.0       0.0        25.1       759.8
std      116.6       0.4       0.0       0.2        30.8       108.7
min        0.0       0.0       0.0       0.0         0.0       200.0
25%       55.0       0.0       0.0       0.0         0.0       800.0
50%      177.0       0.0       0.0       0.0        20.0       800.0
75%      270.0       0.0       0.0       0.0        40.0       801.0
max      360.0      12.0       2.3      21.5       100.0       804.0
```

[33]:
```
weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178396 entries, 0 to 178395
Data columns (total 17 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   dt_iso               178396 non-null  object
 1   city_name            178396 non-null  object
 2   temp                 178396 non-null  float64
 3   temp_min             178396 non-null  float64
 4   temp_max             178396 non-null  float64
 5   pressure             178396 non-null  int64
 6   humidity             178396 non-null  int64
 7   wind_speed           178396 non-null  int64
 8   wind_deg             178396 non-null  int64
 9   rain_1h              178396 non-null  float64
 10  rain_3h              178396 non-null  float64
 11  snow_3h              178396 non-null  float64
 12  clouds_all           178396 non-null  int64
 13  weather_id           178396 non-null  int64
 14  weather_main         178396 non-null  object
 15  weather_description  178396 non-null  object
```

```
 16  weather_icon         178396 non-null  object
dtypes: float64(6), int64(6), object(5)
memory usage: 23.1+ MB
```

First we started to check the values within the dataset to see if there is any outliers as it has direct effect on generation in terms due to the following reasons

1- temperature and Humidity Demand due to air conditioning and heating

2- temperature and Humidity affects the generating thermal as it affects the cooling process for the equipment and efficiency of the plants

3- Wind speed affects the wind generation

4- Rain and clouds have direct effect on generation through solar generation farms.

so first we will look at the max and mi of all the values and see if they make sense or not

1- For temperature the max is 321 k which is equivalent to 47.85 C which might happen in some rare very summer days

2- For Temperature the low is 262.2 K which is equivalent to -10.95 chick still make sense to reach this temperature during winter coldest days in northern Spain

3- Pressure: it seems we have some problems with the pressure max. and min in hPA as it has up normal pressure like 0 and 1008371.0 which cannot happen unless there is a problem in the measurement. Though my knowledge in the domain of the power generation the pressure factor can be neglected from this detail as it very minor to neglected effect so we will drop it from the data

4- Humidity: has 0 which is very highly unlikely to happen so will cap the humidity at max 90 % and low 35%

5- Wind speed: looking at the wind speed it seems that there are some strange values that is reaching 133 m/s which is impossible the max wind speed recorded in Madrid was 7.8m/s over the last 10 years so we will need to clean this data.

6- we will drop weather_id, weather_main, weather_description, weather_icon as the details in the other columns are enoughto indicate all waether condtions.

[34]:
```python
# Drop the columns that are not required for the analysis

weather.drop(columns=['pressure','temp_min','temp_max',
                      'weather_id','weather_main',
                      'weather_description',
                      'weather_icon','rain_1h','rain_3h','snow_3h' ], inplace␣
 ↪=True)
```

[35]:
```python
weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178396 entries, 0 to 178395
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
```

```
---  ------      --------------   -----
 0   dt_iso      178396 non-null  object
 1   city_name   178396 non-null  object
 2   temp        178396 non-null  float64
 3   humidity    178396 non-null  int64
 4   wind_speed  178396 non-null  int64
 5   wind_deg    178396 non-null  int64
 6   clouds_all  178396 non-null  int64
dtypes: float64(1), int64(4), object(2)
memory usage: 9.5+ MB
```

[36]:
```python
# Checking the null values
weather.isnull().sum()
```

[36]:
```
dt_iso        0
city_name     0
temp          0
humidity      0
wind_speed    0
wind_deg      0
clouds_all    0
dtype: int64
```

[37]:
```python
# adjusting the types of data
weather['humidity'] = weather['humidity'].astype(np.float64)
weather['wind_speed'] = weather['wind_speed'].astype(np.float64)
weather['wind_deg'] = weather['wind_deg'].astype(np.float64)
weather['clouds_all'] = weather['clouds_all'].astype(np.float64)
```

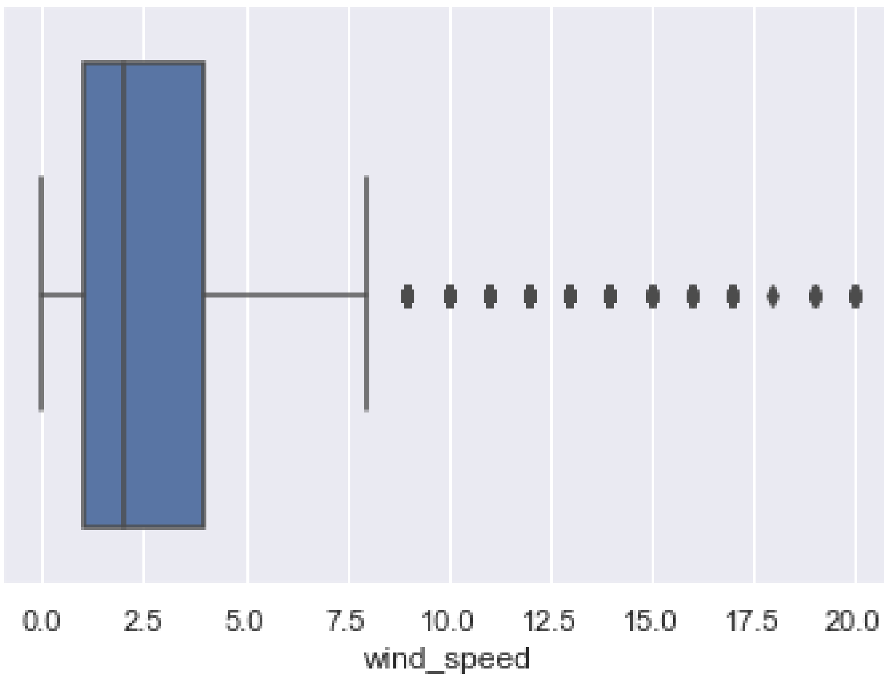
[38]:
```python
weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178396 entries, 0 to 178395
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   dt_iso      178396 non-null  object
 1   city_name   178396 non-null  object
 2   temp        178396 non-null  float64
 3   humidity    178396 non-null  float64
 4   wind_speed  178396 non-null  float64
 5   wind_deg    178396 non-null  float64
 6   clouds_all  178396 non-null  float64
dtypes: float64(5), object(2)
memory usage: 9.5+ MB
```

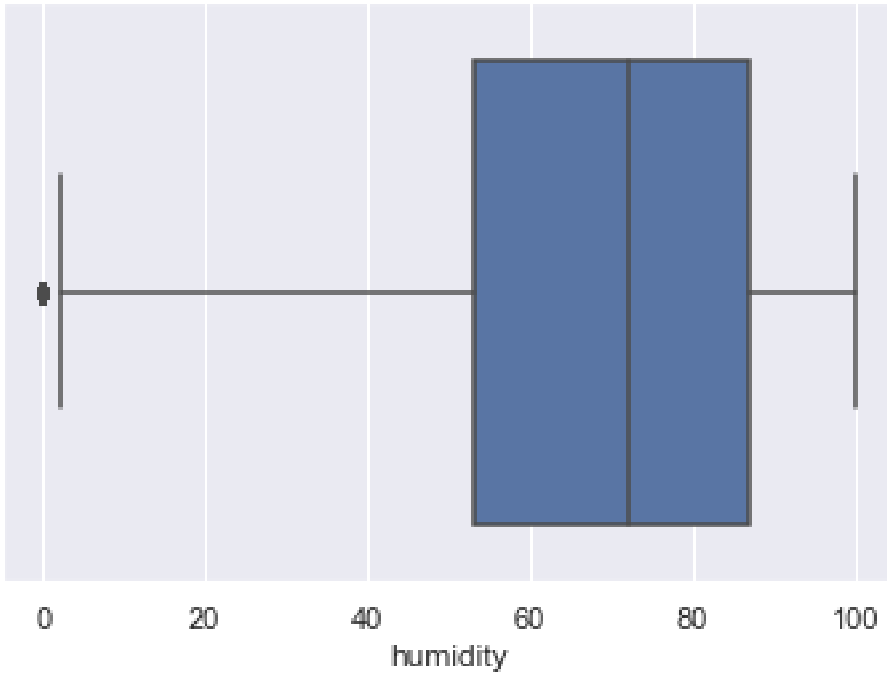
[621]:
```python
# Cleaning the outliers
```

```python
weather.loc[weather.wind_speed > 20, 'wind_speed'] = np.nan
```

[40]:
```python
sns.boxplot(x=weather['wind_speed'])
plt.show()
```

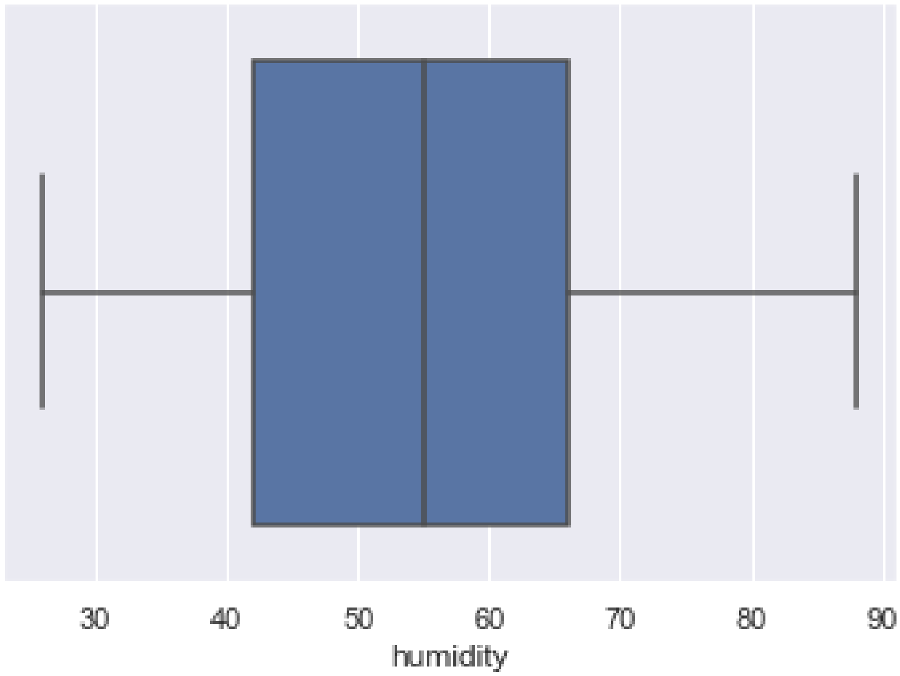


[41]:
```python
sns.boxplot(x=weather['humidity'])
plt.show()
```



[42]:
```
#Removing all values less than 15 and more than 90 for humidity
weather.loc[weather.wind_speed > 90, 'humidity'] = np.nan
weather.loc[weather.wind_speed < 15, 'humidity'] = np.nan
```

[43]:
```
sns.boxplot(x=weather['humidity'])
plt.show()
```



[44]:
```
weather.isnull().sum()
```

[44]:
```
dt_iso             0
city_name          0
temp               0
humidity      178283
wind_speed        31
wind_deg           0
clouds_all         0
dtype: int64
```

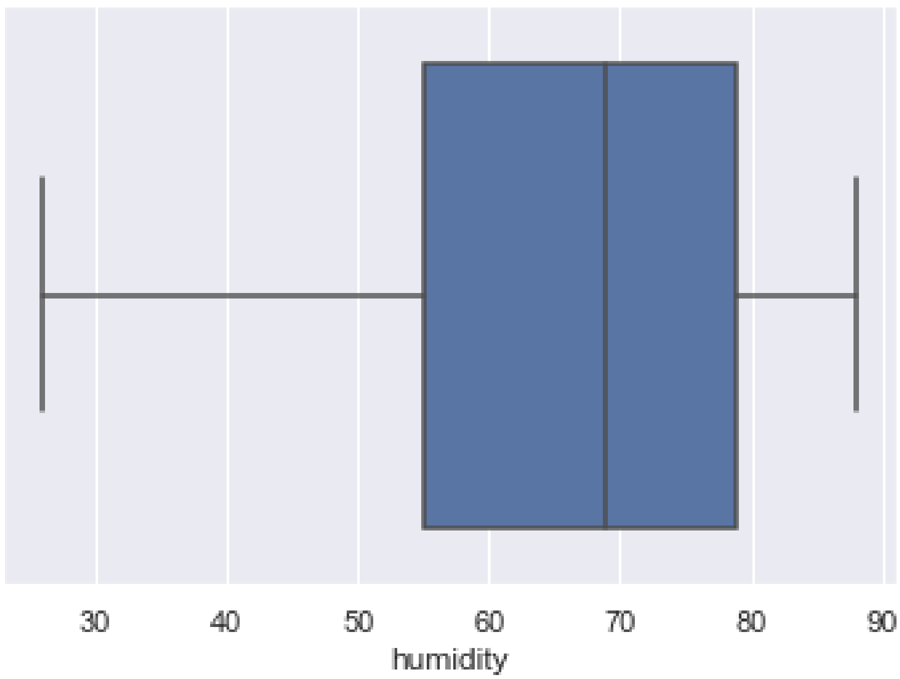
[45]:
```
# we will fill the missing values through interpolation
# Fill null values using interpolation with both directions forword and backword

weather.interpolate(method='linear', limit_direction='both', inplace=True,␣
 ↪axis=0)
```

[46]:
```
weather.isnull().sum()
```

[46]:
```
dt_iso        0
city_name     0
temp          0
humidity      0
wind_speed    0
```

```
wind_deg      0
clouds_all    0
dtype: int64
```

[47]:
```python
sns.boxplot(x=weather['humidity'])
plt.show()
```



[48]:
```python
# we will check the cities in the dataset

cities = weather['city_name'].unique()

print(cities)
```

```
['Valencia' 'Madrid' 'Bilbao' ' Barcelona' 'Seville']
```

[49]:
```python
#we will split the datasets into sities to do this we need to find out how many␣
↪rows are for each city
cities = weather['city_name'].unique()
grouped_weather = weather.groupby('city_name')

for city in cities:
    print('There are {} observations in eather'.format(grouped_weather.
↪get_group('{}'.format(city)).shape[0]),
          'about city: {}.'.format(city))
```

```
There are 35145 observations in eather about city: Valencia.
There are 36267 observations in eather about city: Madrid.
There are 35951 observations in eather about city: Bilbao.
There are 35476 observations in eather about city:  Barcelona.
There are 35557 observations in eather about city: Seville.
```

Based on the above finiding the number of rows are not equal for all cities so we will have to dig more to find if the dates are for the same range or different ranges also if there is duplicates or not

[50]:
```
# first step is to check for duplicates
temp_weather = weather.duplicated(keep='first').sum()

print('There are {} duplicate rows in weather except first occurrence based on␣
↪all columns.'.format(temp_weather))
```

```
There are 395 duplicate rows in weather except first occurrence based on all
columns.
```

[51]:
```
weather = weather.reset_index().drop_duplicates(subset=['dt_iso', 'city_name'],␣
↪keep='first')
```

[52]:
```
#replacing the name of the time column to match the energy dataset
weather.rename(columns = {'dt_iso':'time'}, inplace = True)
```

[53]:
```
parse_dates=[weather.time]
weather['time'] = pd.to_datetime(weather['time'], utc=True,␣
↪infer_datetime_format=True)
weather = weather.set_index('time')
```

[54]:
```
weather.drop(columns=['index'], inplace =True)
weather.head()
```

[54]:
```
                           city_name     temp  humidity  wind_speed  wind_deg  \
time
2014-12-31 23:00:00+00:00   Valencia  270.475      26.0         1.0      62.0
2015-01-01 00:00:00+00:00   Valencia  270.475      26.0         1.0      62.0
2015-01-01 01:00:00+00:00   Valencia  269.686      26.0         0.0      23.0
2015-01-01 02:00:00+00:00   Valencia  269.686      26.0         0.0      23.0
2015-01-01 03:00:00+00:00   Valencia  269.686      26.0         0.0      23.0

                           clouds_all
time
2014-12-31 23:00:00+00:00         0.0
2015-01-01 00:00:00+00:00         0.0
2015-01-01 01:00:00+00:00         0.0
2015-01-01 02:00:00+00:00         0.0
2015-01-01 03:00:00+00:00         0.0
```

[55]:
```
weather_Val= weather[weather['city_name']== "Valencia"]
weather_Mad= weather[weather['city_name']== "Madrid"]
weather_Bil=weather[weather['city_name']== "Bilbao"]
weather_Bar=weather[weather['city_name']== " Barcelona"]
weather_Sev=weather[weather['city_name']== "Seville"]
```

[56]:
```
df1 = weather_Val.copy()
df2 = weather_Mad.copy()
df3 = weather_Bil.copy()
df4 = weather_Bar.copy()
df5 = weather_Sev.copy()
```

[57]:
```
df1.drop(columns=['city_name'], inplace = True)
df2.drop(columns=['city_name'], inplace =True)
df3.drop(columns=['city_name'], inplace =True)
df4.drop(columns=['city_name'], inplace =True)
df5.drop(columns=['city_name'], inplace =True)
```

[58]:
```
df1 = df1.add_suffix('_Val')
df2 = df2.add_suffix('_Mad')
df3 = df3.add_suffix('_Bil')
df4 = df4.add_suffix('_Bar')
df5 = df5.add_suffix('_Sev')
```

Now the 5 dataframes for the weather are ready to be merged with the enrgy dataframe

[59]:
```
dfs = pd.merge(df1,df2, left_index=True, right_index=True )
dfs = pd.merge(dfs,df3, left_index=True, right_index=True )
dfs = pd.merge(dfs,df4, left_index=True, right_index=True )
dfs = pd.merge(dfs,df5, left_index=True, right_index=True )
df_energy = pd.merge(energy,dfs, left_index=True, right_index=True )
```

[60]:
```
df_energy.isnull().sum()
```

[60]:
```
generation biomass                               0
generation fossil brown coal/lignite             0
generation fossil gas                            0
generation fossil hard coal                      0
generation fossil oil                            0
generation hydro pumped storage consumption      0
generation hydro run-of-river and poundage       0
generation hydro water reservoir                 0
generation nuclear                               0
generation other                                 0
generation other renewable                       0
generation solar                                 0
generation waste                                 0
generation wind onshore                          0
```

```
forecast solar day ahead                        0
forecast wind onshore day ahead                 0
total load forecast                             0
total load actual                               0
price day ahead                                 0
price actual                                    0
temp_Val                                        0
humidity_Val                                    0
wind_speed_Val                                  0
wind_deg_Val                                    0
clouds_all_Val                                  0
temp_Mad                                        0
humidity_Mad                                    0
wind_speed_Mad                                  0
wind_deg_Mad                                    0
clouds_all_Mad                                  0
temp_Bil                                        0
humidity_Bil                                    0
wind_speed_Bil                                  0
wind_deg_Bil                                    0
clouds_all_Bil                                  0
temp_Bar                                        0
humidity_Bar                                    0
wind_speed_Bar                                  0
wind_deg_Bar                                    0
clouds_all_Bar                                  0
temp_Sev                                        0
humidity_Sev                                    0
wind_speed_Sev                                  0
wind_deg_Sev                                    0
clouds_all_Sev                                  0
dtype: int64
```

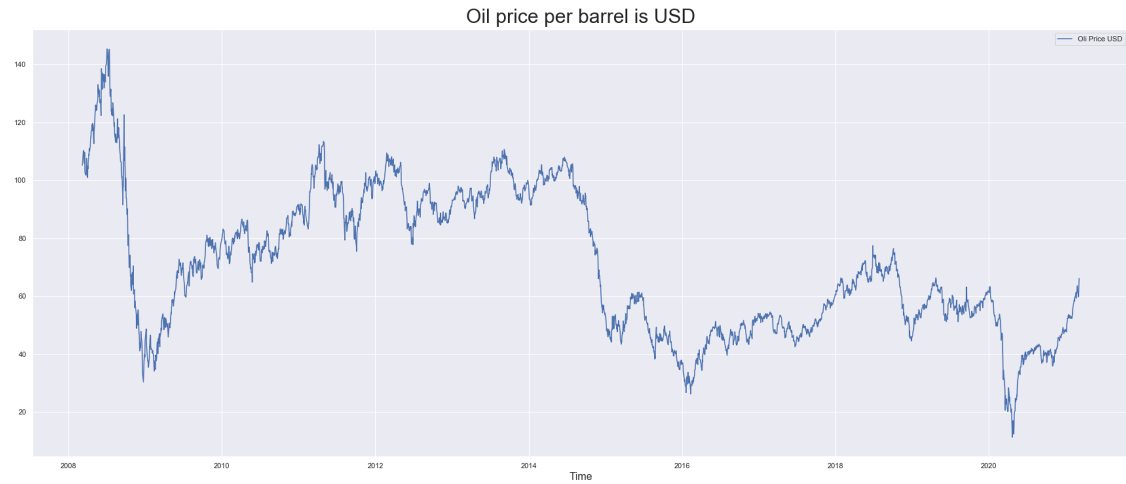
1.4.3 Oil Price

[628]:
```
oil_price.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3289 entries, 0 to 3288
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    3289 non-null   object
 1    value  3268 non-null   float64
dtypes: float64(1), object(1)
memory usage: 51.5+ KB
```

[629]:
```
# Cleaning  Oil prices
oil_price.rename(columns = {'date':'time'}, inplace = True)
oil_price.rename(columns = {' value':'oli_price'}, inplace = True)
oil_price['time'] = pd.to_datetime(oil_price['time'], utc=True,␣
↪infer_datetime_format=True)
oil_price = oil_price.set_index('time')
```

[633]:
```
ax = plot_series(df=oil_price, column=['oli_price',],
                 title='Oil price per barrel is USD', end=24*7*30)
ax.legend(([’Oli Price USD']))
plt.show()
```



1.4.4 Natural Gas Prices

[63]:
```
ntrl_gas_price.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3311 entries, 0 to 3310
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    3311 non-null   object
 1    value  3290 non-null   float64
dtypes: float64(1), object(1)
memory usage: 51.9+ KB
```

[641]:
```
ntrl_gas_price.rename(columns = {'date':'time'}, inplace = True)
ntrl_gas_price.rename(columns = {' value':'NG_price'}, inplace = True)
ntrl_gas_price['time'] = pd.to_datetime(ntrl_gas_price['time'], utc=True,␣
↪infer_datetime_format=True)
ntrl_gas_price = ntrl_gas_price.set_index('time')
```

[642]:
```
ax = plot_series(df=ntrl_gas_price, column=['NG_price',],
                 title='NG Price in  USD', end=24*7*30)
ax.legend((['NG_price']))
plt.show()
```

1.4.5 Coal prices

Since the Data set of the Coal is monthly average will have to fill the hourly values with the same monthly average

[644]:
```
coal_price.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49 entries, 0 to 48
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   time        49 non-null     object
 1   coal_price  49 non-null     float64
dtypes: float64(1), object(1)
memory usage: 912.0+ bytes
```

[645]:
```
coal_price['time'] = pd.to_datetime(coal_price['time'], utc=True,␣
 ↪infer_datetime_format=True)
coal_price = coal_price.set_index('time')
```

[646]:
```
rng = pd.date_range(coal_price.index.min(), coal_price.index.max() + pd.
 ↪Timedelta(23, 'H'), freq='H')
coal_price = coal_price.reindex(rng, method='ffill')
```

[647]:
```
coal_price.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 35088 entries, 2015-01-01 00:00:00+00:00 to 2019-01-01
23:00:00+00:00
Freq: H
Data columns (total 1 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   coal_price  35088 non-null  float64
dtypes: float64(1)
memory usage: 548.2 KB
```

[648]:
```
coal_price = coal_price.loc['2014-12-31 23:00:00+00:00':'2018-12-31 23:00:00+00:
↪00'].copy()
```

[653]:
```
ax = plot_series(df=coal_price, column=['coal_price',],
                 title='Coal Price in  USD', end=24*7*200)
ax.legend((['Average Coal price in USD']))
plt.show()
```

1.4.6 Exchange rate of USD and EURO

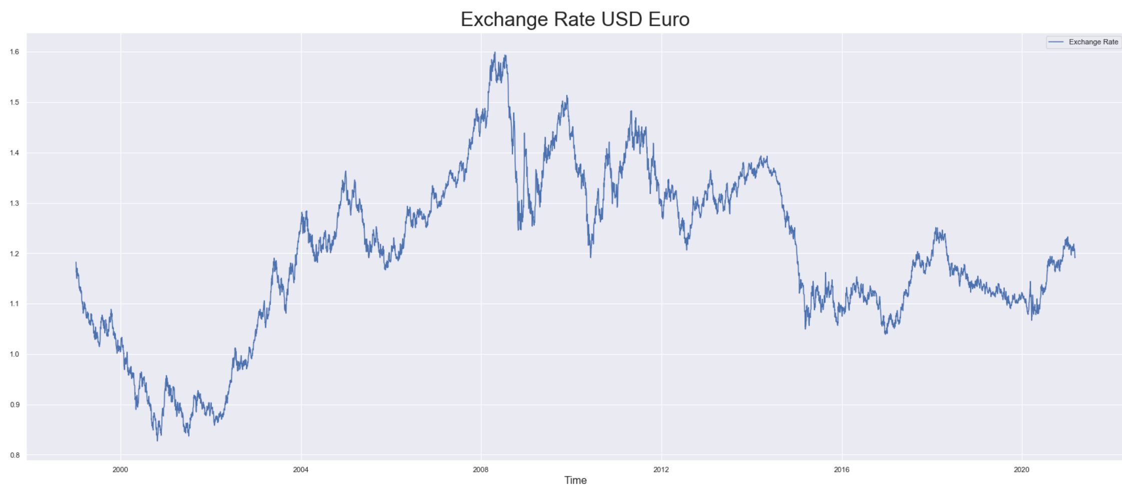
[654]:
```
xchg_rate.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5825 entries, 0 to 5824
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    5825 non-null   object
 1    value  5825 non-null   float64
dtypes: float64(1), object(1)
```

```
memory usage: 91.1+ KB
```

[655]:
```
xchg_rate.rename(columns = {'date':'time'}, inplace = True)
xchg_rate.rename(columns = {' value':'XG_rate'}, inplace = True)
xchg_rate['time'] = pd.to_datetime(xchg_rate['time'], utc=True,␣
 ↪infer_datetime_format=True)
xchg_rate = xchg_rate.set_index('time')
```

[657]:
```
ax = plot_series(df=xchg_rate, column=['XG_rate',],
                 title='Exchange Rate USD Euro', end=24*7*200)
ax.legend((['Exchange Rate']))
plt.show()
```



1.4.7 Combining the Datasets

[72]:
```
df_energy
```

[72]:
```
                           generation biomass  \
time
2014-12-31 23:00:00+00:00               447.0
2015-01-01 00:00:00+00:00               449.0
2015-01-01 01:00:00+00:00               448.0
2015-01-01 02:00:00+00:00               438.0
2015-01-01 03:00:00+00:00               428.0
…                                       …
2018-12-31 18:00:00+00:00               297.0
2018-12-31 19:00:00+00:00               296.0
2018-12-31 20:00:00+00:00               292.0
2018-12-31 21:00:00+00:00               293.0
2018-12-31 22:00:00+00:00               290.0
```

```
                           generation fossil brown coal/lignite  \
time
2014-12-31 23:00:00+00:00                               329.0
2015-01-01 00:00:00+00:00                               328.0
2015-01-01 01:00:00+00:00                               323.0
2015-01-01 02:00:00+00:00                               254.0
2015-01-01 03:00:00+00:00                               187.0
…                                                       …
2018-12-31 18:00:00+00:00                                 0.0
2018-12-31 19:00:00+00:00                                 0.0
2018-12-31 20:00:00+00:00                                 0.0
2018-12-31 21:00:00+00:00                                 0.0
2018-12-31 22:00:00+00:00                                 0.0

                           generation fossil gas  generation fossil hard coal  \
time
2014-12-31 23:00:00+00:00                 4844.0                       4821.0
2015-01-01 00:00:00+00:00                 5196.0                       4755.0
2015-01-01 01:00:00+00:00                 4857.0                       4581.0
2015-01-01 02:00:00+00:00                 4314.0                       4131.0
2015-01-01 03:00:00+00:00                 4130.0                       3840.0
…                                          …                            …
2018-12-31 18:00:00+00:00                 7634.0                       2628.0
2018-12-31 19:00:00+00:00                 7241.0                       2566.0
2018-12-31 20:00:00+00:00                 7025.0                       2422.0
2018-12-31 21:00:00+00:00                 6562.0                       2293.0
2018-12-31 22:00:00+00:00                 6926.0                       2166.0

                           generation fossil oil  \
time
2014-12-31 23:00:00+00:00                  162.0
2015-01-01 00:00:00+00:00                  158.0
2015-01-01 01:00:00+00:00                  157.0
2015-01-01 02:00:00+00:00                  160.0
2015-01-01 03:00:00+00:00                  156.0
…                                          …
2018-12-31 18:00:00+00:00                  178.0
2018-12-31 19:00:00+00:00                  174.0
2018-12-31 20:00:00+00:00                  168.0
2018-12-31 21:00:00+00:00                  163.0
2018-12-31 22:00:00+00:00                  163.0

                           generation hydro pumped storage consumption  \
time
2014-12-31 23:00:00+00:00                                        863.0
2015-01-01 00:00:00+00:00                                        920.0
2015-01-01 01:00:00+00:00                                       1164.0
```

```
2015-01-01 02:00:00+00:00                                               1503.0
2015-01-01 03:00:00+00:00                                               1826.0
…                                                                          …
2018-12-31 18:00:00+00:00                                                  1.0
2018-12-31 19:00:00+00:00                                                  1.0
2018-12-31 20:00:00+00:00                                                 50.0
2018-12-31 21:00:00+00:00                                                108.0
2018-12-31 22:00:00+00:00                                                108.0

                           generation hydro run-of-river and poundage  \
time
2014-12-31 23:00:00+00:00                                      1051.0
2015-01-01 00:00:00+00:00                                      1009.0
2015-01-01 01:00:00+00:00                                       973.0
2015-01-01 02:00:00+00:00                                       949.0
2015-01-01 03:00:00+00:00                                       953.0
…                                                                 …
2018-12-31 18:00:00+00:00                                      1135.0
2018-12-31 19:00:00+00:00                                      1172.0
2018-12-31 20:00:00+00:00                                      1148.0
2018-12-31 21:00:00+00:00                                      1128.0
2018-12-31 22:00:00+00:00                                      1069.0

                           generation hydro water reservoir  \
time
2014-12-31 23:00:00+00:00                            1899.0
2015-01-01 00:00:00+00:00                            1658.0
2015-01-01 01:00:00+00:00                            1371.0
2015-01-01 02:00:00+00:00                             779.0
2015-01-01 03:00:00+00:00                             720.0
…                                                     …
2018-12-31 18:00:00+00:00                            4836.0
2018-12-31 19:00:00+00:00                            3931.0
2018-12-31 20:00:00+00:00                            2831.0
2018-12-31 21:00:00+00:00                            2068.0
2018-12-31 22:00:00+00:00                            1686.0

                           generation nuclear  generation other  …  \
time                                                             …
2014-12-31 23:00:00+00:00              7096.0              43.0  …
2015-01-01 00:00:00+00:00              7096.0              43.0  …
2015-01-01 01:00:00+00:00              7099.0              43.0  …
2015-01-01 02:00:00+00:00              7098.0              43.0  …
2015-01-01 03:00:00+00:00              7097.0              43.0  …
…                                        …                 …  …
2018-12-31 18:00:00+00:00              6073.0              63.0  …
2018-12-31 19:00:00+00:00              6074.0              62.0  …
```

```
2018-12-31 20:00:00+00:00              6076.0              61.0  …
2018-12-31 21:00:00+00:00              6075.0              61.0  …
2018-12-31 22:00:00+00:00              6075.0              61.0  …

                           temp_Bar  humidity_Bar  wind_speed_Bar  \
time
2014-12-31 23:00:00+00:00   281.625     79.540056             7.0
2015-01-01 00:00:00+00:00   281.625     79.540444             7.0
2015-01-01 01:00:00+00:00   281.286     79.540833             7.0
2015-01-01 02:00:00+00:00   281.286     79.541221             7.0
2015-01-01 03:00:00+00:00   281.286     79.541609             7.0
…                               …           …               …
2018-12-31 18:00:00+00:00   284.130     67.133761             1.0
2018-12-31 19:00:00+00:00   282.640     67.132539             3.0
2018-12-31 20:00:00+00:00   282.140     67.131318             4.0
2018-12-31 21:00:00+00:00   281.130     67.130096             5.0
2018-12-31 22:00:00+00:00   280.130     67.128875             5.0

                           wind_deg_Bar  clouds_all_Bar  temp_Sev  \
time
2014-12-31 23:00:00+00:00          58.0             0.0   273.375
2015-01-01 00:00:00+00:00          58.0             0.0   273.375
2015-01-01 01:00:00+00:00          48.0             0.0   274.086
2015-01-01 02:00:00+00:00          48.0             0.0   274.086
2015-01-01 03:00:00+00:00          48.0             0.0   274.086
…                                   …             …       …
2018-12-31 18:00:00+00:00         250.0             0.0   287.760
2018-12-31 19:00:00+00:00         270.0             0.0   285.760
2018-12-31 20:00:00+00:00         300.0             0.0   285.150
2018-12-31 21:00:00+00:00         320.0             0.0   284.150
2018-12-31 22:00:00+00:00         310.0             0.0   283.970

                           humidity_Sev  wind_speed_Sev  wind_deg_Sev  \
time
2014-12-31 23:00:00+00:00     67.127653             1.0          21.0
2015-01-01 00:00:00+00:00     67.126432             1.0          21.0
2015-01-01 01:00:00+00:00     67.125210             3.0          27.0
2015-01-01 02:00:00+00:00     67.123988             3.0          27.0
2015-01-01 03:00:00+00:00     67.122767             3.0          27.0
…                                   …             …            …
2018-12-31 18:00:00+00:00     55.000000             3.0          30.0
2018-12-31 19:00:00+00:00     55.000000             3.0          30.0
2018-12-31 20:00:00+00:00     55.000000             4.0          50.0
2018-12-31 21:00:00+00:00     55.000000             4.0          60.0
2018-12-31 22:00:00+00:00     55.000000             3.0          50.0

                           clouds_all_Sev
```

```
time
2014-12-31 23:00:00+00:00          0.0
2015-01-01 00:00:00+00:00          0.0
2015-01-01 01:00:00+00:00          0.0
2015-01-01 02:00:00+00:00          0.0
2015-01-01 03:00:00+00:00          0.0
…                                 …
2018-12-31 18:00:00+00:00          0.0
2018-12-31 19:00:00+00:00          0.0
2018-12-31 20:00:00+00:00          0.0
2018-12-31 21:00:00+00:00          0.0
2018-12-31 22:00:00+00:00          0.0

[35064 rows x 45 columns]
```

[180]:
```
df_price = pd.merge(oil_price,ntrl_gas_price, left_index=True, right_index=True␣
 ↪)
```

[181]:
```
df_price = pd.merge(df_price,xchg_rate, left_index=True, right_index=True )
```

[182]:
```
df_price.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3264 entries, 2008-03-07 00:00:00+00:00 to 2021-03-06
00:00:00+00:00
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   oli_price  3263 non-null   float64
 1   NG_price   3260 non-null   float64
 2   XG_rate    3264 non-null   float64
dtypes: float64(3)
memory usage: 102.0 KB
```

We will copy the range of the data to match energy dataset

[183]:
```
df_price_fn = df_price.loc['2014-12-31':'2019-01-03'].copy()
```

[184]:
```
df_price_fn
```

[184]:
```
                           oli_price  NG_price  XG_rate
time
2014-12-31 00:00:00+00:00      53.45      3.14   1.2097
2015-01-02 00:00:00+00:00      52.72      3.01   1.2000
2015-01-05 00:00:00+00:00      50.05      3.22   1.1932
2015-01-06 00:00:00+00:00      47.98      2.98   1.1891
2015-01-07 00:00:00+00:00      48.69      3.08   1.1839
…                              …         …        …
```

```
2018-12-26 00:00:00+00:00      46.04      3.42   1.1356
2018-12-27 00:00:00+00:00      44.48      3.10   1.1429
2018-12-28 00:00:00+00:00      45.15      3.25   1.1438
2019-01-02 00:00:00+00:00      46.31      3.25   1.1338
2019-01-03 00:00:00+00:00      46.92      2.72   1.1394

[1005 rows x 3 columns]
```

[185]:
```
rng = pd.date_range(df_price_fn.index.min(), df_price_fn.index.max() + pd.
 ↪Timedelta(23, 'H'), freq='H')
df_price_fn2 = df_price_fn.reindex(rng, method='ffill')
```

[186]:
```
df_price_fn2
```

[186]:
```
                           oli_price  NG_price  XG_rate
2014-12-31 00:00:00+00:00      53.45      3.14   1.2097
2014-12-31 01:00:00+00:00      53.45      3.14   1.2097
2014-12-31 02:00:00+00:00      53.45      3.14   1.2097
2014-12-31 03:00:00+00:00      53.45      3.14   1.2097
2014-12-31 04:00:00+00:00      53.45      3.14   1.2097
…                                …        …       …
2019-01-03 19:00:00+00:00      46.92      2.72   1.1394
2019-01-03 20:00:00+00:00      46.92      2.72   1.1394
2019-01-03 21:00:00+00:00      46.92      2.72   1.1394
2019-01-03 22:00:00+00:00      46.92      2.72   1.1394
2019-01-03 23:00:00+00:00      46.92      2.72   1.1394

[35160 rows x 3 columns]
```

[187]:
```
df_price_fn3 = df_price_fn2.loc['2014-12-31 23:00:00+00:00':'2018-12-31 23:00:
 ↪00+00:00'].copy()
```

[188]:
```
df_price_fn4 = pd.merge(df_price_fn3,coal_price, left_index=True,␣
 ↪right_index=True )
```

[189]:
```
df_price_fn4
```

[189]:
```
                           oli_price  NG_price  XG_rate  coal_price
2015-01-01 00:00:00+00:00      53.45      3.14   1.2097   64.716327
2015-01-01 01:00:00+00:00      53.45      3.14   1.2097   64.716327
2015-01-01 02:00:00+00:00      53.45      3.14   1.2097   64.716327
2015-01-01 03:00:00+00:00      53.45      3.14   1.2097   64.716327
2015-01-01 04:00:00+00:00      53.45      3.14   1.2097   64.716327
…                                …        …       …          …
2018-12-31 19:00:00+00:00      45.15      3.25   1.1438  106.143609
2018-12-31 20:00:00+00:00      45.15      3.25   1.1438  106.143609
2018-12-31 21:00:00+00:00      45.15      3.25   1.1438  106.143609
```

```
2018-12-31 22:00:00+00:00      45.15      3.25    1.1438   106.143609
2018-12-31 23:00:00+00:00      45.15      3.25    1.1438   106.143609

[35064 rows x 4 columns]
```

[190]:
```
df_energy_fin = pd.merge(df_energy,df_price_fn4, left_index=True,␣
↪right_index=True )
```

[191]:
```
df_energy_fin
```

[191]:
```
                           generation biomass  \
2015-01-01 00:00:00+00:00               449.0
2015-01-01 01:00:00+00:00               448.0
2015-01-01 02:00:00+00:00               438.0
2015-01-01 03:00:00+00:00               428.0
2015-01-01 04:00:00+00:00               410.0
…                                       …
2018-12-31 18:00:00+00:00               297.0
2018-12-31 19:00:00+00:00               296.0
2018-12-31 20:00:00+00:00               292.0
2018-12-31 21:00:00+00:00               293.0
2018-12-31 22:00:00+00:00               290.0

                           generation fossil brown coal/lignite  \
2015-01-01 00:00:00+00:00                                 328.0
2015-01-01 01:00:00+00:00                                 323.0
2015-01-01 02:00:00+00:00                                 254.0
2015-01-01 03:00:00+00:00                                 187.0
2015-01-01 04:00:00+00:00                                 178.0
…                                                         …
2018-12-31 18:00:00+00:00                                   0.0
2018-12-31 19:00:00+00:00                                   0.0
2018-12-31 20:00:00+00:00                                   0.0
2018-12-31 21:00:00+00:00                                   0.0
2018-12-31 22:00:00+00:00                                   0.0

                           generation fossil gas  generation fossil hard coal  \
2015-01-01 00:00:00+00:00                 5196.0                       4755.0
2015-01-01 01:00:00+00:00                 4857.0                       4581.0
2015-01-01 02:00:00+00:00                 4314.0                       4131.0
2015-01-01 03:00:00+00:00                 4130.0                       3840.0
2015-01-01 04:00:00+00:00                 4038.0                       3590.0
…                                           …                           …
2018-12-31 18:00:00+00:00                 7634.0                       2628.0
2018-12-31 19:00:00+00:00                 7241.0                       2566.0
2018-12-31 20:00:00+00:00                 7025.0                       2422.0
2018-12-31 21:00:00+00:00                 6562.0                       2293.0
```

```
2018-12-31 22:00:00+00:00                  6926.0                      2166.0

                           generation fossil oil  \
2015-01-01 00:00:00+00:00                  158.0
2015-01-01 01:00:00+00:00                  157.0
2015-01-01 02:00:00+00:00                  160.0
2015-01-01 03:00:00+00:00                  156.0
2015-01-01 04:00:00+00:00                  156.0
…                                          …
2018-12-31 18:00:00+00:00                  178.0
2018-12-31 19:00:00+00:00                  174.0
2018-12-31 20:00:00+00:00                  168.0
2018-12-31 21:00:00+00:00                  163.0
2018-12-31 22:00:00+00:00                  163.0

                           generation hydro pumped storage consumption  \
2015-01-01 00:00:00+00:00                                        920.0
2015-01-01 01:00:00+00:00                                       1164.0
2015-01-01 02:00:00+00:00                                       1503.0
2015-01-01 03:00:00+00:00                                       1826.0
2015-01-01 04:00:00+00:00                                       2109.0
…                                                                …
2018-12-31 18:00:00+00:00                                          1.0
2018-12-31 19:00:00+00:00                                          1.0
2018-12-31 20:00:00+00:00                                         50.0
2018-12-31 21:00:00+00:00                                        108.0
2018-12-31 22:00:00+00:00                                        108.0

                           generation hydro run-of-river and poundage  \
2015-01-01 00:00:00+00:00                                      1009.0
2015-01-01 01:00:00+00:00                                       973.0
2015-01-01 02:00:00+00:00                                       949.0
2015-01-01 03:00:00+00:00                                       953.0
2015-01-01 04:00:00+00:00                                       952.0
…                                                               …
2018-12-31 18:00:00+00:00                                      1135.0
2018-12-31 19:00:00+00:00                                      1172.0
2018-12-31 20:00:00+00:00                                      1148.0
2018-12-31 21:00:00+00:00                                      1128.0
2018-12-31 22:00:00+00:00                                      1069.0

                           generation hydro water reservoir  \
2015-01-01 00:00:00+00:00                            1658.0
2015-01-01 01:00:00+00:00                            1371.0
2015-01-01 02:00:00+00:00                             779.0
2015-01-01 03:00:00+00:00                             720.0
2015-01-01 04:00:00+00:00                             743.0
```

```
…                                                       …
2018-12-31 18:00:00+00:00                            4836.0
2018-12-31 19:00:00+00:00                            3931.0
2018-12-31 20:00:00+00:00                            2831.0
2018-12-31 21:00:00+00:00                            2068.0
2018-12-31 22:00:00+00:00                            1686.0

                           generation nuclear  generation other  …  \
2015-01-01 00:00:00+00:00              7096.0              43.0  …
2015-01-01 01:00:00+00:00              7099.0              43.0  …
2015-01-01 02:00:00+00:00              7098.0              43.0  …
2015-01-01 03:00:00+00:00              7097.0              43.0  …
2015-01-01 04:00:00+00:00              7098.0              43.0  …
…                                         …               …  …
2018-12-31 18:00:00+00:00              6073.0              63.0  …
2018-12-31 19:00:00+00:00              6074.0              62.0  …
2018-12-31 20:00:00+00:00              6076.0              61.0  …
2018-12-31 21:00:00+00:00              6075.0              61.0  …
2018-12-31 22:00:00+00:00              6075.0              61.0  …

                           clouds_all_Bar  temp_Sev  humidity_Sev  \
2015-01-01 00:00:00+00:00             0.0   273.375     67.126432
2015-01-01 01:00:00+00:00             0.0   274.086     67.125210
2015-01-01 02:00:00+00:00             0.0   274.086     67.123988
2015-01-01 03:00:00+00:00             0.0   274.086     67.122767
2015-01-01 04:00:00+00:00             0.0   274.592     67.121545
…                                        …         …           …
2018-12-31 18:00:00+00:00             0.0   287.760     55.000000
2018-12-31 19:00:00+00:00             0.0   285.760     55.000000
2018-12-31 20:00:00+00:00             0.0   285.150     55.000000
2018-12-31 21:00:00+00:00             0.0   284.150     55.000000
2018-12-31 22:00:00+00:00             0.0   283.970     55.000000

                           wind_speed_Sev  wind_deg_Sev  clouds_all_Sev  \
2015-01-01 00:00:00+00:00             1.0          21.0             0.0
2015-01-01 01:00:00+00:00             3.0          27.0             0.0
2015-01-01 02:00:00+00:00             3.0          27.0             0.0
2015-01-01 03:00:00+00:00             3.0          27.0             0.0
2015-01-01 04:00:00+00:00             4.0          57.0             0.0
…                                        …             …               …
2018-12-31 18:00:00+00:00             3.0          30.0             0.0
2018-12-31 19:00:00+00:00             3.0          30.0             0.0
2018-12-31 20:00:00+00:00             4.0          50.0             0.0
2018-12-31 21:00:00+00:00             4.0          60.0             0.0
2018-12-31 22:00:00+00:00             3.0          50.0             0.0

                           oli_price  NG_price  XG_rate  coal_price
```

```
2015-01-01 00:00:00+00:00      53.45      3.14   1.2097   64.716327
2015-01-01 01:00:00+00:00      53.45      3.14   1.2097   64.716327
2015-01-01 02:00:00+00:00      53.45      3.14   1.2097   64.716327
2015-01-01 03:00:00+00:00      53.45      3.14   1.2097   64.716327
2015-01-01 04:00:00+00:00      53.45      3.14   1.2097   64.716327
…                              …        …       …          …
2018-12-31 18:00:00+00:00      45.15      3.25   1.1438  106.143609
2018-12-31 19:00:00+00:00      45.15      3.25   1.1438  106.143609
2018-12-31 20:00:00+00:00      45.15      3.25   1.1438  106.143609
2018-12-31 21:00:00+00:00      45.15      3.25   1.1438  106.143609
2018-12-31 22:00:00+00:00      45.15      3.25   1.1438  106.143609

[35063 rows x 49 columns]
```

[192]:
```
df_energy_fin.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 35063 entries, 2015-01-01 00:00:00+00:00 to 2018-12-31
22:00:00+00:00
Data columns (total 49 columns):
 #   Column                                       Non-Null Count  Dtype
---  ------                                       --------------  -----
 0   generation biomass                           35063 non-null  float64
 1   generation fossil brown coal/lignite         35063 non-null  float64
 2   generation fossil gas                        35063 non-null  float64
 3   generation fossil hard coal                  35063 non-null  float64
 4   generation fossil oil                        35063 non-null  float64
 5   generation hydro pumped storage consumption  35063 non-null  float64
 6   generation hydro run-of-river and poundage   35063 non-null  float64
 7   generation hydro water reservoir             35063 non-null  float64
 8   generation nuclear                           35063 non-null  float64
 9   generation other                             35063 non-null  float64
 10  generation other renewable                   35063 non-null  float64
 11  generation solar                             35063 non-null  float64
 12  generation waste                             35063 non-null  float64
 13  generation wind onshore                      35063 non-null  float64
 14  forecast solar day ahead                     35063 non-null  float64
 15  forecast wind onshore day ahead              35063 non-null  float64
 16  total load forecast                          35063 non-null  float64
 17  total load actual                            35063 non-null  float64
 18  price day ahead                              35063 non-null  float64
 19  price actual                                 35063 non-null  float64
 20  temp_Val                                     35063 non-null  float64
 21  humidity_Val                                 35063 non-null  float64
 22  wind_speed_Val                               35063 non-null  float64
 23  wind_deg_Val                                 35063 non-null  float64
 24  clouds_all_Val                               35063 non-null  float64
 25  temp_Mad                                     35063 non-null  float64
```

```
 26  humidity_Mad                                   35063 non-null  float64
 27  wind_speed_Mad                                 35063 non-null  float64
 28  wind_deg_Mad                                   35063 non-null  float64
 29  clouds_all_Mad                                 35063 non-null  float64
 30  temp_Bil                                       35063 non-null  float64
 31  humidity_Bil                                   35063 non-null  float64
 32  wind_speed_Bil                                 35063 non-null  float64
 33  wind_deg_Bil                                   35063 non-null  float64
 34  clouds_all_Bil                                 35063 non-null  float64
 35  temp_Bar                                       35063 non-null  float64
 36  humidity_Bar                                   35063 non-null  float64
 37  wind_speed_Bar                                 35063 non-null  float64
 38  wind_deg_Bar                                   35063 non-null  float64
 39  clouds_all_Bar                                 35063 non-null  float64
 40  temp_Sev                                       35063 non-null  float64
 41  humidity_Sev                                   35063 non-null  float64
 42  wind_speed_Sev                                 35063 non-null  float64
 43  wind_deg_Sev                                   35063 non-null  float64
 44  clouds_all_Sev                                 35063 non-null  float64
 45  oli_price                                      35063 non-null  float64
 46  NG_price                                       35063 non-null  float64
 47  XG_rate                                        35063 non-null  float64
 48  coal_price                                     35063 non-null  float64
dtypes: float64(49)
memory usage: 13.4 MB
```

[193]:
```
df_energy_fin.describe()
```

[193]:
```
       generation biomass  generation fossil brown coal/lignite  \
count        35063.000000                           35063.000000
mean           383.529533                             448.097967
std             85.346810                             354.622756
min              0.000000                               0.000000
25%            333.000000                               0.000000
50%            367.000000                             509.000000
75%            433.000000                             757.000000
max            592.000000                             999.000000

       generation fossil gas  generation fossil hard coal  \
count           35063.000000                 35063.000000
mean             5622.722856                  4256.515173
std              2201.538451                  1962.014599
min                 0.000000                     0.000000
25%              4126.000000                  2527.000000
50%              4970.000000                  4475.000000
75%              6429.000000                  5839.000000
max             20034.000000                  8359.000000
```

	generation fossil oil	generation hydro pumped storage consumption
count	35063.000000	35063.000000
mean	298.346305	475.571657
std	52.515628	792.321301
min	0.000000	0.000000
25%	263.000000	0.000000
50%	300.000000	68.000000
75%	330.000000	616.000000
max	449.000000	4523.000000

	generation hydro run-of-river and poundage
count	35063.000000
mean	972.199655
std	400.717797
min	0.000000
25%	637.000000
50%	906.000000
75%	1250.000000
max	2000.000000

	generation hydro water reservoir	generation nuclear	generation other
count	35063.000000	35063.000000	35063.000000
mean	2605.554274	6263.459687	60.226521
std	1835.197370	840.272553	20.238871
min	0.000000	0.000000	0.000000
25%	1078.000000	5759.000000	53.000000
50%	2165.000000	6564.000000	57.000000
75%	3758.000000	7025.000000	80.000000
max	9728.000000	7117.000000	106.000000

	…	clouds_all_Bar	temp_Sev	humidity_Sev	wind_speed_Sev
count	…	35063.000000	35063.000000	35063.000000	35063.000000
mean	…	22.715341	293.167106	56.709694	2.482760
std	…	27.328563	8.081899	3.305923	1.868513
min	…	0.000000	271.050000	55.000000	0.000000
25%	…	0.000000	287.330000	55.000000	1.000000
50%	…	20.000000	292.440000	55.000000	2.000000
75%	…	36.000000	298.880000	56.352878	3.000000
max	…	100.000000	315.600000	67.126432	15.000000

	wind_deg_Sev	clouds_all_Sev	oli_price	NG_price	XG_rate
count	35063.000000	35063.000000	35063.000000	35063.000000	35063.000000
mean	151.889542	14.165474	51.941631	2.821767	1.132108
std	104.327901	26.169960	10.114652	0.529946	0.047399
min	0.000000	0.000000	26.190000	1.490000	1.038800
25%	50.000000	0.000000	45.690000	2.630000	1.097700

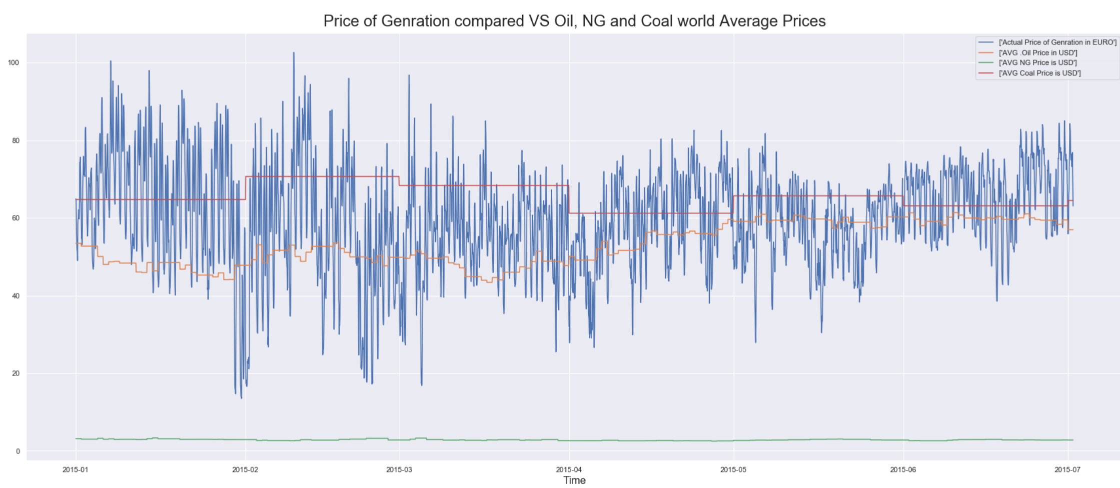
```
50%      175.000000        0.000000     49.900000       2.850000       1.127000
75%      230.000000       20.000000     59.110000       3.020000       1.166000
max      360.000000      100.000000     77.410000       6.240000       1.251000

         coal_price
count  35063.000000
mean      85.042032
std       23.077124
min       53.428929
25%       63.393214
50%       86.333851
75%      105.728061
max      125.085877

[8 rows x 49 columns]
```

[194]:
```
print(df_energy_fin.isnull().sum())
```

```
generation biomass                             0
generation fossil brown coal/lignite           0
generation fossil gas                          0
generation fossil hard coal                    0
generation fossil oil                          0
generation hydro pumped storage consumption    0
generation hydro run-of-river and poundage     0
generation hydro water reservoir               0
generation nuclear                             0
generation other                               0
generation other renewable                     0
generation solar                               0
generation waste                               0
generation wind onshore                        0
forecast solar day ahead                       0
forecast wind onshore day ahead                0
total load forecast                            0
total load actual                              0
price day ahead                                0
price actual                                   0
temp_Val                                       0
humidity_Val                                   0
wind_speed_Val                                 0
wind_deg_Val                                   0
clouds_all_Val                                 0
temp_Mad                                       0
humidity_Mad                                   0
wind_speed_Mad                                 0
wind_deg_Mad                                   0
clouds_all_Mad                                 0
```

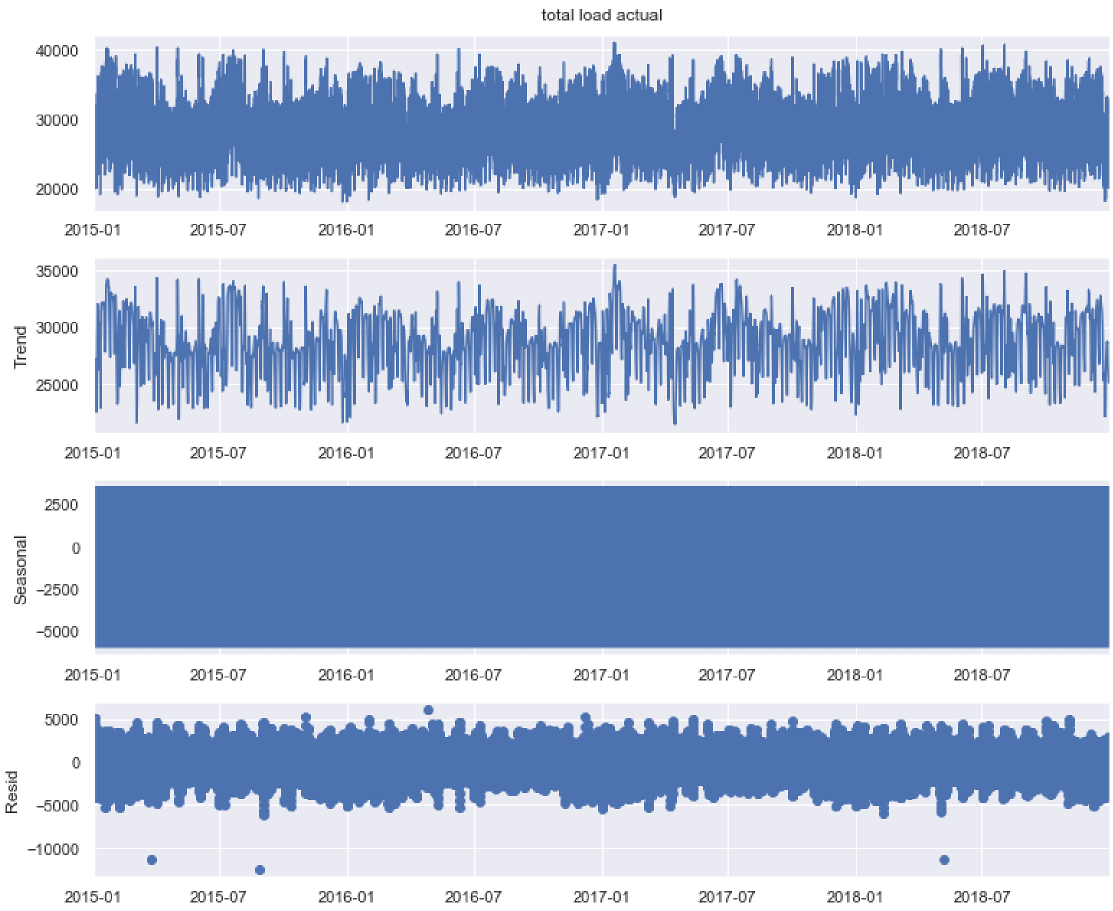
```
temp_Bil                                               0
humidity_Bil                                           0
wind_speed_Bil                                         0
wind_deg_Bil                                           0
clouds_all_Bil                                         0
temp_Bar                                               0
humidity_Bar                                           0
wind_speed_Bar                                         0
wind_deg_Bar                                           0
clouds_all_Bar                                         0
temp_Sev                                               0
humidity_Sev                                           0
wind_speed_Sev                                         0
wind_deg_Sev                                           0
clouds_all_Sev                                         0
oli_price                                              0
NG_price                                               0
XG_rate                                                0
coal_price                                             0
dtype: int64
```

[195]:
```python
# Plting the Price of the MWH against the prices of the fuels to see if we can␣
↪spot any relation just by looking to the plot
ax = plot_series(df=df_energy_fin, column=['price␣
↪actual','oli_price','NG_price','coal_price'],
                 title='Price of Genration compared VS Oil, NG and Coal world␣
↪Average Prices', end=24*7*26)
ax.legend((['Actual Price of Genration in EURO'], ['AVG .Oil Price in␣
↪USD'],['AVG NG Price is USD'], ['AVG Coal Price is USD']))
plt.show()
```



1.5 Predection using Univariate ARIMA model for the load

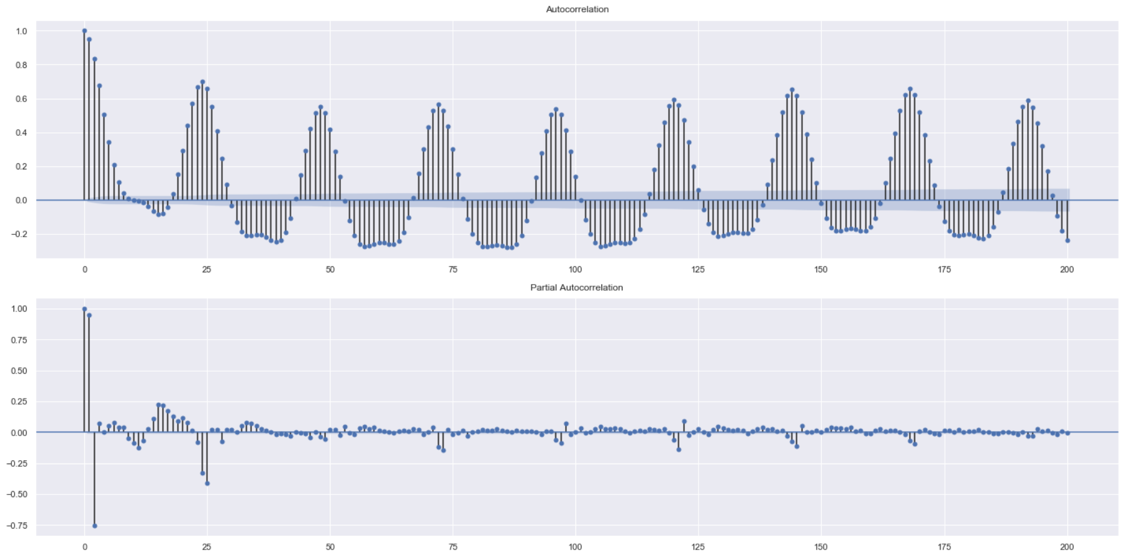
[682]:
```
#Looking at the decompostion plot
rcParams['figure.figsize'] = 11, 9
decomp = seasonal_decompose(df_energy_fin['total load actual'])
fig = decomp.plot()
fig = plt.figure()
plt.show()
```




```
<Figure size 792x648 with 0 Axes>
```

From the above Graphs it seems tha there is no increasing or deacresing trend in the load and the data seems to be a stationary

[687]:
```
fig, (ax1, ax2) = plt.subplots(nrows=2, figsize=(20, 10))
plot_acf(df_energy_fin['total load actual'], lags=200, ax=ax1)
plot_pacf(df_energy_fin['total load actual'], lags=200, ax=ax2)
plt.tight_layout()
plt.show()
```



ACF and PACF for the Actual load , respectively indicate that there is a high autocorrelation between the values in an oscillating manne.

[329]:
```
df_load_forcast = pd.DataFrame(df_energy_fin)
df_load_forcast.drop(columns=['total load forecast'], inplace = True)
```

[202]:
```
df_load_forcast.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 35063 entries, 2015-01-01 00:00:00+00:00 to 2018-12-31
22:00:00+00:00
Data columns (total 48 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   generation biomass                          35063 non-null  float64
 1   generation fossil brown coal/lignite        35063 non-null  float64
 2   generation fossil gas                       35063 non-null  float64
 3   generation fossil hard coal                 35063 non-null  float64
 4   generation fossil oil                       35063 non-null  float64
 5   generation hydro pumped storage consumption 35063 non-null  float64
 6   generation hydro run-of-river and poundage  35063 non-null  float64
 7   generation hydro water reservoir            35063 non-null  float64
 8   generation nuclear                          35063 non-null  float64
 9   generation other                            35063 non-null  float64
 10  generation other renewable                  35063 non-null  float64
 11  generation solar                            35063 non-null  float64
 12  generation waste                            35063 non-null  float64
 13  generation wind onshore                     35063 non-null  float64
 14  forecast solar day ahead                    35063 non-null  float64
 15  forecast wind onshore day ahead             35063 non-null  float64
```

```
 16  total load actual                             35063 non-null  float64
 17  price day ahead                               35063 non-null  float64
 18  price actual                                  35063 non-null  float64
 19  temp_Val                                      35063 non-null  float64
 20  humidity_Val                                  35063 non-null  float64
 21  wind_speed_Val                                35063 non-null  float64
 22  wind_deg_Val                                  35063 non-null  float64
 23  clouds_all_Val                                35063 non-null  float64
 24  temp_Mad                                      35063 non-null  float64
 25  humidity_Mad                                  35063 non-null  float64
 26  wind_speed_Mad                                35063 non-null  float64
 27  wind_deg_Mad                                  35063 non-null  float64
 28  clouds_all_Mad                                35063 non-null  float64
 29  temp_Bil                                      35063 non-null  float64
 30  humidity_Bil                                  35063 non-null  float64
 31  wind_speed_Bil                                35063 non-null  float64
 32  wind_deg_Bil                                  35063 non-null  float64
 33  clouds_all_Bil                                35063 non-null  float64
 34  temp_Bar                                      35063 non-null  float64
 35  humidity_Bar                                  35063 non-null  float64
 36  wind_speed_Bar                                35063 non-null  float64
 37  wind_deg_Bar                                  35063 non-null  float64
 38  clouds_all_Bar                                35063 non-null  float64
 39  temp_Sev                                      35063 non-null  float64
 40  humidity_Sev                                  35063 non-null  float64
 41  wind_speed_Sev                                35063 non-null  float64
 42  wind_deg_Sev                                  35063 non-null  float64
 43  clouds_all_Sev                                35063 non-null  float64
 44  oli_price                                     35063 non-null  float64
 45  NG_price                                      35063 non-null  float64
 46  XG_rate                                       35063 non-null  float64
 47  coal_price                                    35063 non-null  float64
dtypes: float64(48)
memory usage: 13.1 MB
```

[494]:
```
df_load_arima  = pd.DataFrame(df_load_forcast['total load actual'])
y = pd.DataFrame(df_load_arima['total load actual'])
```

[494]:
```
total load actual    0
dtype: int64
```

[442]:
```
# Using AD Fuller test to check for stationary  data
from statsmodels.tsa.stattools import adfuller

def adf_test(dataset):
  dftest = adfuller(dataset, autolag = 'AIC')
  print("1. ADF : ",dftest[0])
  print("2. P-Value : ", dftest[1])
```

```python
    print("3. Num Of Lags : ", dftest[2])
    print("4. Num Of Observations Used For ADF Regression and Critical Values␣
↪Calculation :", dftest[3])
    print("5. Critical Values :")
    for key, val in dftest[4].items():
        print("\t",key, ": ", val)
```

[448]:
```python
adf_test(y['total load actual'])
```

```
1. ADF :  -21.418561804776697
2. P-Value :  0.0
3. Num Of Lags :  52
4. Num Of Observations Used For ADF Regression and Critical Values Calculation :
35010
5. Critical Values :
         1% :  -3.430536797472945
         5% :  -2.8616225598677394
         10% :  -2.5668139440226856
```

As P is 0 then the data set is stationary and no need to apply the differencing on it

[450]:
```python
#Trying to find the lowes  AIC for the pdq values of the ARIMA model so first␣
↪we will try the Auto Arima method
from pmdarima import auto_arima
stepwise_fit = auto_arima(y['total load actual'], trace=True,
suppress_warnings=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=578763.360, Time=89.74 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=608930.956, Time=1.71 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=584632.091, Time=4.12 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=590377.152, Time=19.53 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=608928.956, Time=1.07 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=583251.906, Time=15.04 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=578852.653, Time=59.02 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=578764.747, Time=99.47 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=578758.573, Time=95.10 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=583084.316, Time=23.34 sec
 ARIMA(3,1,3)(0,0,0)[0] intercept   : AIC=578791.046, Time=92.28 sec
 ARIMA(2,1,4)(0,0,0)[0] intercept   : AIC=577432.855, Time=102.54 sec
 ARIMA(1,1,4)(0,0,0)[0] intercept   : AIC=inf, Time=81.73 sec
 ARIMA(3,1,4)(0,0,0)[0] intercept   : AIC=inf, Time=119.75 sec
 ARIMA(2,1,5)(0,0,0)[0] intercept   : AIC=inf, Time=124.40 sec
 ARIMA(1,1,5)(0,0,0)[0] intercept   : AIC=inf, Time=100.97 sec
 ARIMA(3,1,5)(0,0,0)[0] intercept   : AIC=inf, Time=138.28 sec
 ARIMA(2,1,4)(0,0,0)[0]             : AIC=577430.801, Time=35.51 sec
 ARIMA(1,1,4)(0,0,0)[0]             : AIC=inf, Time=40.69 sec
 ARIMA(2,1,3)(0,0,0)[0]             : AIC=578793.750, Time=14.30 sec
 ARIMA(3,1,4)(0,0,0)[0]             : AIC=inf, Time=50.57 sec
```

```
 ARIMA(2,1,5)(0,0,0)[0]             : AIC=inf, Time=52.83 sec
 ARIMA(1,1,3)(0,0,0)[0]             : AIC=583082.316, Time=9.72 sec
 ARIMA(1,1,5)(0,0,0)[0]             : AIC=inf, Time=22.27 sec
 ARIMA(3,1,3)(0,0,0)[0]             : AIC=578774.486, Time=43.64 sec
 ARIMA(3,1,5)(0,0,0)[0]             : AIC=inf, Time=58.72 sec

Best model:  ARIMA(2,1,4)(0,0,0)[0]
Total fit time: 1496.426 seconds
```

[445]:

```
#Then we will try t compute the PDQ throgh testing the the data for the lowes␣
 ↪AIC
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
```

[446]:

```
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d,␣
 ↪q))]
```

[447]:

```
print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA…
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

[455]:

```
warnings.filterwarnings("ignore")
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False, freq=␣
 ↪'H')

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.
 ↪aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:820178.6805117446
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:800664.6535651962
```

```
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:715204.2845748094
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:685838.9985359984
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:714783.1217399709
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:695502.918763745
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:669241.8416563898
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:659910.4944190428
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:794021.0355602893
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:788798.2699052686
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:687589.1901942658
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:645672.3519714368
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:785227.1772129472
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:784938.806365282
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:629692.0473348373
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:621354.6390478195
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:608911.2584422461
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:608532.0047119465
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:628697.605957422
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:591253.1573766256
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:608257.7862475851
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:594891.5395960734
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:573976.4747494131
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:573765.1324649397
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:590279.1454521646
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:590272.481597959
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:611942.8917654434
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:577428.2671584324
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:590174.8079914463
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:577636.4399254421
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:570125.590462947
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:569522.3018574824
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:608909.7611971542
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:608527.3500220331
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:627966.6263314241
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:590754.826910983
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:608236.342541166
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:595076.8618569295
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:573398.561211734
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:573245.9881244474
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:590381.8826668598
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:590076.3550736893
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:610019.2808955191
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:576570.0548547711
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:589952.4702812466
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:580234.3261480371
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:569244.9759834239
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:568750.6231474527
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:584485.2429516664
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:583879.0109161588
```

```
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:602496.6512712637
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:572174.653837
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:583409.5173444976
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:574203.0514621738
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:569478.8309671521
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:568639.1892915913
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:583309.0973008897
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:582911.8974641787
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:602475.8957381805
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:572143.2988505333
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:582698.0774687254
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:573908.8412251133
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:569479.0748083513
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:568623.2164110676
```

The best lowes AIC is 568623.216 from PDQ (1,1,1) (1,1,1,12)

1.5.1 ARIMA Model Fitting

[457]:
```python
#Fit the ARIMA model
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.4329      0.008     57.062      0.000       0.418       0.448
ma.L1         -0.0168      0.008     -2.225      0.026      -0.032      -0.002
ar.S.L12      -0.6922      0.002   -334.954      0.000      -0.696      -0.688
ma.S.L12      -0.3594      0.003   -121.847      0.000      -0.365      -0.354
sigma2      6.543e+05   1483.185    441.138      0.000    6.51e+05    6.57e+05
==============================================================================
```

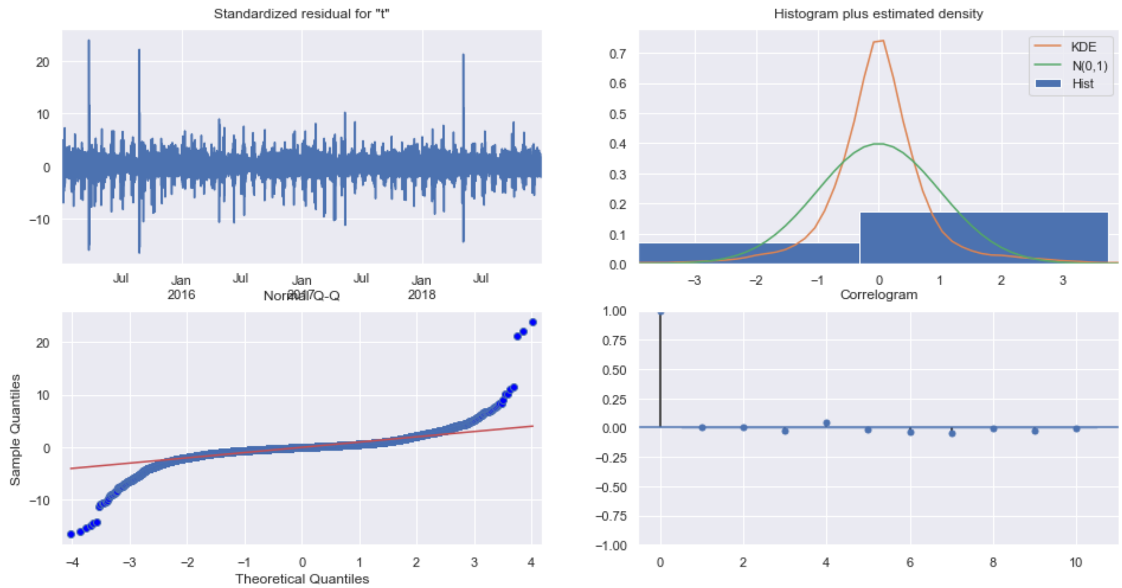
[458]:
```python
print(results.summary())
```

```
                                     SARIMAX Results
================================================================================
==========
Dep. Variable:                 total load actual   No. Observations:
35063
Model:             SARIMAX(1, 1, 1)x(1, 1, 1, 12)   Log Likelihood
-284306.608
```

```
Date:                             Mon, 08 Mar 2021   AIC
568623.216
Time:                                     22:45:36   BIC
568665.537
Sample:                                 01-01-2015   HQIC
568636.696
                                      - 12-31-2018
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.4329      0.008     57.062      0.000       0.418       0.448
ma.L1         -0.0168      0.008     -2.225      0.026      -0.032      -0.002
ar.S.L12      -0.6922      0.002   -334.954      0.000      -0.696      -0.688
ma.S.L12      -0.3594      0.003   -121.847      0.000      -0.365      -0.354
sigma2      6.543e+05   1483.185    441.138      0.000    6.51e+05    6.57e+05
================================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
2812621.68
Prob(Q):                              1.00   Prob(JB):
0.00
Heteroskedasticity (H):               0.78   Skew:
0.17
Prob(H) (two-sided):                  0.00   Kurtosis:
46.89
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```

[479]:

```
#Running model diagnostic to investigate unusual behaviour
results.plot_diagnostics(figsize=(16, 8))
plt.show()
```



So how to interpret the plot diagnostics?

Top left: The residual errors seem to fluctuate around a mean of zero and have a uniform variance.

Top Right: The density plot suggest normal distribution with mean zero.

Bottom left: The middle values of the sample are close to what we would expect from normally distributed data, as it follows the straight line from the diagram closely.However, it seems the underlying data distribution presents extreme values more often than a normal one, that's why you see the points going under the line for big negative values and over it for big positive ones.

Bottom Right: The Correlogram, aka, ACF plot shows the residual errors are not autocorrelated. Any autocorrelation would imply that there is some pattern in the residual errors which are not explained in the model. So you will need to look for more X's (predictors) to the model.

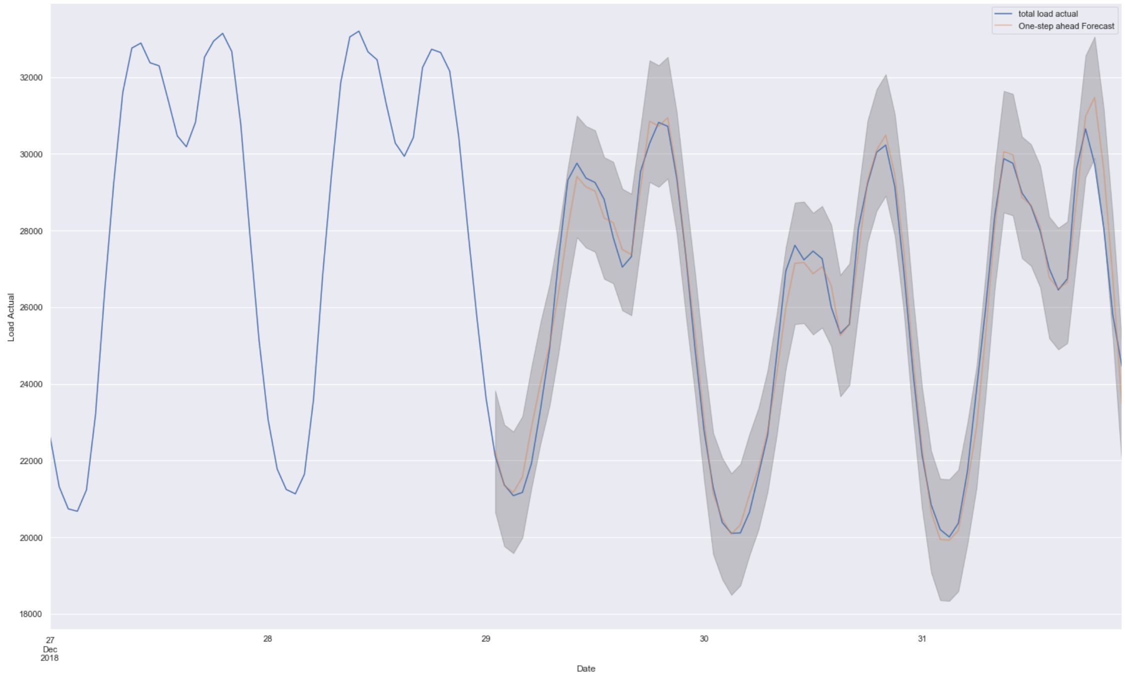
Those observations lead us to conclude that our model produces a satisfactory fit that could help us understand our time series data and forecast future values.

Predection

[490]:
```
pred = results.get_prediction(start=pd.to_datetime('2018-12-29 01:00:00+00:
→00'), dynamic=False)
pred_ci = pred.conf_int()
```

[491]:
```
ax = y['2018-12-27 00:00:00+00:00':].plot(label='Actual Load')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.5,␣
→figsize=(25, 15))

ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.2)
```

```
ax.set_xlabel('Date')
ax.set_ylabel('Load Actual')
plt.legend()
plt.show()
```



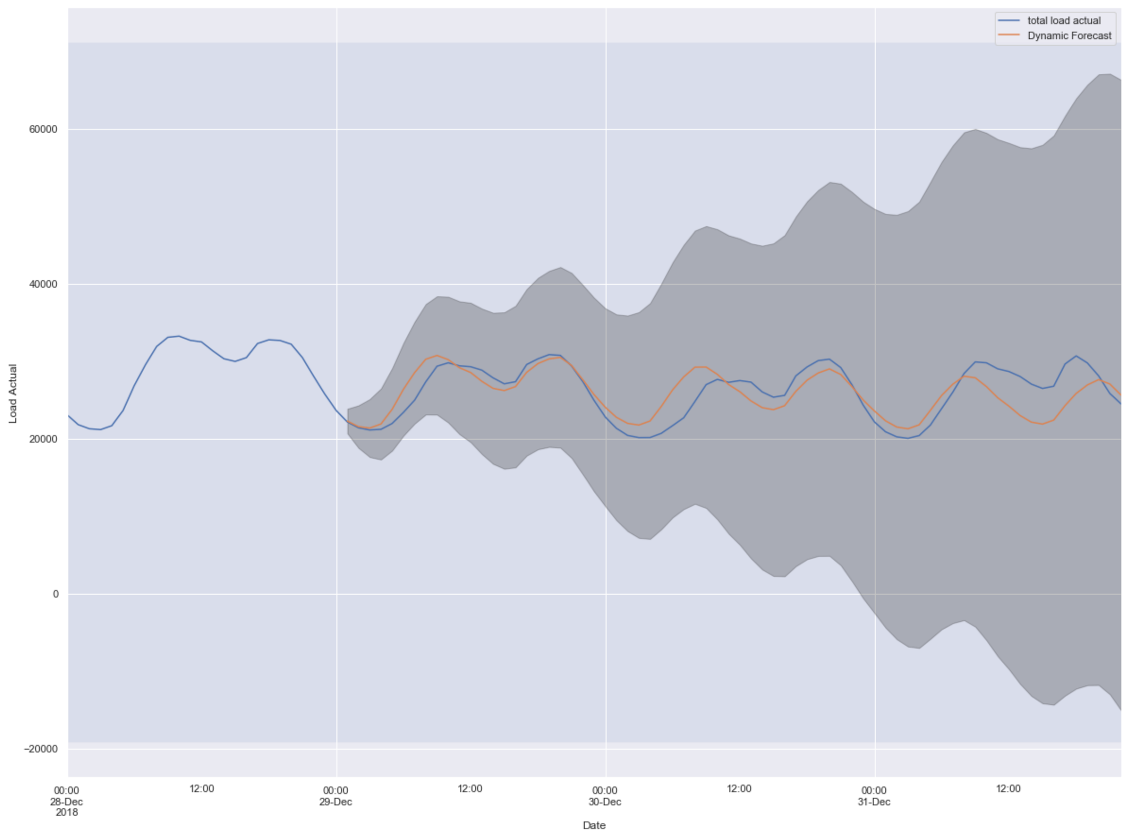
[513]:
```
y_forecasted = pred.predicted_mean
y_truth = y['2018-12-29 0:00:00+00:00':]

# Compute the mean square error
mse = ((y_forecasted - y_truth['total load actual']) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
```

```
The Mean Squared Error of our forecasts is 271776.81
```

[476]:
```
pred_dynamic = results.get_prediction(start=pd.to_datetime('2018-12-29 01:00:
↪00+00:00'), dynamic=True, full_results=True)
pred_dynamic_ci = pred_dynamic.conf_int()
```

[477]:
```
ax = y['2018-12-28 00:00:00+00:00':].plot(label='observed', figsize=(20, 15))
pred_dynamic.predicted_mean.plot(label='Dynamic Forecast', ax=ax)

ax.fill_between(pred_dynamic_ci.index,
                pred_dynamic_ci.iloc[:, 0],
                pred_dynamic_ci.iloc[:, 1], color='k', alpha=.25)
```

```
ax.fill_betweenx(ax.get_ylim(), pd.to_datetime('2017-12-30 00:00:00+00:00'), y.
↪index[-1],
                 alpha=.1, zorder=-1)

ax.set_xlabel('Date')
ax.set_ylabel('Load Actual')

plt.legend()
plt.show()
```




```
[517]: # Extract the predicted and true values of our time series
y_forecasted = pred_dynamic.predicted_mean
y_truth = y['2018-12-29 01:00:00+00:00': ]
```

The Mean Squared Error of our forecasts is 5568514.6

```
[516]: mse = mean_squared_error(y_truth, y_forecasted)
print('MSE: '+str(mse))
mae = mean_absolute_error(y_truth, y_forecasted)
print('MAE: '+str(mae))
```

```
rmse = math.sqrt(mean_squared_error(y_truth, y_forecasted))
print('RMSE: '+str(rmse))
mape = np.mean(np.abs(y_forecasted - y_truth['total load actual'])/np.
 ↪abs(y_truth['total load actual']))
print('MAPE: '+str(mape))
```

```
MSE: 5568514.601035193
MAE: 1842.9796334341083
RMSE: 2359.7700313876335
MAPE: 0.07207225388458076
```

1.6 Forcasting the Cost of the MWH with Tensorflow LSTM

In the following section we will explore the full dataset and the target variable whi is the cost of the MW through the following steps

1- Finding the coorelations to the target variable

2- Understaing the features and selecting what will be included in the mdoel

[146]:
```
# Testing the Price of MW with ADF
y = df_energy_fin['price actual']
adf_test = adfuller(y, regression='c')
print('ADF Statistic: {:.6f}\np-value: {:.6f}\n#Lags used: {}'
      .format(adf_test[0], adf_test[1], adf_test[2]))
for key, value in adf_test[4].items():
    print('Critical Value ({}): {:.6f}'.format(key, value))
```

```
ADF Statistic: -9.148003
p-value: 0.000000
#Lags used: 50
Critical Value (1%): -3.430537
Critical Value (5%): -2.861623
Critical Value (10%): -2.566814
```

Since the P value is 0 the dataset is sattionary

[147]:
```
# Find the correlations between the energy price and the rest of the features

correlations = df_energy_fin.corr(method='pearson')
print(correlations['price actual'].sort_values(ascending=False).to_string())
```

```
price actual                                   1.000000
price day ahead                                0.732158
generation fossil hard coal                    0.465635
generation fossil gas                          0.461460
total load forecast                            0.435876
total load actual                              0.435269
oli_price                                      0.383089
coal_price                                     0.378999
generation fossil brown coal/lignite           0.364000
```

```
NG_price                                       0.360450
generation fossil oil                          0.285118
generation other renewable                     0.255568
humidity_Val                                   0.212435
generation waste                               0.168738
generation biomass                             0.142662
humidity_Sev                                   0.106835
forecast solar day ahead                       0.101417
generation other                               0.099928
generation solar                               0.098542
temp_Val                                       0.090558
temp_Mad                                       0.088035
temp_Bar                                       0.085877
temp_Bil                                       0.073061
generation hydro water reservoir               0.071916
temp_Sev                                       0.050315
clouds_all_Val                                 0.040068
XG_rate                                        0.018651
clouds_all_Bar                                -0.027587
generation nuclear                            -0.053032
wind_speed_Sev                                -0.078458
clouds_all_Mad                                -0.079405
wind_deg_Mad                                  -0.082775
clouds_all_Sev                                -0.086226
wind_deg_Val                                  -0.092699
wind_deg_Bar                                  -0.096232
wind_deg_Bil                                  -0.103106
clouds_all_Bil                                -0.132653
generation hydro run-of-river and poundage    -0.136663
wind_deg_Sev                                  -0.137083
wind_speed_Bar                                -0.138699
humidity_Bil                                  -0.141645
wind_speed_Bil                                -0.143314
wind_speed_Val                                -0.145730
generation wind onshore                       -0.220503
forecast wind onshore day ahead               -0.221711
humidity_Mad                                  -0.229526
wind_speed_Mad                                -0.245852
humidity_Bar                                  -0.315769
generation hydro pumped storage consumption   -0.426206
```

[148]:
```
print(correlations['total load actual'].sort_values(ascending=False).
 ↪to_string())
```

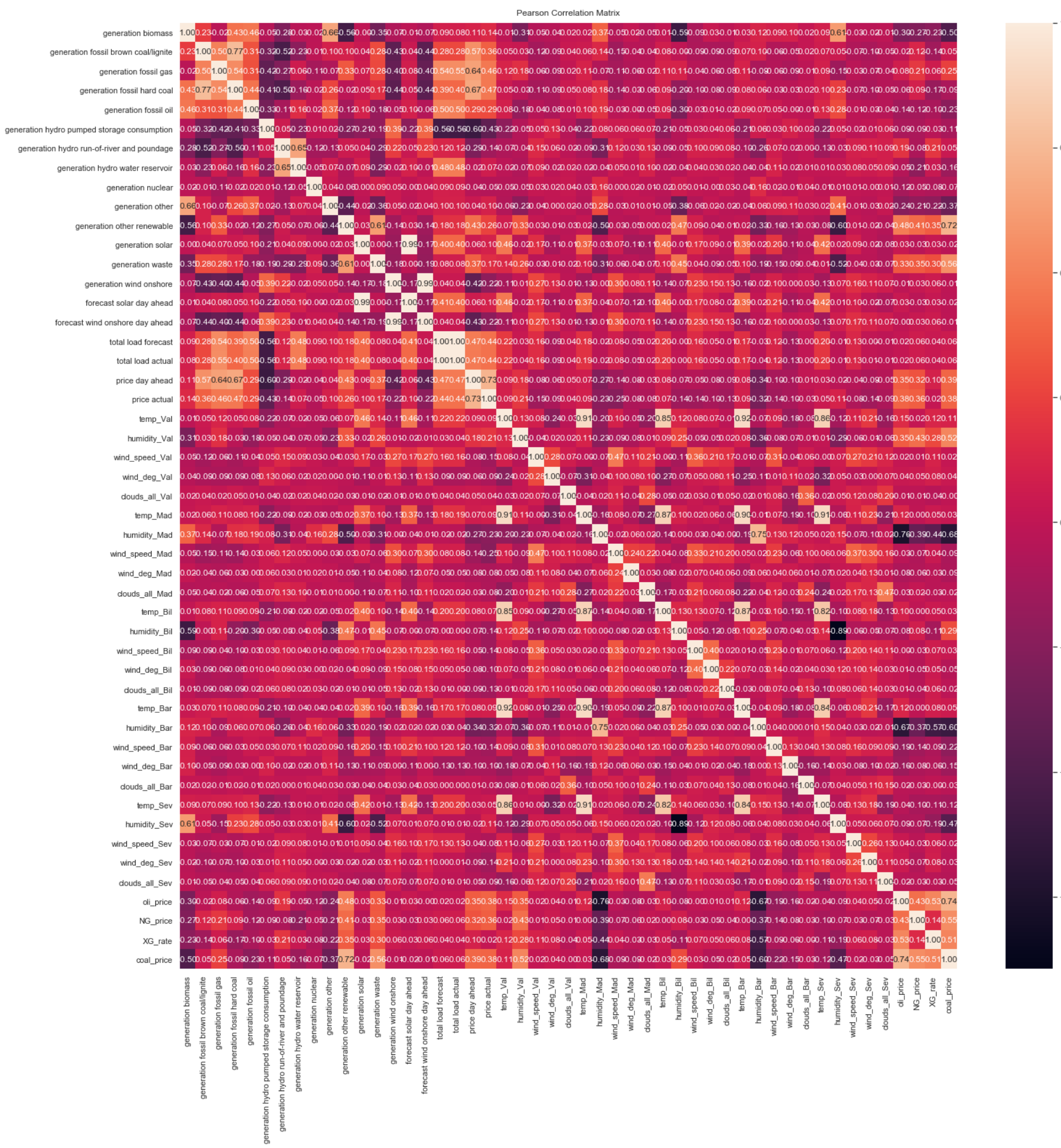
```
total load actual                              1.000000
total load forecast                            0.995096
generation fossil gas                          0.548983
generation fossil oil                          0.496136
```

generation hydro water reservoir	0.479488
price day ahead	0.474277
price actual	0.435269
forecast solar day ahead	0.403825
generation fossil hard coal	0.397088
generation solar	0.395501
generation fossil brown coal/lignite	0.280731
temp_Val	0.220758
temp_Sev	0.204539
temp_Bil	0.196503
temp_Mad	0.185356
generation other renewable	0.180799
temp_Bar	0.167318
wind_speed_Val	0.157213
wind_speed_Bil	0.155880
wind_speed_Sev	0.132959
wind_speed_Bar	0.120772
generation hydro run-of-river and poundage	0.118487
generation other	0.100651
generation nuclear	0.086108
wind_speed_Mad	0.083655
generation biomass	0.083523
generation waste	0.076944
NG_price	0.063514
coal_price	0.061192
wind_deg_Bil	0.050119
clouds_all_Val	0.040345
XG_rate	0.039712
generation wind onshore	0.039674
humidity_Val	0.038178
forecast wind onshore day ahead	0.037243
clouds_all_Mad	0.022158
oli_price	0.017605
wind_deg_Sev	0.005081
clouds_all_Bar	0.003202
humidity_Bil	0.001878
clouds_all_Bil	-0.004998
clouds_all_Sev	-0.008502
humidity_Sev	-0.014725
humidity_Mad	-0.021453
humidity_Bar	-0.035407
wind_deg_Mad	-0.051253
wind_deg_Val	-0.093775
wind_deg_Bar	-0.130585
generation hydro pumped storage consumption	-0.562930

1.6.1 Feature Selection for Multivarite Forcasting

First step is bulding the correlation matrix

[149]:
```python
correlations = df_energy_fin.corr(method='pearson')
fig = plt.figure(figsize=(24, 24))
sns.heatmap(correlations, annot=True, fmt='.2f')
plt.title('Pearson Correlation Matrix')
plt.show()
```



[203]:
```
#spliting the dataset into train and test
train_data = df_load_forcast.loc['2014-12-31 23:00:00+00:00':'2017-12-31 23:00:
↪00+00:00'].copy()
test_data = df_load_forcast.loc['2018-01-01 00:00:00+00:00':'2018-12-31 23:00:
↪00+00:00'].copy()
```

1.6.2 Feature Selection

we will use XGBoost for unsderstanding feature importance

Feature Importance in Gradient Boosting A benefit of using gradient boosting is that after the boosted trees are constructed, it is relatively straightforward to retrieve importance scores for each attribute.

Generally, importance provides a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. The more an attribute is used to make key decisions with decision trees, the higher its relative importance.

This importance is calculated explicitly for each attribute in the dataset, allowing attributes to be ranked and compared to each other.

Importance is calculated for a single decision tree by the amount that each attribute split point improves the performance measure, weighted by the number of observations the node is responsible for. The performance measure may be the purity (Gini index) used to select the split points or another more specific error function.

The feature importances are then averaged across all of the the decision trees within the model.

[255]:
```
#spliting the dataset into train and test
train_end_idx = 27048
cv_end_idx = 31056
test_end_idx = 35063

df_train = df_load_forcast[:27048]

df_cv = df_load_forcast[27048:31056]
```

[257]:
```
y_train = df_train['price actual'].values
y_cv = df_cv['price actual'].values
y_train = y_train.reshape(-1, 1)
y_cv = y_cv.reshape(-1, 1)

X_train = df_train.drop(['price actual', 'price day ahead'], axis=1)
X_cv = df_cv.drop(['price actual', 'price day ahead'], axis=1)
names = X_train.columns.values
```

[258]:
```
scaler_price = MinMaxScaler(feature_range=(0, 1))
scaler_mult = MinMaxScaler(feature_range=(0, 1))

scaler_price.fit(y_train)
```

```
scaler_price.transform(y_train)
scaler_price.transform(y_cv)

scaler_mult.fit(X_train)
scaler_mult.transform(X_train)
scaler_mult.transform(X_cv)
```

[258]:
```
array([[0.44932432, 0.        , 0.12070642, …, 0.33052632, 1.04070623,
        0.99221744],
       [0.4527027 , 0.        , 0.11860013, …, 0.33052632, 1.04070623,
        0.99221744],
       [0.44256757, 0.        , 0.11876215, …, 0.33052632, 1.04070623,
        0.99221744],
       …,
       [0.55067568, 0.63863864, 0.19944913, …, 0.27789474, 0.62187347,
        1.2248276 ],
       [0.5625    , 0.64564565, 0.17298553, …, 0.27789474, 0.62187347,
        1.2248276 ],
       [0.56756757, 0.65665666, 0.14916829, …, 0.27789474, 0.62187347,
        1.2248276 ]])
```

[263]:
```
param = {'eta': 0.03, 'max_depth': 2,
         'subsample': 1.0, 'colsample_bytree': 0.80,
         'alpha': 1.5, 'lambda': 1.5, 'gamma': 1.5,
         'objective': 'reg:linear', 'eval_metric': 'rmse',
         'silent': 1, 'min_child_weight': 5, 'n_jobs': -1}

dtrain = xgb.DMatrix(X_train, y_train, feature_names=X_train.columns.values)
dtest = xgb.DMatrix(X_cv, y_cv, feature_names=X_cv.columns.values)
eval_list = [(dtrain, 'train'), (dtest, 'eval')]

xgb_model = xgb.train(param, dtrain, 200, eval_list)
```

```
[13:45:33] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/objective/regression_obj.cu:170: reg:linear is now
deprecated in favor of reg:squarederror.
[13:45:33] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[0]     train-rmse:55.71547     eval-rmse:57.16536
```

```
[1]	train-rmse:54.11033	eval-rmse:55.48377
[2]	train-rmse:52.55515	eval-rmse:53.96031
[3]	train-rmse:51.05350	eval-rmse:52.46596
[4]	train-rmse:49.59269	eval-rmse:50.93366
[5]	train-rmse:48.17777	eval-rmse:49.45105
[6]	train-rmse:46.80685	eval-rmse:48.10619
[7]	train-rmse:45.47900	eval-rmse:46.71223
[8]	train-rmse:44.19274	eval-rmse:45.45274
[9]	train-rmse:42.94705	eval-rmse:44.14606
[10]	train-rmse:41.74060	eval-rmse:42.97431
[11]	train-rmse:40.57504	eval-rmse:41.82203
[12]	train-rmse:39.44326	eval-rmse:40.63075
[13]	train-rmse:38.34759	eval-rmse:39.48051
[14]	train-rmse:37.28824	eval-rmse:38.27469
[15]	train-rmse:36.26070	eval-rmse:37.27470
[16]	train-rmse:35.26641	eval-rmse:36.22750
[17]	train-rmse:34.30407	eval-rmse:35.29528
[18]	train-rmse:33.37283	eval-rmse:34.31818
[19]	train-rmse:32.47190	eval-rmse:33.44094
[20]	train-rmse:31.59790	eval-rmse:32.59365
[21]	train-rmse:30.75457	eval-rmse:31.82772
[22]	train-rmse:29.93805	eval-rmse:30.96984
[23]	train-rmse:29.14669	eval-rmse:30.20463
[24]	train-rmse:28.38114	eval-rmse:29.46577
[25]	train-rmse:27.64082	eval-rmse:28.75709
[26]	train-rmse:26.92591	eval-rmse:28.00501
[27]	train-rmse:26.23438	eval-rmse:27.34351
[28]	train-rmse:25.56797	eval-rmse:26.71794
[29]	train-rmse:24.92058	eval-rmse:26.00683
[30]	train-rmse:24.29352	eval-rmse:25.41768
[31]	train-rmse:23.68928	eval-rmse:24.78286
[32]	train-rmse:23.10479	eval-rmse:24.13963
[33]	train-rmse:22.54265	eval-rmse:23.59106
[34]	train-rmse:21.99588	eval-rmse:23.06483
[35]	train-rmse:21.46956	eval-rmse:22.64216
[36]	train-rmse:20.96264	eval-rmse:22.03415
[37]	train-rmse:20.47062	eval-rmse:21.48480
[38]	train-rmse:19.99699	eval-rmse:21.01609
[39]	train-rmse:19.53752	eval-rmse:20.59605
[40]	train-rmse:19.09814	eval-rmse:20.18201
[41]	train-rmse:18.67212	eval-rmse:19.68980
[42]	train-rmse:18.25501	eval-rmse:19.17932
[43]	train-rmse:17.85899	eval-rmse:18.81402
[44]	train-rmse:17.47693	eval-rmse:18.52249
[45]	train-rmse:17.10767	eval-rmse:18.17800
[46]	train-rmse:16.75302	eval-rmse:17.76352
[47]	train-rmse:16.40614	eval-rmse:17.39310
[48]	train-rmse:16.07573	eval-rmse:17.10049
```

```
[49]    train-rmse:15.76101    eval-rmse:16.75499
[50]    train-rmse:15.45591    eval-rmse:16.39964
[51]    train-rmse:15.16211    eval-rmse:16.18701
[52]    train-rmse:14.87973    eval-rmse:15.93727
[53]    train-rmse:14.60203    eval-rmse:15.59576
[54]    train-rmse:14.34027    eval-rmse:15.34333
[55]    train-rmse:14.08758    eval-rmse:15.12889
[56]    train-rmse:13.84622    eval-rmse:14.84450
[57]    train-rmse:13.61385    eval-rmse:14.63599
[58]    train-rmse:13.39216    eval-rmse:14.38428
[59]    train-rmse:13.17160    eval-rmse:14.16177
[60]    train-rmse:12.96617    eval-rmse:13.93519
[61]    train-rmse:12.76963    eval-rmse:13.71907
[62]    train-rmse:12.58217    eval-rmse:13.56480
[63]    train-rmse:12.39856    eval-rmse:13.41677
[64]    train-rmse:12.22427    eval-rmse:13.26116
[65]    train-rmse:12.05691    eval-rmse:13.12343
[66]    train-rmse:11.89005    eval-rmse:12.96088
[67]    train-rmse:11.73593    eval-rmse:12.77820
[68]    train-rmse:11.58694    eval-rmse:12.66196
[69]    train-rmse:11.44506    eval-rmse:12.52704
[70]    train-rmse:11.30922    eval-rmse:12.44839
[71]    train-rmse:11.17854    eval-rmse:12.28768
[72]    train-rmse:11.05386    eval-rmse:12.19099
[73]    train-rmse:10.93476    eval-rmse:12.08192
[74]    train-rmse:10.81220    eval-rmse:11.97124
[75]    train-rmse:10.70358    eval-rmse:11.87037
[76]    train-rmse:10.59768    eval-rmse:11.75879
[77]    train-rmse:10.49755    eval-rmse:11.68529
[78]    train-rmse:10.40179    eval-rmse:11.59547
[79]    train-rmse:10.30159    eval-rmse:11.50884
[80]    train-rmse:10.19741    eval-rmse:11.42085
[81]    train-rmse:10.11127    eval-rmse:11.36050
[82]    train-rmse:10.02944    eval-rmse:11.29848
[83]    train-rmse:9.95142     eval-rmse:11.23195
[84]    train-rmse:9.86219     eval-rmse:11.15884
[85]    train-rmse:9.79071     eval-rmse:11.08067
[86]    train-rmse:9.70844     eval-rmse:11.01567
[87]    train-rmse:9.64282     eval-rmse:10.94745
[88]    train-rmse:9.57979     eval-rmse:10.90013
[89]    train-rmse:9.51827     eval-rmse:10.84966
[90]    train-rmse:9.44582     eval-rmse:10.79717
[91]    train-rmse:9.38987     eval-rmse:10.75644
[92]    train-rmse:9.33553     eval-rmse:10.72078
[93]    train-rmse:9.28403     eval-rmse:10.67642
[94]    train-rmse:9.23346     eval-rmse:10.62036
[95]    train-rmse:9.18473     eval-rmse:10.57819
[96]    train-rmse:9.13915     eval-rmse:10.54193
```

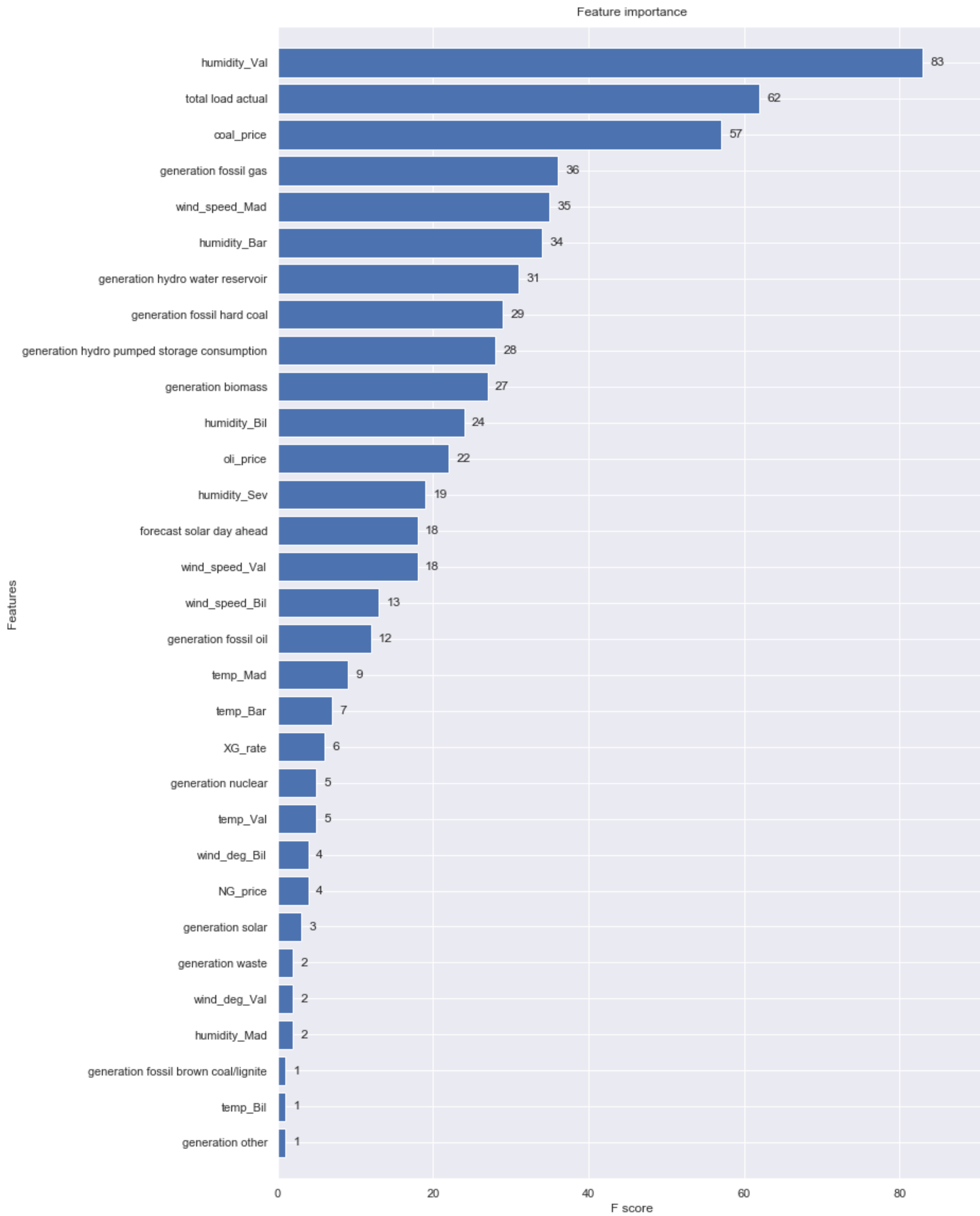
```
[97]	train-rmse:9.09256	eval-rmse:10.53151
[98]	train-rmse:9.05049	eval-rmse:10.50074
[99]	train-rmse:9.01008	eval-rmse:10.45192
[100]	train-rmse:8.96635	eval-rmse:10.46945
[101]	train-rmse:8.92862	eval-rmse:10.43313
[102]	train-rmse:8.89268	eval-rmse:10.40788
[103]	train-rmse:8.85824	eval-rmse:10.38122
[104]	train-rmse:8.81156	eval-rmse:10.36678
[105]	train-rmse:8.77885	eval-rmse:10.35142
[106]	train-rmse:8.74724	eval-rmse:10.32443
[107]	train-rmse:8.71748	eval-rmse:10.30273
[108]	train-rmse:8.68636	eval-rmse:10.27537
[109]	train-rmse:8.64719	eval-rmse:10.25306
[110]	train-rmse:8.62040	eval-rmse:10.23893
[111]	train-rmse:8.59385	eval-rmse:10.20873
[112]	train-rmse:8.56895	eval-rmse:10.18223
[113]	train-rmse:8.54289	eval-rmse:10.16025
[114]	train-rmse:8.51878	eval-rmse:10.15685
[115]	train-rmse:8.49605	eval-rmse:10.13839
[116]	train-rmse:8.47222	eval-rmse:10.13709
[117]	train-rmse:8.44912	eval-rmse:10.13600
[118]	train-rmse:8.42764	eval-rmse:10.10388
[119]	train-rmse:8.40908	eval-rmse:10.10004
[120]	train-rmse:8.38915	eval-rmse:10.09824
[121]	train-rmse:8.36817	eval-rmse:10.08076
[122]	train-rmse:8.34943	eval-rmse:10.06537
[123]	train-rmse:8.33307	eval-rmse:10.05714
[124]	train-rmse:8.31589	eval-rmse:10.03265
[125]	train-rmse:8.29853	eval-rmse:10.02690
[126]	train-rmse:8.28093	eval-rmse:10.01295
[127]	train-rmse:8.26319	eval-rmse:10.00310
[128]	train-rmse:8.24626	eval-rmse:9.98561
[129]	train-rmse:8.22994	eval-rmse:9.98596
[130]	train-rmse:8.21406	eval-rmse:9.96407
[131]	train-rmse:8.19214	eval-rmse:9.95554
[132]	train-rmse:8.17924	eval-rmse:9.94324
[133]	train-rmse:8.16399	eval-rmse:9.92685
[134]	train-rmse:8.14917	eval-rmse:9.91395
[135]	train-rmse:8.13547	eval-rmse:9.91436
[136]	train-rmse:8.12384	eval-rmse:9.90957
[137]	train-rmse:8.11113	eval-rmse:9.89038
[138]	train-rmse:8.09188	eval-rmse:9.88384
[139]	train-rmse:8.07840	eval-rmse:9.86507
[140]	train-rmse:8.06778	eval-rmse:9.86386
[141]	train-rmse:8.05549	eval-rmse:9.85811
[142]	train-rmse:8.04362	eval-rmse:9.84768
[143]	train-rmse:8.03122	eval-rmse:9.83788
[144]	train-rmse:8.01906	eval-rmse:9.83933
```

```
[145]	train-rmse:7.99866	eval-rmse:9.83412
[146]	train-rmse:7.98663	eval-rmse:9.81843
[147]	train-rmse:7.97505	eval-rmse:9.80364
[148]	train-rmse:7.96486	eval-rmse:9.78899
[149]	train-rmse:7.94684	eval-rmse:9.79398
[150]	train-rmse:7.93588	eval-rmse:9.78146
[151]	train-rmse:7.92719	eval-rmse:9.78103
[152]	train-rmse:7.91684	eval-rmse:9.77896
[153]	train-rmse:7.90588	eval-rmse:9.76916
[154]	train-rmse:7.89731	eval-rmse:9.76898
[155]	train-rmse:7.87765	eval-rmse:9.76473
[156]	train-rmse:7.86803	eval-rmse:9.75217
[157]	train-rmse:7.85907	eval-rmse:9.74317
[158]	train-rmse:7.84872	eval-rmse:9.74141
[159]	train-rmse:7.84085	eval-rmse:9.73934
[160]	train-rmse:7.83097	eval-rmse:9.72142
[161]	train-rmse:7.82176	eval-rmse:9.72234
[162]	train-rmse:7.80521	eval-rmse:9.72057
[163]	train-rmse:7.79639	eval-rmse:9.71292
[164]	train-rmse:7.78786	eval-rmse:9.70553
[165]	train-rmse:7.77775	eval-rmse:9.67102
[166]	train-rmse:7.76886	eval-rmse:9.66058
[167]	train-rmse:7.76211	eval-rmse:9.65880
[168]	train-rmse:7.75353	eval-rmse:9.66046
[169]	train-rmse:7.74470	eval-rmse:9.64895
[170]	train-rmse:7.73691	eval-rmse:9.64116
[171]	train-rmse:7.72538	eval-rmse:9.62891
[172]	train-rmse:7.71860	eval-rmse:9.62917
[173]	train-rmse:7.70903	eval-rmse:9.62368
[174]	train-rmse:7.70133	eval-rmse:9.61558
[175]	train-rmse:7.69229	eval-rmse:9.61601
[176]	train-rmse:7.68482	eval-rmse:9.60840
[177]	train-rmse:7.67411	eval-rmse:9.59851
[178]	train-rmse:7.66215	eval-rmse:9.59570
[179]	train-rmse:7.65334	eval-rmse:9.59304
[180]	train-rmse:7.64542	eval-rmse:9.57862
[181]	train-rmse:7.63926	eval-rmse:9.57284
[182]	train-rmse:7.63219	eval-rmse:9.56679
[183]	train-rmse:7.61672	eval-rmse:9.56449
[184]	train-rmse:7.60917	eval-rmse:9.55199
[185]	train-rmse:7.59720	eval-rmse:9.58905
[186]	train-rmse:7.59097	eval-rmse:9.56580
[187]	train-rmse:7.58516	eval-rmse:9.56637
[188]	train-rmse:7.57903	eval-rmse:9.55823
[189]	train-rmse:7.57155	eval-rmse:9.55210
[190]	train-rmse:7.55814	eval-rmse:9.55090
[191]	train-rmse:7.55149	eval-rmse:9.54927
[192]	train-rmse:7.54529	eval-rmse:9.54699
```

```
[193]	train-rmse:7.53769	eval-rmse:9.54504
[194]	train-rmse:7.52690	eval-rmse:9.58561
[195]	train-rmse:7.51987	eval-rmse:9.57174
[196]	train-rmse:7.51375	eval-rmse:9.56576
[197]	train-rmse:7.50775	eval-rmse:9.56110
[198]	train-rmse:7.50168	eval-rmse:9.55966
[199]	train-rmse:7.49662	eval-rmse:9.55580
```

[264]:

```
fig, ax = plt.subplots(figsize=(12, 20))
xgb.plot_importance(xgb_model, max_num_features=70, height=0.8, ax=ax)
plt.show()
```



[265]:
```
correlations = df_train.corr(method='pearson')
correlations_price = abs(correlations['price actual'])
print(correlations_price[correlations_price > 0.20]
      .sort_values(ascending=False).to_string())
```

```
price actual                                    1.000000
```

```
price day ahead                                     0.725862
generation fossil hard coal                         0.536698
generation fossil gas                               0.458971
total load actual                                   0.450557
generation fossil brown coal/lignite                0.432433
generation hydro pumped storage consumption         0.411134
oli_price                                           0.364136
generation fossil oil                               0.351321
NG_price                                            0.338010
coal_price                                          0.291640
generation hydro run-of-river and poundage          0.242462
wind_speed_Mad                                      0.233457
generation biomass                                  0.228573
humidity_Sev                                        0.204557
forecast wind onshore day ahead                     0.203511
generation wind onshore                             0.202335
```

By comparing the coorelation matrix and the result of the XGBoost model we selected some features that we intailly expected these feature will have an effect on the forecasting we ignored sevral features for example the exchange rate was low on the importance grediant and the coorealtion matrix.

We can see that the Hard Coal genration and the coal price has a higer importance and score in the correlation matrix as 20% of the Power genration in Spain comes from Coal

[267]:
```
considered_features = ['generation fossil hard coal', 'generation fossil gas',␣
↪'total load actual', 'generation fossil brown coal/lignite',
                       'generation hydro pumped storage consumption',␣
↪'oli_price',
                       'generation fossil oil', 'NG_price',
                       'coal_price', 'humidity_Val',
                       'total load actual','wind_speed_Mad']
```

[273]:
```
len(considered_features)

values = df_load_forcast[considered_features].values
```

1.6.3 Checking the RMSE for basline forcast provided with the datset

We will compare the Price forcast to the actual value that are both provided in the dataset use and we will use this a baseline for the model we are bulding

[277]:
```
y = df_load_forcast['price actual'].values
y_forecast = df_load_forcast['price day ahead'].values
```

[278]:
```
#spliting the dataset
y_train = y[:27048]
y_cv = y[27048 : 31056]
y_test = y[31056:]
```

[279]:
```
rmse_tso_day = sqrt(mean_squared_error(y_test, y_forecast[31056:]))

print('RMSE of day-ahead electricity price forecast by TSO: {}'.
 ↪format(round(rmse_tso_day, 3)))
```

```
RMSE of day-ahead electricity price forecast by TSO: 12.334
```

1.6.4 Multivariate forecasts using Tensorflow keras LSTM

[295]:
```
def multivariate_data(dataset, target, start_index, end_index, history_size,
                      target_size, step, single_step=False):
    data = []
    labels = []

    start_index = start_index + history_size
    if end_index is None:
        end_index = len(dataset) - target_size

    for i in range(start_index, end_index):
        indices = range(i-history_size, i, step)
        data.append(dataset[indices])

        if single_step:
            labels.append(target[i + target_size])
        else:
            labels.append(target[i : i + target_size])

    return np.array(data), np.array(labels)
```

[296]:
```
#the spliting ratio for the test / train data
train_end_idx = 27048
cv_end_idx = 31056
test_end_idx = 35063
```

[297]:
```
#Assiging the testing dataset
dataset = df_load_forcast[considered_features]
```

[298]:
```
#scaling the values between 0 to 1  to reduse the time of computation
scaler_mult = MinMaxScaler(feature_range=(0, 1))

scaler_mult.fit(dataset[:train_end_idx])
```

[298]:
```
MinMaxScaler()
```

[299]:
```
# prepraing the dataset and transforming its shape
y = df_load_forcast['price actual'].values

scaler = MinMaxScaler(feature_range=(0, 1))
```

```
y_reshaped = y.reshape(-1, 1)
scaler.fit(y_reshaped[:train_end_idx])

scaled_price = scaler.transform(y_reshaped)
```

[300]:
```
scaled_dataset = scaler_mult.transform(dataset)
```

[301]:
```
scaled_dataset = np.concatenate((scaled_dataset, scaled_price), axis=1)
```

[302]:
```
# Set the number of previous time-lags that will be used

#multivariate_past_history = 3
#multivariate_past_history = 10
multivariate_past_history = 25
```

[303]:
```
multivariate_future_target = 0
```

[304]:
```
X_train_mult, y_train_mult = multivariate_data(scaled_dataset, scaled_dataset[:
 ↪, -1],
                                               0, train_end_idx,
                                               multivariate_past_history,
                                               multivariate_future_target,
                                               step=1, single_step=True)
```

[305]:
```
X_val_mult, y_val_mult = multivariate_data(scaled_dataset, scaled_dataset[:,␣
 ↪-1],
                                           train_end_idx, cv_end_idx,
                                           multivariate_past_history,
                                           multivariate_future_target,
                                           step=1, single_step=True)
```

[306]:
```
X_test_mult, y_test_mult = multivariate_data(scaled_dataset, scaled_dataset[:,␣
 ↪-1],
                                             cv_end_idx, test_end_idx,
                                             multivariate_past_history,
                                             multivariate_future_target,
                                             step=1, single_step=True)
```

[307]:
```
batch_size = 32
buffer_size = 1000
```

[311]:
```
train_mult = tf.data.Dataset.from_tensor_slices((X_train_mult, y_train_mult))
train_mult = train_mult.cache().shuffle(buffer_size).batch(batch_size).
 ↪prefetch(1)
```

[312]:
```
val_mult = tf.data.Dataset.from_tensor_slices((X_val_mult, y_val_mult))
val_mult = val_mult.batch(batch_size).prefetch(1)
```

[313]:
```
# Define some common parameters

input_shape_mult = X_train_mult.shape[-2:]
loss = tf.keras.losses.MeanSquaredError()
metric = [tf.keras.metrics.RootMeanSquaredError()]
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
              lambda epoch: 1e-4 * 10**(epoch / 10))
early_stopping = tf.keras.callbacks.EarlyStopping(patience=10)
```

[314]:
```
y_test_mult_reshaped = y_test_mult.reshape(-1, 1)
y_test_mult_inv = scaler.inverse_transform(y_test_mult_reshaped)
```

[315]:
```
#using Keras Sequential using the optimizer to find the best learning rate
tf.keras.backend.clear_session()

multivariate_lstm = tf.keras.models.Sequential([
    LSTM(80, input_shape=input_shape_mult, return_sequences=True),
    Flatten(),
    Dense(160, activation='relu'),
    Dropout(0.1),
    Dense(1)
])

optimizer = tf.keras.optimizers.Adam(lr=1e-4, amsgrad=True)
multivariate_lstm.compile(loss=loss,
                          optimizer=optimizer,
                          metrics=metric)
```

[316]:
```
history_lr = multivariate_lstm.fit(train_mult, epochs=50,
                                   validation_data=val_mult,
                                   callbacks=[lr_schedule])
```

```
Epoch 1/50
845/845 [==============================] - 20s 23ms/step - loss: 0.0060 -
root_mean_squared_error: 0.0773 - val_loss: 0.0035 -
val_root_mean_squared_error: 0.0590
Epoch 2/50
845/845 [==============================] - 19s 23ms/step - loss: 0.0031 -
root_mean_squared_error: 0.0558 - val_loss: 0.0035 -
val_root_mean_squared_error: 0.0590
Epoch 3/50
845/845 [==============================] - 21s 25ms/step - loss: 0.0026 -
root_mean_squared_error: 0.0506 - val_loss: 0.0032 -
val_root_mean_squared_error: 0.0568
Epoch 4/50
```

```
845/845 [==============================] - 18s 21ms/step - loss: 0.0022 -
root_mean_squared_error: 0.0470 - val_loss: 0.0017 -
val_root_mean_squared_error: 0.0418
Epoch 5/50
845/845 [==============================] - 17s 20ms/step - loss: 0.0020 -
root_mean_squared_error: 0.0443 - val_loss: 0.0017 -
val_root_mean_squared_error: 0.0407
Epoch 6/50
845/845 [==============================] - 17s 20ms/step - loss: 0.0017 -
root_mean_squared_error: 0.0415 - val_loss: 0.0023 -
val_root_mean_squared_error: 0.0477
Epoch 7/50
845/845 [==============================] - 18s 21ms/step - loss: 0.0016 -
root_mean_squared_error: 0.0395 - val_loss: 0.0011 -
val_root_mean_squared_error: 0.0335
Epoch 8/50
845/845 [==============================] - 17s 21ms/step - loss: 0.0014 -
root_mean_squared_error: 0.0370 - val_loss: 0.0010 -
val_root_mean_squared_error: 0.0321
Epoch 9/50
845/845 [==============================] - 17s 20ms/step - loss: 0.0013 -
root_mean_squared_error: 0.0354 - val_loss: 0.0010 -
val_root_mean_squared_error: 0.0324
Epoch 10/50
845/845 [==============================] - 18s 21ms/step - loss: 0.0011 -
root_mean_squared_error: 0.0330 - val_loss: 9.1657e-04 -
val_root_mean_squared_error: 0.0303
Epoch 11/50
845/845 [==============================] - 17s 20ms/step - loss: 0.0011 -
root_mean_squared_error: 0.0325 - val_loss: 0.0011 -
val_root_mean_squared_error: 0.0328
Epoch 12/50
845/845 [==============================] - 17s 20ms/step - loss: 9.4123e-04 -
root_mean_squared_error: 0.0307 - val_loss: 8.9426e-04 -
val_root_mean_squared_error: 0.0299
Epoch 13/50
845/845 [==============================] - 17s 20ms/step - loss: 9.0283e-04 -
root_mean_squared_error: 0.0300 - val_loss: 7.3517e-04 -
val_root_mean_squared_error: 0.0271
Epoch 14/50
845/845 [==============================] - 17s 20ms/step - loss: 8.1828e-04 -
root_mean_squared_error: 0.0286 - val_loss: 6.8469e-04 -
val_root_mean_squared_error: 0.0262
Epoch 15/50
845/845 [==============================] - 17s 20ms/step - loss: 7.7358e-04 -
root_mean_squared_error: 0.0278 - val_loss: 7.2172e-04 -
val_root_mean_squared_error: 0.0269
Epoch 16/50
```

```
845/845 [==============================] - 17s 20ms/step - loss: 7.7909e-04 -
root_mean_squared_error: 0.0279 - val_loss: 0.0011 -
val_root_mean_squared_error: 0.0329
Epoch 17/50
845/845 [==============================] - 18s 21ms/step - loss: 7.3718e-04 -
root_mean_squared_error: 0.0272 - val_loss: 6.9570e-04 -
val_root_mean_squared_error: 0.0264
Epoch 18/50
845/845 [==============================] - 18s 21ms/step - loss: 7.3329e-04 -
root_mean_squared_error: 0.0271 - val_loss: 0.0011 -
val_root_mean_squared_error: 0.0324
Epoch 19/50
845/845 [==============================] - 17s 21ms/step - loss: 7.1931e-04 -
root_mean_squared_error: 0.0268 - val_loss: 7.1971e-04 -
val_root_mean_squared_error: 0.0268
Epoch 20/50
845/845 [==============================] - 17s 20ms/step - loss: 6.8313e-04 -
root_mean_squared_error: 0.0261 - val_loss: 8.1349e-04 -
val_root_mean_squared_error: 0.0285
Epoch 21/50
845/845 [==============================] - 18s 21ms/step - loss: 7.4177e-04 -
root_mean_squared_error: 0.0272 - val_loss: 6.5861e-04 -
val_root_mean_squared_error: 0.0257
Epoch 22/50
845/845 [==============================] - 17s 20ms/step - loss: 0.0015 -
root_mean_squared_error: 0.0390 - val_loss: 8.1641e-04 -
val_root_mean_squared_error: 0.0286
Epoch 23/50
845/845 [==============================] - 17s 20ms/step - loss: 0.0017 -
root_mean_squared_error: 0.0414 - val_loss: 0.0073 -
val_root_mean_squared_error: 0.0855
Epoch 24/50
845/845 [==============================] - 17s 20ms/step - loss: 0.0017 -
root_mean_squared_error: 0.0415 - val_loss: 9.1431e-04 -
val_root_mean_squared_error: 0.0302
Epoch 25/50
845/845 [==============================] - 17s 20ms/step - loss: 0.0011 -
root_mean_squared_error: 0.0333 - val_loss: 0.0010 -
val_root_mean_squared_error: 0.0321
Epoch 26/50
845/845 [==============================] - 18s 21ms/step - loss: 0.0010 -
root_mean_squared_error: 0.0320 - val_loss: 8.4669e-04 -
val_root_mean_squared_error: 0.0291
Epoch 27/50
845/845 [==============================] - 18s 21ms/step - loss: 0.0013 -
root_mean_squared_error: 0.0364 - val_loss: 9.9573e-04 -
val_root_mean_squared_error: 0.0316
Epoch 28/50
```

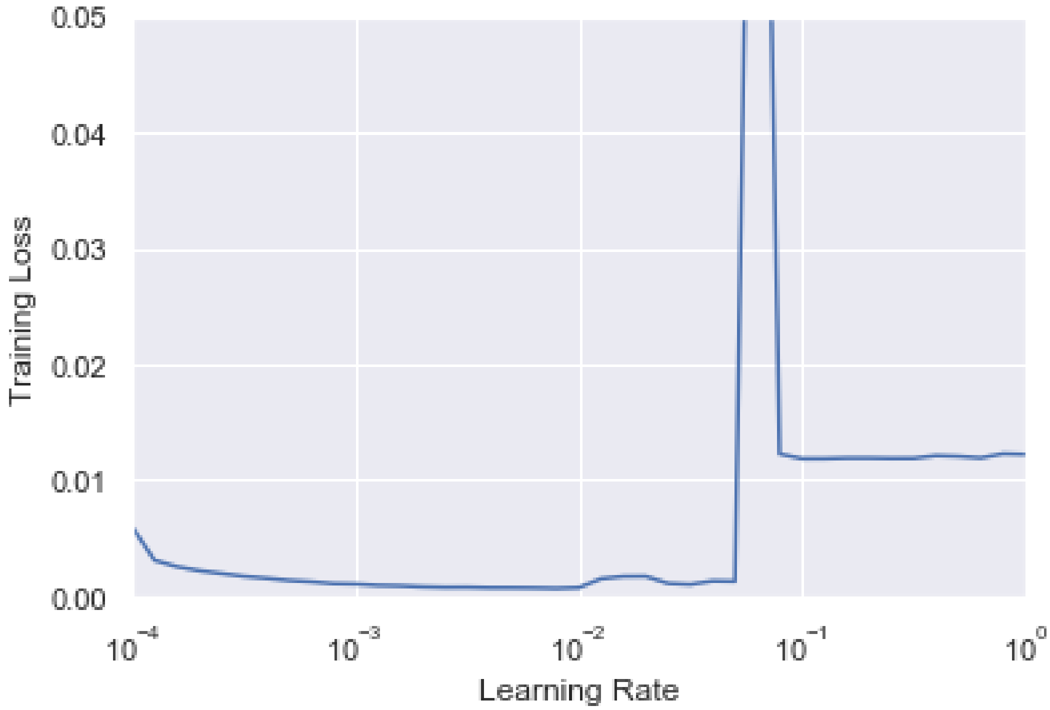
```
845/845 [==============================] - 17s 20ms/step - loss: 0.0013 -
root_mean_squared_error: 0.0359 - val_loss: 0.0017 -
val_root_mean_squared_error: 0.0409
Epoch 29/50
845/845 [==============================] - 17s 20ms/step - loss: 0.1166 -
root_mean_squared_error: 0.3415 - val_loss: 0.0911 -
val_root_mean_squared_error: 0.3018
Epoch 30/50
845/845 [==============================] - 18s 21ms/step - loss: 0.0123 -
root_mean_squared_error: 0.1108 - val_loss: 0.0125 -
val_root_mean_squared_error: 0.1119
Epoch 31/50
845/845 [==============================] - 18s 21ms/step - loss: 0.0119 -
root_mean_squared_error: 0.1089 - val_loss: 0.0126 -
val_root_mean_squared_error: 0.1123
Epoch 32/50
845/845 [==============================] - 17s 21ms/step - loss: 0.0119 -
root_mean_squared_error: 0.1090 - val_loss: 0.0131 -
val_root_mean_squared_error: 0.1143
Epoch 33/50
845/845 [==============================] - 17s 21ms/step - loss: 0.0119 -
root_mean_squared_error: 0.1092 - val_loss: 0.0123 -
val_root_mean_squared_error: 0.1111
Epoch 34/50
845/845 [==============================] - 17s 21ms/step - loss: 0.0119 -
root_mean_squared_error: 0.1092 - val_loss: 0.0124 -
val_root_mean_squared_error: 0.1113
Epoch 35/50
845/845 [==============================] - 18s 21ms/step - loss: 0.0119 -
root_mean_squared_error: 0.1091 - val_loss: 0.0129 -
val_root_mean_squared_error: 0.1135
Epoch 36/50
845/845 [==============================] - 18s 21ms/step - loss: 0.0119 -
root_mean_squared_error: 0.1091 - val_loss: 0.0144 -
val_root_mean_squared_error: 0.1200
Epoch 37/50
845/845 [==============================] - 18s 21ms/step - loss: 0.0121 -
root_mean_squared_error: 0.1101 - val_loss: 0.0123 -
val_root_mean_squared_error: 0.1107
Epoch 38/50
845/845 [==============================] - 17s 20ms/step - loss: 0.0121 -
root_mean_squared_error: 0.1099 - val_loss: 0.0127 -
val_root_mean_squared_error: 0.1128
Epoch 39/50
845/845 [==============================] - 17s 21ms/step - loss: 0.0119 -
root_mean_squared_error: 0.1092 - val_loss: 0.0132 -
val_root_mean_squared_error: 0.1148
Epoch 40/50
```

```
845/845 [==============================] - 17s 21ms/step - loss: 0.0123 -
root_mean_squared_error: 0.1109 - val_loss: 0.0134 -
val_root_mean_squared_error: 0.1156
Epoch 41/50
845/845 [==============================] - 17s 20ms/step - loss: 0.0122 -
root_mean_squared_error: 0.1106 - val_loss: 0.0125 -
val_root_mean_squared_error: 0.1119
Epoch 42/50
845/845 [==============================] - 17s 20ms/step - loss: 0.0124 -
root_mean_squared_error: 0.1115 - val_loss: 0.0123 -
val_root_mean_squared_error: 0.1107
Epoch 43/50
845/845 [==============================] - 18s 21ms/step - loss: 0.0129 -
root_mean_squared_error: 0.1135 - val_loss: 0.0139 -
val_root_mean_squared_error: 0.1177
Epoch 44/50
845/845 [==============================] - 17s 21ms/step - loss: 0.0132 -
root_mean_squared_error: 0.1151 - val_loss: 0.0124 -
val_root_mean_squared_error: 0.1115
Epoch 45/50
845/845 [==============================] - 18s 21ms/step - loss: 0.0138 -
root_mean_squared_error: 0.1175 - val_loss: 0.0143 -
val_root_mean_squared_error: 0.1194
Epoch 46/50
845/845 [==============================] - 17s 21ms/step - loss: 0.0146 -
root_mean_squared_error: 0.1210 - val_loss: 0.0210 -
val_root_mean_squared_error: 0.1449
Epoch 47/50
845/845 [==============================] - 18s 22ms/step - loss: 0.0154 -
root_mean_squared_error: 0.1242 - val_loss: 0.0163 -
val_root_mean_squared_error: 0.1276
Epoch 48/50
845/845 [==============================] - 21s 25ms/step - loss: 0.0176 -
root_mean_squared_error: 0.1329 - val_loss: 0.0561 -
val_root_mean_squared_error: 0.2369
Epoch 49/50
845/845 [==============================] - 18s 21ms/step - loss: 0.0179 -
root_mean_squared_error: 0.1339 - val_loss: 0.0124 -
val_root_mean_squared_error: 0.1112
Epoch 50/50
845/845 [==============================] - 17s 21ms/step - loss: 0.0206 -
root_mean_squared_error: 0.1436 - val_loss: 0.0531 -
val_root_mean_squared_error: 0.2305
```

[319]:
```
def plot_learning_rate_schedule(history_lr, max_loss):
    plt.semilogx(history_lr.history['lr'], history_lr.history['loss'])
    plt.axis([1e-4, 1, 0, max_loss])
```

```
    plt.xlabel('Learning Rate')
    plt.ylabel('Training Loss')
    plt.show()
```

[320]:
```
plot_learning_rate_schedule(history_lr, 0.05)
```



The learning rate that we will use for Adam is equal to 5e-3.

[321]:
```
tf.keras.backend.clear_session()

multivariate_lstm = tf.keras.models.Sequential([
    LSTM(80, input_shape=input_shape_mult, return_sequences=True),
    Flatten(),
    Dense(160, activation='relu'),
    Dropout(0.1),
    Dense(1)
])

model_checkpoint = tf.keras.callbacks.ModelCheckpoint(
                    'multivariate_lstm.h5', monitor=('val_loss'),␣
 ↪save_best_only=True)
optimizer = tf.keras.optimizers.Adam(lr=5e-3, amsgrad=True)

multivariate_lstm.compile(loss=loss,
```

```
                              optimizer=optimizer,
                              metrics=metric)
```

[322]:
```
history = multivariate_lstm.fit(train_mult, epochs=120,
                                validation_data=val_mult,
                                callbacks=[early_stopping,
                                           model_checkpoint])
```

```
Epoch 1/120
845/845 [==============================] - 20s 23ms/step - loss: 0.0427 -
root_mean_squared_error: 0.2099 - val_loss: 0.0032 -
val_root_mean_squared_error: 0.0562
Epoch 2/120
845/845 [==============================] - 17s 21ms/step - loss: 0.0033 -
root_mean_squared_error: 0.0573 - val_loss: 0.0032 -
val_root_mean_squared_error: 0.0566
Epoch 3/120
845/845 [==============================] - 18s 21ms/step - loss: 0.0025 -
root_mean_squared_error: 0.0503 - val_loss: 0.0019 -
val_root_mean_squared_error: 0.0430
Epoch 4/120
845/845 [==============================] - 18s 21ms/step - loss: 0.0019 -
root_mean_squared_error: 0.0438 - val_loss: 0.0013 -
val_root_mean_squared_error: 0.0363
Epoch 5/120
845/845 [==============================] - 18s 22ms/step - loss: 0.0016 -
root_mean_squared_error: 0.0397 - val_loss: 0.0010 -
val_root_mean_squared_error: 0.0322
Epoch 6/120
845/845 [==============================] - 20s 24ms/step - loss: 0.0013 -
root_mean_squared_error: 0.0366 - val_loss: 9.2521e-04 -
val_root_mean_squared_error: 0.0304
Epoch 7/120
845/845 [==============================] - 18s 22ms/step - loss: 0.0012 -
root_mean_squared_error: 0.0340 - val_loss: 7.8683e-04 -
val_root_mean_squared_error: 0.0281
Epoch 8/120
845/845 [==============================] - 18s 21ms/step - loss: 0.0010 -
root_mean_squared_error: 0.0323 - val_loss: 7.2958e-04 -
val_root_mean_squared_error: 0.0270
Epoch 9/120
845/845 [==============================] - 17s 21ms/step - loss: 9.8684e-04 -
root_mean_squared_error: 0.0314 - val_loss: 9.8580e-04 -
val_root_mean_squared_error: 0.0314
Epoch 10/120
845/845 [==============================] - 17s 20ms/step - loss: 9.3093e-04 -
root_mean_squared_error: 0.0305 - val_loss: 0.0013 -
val_root_mean_squared_error: 0.0364
```

```
Epoch 11/120
845/845 [==============================] - 17s 20ms/step - loss: 9.0318e-04 -
root_mean_squared_error: 0.0301 - val_loss: 6.9621e-04 -
val_root_mean_squared_error: 0.0264
Epoch 12/120
845/845 [==============================] - 17s 20ms/step - loss: 8.3860e-04 -
root_mean_squared_error: 0.0290 - val_loss: 8.9050e-04 -
val_root_mean_squared_error: 0.0298
Epoch 13/120
845/845 [==============================] - 20s 23ms/step - loss: 8.1655e-04 -
root_mean_squared_error: 0.0286 - val_loss: 6.3760e-04 -
val_root_mean_squared_error: 0.0253
Epoch 14/120
845/845 [==============================] - 19s 22ms/step - loss: 7.9522e-04 -
root_mean_squared_error: 0.0282 - val_loss: 5.8895e-04 -
val_root_mean_squared_error: 0.0243
Epoch 15/120
845/845 [==============================] - 21s 24ms/step - loss: 7.8261e-04 -
root_mean_squared_error: 0.0280 - val_loss: 6.2013e-04 -
val_root_mean_squared_error: 0.0249
Epoch 16/120
845/845 [==============================] - 18s 22ms/step - loss: 7.5984e-04 -
root_mean_squared_error: 0.0276 - val_loss: 6.0285e-04 -
val_root_mean_squared_error: 0.0246
Epoch 17/120
845/845 [==============================] - 19s 23ms/step - loss: 7.5885e-04 -
root_mean_squared_error: 0.0275 - val_loss: 6.9881e-04 -
val_root_mean_squared_error: 0.0264
Epoch 18/120
845/845 [==============================] - 18s 22ms/step - loss: 7.2673e-04 -
root_mean_squared_error: 0.0270 - val_loss: 6.7495e-04 -
val_root_mean_squared_error: 0.0260
Epoch 19/120
845/845 [==============================] - 17s 21ms/step - loss: 7.2804e-04 -
root_mean_squared_error: 0.0270 - val_loss: 5.8134e-04 -
val_root_mean_squared_error: 0.0241
Epoch 20/120
845/845 [==============================] - 18s 22ms/step - loss: 7.1943e-04 -
root_mean_squared_error: 0.0268 - val_loss: 5.9511e-04 -
val_root_mean_squared_error: 0.0244
Epoch 21/120
845/845 [==============================] - 18s 22ms/step - loss: 7.0535e-04 -
root_mean_squared_error: 0.0266 - val_loss: 5.8799e-04 -
val_root_mean_squared_error: 0.0242
Epoch 22/120
845/845 [==============================] - 18s 21ms/step - loss: 7.1167e-04 -
root_mean_squared_error: 0.0267 - val_loss: 5.6601e-04 -
val_root_mean_squared_error: 0.0238
```

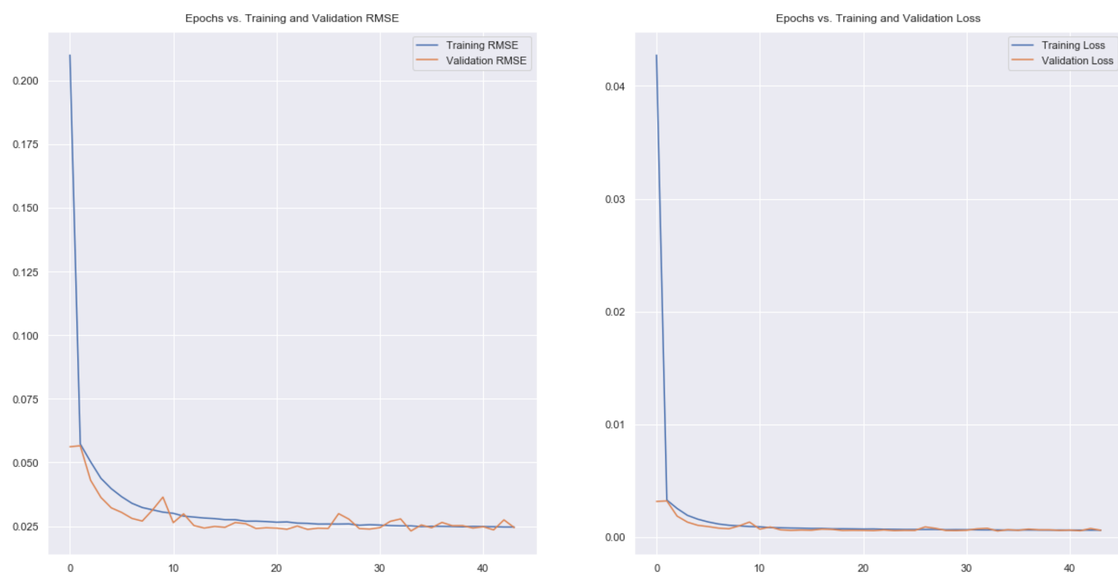
```
Epoch 23/120
845/845 [==============================] - 18s 21ms/step - loss: 6.8696e-04 -
root_mean_squared_error: 0.0262 - val_loss: 6.2731e-04 -
val_root_mean_squared_error: 0.0250
Epoch 24/120
845/845 [==============================] - 18s 21ms/step - loss: 6.8188e-04 -
root_mean_squared_error: 0.0261 - val_loss: 5.6352e-04 -
val_root_mean_squared_error: 0.0237
Epoch 25/120
845/845 [==============================] - 18s 21ms/step - loss: 6.6761e-04 -
root_mean_squared_error: 0.0258 - val_loss: 5.8710e-04 -
val_root_mean_squared_error: 0.0242
Epoch 26/120
845/845 [==============================] - 18s 21ms/step - loss: 6.6832e-04 -
root_mean_squared_error: 0.0259 - val_loss: 5.8210e-04 -
val_root_mean_squared_error: 0.0241
Epoch 27/120
845/845 [==============================] - 18s 21ms/step - loss: 6.6766e-04 -
root_mean_squared_error: 0.0258 - val_loss: 8.9526e-04 -
val_root_mean_squared_error: 0.0299
Epoch 28/120
845/845 [==============================] - 19s 22ms/step - loss: 6.7037e-04 -
root_mean_squared_error: 0.0259 - val_loss: 7.7024e-04 -
val_root_mean_squared_error: 0.0278
Epoch 29/120
845/845 [==============================] - 18s 22ms/step - loss: 6.4410e-04 -
root_mean_squared_error: 0.0254 - val_loss: 5.7938e-04 -
val_root_mean_squared_error: 0.0241
Epoch 30/120
845/845 [==============================] - 18s 21ms/step - loss: 6.5422e-04 -
root_mean_squared_error: 0.0256 - val_loss: 5.6937e-04 -
val_root_mean_squared_error: 0.0239
Epoch 31/120
845/845 [==============================] - 17s 20ms/step - loss: 6.4720e-04 -
root_mean_squared_error: 0.0254 - val_loss: 5.9681e-04 -
val_root_mean_squared_error: 0.0244
Epoch 32/120
845/845 [==============================] - 18s 21ms/step - loss: 6.3620e-04 -
root_mean_squared_error: 0.0252 - val_loss: 7.2393e-04 -
val_root_mean_squared_error: 0.0269
Epoch 33/120
845/845 [==============================] - 21s 25ms/step - loss: 6.3105e-04 -
root_mean_squared_error: 0.0251 - val_loss: 7.7558e-04 -
val_root_mean_squared_error: 0.0278
Epoch 34/120
845/845 [==============================] - 23s 27ms/step - loss: 6.3090e-04 -
root_mean_squared_error: 0.0251 - val_loss: 5.3433e-04 -
val_root_mean_squared_error: 0.0231
```

```
Epoch 35/120
845/845 [==============================] - 26s 30ms/step - loss: 6.1618e-04 -
root_mean_squared_error: 0.0248 - val_loss: 6.4962e-04 -
val_root_mean_squared_error: 0.0255
Epoch 36/120
845/845 [==============================] - 23s 27ms/step - loss: 6.2346e-04 -
root_mean_squared_error: 0.0250 - val_loss: 5.9457e-04 -
val_root_mean_squared_error: 0.0244
Epoch 37/120
845/845 [==============================] - 21s 25ms/step - loss: 6.2134e-04 -
root_mean_squared_error: 0.0249 - val_loss: 7.0092e-04 -
val_root_mean_squared_error: 0.0265
Epoch 38/120
845/845 [==============================] - 20s 24ms/step - loss: 6.1813e-04 -
root_mean_squared_error: 0.0249 - val_loss: 6.3443e-04 -
val_root_mean_squared_error: 0.0252
Epoch 39/120
845/845 [==============================] - 19s 22ms/step - loss: 6.1342e-04 -
root_mean_squared_error: 0.0248 - val_loss: 6.3594e-04 -
val_root_mean_squared_error: 0.0252
Epoch 40/120
845/845 [==============================] - 19s 22ms/step - loss: 6.2123e-04 -
root_mean_squared_error: 0.0249 - val_loss: 5.8864e-04 -
val_root_mean_squared_error: 0.0243
Epoch 41/120
845/845 [==============================] - 20s 23ms/step - loss: 6.1701e-04 -
root_mean_squared_error: 0.0248 - val_loss: 6.1441e-04 -
val_root_mean_squared_error: 0.0248
Epoch 42/120
845/845 [==============================] - 21s 24ms/step - loss: 6.1455e-04 -
root_mean_squared_error: 0.0248 - val_loss: 5.5417e-04 -
val_root_mean_squared_error: 0.0235
Epoch 43/120
845/845 [==============================] - 21s 25ms/step - loss: 6.0894e-04 -
root_mean_squared_error: 0.0247 - val_loss: 7.5497e-04 -
val_root_mean_squared_error: 0.0275
Epoch 44/120
845/845 [==============================] - 21s 25ms/step - loss: 6.0843e-04 -
root_mean_squared_error: 0.0247 - val_loss: 5.9664e-04 -
val_root_mean_squared_error: 0.0244
```

[324]:
```
def plot_model_rmse_and_loss(history):

    # Evaluate train and validation accuracies and losses

    train_rmse = history.history['root_mean_squared_error']
    val_rmse = history.history['val_root_mean_squared_error']
```

```
    train_loss = history.history['loss']
    val_loss = history.history['val_loss']

    # Visualize epochs vs. train and validation accuracies and losses

    plt.figure(figsize=(20, 10))
    plt.subplot(1, 2, 1)
    plt.plot(train_rmse, label='Training RMSE')
    plt.plot(val_rmse, label='Validation RMSE')
    plt.legend()
    plt.title('Epochs vs. Training and Validation RMSE')

    plt.subplot(1, 2, 2)
    plt.plot(train_loss, label='Training Loss')
    plt.plot(val_loss, label='Validation Loss')
    plt.legend()
    plt.title('Epochs vs. Training and Validation Loss')

    plt.show()
```

[325]:
```
plot_model_rmse_and_loss(history)
```

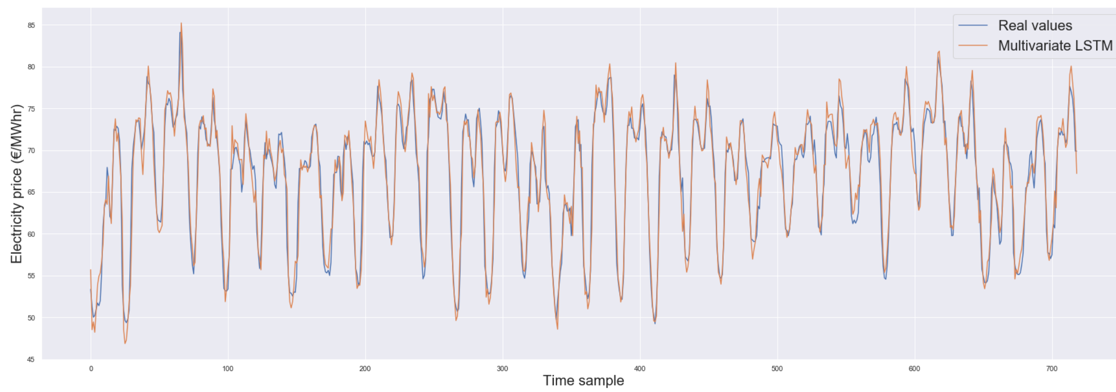


[326]:
```
multivariate_lstm = tf.keras.models.load_model('multivariate_lstm.h5')

forecast = multivariate_lstm.predict(X_test_mult)
multivariate_lstm_forecast = scaler.inverse_transform(forecast)
```

```
rmse_mult_lstm = sqrt(mean_squared_error(y_test_mult_inv,
                                          multivariate_lstm_forecast))
print('RMSE of hour-ahead electricity price multivariate LSTM forecast: {}'
      .format(round(rmse_mult_lstm, 3)))
```

```
RMSE of hour-ahead electricity price multivariate LSTM forecast: 2.257
```

[328]:
```
fig, ax = plt.subplots(figsize=(30, 10))
ax.set_xlabel('Time sample', fontsize=22)
ax.set_ylabel('Electricity price (€/MWhr)', fontsize=22)
ax.plot(y_test_mult_inv[3263:], label='Real values')
ax.plot(multivariate_lstm_forecast[3263:], label='Multivariate LSTM')
ax.legend(prop={'size':22})
plt.show()
```



1.7 Conclusion

Based on the results of the RMSE of 2.257 which is far better than the prediction values provided by the TSO which is 12.334 From the analysis We will need to add more features related to the time as the Week day or weekend also the hour of the week as this has a high effect on the generation

Many variables that had a high prediction power was actually expected and made full since as they have a direct effect either on the load requirements or the cost of generation

I was expecting to see more of the temperature variables having effects on the prediction power however it was not, and this may be because the temperature difference in Spain from summer to winter is not so huge.

I was expecting to see very clear seasonality in the load, but it was not very clear and this also can be because the difference in temperatures between winter and summer is not as huge as in Canada for an example.

1.7.1 App Deployment

An app that uses the model will be developed on using Dash to help in predicting the load an the cost of MWH and it will be taking into consideration the weather conditions and the price of Fossil fuels of the day.

```
[ ]:
```