Energy price

March 13, 2022

0.0.1 Imports

[116]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Read files

[117]:
```
epex_df = pd.read_csv("epex_day_ahead_price.csv")
spot_df = pd.read_csv("spot_intraday_price.csv")
system_df = pd.read_csv("systemprice.csv")
```

1 1. Inspect Datasets and fix errors

1.0.1 Epex Prices

[118]:
```
epex_df
```

[118]:

	timestamp	apx_da_hourly
0	2019-03-31 23:00:00+00:00	26.43
1	2019-03-31 23:30:00+00:00	26.43
2	2019-04-01 00:00:00+00:00	29.24
3	2019-04-01 00:30:00+00:00	29.24
4	2019-04-01 01:00:00+00:00	35.10
…	…	…
47853	2021-12-22 21:30:00+00:00	315.00
47854	2021-12-22 22:00:00+00:00	276.85
47855	2021-12-22 22:30:00+00:00	276.85
47856	2021-12-22 23:00:00+00:00	325.40
47857	2021-12-22 23:30:00+00:00	325.40

[47858 rows x 2 columns]

Get rid of first and last dates to match with other dataframes

[119]:
```
epex_df = epex_df.iloc[338:-336].copy()
epex_df
```

[119]:
```
                       timestamp  apx_da_hourly
338    2019-04-08 00:00:00+00:00          33.41
339    2019-04-08 00:30:00+00:00          33.41
340    2019-04-08 01:00:00+00:00          41.03
341    2019-04-08 01:30:00+00:00          41.03
342    2019-04-08 02:00:00+00:00          39.00
…                            …              …
47517  2021-12-15 21:30:00+00:00         258.30
47518  2021-12-15 22:00:00+00:00         231.80
47519  2021-12-15 22:30:00+00:00         231.80
47520  2021-12-15 23:00:00+00:00         231.00
47521  2021-12-15 23:30:00+00:00         231.00

[47184 rows x 2 columns]
```

[120]:
```
epex_df.describe()
```

[120]:
```
       apx_da_hourly
count   47184.000000
mean       63.408615
std        65.787193
min       -38.800000
25%        33.100000
50%        44.300000
75%        68.000000
max      1860.000000
```

Check if there are any nan values

[121]:
```
epex_df.isnull().values.any()
```

[121]:
```
False
```

1.0.2 Spot Prices

[122]:
```
spot_df
```

[122]:
```
                       timestamp    SP  spot_price
0      2019-01-02 00:00:00+00:00   1.0       40.01
1      2019-01-02 00:30:00+00:00   2.0       43.27
2      2019-01-02 01:00:00+00:00   3.0       42.72
3      2019-01-02 01:30:00+00:00   4.0       52.17
4      2019-01-02 02:00:00+00:00   5.0       53.44
…                            …   …           …
51835  2021-12-16 21:30:00+00:00  44.0      232.25
51836  2021-12-16 22:00:00+00:00  45.0      242.44
51837  2021-12-16 22:30:00+00:00  46.0      220.12
51838  2021-12-16 23:00:00+00:00  47.0      231.87
```

```
51839  2021-12-16 23:30:00+00:00  48.0      246.75

[51840 rows x 3 columns]
```

Get rid of first and last dates to match with other dataframes

[123]:
```
spot_df = spot_df.iloc[4608:-48].copy()
spot_df
```

[123]:
```
                      timestamp    SP  spot_price
4608   2019-04-08 00:00:00+00:00   3.0       40.45
4609   2019-04-08 00:30:00+00:00   4.0       43.69
4610   2019-04-08 01:00:00+00:00   5.0       46.13
4611   2019-04-08 01:30:00+00:00   6.0       43.96
4612   2019-04-08 02:00:00+00:00   7.0       44.36
…                            …   …          …
51787  2021-12-15 21:30:00+00:00  44.0      231.98
51788  2021-12-15 22:00:00+00:00  45.0      238.32
51789  2021-12-15 22:30:00+00:00  46.0      213.15
51790  2021-12-15 23:00:00+00:00  47.0      218.80
51791  2021-12-15 23:30:00+00:00  48.0      157.07

[47184 rows x 3 columns]
```

[124]:
```
spot_df.describe()
```

[124]:
```
                 SP    spot_price
count  46810.000000  46810.000000
mean      24.508182     60.966664
std       13.851154     68.274426
min        1.000000   -101.620000
25%       13.000000     31.510000
50%       25.000000     43.470000
75%       37.000000     68.487500
max       48.000000   2975.060000
```

There seems to be something wrong wth the SP column, there is no need to deal with this now as we will merge datasets and use the other, correct SP column Check if there are any nan values

[125]:
```
spot_df.isnull().values.any()
```

[125]:
```
True
```

Replace missing values with median, not mean - because the data is very skewed

[126]:
```
spot_df.fillna(value = spot_df["spot_price"].median(),inplace=True)
spot_df.isnull().values.any()
```

[126]:
```
False
```

1.0.3 System Prices

[127]:
```
system_df
```

[127]:
```
      Settlement Date  Settlement Period  System Sell Price(£/MWh)  \
0          08/04/2019                  1                     52.25
1          08/04/2019                  2                     51.90
2          08/04/2019                  3                     32.76
3          08/04/2019                  4                     50.85
4          08/04/2019                  5                     51.40
…                 …                  …                         …
47181      15/12/2021                 44                    295.00
47182      15/12/2021                 45                    176.55
47183      15/12/2021                 46                    176.55
47184      15/12/2021                 47                    350.00
47185      15/12/2021                 48                    350.00

       System Buy Price(£/MWh)  Net Imbalance Volume(MWh)
0                        52.25                   195.4258
1                        51.90                    62.2486
2                        32.76                   -40.7968
3                        50.85                    22.6933
4                        51.40                   186.5092
…                            …                          …
47181                   295.00                    47.1667
47182                   176.55                  -239.7501
47183                   176.55                  -297.1255
47184                   350.00                    65.4437
47185                   350.00                   336.3496

[47186 rows x 5 columns]
```

Check if there are any nan values

[128]:
```
system_df.isnull().values.any()
```

[128]:
```
False
```

1.0.4 Deal with the extra data in the system prices dataset

Row count does not match the others. Inspect:

[129]:
```
system_df["Settlement Period"].value_counts()
```

[129]:
```
1     983
36    983
```

27	983
28	983
29	983
30	983
31	983
32	983
33	983
34	983
35	983
37	983
25	983
38	983
39	983
40	983
41	983
42	983
43	983
44	983
45	983
46	983
2	983
26	983
24	983
12	983
3	983
4	983
5	983
6	983
7	983
8	983
9	983
10	983
23	983
11	983
13	983
14	983
15	983
16	983
17	983
18	983
19	983
20	983
21	983
22	983
47	981
48	981
49	3

```
50      3
Name: Settlement Period, dtype: int64
```

Seems there is some extra and missing data

[130]:
```
system_df.loc[ system_df["Settlement Period"] == 49]
```

[130]:

	Settlement Date	Settlement Period	System Sell Price(£/MWh)	System Buy Price(£/MWh)	Net Imbalance Volume(MWh)
9744	27/10/2019	49	17.00	17.00	-79.4619
27216	25/10/2020	49	17.34	17.34	-477.2735
45024	31/10/2021	49	16.90	16.90	-424.3871

[131]:
```
system_df.loc[ system_df["Settlement Period"] == 50]
```

[131]:

	Settlement Date	Settlement Period	System Sell Price(£/MWh)	System Buy Price(£/MWh)	Net Imbalance Volume(MWh)
9745	27/10/2019	50	45.00	45.00	105.5561
27217	25/10/2020	50	14.71	14.71	-760.7069
45025	31/10/2021	50	16.90	16.90	-444.9254

[132]:
```
system_df.iloc[9740:9750]
```

[132]:

	Settlement Date	Settlement Period	System Sell Price(£/MWh)	System Buy Price(£/MWh)	Net Imbalance Volume(MWh)
9740	27/10/2019	45	58.92796	58.92796	290.9965
9741	27/10/2019	46	16.25000	16.25000	-84.4829
9742	27/10/2019	47	48.50000	48.50000	376.2955
9743	27/10/2019	48	47.50000	47.50000	247.2096
9744	27/10/2019	49	17.00000		
9745	27/10/2019	50	45.00000		
9746	28/10/2019	1	22.66000		
9747	28/10/2019	2	24.06000		
9748	28/10/2019	3	24.34000		
9749	28/10/2019	4	22.17000		

```
9744                17.00000                -79.4619
9745                45.00000                105.5561
9746                22.66000               -342.0250
9747                24.06000               -212.9334
9748                24.34000               -117.4583
9749                22.17000               -277.2544
```

[133]:
```
system_df.iloc[27210:27220]
```

[133]:
```
      Settlement Date  Settlement Period  System Sell Price(£/MWh)  \
27210      25/10/2020                 43                     2.20
27211      25/10/2020                 44                    18.00
27212      25/10/2020                 45                    20.20
27213      25/10/2020                 46                     3.95
27214      25/10/2020                 47                    20.86
27215      25/10/2020                 48                     5.00
27216      25/10/2020                 49                    17.34
27217      25/10/2020                 50                    14.71
27218      26/10/2020                  1                    13.12
27219      26/10/2020                  2                    15.09

       System Buy Price(£/MWh)  Net Imbalance Volume(MWh)
27210                     2.20                 -603.6744
27211                    18.00                 -534.3983
27212                    20.20                 -967.2610
27213                     3.95                 -842.2514
27214                    20.86                 -603.4282
27215                     5.00                -1016.4918
27216                    17.34                 -477.2735
27217                    14.71                 -760.7069
27218                    13.12                 -716.9055
27219                    15.09                 -330.1523
```

[134]:
```
system_df.iloc[45020:45030]
```

[134]:
```
      Settlement Date  Settlement Period  System Sell Price(£/MWh)  \
45020      31/10/2021                 45                    175.00
45021      31/10/2021                 46                     76.00
45022      31/10/2021                 47                     30.92
45023      31/10/2021                 48                      0.00
45024      31/10/2021                 49                     16.90
45025      31/10/2021                 50                     16.90
45026      01/11/2021                  1                      0.00
45027      01/11/2021                  2                      1.00
45028      01/11/2021                  3                     -3.44
45029      01/11/2021                  4                     -8.52
```

	System Buy Price(£/MWh)	Net Imbalance Volume(MWh)
45020	175.00	341.7950
45021	76.00	-25.3079
45022	30.92	-357.9742
45023	0.00	-616.3338
45024	16.90	-424.3871
45025	16.90	-444.9254
45026	0.00	-516.4873
45027	1.00	-436.4728
45028	-3.44	-424.9166
45029	-8.52	-529.8139

Drop the aforementioned rows

[135]:
```
system_df_copy = system_df.copy()
```

[136]:
```
system_df_copy.drop([9744,9745,27216,27217,45024,45025], inplace=True)
```

[137]:
```
system_df_copy.reset_index(inplace=True)
system_df_copy.drop("index", axis=1,inplace=True)
```

[138]:
```
system_df_copy
```

[138]:

	Settlement Date	Settlement Period	System Sell Price(£/MWh)	\
0	08/04/2019	1	52.25	
1	08/04/2019	2	51.90	
2	08/04/2019	3	32.76	
3	08/04/2019	4	50.85	
4	08/04/2019	5	51.40	
…	…	…	…	
47175	15/12/2021	44	295.00	
47176	15/12/2021	45	176.55	
47177	15/12/2021	46	176.55	
47178	15/12/2021	47	350.00	
47179	15/12/2021	48	350.00	

	System Buy Price(£/MWh)	Net Imbalance Volume(MWh)
0	52.25	195.4258
1	51.90	62.2486
2	32.76	-40.7968
3	50.85	22.6933
4	51.40	186.5092
…	…	…
47175	295.00	47.1667
47176	176.55	-239.7501
47177	176.55	-297.1255
47178	350.00	65.4437
47179	350.00	336.3496

```
[47180 rows x 5 columns]
```

[139]:
```
k = 0
for i in range(system_df_copy.shape[0]):
    k += 1
    if system_df_copy["Settlement Period"][i] != k:
        print(k, system_df_copy["Settlement Period"][i], i)
    if k == 48:
        k = 0
```

```
47 1 17134
48 2 17135
1 3 17136
2 4 17137
3 5 17138
4 6 17139
5 7 17140
6 8 17141
7 9 17142
8 10 17143
9 11 17144
10 12 17145
11 13 17146
12 14 17147
13 15 17148
14 16 17149
15 17 17150
16 18 17151
17 19 17152
18 20 17153
19 21 17154
20 22 17155
21 23 17156
22 24 17157
23 25 17158
24 26 17159
25 27 17160
26 28 17161
27 29 17162
28 30 17163
29 31 17164
30 32 17165
31 33 17166
32 34 17167
33 35 17168
34 36 17169
35 37 17170
```

```
7 9 47140
8 10 47141
9 11 47142
10 12 47143
11 13 47144
12 14 47145
13 15 47146
14 16 47147
15 17 47148
16 18 47149
17 19 47150
18 20 47151
19 21 47152
20 22 47153
21 23 47154
22 24 47155
23 25 47156
24 26 47157
25 27 47158
26 28 47159
27 29 47160
28 30 47161
29 31 47162
30 32 47163
31 33 47164
32 34 47165
33 35 47166
34 36 47167
35 37 47168
36 38 47169
37 39 47170
38 40 47171
39 41 47172
40 42 47173
41 43 47174
42 44 47175
43 45 47176
44 46 47177
45 47 47178
46 48 47179
```

[142]:
```
system_df_copy.iloc[34600:34608]
```

[142]:

	Settlement Date	Settlement Period	System Sell Price(£/MWh)	\
34600	28/03/2021	43	74.95	
34601	28/03/2021	44	-61.00	
34602	28/03/2021	45	7.00	

```
34603      28/03/2021                46                    -42.00
34604      29/03/2021                 1                     29.67
34605      29/03/2021                 2                     27.53
34606      29/03/2021                 3                      3.07
34607      29/03/2021                 4                     24.44

       System Buy Price(£/MWh)  Net Imbalance Volume(MWh)
34600                    74.95                    90.3904
34601                   -61.00                  -390.3229
34602                     7.00                  -128.3773
34603                   -42.00                  -230.1941
34604                    29.67                   366.2934
34605                    27.53                   106.1758
34606                     3.07                   -11.2173
34607                    24.44                   169.7093
```

Add rows to replace missing values

[143]:
```
row1 =  pd.DataFrame({
    "Settlement Date": "29/03/2020",
    "Settlement Period": 47,
    "System Sell Price(£/MWh)":41.80,
    "Net Imbalance Volume(MWh)":383.3224
}, index=[17133])
```

[144]:
```
row2 =  pd.DataFrame({
    "Settlement Date": "29/03/2020",
    "Settlement Period": 48,
    "System Sell Price(£/MWh)":41.80,
    "Net Imbalance Volume(MWh)":383.3224
}, index=[17134])
```

[145]:
```
row3 =  pd.DataFrame({
    "Settlement Date": "28/03/2021",
    "Settlement Period": 47,
    "System Sell Price(£/MWh)":-42.00,
    "Net Imbalance Volume(MWh)":-230.1941
}, index=[34604])
```

[146]:
```
row4 =  pd.DataFrame({
    "Settlement Date": "28/03/2021",
    "Settlement Period": 48,
    "System Sell Price(£/MWh)":-42.00,
    "Net Imbalance Volume(MWh)":-230.1941
}, index=[34605])
```

Insert new rows

[147]:
```
df2 = pd.concat([system_df_copy.iloc[:17134],row1,system_df_copy.iloc[17134:]]).
↪reset_index(drop=True)
df2 = pd.concat([df2.iloc[:17135],row2,df2.iloc[17135:]]).reset_index(drop=True)
df2 = pd.concat([df2.iloc[:34606],row3,df2.iloc[34606:]]).reset_index(drop=True)
df2 = pd.concat([df2.iloc[:34607],row4,df2.iloc[34607:]]).reset_index(drop=True)
```

[148]:
```
df2.iloc[17130:17140]
```

[148]:

	Settlement Date	Settlement Period	System Sell Price(£/MWh)	System Buy Price(£/MWh)	Net Imbalance Volume(MWh)
17130	29/03/2020	43	38.50	38.50	21.5250
17131	29/03/2020	44	39.25	39.25	170.9372
17132	29/03/2020	45	41.80	41.80	482.7865
17133	29/03/2020	46	41.80	41.80	383.3224
17134	29/03/2020	47	41.80	NaN	383.3224
17135	29/03/2020	48	41.80	NaN	383.3224
17136	30/03/2020	1	39.00	39.00	203.3983
17137	30/03/2020	2	19.20	19.20	-1.8051
17138	30/03/2020	3	19.19	19.19	-106.6651
17139	30/03/2020	4	38.25	38.25	176.6504

[149]:
```
df2.iloc[34600:34610]
```

[149]:

	Settlement Date	Settlement Period	System Sell Price(£/MWh)	System Buy Price(£/MWh)	Net Imbalance Volume(MWh)
34600	28/03/2021	41	74.95		
34601	28/03/2021	42	8.00		
34602	28/03/2021	43	74.95		
34603	28/03/2021	44	-61.00		
34604	28/03/2021	45	7.00		
34605	28/03/2021	46	-42.00		
34606	28/03/2021	47	-42.00		
34607	28/03/2021	48	-42.00		
34608	29/03/2021	1	29.67		
34609	29/03/2021	2	27.53		

```
34600                74.95                 5.5144
34601                 8.00              -302.4828
34602                74.95                90.3904
34603               -61.00              -390.3229
34604                 7.00              -128.3773
34605               -42.00              -230.1941
34606                  NaN              -230.1941
34607                  NaN              -230.1941
34608                29.67               366.2934
34609                27.53               106.1758
```

[150]:
```
system_df = df2.copy()
```

1.1 2. Combine datasets and make adjustments

i.Drop extra price column from system price

[151]:
```
system_df.drop(labels=["System Buy Price(£/MWh)"],axis=1, inplace=True)
```

1.1.1 ii. Merge the dataframes

First merge spot and epex

[152]:
```
spot_df.reset_index(inplace=True)
spot_df.drop("index",axis=1,inplace=True)
spot_df.drop("timestamp",axis=1,inplace=True)
spot_df.head(1)
```

[152]:
```
    SP  spot_price
0  3.0       40.45
```

[153]:
```
epex_df.reset_index(inplace=True)
epex_df.drop("index",axis=1,inplace=True)
epex_df.head(1)
```

[153]:
```
                  timestamp  apx_da_hourly
0  2019-04-08 00:00:00+00:00          33.41
```

[154]:
```
combined = pd.concat([epex_df,system_df],axis=1)
combined = pd.concat([combined,spot_df],axis=1)
combined.head(1)
```

[154]:
```
                  timestamp  apx_da_hourly Settlement Date  \
0  2019-04-08 00:00:00+00:00          33.41      08/04/2019

   Settlement Period  System Sell Price(£/MWh)  Net Imbalance Volume(MWh)  \
0                  1                     52.25                   195.4258

    SP  spot_price
0  3.0       40.45
```

Drop duplicate columns

[155]:
```
combined.drop(["timestamp","SP"],axis=1,inplace=True)
combined
```

[155]:
```
       apx_da_hourly Settlement Date  Settlement Period  \
0              33.41      08/04/2019                  1
1              33.41      08/04/2019                  2
2              41.03      08/04/2019                  3
3              41.03      08/04/2019                  4
4              39.00      08/04/2019                  5
…                …               …                  …
47179         258.30      15/12/2021                 44
47180         231.80      15/12/2021                 45
47181         231.80      15/12/2021                 46
47182         231.00      15/12/2021                 47
47183         231.00      15/12/2021                 48

       System Sell Price(£/MWh)  Net Imbalance Volume(MWh)  spot_price
0                         52.25                   195.4258       40.45
1                         51.90                    62.2486       43.69
2                         32.76                   -40.7968       46.13
3                         50.85                    22.6933       43.96
4                         51.40                   186.5092       44.36
…                           …                         …          …
47179                    295.00                    47.1667      231.98
47180                    176.55                  -239.7501      238.32
47181                    176.55                  -297.1255      213.15
47182                    350.00                    65.4437      218.80
47183                    350.00                   336.3496      157.07

[47184 rows x 6 columns]
```

[156]:
```
cols = combined.columns
cols
```

[156]:
```
Index(['apx_da_hourly', 'Settlement Date', 'Settlement Period',
       'System Sell Price(£/MWh)', 'Net Imbalance Volume(MWh)', 'spot_price'],
      dtype='object')
```

[157]:
```
new_combined = combined[['Settlement Date','Settlement␣
↪Period','apx_da_hourly','spot_price','System Sell Price(£/MWh)', 'Net␣
↪Imbalance Volume(MWh)']]
```

Rename columns

[158]:
```
column_rename = {
    'apx_da_hourly': "EpexHourly"
    , 'Settlement Date': "Date"
```

```
    , 'Settlement Period': "Period"
    ,'System Sell Price(£/MWh)':"SystemPrice"
    , 'Net Imbalance Volume(MWh)':"ImbalanceVolume"
    , 'spot_price':"SpotPrice"
}
```

[159]:
```
new_combined.rename(columns=column_rename,inplace=True)
new_combined.head(1)
```

[159]:

	Date	Period	EpexHourly	SpotPrice	SystemPrice	ImbalanceVolume
0	08/04/2019	1	33.41	40.45	52.25	195.4258

Rename

[160]:
```
df = new_combined.copy()
```

1.1.2 iii.Add external data

- Add temperature data

[161]:
```
t1 = pd.read_csv("Temperature1.csv")
t1.head(1)
```

[161]:

	date	Average
0	20190408	9.0

Check if lengths match

[162]:
```
t1.shape[0] * 48 == df.shape[0]
```

[162]:
```
True
```

Duplicate values so temperature matches

[163]:
```
for i in range(t1.shape[0]):
    for k in range(i*48,i*48+48):
        df.loc[k, ("Temp")] = t1.iloc[i]["Average"]
```

[164]:
```
df.head()
```

[164]:

	Date	Period	EpexHourly	SpotPrice	SystemPrice	ImbalanceVolume	\
0	08/04/2019	1	33.41	40.45	52.25	195.4258	
1	08/04/2019	2	33.41	43.69	51.90	62.2486	
2	08/04/2019	3	41.03	46.13	32.76	-40.7968	
3	08/04/2019	4	41.03	43.96	50.85	22.6933	
4	08/04/2019	5	39.00	44.36	51.40	186.5092	

	Temp
0	9.0
1	9.0

```
2   9.0
3   9.0
4   9.0
```

1.1.3 iv. Create column for future system price

Make label column : System Price shifted by one

[165]:
```
df_new = df.copy()
df_new['NextSystemPrice'] = df['SystemPrice'].shift(-1)
```

[166]:
```
df = df_new.copy()
df.head()
```

[166]:

	Date	Period	EpexHourly	SpotPrice	SystemPrice	ImbalanceVolume	Temp	NextSystemPrice
0	08/04/2019	1	33.41	40.45	52.25	195.4258	9.0	51.90
1	08/04/2019	2	33.41	43.69	51.90	62.2486	9.0	32.76
2	08/04/2019	3	41.03	46.13	32.76	-40.7968	9.0	50.85
3	08/04/2019	4	41.03	43.96	50.85	22.6933	9.0	51.40
4	08/04/2019	5	39.00	44.36	51.40	186.5092	9.0	50.85

[167]:
```
df.fillna(value= y_df["NextSystemPrice"].median(),inplace=True)
```

1.1.4 Save file

[168]:
```
df.to_csv("combined_prices.csv")
```

1.1.5 v. Get extra data from API

Rolling demand Data

[169]:
```
import requests
```

API call parameters:

[170]:
```
key = "vo2mj812lhmtnhp"
host = "https://api.bmreports.com"
date_start = "2019-04-08"
date_end = "2021-12-15"
```

[171]:
```
end = "ROLSYSDEM"
base = f"https://api.bmreports.com/BMRS/{end}/V1?APIKey={key}"
date = f"&FromDateTime={date_start}&ToDateTime={date_end}&ServiceType=csv"
```

[172]:
```
rolling_demand = requests.get(base + date)
```

[173]:
```
rolling_demand
```

[173]:
```
<Response [200]>
```

Get wind forecast

[174]:
```
base = f"https://api.bmreports.com/BMRS/WINDFORFUELHH/V1?APIKey={key}"
date = f"&FromDateTime={date_start}&ToDateTime={date_end}&ServiceType=csv"
```

[175]:
```
wind = requests.get(base + date)
```

[176]:
```
wind
```

[176]:
```
<Response [200]>
```

————-Could not figure out how to parse this data so will leave them for now————

1.1.6 vi. Get correlation matrix for the prices

[177]:
```
df[["EpexHourly", "SpotPrice", "SystemPrice", "NextSystemPrice"]].corr()
```

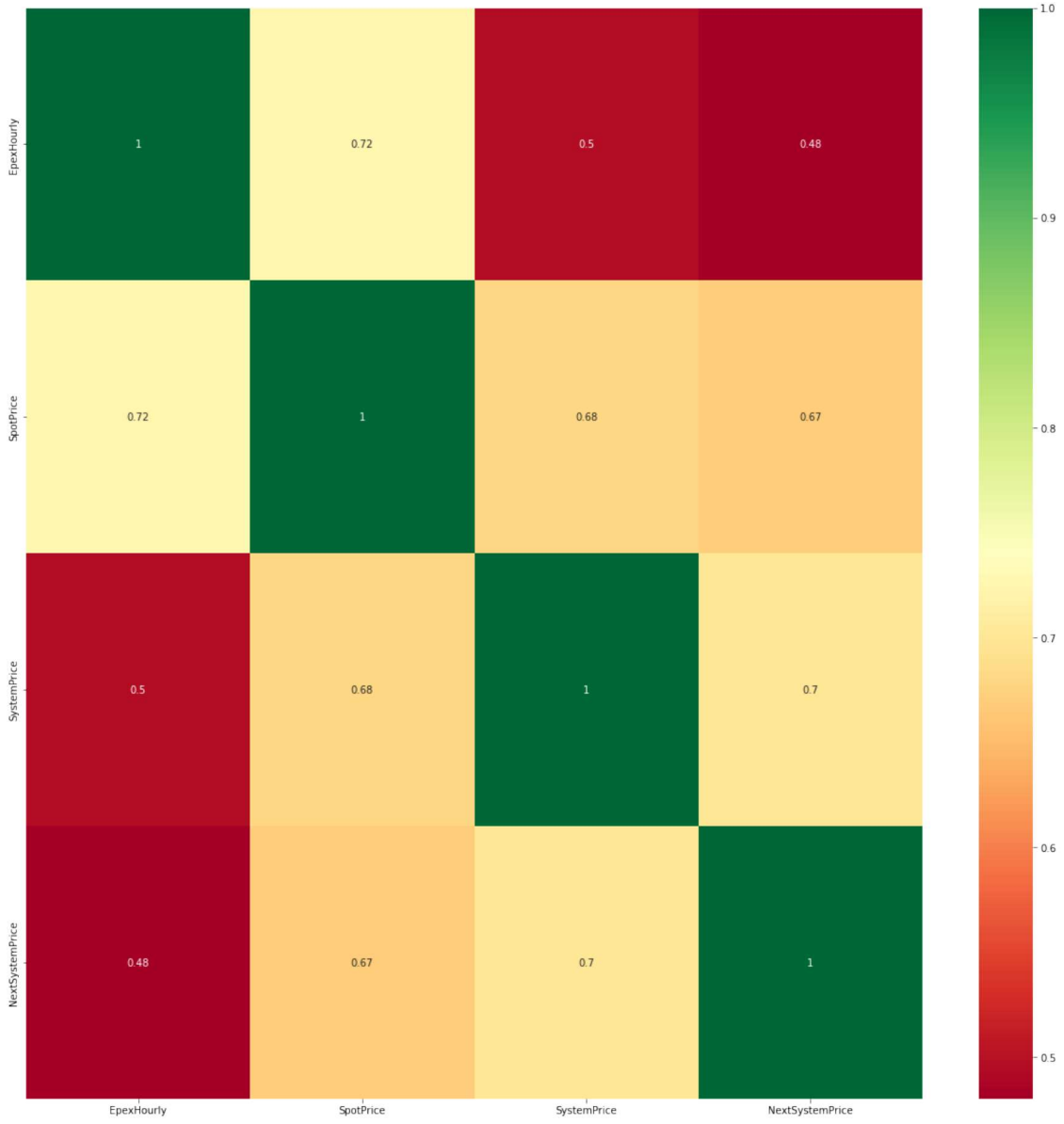
[177]:

	EpexHourly	SpotPrice	SystemPrice	NextSystemPrice
EpexHourly	1.000000	0.724093	0.495713	0.480205
SpotPrice	0.724093	1.000000	0.680486	0.669663
SystemPrice	0.495713	0.680486	1.000000	0.701251
NextSystemPrice	0.480205	0.669663	0.701251	1.000000

Get a Visual Correlation Matrix

[178]:
```
import seaborn as sns

data = df[["EpexHourly", "SpotPrice", "SystemPrice","NextSystemPrice"]]
X_cor = data.iloc[:,[0,1,2]]  #independent columns
y_cor = data.iloc[:,3]    #target column i.e price range
#get correlations of each features in dataset
corrmat = data.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



1.2 3. More feature preparation

1.2.1 i.Make seperate columns for year, month, day

[179]:
```
dates = df["Date"]
day = []
month = []
year = []

for i in range(len(dates)):
    day += [int(dates[i][0:2])]
```

```
    month += [int(dates[i][3:5])]
    year += [int(dates[i][6:10])]
```

Remove date column and add these as columns

[180]:
```
df.drop("Date",axis=1,inplace=True)
df["Day"] = day
df["Month"] = month
df["Year"] = year
df.head(1)
```

[180]:
```
   Period  EpexHourly  SpotPrice  SystemPrice  ImbalanceVolume  Temp  \
0       1       33.41      40.45        52.25         195.4258   9.0

   NextSystemPrice  Day  Month  Year
0             51.9    8      4  2019
```

Reorder columns

[181]:
```
df = df[['Year', 'Month','Day', 'Period', 'EpexHourly', 'SpotPrice',␣
 ↪'SystemPrice', 'ImbalanceVolume',
       'Temp', 'NextSystemPrice']]
df.head(1)
```

[181]:
```
   Year  Month  Day  Period  EpexHourly  SpotPrice  SystemPrice  \
0  2019      4    8       1       33.41      40.45        52.25

   ImbalanceVolume  Temp  NextSystemPrice
0         195.4258   9.0             51.9
```

1.2.2 iii. Scale Non-Date Data with Standard Scaler

[182]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
```

[183]:
```
scaled_features = df.copy()
col_names =␣
 ↪["Period","EpexHourly","SpotPrice","SystemPrice","ImbalanceVolume","Temp"]
features = scaled_features[col_names]
scaler = StandardScaler().fit(features.values)
features = scaler.transform(features.values)
```

[184]:
```
scaled_features[col_names] = features
scaled_features.head(1)
```

[184]:
```
   Year  Month  Day    Period  EpexHourly  SpotPrice  SystemPrice  \
0  2019      4    8 -1.696335      -0.456  -0.299587    -0.099008

   ImbalanceVolume      Temp  NextSystemPrice
```

```
0          0.541768 -0.655357              51.9
```

1.2.3 ii.Seperate features and labels

[185]:
```
x_df = df.drop("NextSystemPrice",axis=1)
y_df = df["NextSystemPrice"].to_frame()
```

[186]:
```
x_df.info()
y_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47184 entries, 0 to 47183
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Year             47184 non-null  int64
 1   Month            47184 non-null  int64
 2   Day              47184 non-null  int64
 3   Period           47184 non-null  int64
 4   EpexHourly       47184 non-null  float64
 5   SpotPrice        47184 non-null  float64
 6   SystemPrice      47184 non-null  float64
 7   ImbalanceVolume  47184 non-null  float64
 8   Temp             47184 non-null  float64
dtypes: float64(5), int64(4)
memory usage: 3.2 MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47184 entries, 0 to 47183
Data columns (total 1 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   NextSystemPrice  47184 non-null  float64
dtypes: float64(1)
memory usage: 368.8 KB
```

Fill in NAN Data in labels

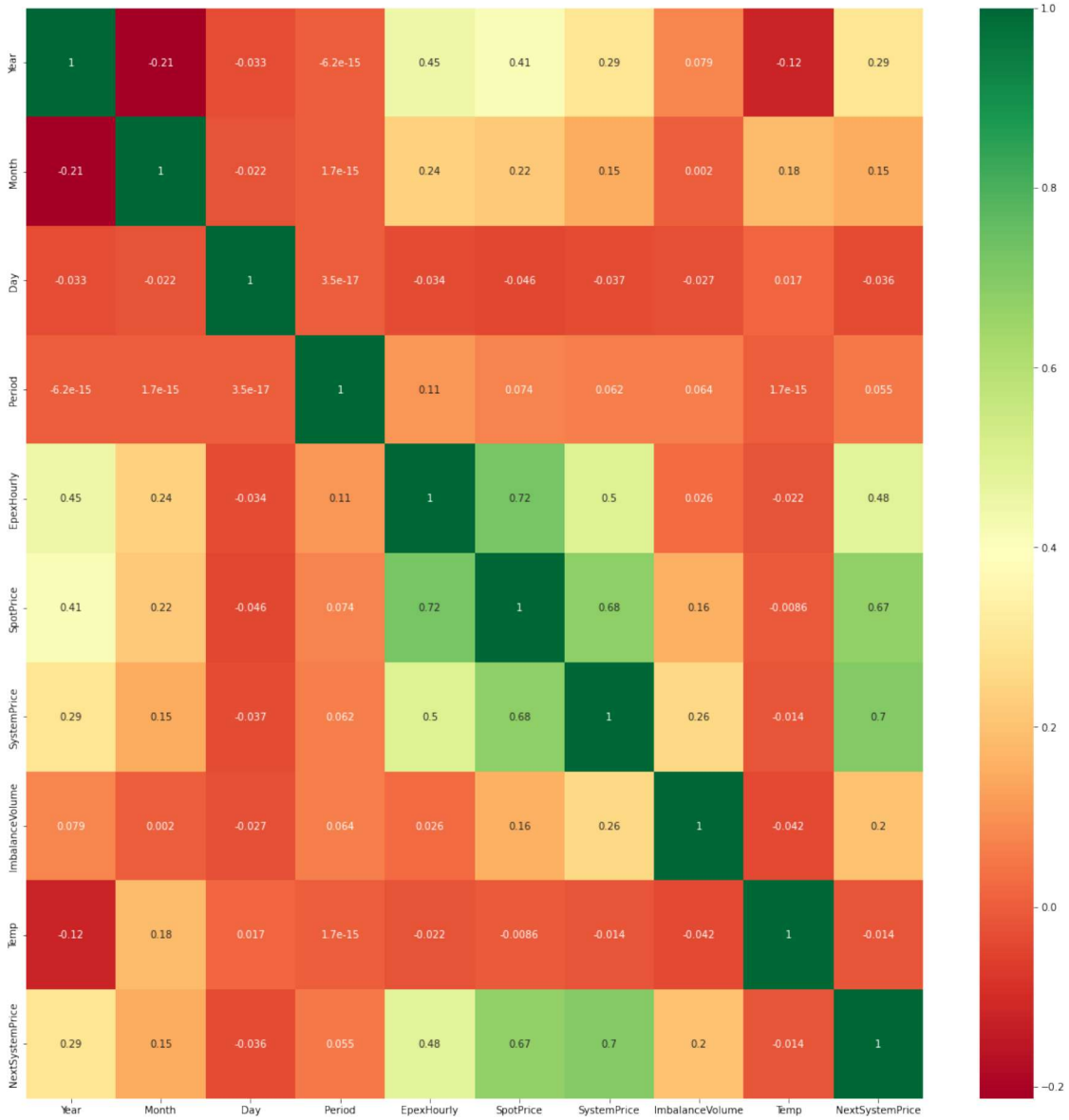
[187]:
```
y_df.fillna(value= y_df["NextSystemPrice"].median(),inplace=True)
```

[188]:
```
y_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47184 entries, 0 to 47183
Data columns (total 1 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   NextSystemPrice  47184 non-null  float64
dtypes: float64(1)
memory usage: 368.8 KB
```

1.2.4 iv.Correlation matrix for whole dataframe

[189]:
```
data = df
X_cor = data.iloc[:,[0,1,2,3,4,5,6]]  #independent columns
y_cor = data.iloc[:,7]    #target column i.e price range
#get correlations of each features in dataset
corrmat = data.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



1.2.5 iv. Split into test train sets

[204]:
```
X = x_df.to_numpy()
Y = y_df.to_numpy()
```

[205]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
↪random_state=42)
```

Save these as csv

[206]:
```
np.save("X_train",X_train)
np.save("Y_train",Y_train)
np.save("X_test",X_test)
np.save("Y_test",Y_test)
```

2 4.Apply ML

2.1 i.Run Linear Regression Model

Fit Model

[208]:
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X_train, Y_train)
```

[208]:
```
LinearRegression()
```

Import metric and calculate mse

[209]:
```
from sklearn.metrics import mean_squared_error
Y_train_predictions = lin_reg.predict(X_train)
lin_mse = mean_squared_error(Y_train, Y_train_predictions)
lin_rmse = np.sqrt(lin_mse)
lin_rmse
```

[209]:
```
63.45598190685875
```

Very poor performance

[210]:
```
lin_reg.coef_
```

[210]:
```
array([[ 3.43013732,  0.56431462, -0.02091173, -0.02068829,  0.02512883,
         0.47360038,  0.4084637 ,  0.01299131, -0.06292201]])
```

Test with test set

[212]:
```
Y_test_predictions = lin_reg.predict(X_test)
lin_mse = mean_squared_error(Y_test, Y_test_predictions)
lin_rmse = np.sqrt(lin_mse)
```

```
lin_rmse
```

[212]:
```
70.42201410659071
```

Performance slightly worse on test set

May need regularization

2.2 ii.Decision Tree

Fit Model

[213]:
```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor(random_state=42)
tree_reg.fit(X_train, Y_train)
```

[213]:
```
DecisionTreeRegressor(random_state=42)
```

Check accuracy

[214]:
```
Y_train_predictions = tree_reg.predict(X_train)
lin_mse = mean_squared_error(Y_train, Y_train_predictions)
lin_rmse = np.sqrt(lin_mse)
lin_mse
```

[214]:
```
1.7253915896639927e-31
```

Very accurate, probably overfitting

[215]:
```
Y_test_predictions = tree_reg.predict(X_test)
lin_mse = mean_squared_error(Y_test, Y_test_predictions)
lin_rmse = np.sqrt(lin_mse)
lin_mse
```

[215]:
```
11298.020174981186
```

Amazing overfitting

2.2.1 Recommendations for further work to improve predictive model

Check correlation matrix

[222]:
```
current_df = scaled_features.copy()
current_df.corr()["NextSystemPrice"]
```

[222]:
```
Year              0.293834
Month             0.149862
Day              -0.036445
Period            0.054667
EpexHourly        0.480205
SpotPrice         0.669663
```

```
SystemPrice        0.701251
ImbalanceVolume    0.202671
Temp              -0.014352
NextSystemPrice    1.000000
Name: NextSystemPrice, dtype: float64
```

- Above matrix suggest we need to drop day and possibly temperature
- This is likely because the day of month does not hold any important informaiton not contained in the month column, and temperature is not a good predictor of solar output,
- Will need to add datapoints that provide information on solar irradiation, cloudiness, and wind speed, ideally in the form of forecasts for the next 30 mins
- May have to timeshift the EPEX Prices to get a more relevant metric
- Although year seems like a relevant metric, the ups and downs in energy prices in the past three years may be peculiar and may cause overfitting when predicting future years
- Need to apply regularization to the ML model
- May add interaction terms
- May try more complicated ML methods than regression, and less complicated ones than decision trees (or at least a regularized decision tree)

[ ]: