datamerging-2

December 3, 2023

#

Forecast Solar Energy Outcome

1 Innovative Solutions for Extended Life of PV Modules

Millions of photovoltaic modules are going to reach the end of their proposed lifespan in close or midterm future, but many are still functional. You are challenged to come up with innovative solutions to evaluate the functionality of these modules and propose concepts for a second life- or re-use. Join forces with SOLARIMO to improve the efficiency and lifetime of PV modules and build a more circular energy industry.

1.1 ANSWER THE QUESTIONS:

1. How to recognize aging photovoltaic modules.
2. How do you use radiation data to assess:
    - Method of calculating losses caused by contamination.
    - Method of calculating losses caused by shading,
    - Method of calculating deviations from MPP.

2 Visualization and further exploration

In this notebook, we seek to understand the behaviour of two solar power plants through the data generated by the photovoltaic modules. To do so, we will talk about:

1. Reminder on photovoltaic systems or PV systems
2. EDA on:

- DC and AC power
- Irradiation
- ambient and module temperature
- yield

3. Correlation of all features
4. Comparison of two power plants

PV system is a power system designed to supply usable solar power by means of photovoltaics.

PV Cell is an electrical device that converts the energy of light directly into electricity by the photovoltaic effect, which is a physical and chemical phenomenon. It is also the basics photovoltaic device that is the building block PV modules.

Photovoltaic effect is the generation of voltage and electric current in a material upon exposure to light.

PV module is a group of PV cell connected in serie and/or parallel and encapsulated in an environmentally protective laminate.

PV panel is a group of modules that is the basic building block of a PV array.

PV array is a group of panels that comprises the complete PV generating unit. # PV inverter

PV inverter convert battery or PV array DC power to AC power for use with conventional utility-powered appliances. It is heart of PV systems because PV array is a DC source, an inverter is required to convert the dc power to normal ac power that is used in our homes and offices.

PV systems are very influenced by weather condition, if the weather is good, we get a maximun yield but if the weather is bad, we get a minimun yield. That is why there is important to know how weather condition can impact on yield of the two solar power plants.

2.0.1 Source:

[1]:
```
from IPython.display import IFrame

# Display PDFs using IFrame
display(IFrame(src='db_sm_ja_365_390m_120_jam60s20_mr_blackframe_1769x1052x35_1000_mc4.
 ↪pdf', width=800, height=600))
display(IFrame(src='SUN2000-60KTL-M0.pdf', width=800, height=600))
display(IFrame(src='SP_59368 Werne, Becklohof 3.pdf', width=800, height=600))
```

```
<IPython.lib.display.IFrame at 0x7f5c57b3b510>
```

```
<IPython.lib.display.IFrame at 0x7f5c57b3b510>
```

```
<IPython.lib.display.IFrame at 0x7f5c57b3b510>
```

According to the notion of PV systems, the important feature are: - DC power - AC power - Yield - ambiant Temperature - module temperature - irradiation

3 Part 1 - Data processing

[2]:
```
import pandas as pd

# Load data from a CSV file into a DataFrame
df = pd.read_csv('WernePlusIrradiationData.csv')

# Display the first few rows of the DataFrame to verify the data loading
df.head(5)
```

```
/tmp/ipykernel_191436/716414227.py:4: DtypeWarning: Columns (34,35,36,37,38,39,4
0,41,42,43,44,45,46,47,48,49,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,94)
have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('WernePlusIrradiationData.csv')
```

[2]:

	Sitename	Verwaltungsdomäne	Gerätename	\
0	59368 Werne Becklohhof 1-7	/Solarimo GmbH	102140090004/6T2199004131	
1	59368 Werne Becklohhof 1-7	/Solarimo GmbH	102140090004/6T2199004131	
2	59368 Werne Becklohhof 1-7	/Solarimo GmbH	102140090004/6T2199004131	
3	59368 Werne Becklohhof 1-7	/Solarimo GmbH	102140090004/6T2199004131	
4	59368 Werne Becklohhof 1-7	/Solarimo GmbH	102140090004/6T2199004131	

	Startzeit	Wechselrichterstatus	\
0	2022-11-01 00:00:00	-	
1	2022-11-01 07:05:00	-	
2	2022-11-01 07:10:00	-	
3	2022-11-01 07:15:00	-	
4	2022-11-01 07:20:00	-	

	Netzspannung/L1L2(AB)-Leitungsspannung im Stromnetz(V)	\
0	-	
1	-	
2	415.5	
3	-	
4	-	

	L2-L3(BC)-Spannung im Stromnetz(V)	L3-L1(CA)-Spannung im Stromnetz(V)	\
0	-	-	
1	-	-	
2	414.2	414.1	
3	-	-	
4	-	-	

	L1(A)-Phasen-Spannung(V)	L2(B)-Phasen-Spannung(V)	…	\
0	0.0	0.0	…	
1	0.0	0.0	…	
2	239.2	238.3	…	
3	239.1	238.3	…	
4	239.1	238.3	…	

	MPPT9 DC kumulierte Produktion(kWh)	MPPT10 DC kumulierte Produktion(kWh)	\
0	-	-	
1	-	-	
2	0	0	
3	-	-	
4	-	-	

	Produktion dieses Monates(kWh)	Produktion dises Jahres(kWh)	\
0	-	-	
1	-	-	
2	0	55643.09	
3	-	-	

```
4                                -                                -

   Startzeit Wechselrichter  Abschaltzeit Wechselrichter  \
0                         -                            -
1                         -                            -
2                         -                            -
3                         -                            -
4                         -                            -

   Spannung zwischen PV- und Erde(V)  Gesamtertrag(kWh)  \
0                               -0.1           56883.34
1                               -0.1           56883.34
2                               -0.1           56883.34
3                               -0.1           56883.34
4                               -0.1           56883.34

   Kumulierte Menge absorbierter Elektrizität(kWh)  \
0                                                -
1                                                -
2                                                -
3                                                -
4                                                -

   Irradiation (Satellite) Wh/m²
0                            NaN
1                            NaN
2                            NaN
3                            NaN
4                            NaN

[5 rows x 98 columns]
```

4 Displaying all column names

[3]:
```
# Displaying all column names
print(df.columns)
```

```
Index(['Sitename', 'Verwaltungsdomäne', 'Gerätename', 'Startzeit',
       'Wechselrichterstatus',
       'Netzspannung/L1L2(AB)-Leitungsspannung im Stromnetz(V)',
       'L2-L3(BC)-Spannung im Stromnetz(V)',
       'L3-L1(CA)-Spannung im Stromnetz(V)', 'L1(A)-Phasen-Spannung(V)',
       'L2(B)-Phasen-Spannung(V)', 'L3(C)-Phasen-Spannung(V)',
       'Netzstrom/L1(A)-Phasen-Strom im Stromnetz(A)',
       'L2(B)-Phasen-Stom im Stromnetz(A)',
       'L3(C)-Phasen-Strom im Stromnetz(A)', 'Wechselrichter-Effizienz(%)',
```

```
 'Innentemperatur(℃)', 'Leistungsfaktor', 'Stromnetzfrequenz(Hz)',
 'Wirkleistung AC(kW)', 'Blindleistung(kvar)',
 'Produktion dieses Tages(kWh)', 'Gesamte DC Eingangsleistung(kW)',
 'PV1 Eingangsspannung(V)', 'PV2 Eingangsspannung(V)',
 'PV3 Eingangsspannung(V)', 'PV4 Eingangsspannung(V)',
 'PV5 Eingangsspannung(V)', 'PV6 Eingangsspannung(V)',
 'PV7 Eingangsspannung(V)', 'PV8 Eingangsspannung(V)',
 'PV9 Eingangsspannung(V)', 'PV10 Eingangsspannung(V)',
 'PV11 Eingangsspannung(V)', 'PV12 Eingangsspannung(V)',
 'PV13 Eingangsspannung(V)', 'PV14 Eingangsspannung(V)',
 'PV15 Eingangsspannung(V)', 'PV16 Eingangsspannung(V)',
 'PV17 Eingangsspannung(V)', 'PV18 Eingangsspannung(V)',
 'PV19 Eingangsspannung(V)', 'PV20 Eingangsspannung(V)',
 'PV21 Eingangsspannung(V)', 'PV22 Eingangsspannung(V)',
 'PV23 Eingangsspannung(V)', 'PV24 Eingangsspannung(V)',
 'PV25 Eingangsspannung(V)', 'PV26 Eingangsspannung(V)',
 'PV27 Eingangsspannung(V)', 'PV28 Eingangsspannung(V)',
 'PV1 Eingangsstrom(A)', 'PV2 Eingangsstrom(A)', 'PV3 Eingangsstrom(A)',
 'PV4 Eingangsstrom(A)', 'PV5 Eingangsstrom(A)', 'PV6 Eingangsstrom(A)',
 'PV7 Eingangsstrom(A)', 'PV8 Eingangsstrom(A)', 'PV9 Eingangsstrom(A)',
 'PV10 Eingangsstrom(A)', 'PV11 Eingangsstrom(A)',
 'PV12 Eingangsstrom(A)', 'PV13 Eingangsstrom(A)',
 'PV14 Eingangsstrom(A)', 'PV15 Eingangsstrom(A)',
 'PV16 Eingangsstrom(A)', 'PV17 Eingangsstrom(A)',
 'PV18 Eingangsstrom(A)', 'PV19 Eingangsstrom(A)',
 'PV20 Eingangsstrom(A)', 'PV21 Eingangsstrom(A)',
 'PV22 Eingangsstrom(A)', 'PV23 Eingangsstrom(A)',
 'PV24 Eingangsstrom(A)', 'PV25 Eingangsstrom(A)',
 'PV26 Eingangsstrom(A)', 'PV27 Eingangsstrom(A)',
 'PV28 Eingangsstrom(A)', 'Der heutige Ertrag(kWh)',
 'Gesamte Produktion(kWh)', 'MPPT1 DC kumulierte Produktion(kWh)',
 'MPPT2 DC kumulierte Produktion(kWh)',
 'MPPT3 DC kumulierte Produktion(kWh)',
 'MPPT4 DC kumulierte Produktion(kWh)',
 'MPPT5 DC kumulierte Produktion(kWh)',
 'MPPT6 DC kumulierte Produktion(kWh)',
 'MPPT7 DC kumulierte Produktion(kWh)',
 'MPPT8 DC kumulierte Produktion(kWh)',
 'MPPT9 DC kumulierte Produktion(kWh)',
 'MPPT10 DC kumulierte Produktion(kWh)',
 'Produktion dieses Monates(kWh)', 'Produktion dises Jahres(kWh)',
 'Startzeit Wechselrichter', 'Abschaltzeit Wechselrichter',
 'Spannung zwischen PV- und Erde(V)', 'Gesamtertrag(kWh)',
 'Kumulierte Menge absorbierter Elektrizität(kWh)',
 'Irradiation (Satellite) Wh/m²'],
dtype='object')
```

GERMAN:ENGLISH

```
                                        'Sitename',:'Site name',
                                    'Verwaltungsdomäne',:'Administrative domain',
                                        'Gerätename',:'Device name',
                                        'Startzeit',:'Start time',
                                    'Wechselrichterstatus',:'Inverter status',
'Netzspannung/L1L2(AB)-Leitungsspannung im Stromnetz(V)',:'Mains voltage/L1L2(AB) line voltage
                    'L2-L3(BC)-Spannung im Stromnetz(V)',:'L2-L3(BC) voltage in the power grid
                    'L3-L1(CA)-Spannung im Stromnetz(V)',:'L3-L1(CA) voltage in the power grid
                            'L1(A)-Phasen-Spannung(V)',:'L1(A)-Phase Voltage(V)',
                            'L2(B)-Phasen-Spannung(V)',:'L2(B)-Phase Voltage(V)',
                            'L3(C)-Phasen-Spannung(V)',:'L3(C)-Phase Voltage(V)',
          'Netzstrom/L1(A)-Phasen-Strom im Stromnetz(A)',:'Mains current/L1(A) phase current in
                     'L2(B)-Phasen-Stom im Stromnetz(A)',:'L2(B) phase electricity in the power
                    'L3(C)-Phasen-Strom im Stromnetz(A)',:'L3(C) phase current in the power gri
                        'Wechselrichter-Effizienz(%)',:'Inverter Efficiency(%)',
                            'Innentemperatur(℃)',:'Indoor temperature(℃)',
                                'Leistungsfaktor',:'Power factor',
                            'Stromnetzfrequenz(Hz)',:'Power grid frequency(Hz)',
                            'Wirkleistung AC(kW)',:'Active power AC(kW)',
                            'Blindleistung(kvar)',:'Reactive power(kvar)',
                        'Produktion dieses Tages(kWh)',:'Production of this day (kWh)',
                            'PV1 Eingangsspannung(V)',:'PV1 input voltage(V)',
                            'PV2 Eingangsspannung(V)',:'PV2 input voltage(V)',
                            'PV3 Eingangsspannung(V)',:'PV3 input voltage(V)',
```

```
                'PV4 Eingangsspannung(V)',:'PV4 input voltage(V)',
                'PV5 Eingangsspannung(V)',:'PV5 input voltage(V)',
                'PV6 Eingangsspannung(V)',:'PV6 input voltage(V)',
                'PV7 Eingangsspannung(V)',:'PV7 input voltage(V)',
                'PV8 Eingangsspannung(V)',:'PV8 input voltage(V)',
                   'PV1 Eingangsstrom(A)',:'PV1 input current(A)',
                   'PV2 Eingangsstrom(A)',:'PV2 input current(A)',
                   'PV3 Eingangsstrom(A)',:'PV3 input current(A)',
                   'PV4 Eingangsstrom(A)',:'PV4 input current(A)',
                   'PV5 Eingangsstrom(A)',:'PV5 input current(A)',
                   'PV6 Eingangsstrom(A)',:'PV6 input current(A)',
                   'PV7 Eingangsstrom(A)',:'PV7 input current(A)',
                   'PV8 Eingangsstrom(A)',:'PV8 input current(A)',
                'Gesamte Produktion(kWh)',:'Total production(kWh)',
              'Startzeit Wechselrichter',:'Inverter start time',
           'Abschaltzeit Wechselrichter',:'Inverter switch-off time',
       'Gesamte DC Eingangsleistung(kW)',:'Total DC input power (kW)',
   'MPPT1 DC kumulierte Produktion(kWh)',:'MPPT1 DC cumulative production(kWh)
   'MPPT2 DC kumulierte Produktion(kWh)',:'MPPT2 DC cumulative production(kWh)
   'MPPT3 DC kumulierte Produktion(kWh)',:'MPPT3 DC cumulative production(kWh)
   'MPPT4 DC kumulierte Produktion(kWh)',:'MPPT4 DC cumulative production(kWh)
        'Produktion dieses Monates(kWh)',:'Production of this month(kWh)',
          'Produktion dises Jahres(kWh)',:'Production of this year (kWh)',
                    'Gesamtertrag(kWh)',:'Total yield(kWh)',
```

```
 'PV9 Eingangsspannung(V)',:'PV9 input voltage(V)',
'PV10 Eingangsspannung(V)',:'PV10 input voltage(V)',
'PV11 Eingangsspannung(V)',:'PV11 input voltage(V)',
'PV12 Eingangsspannung(V)',:'PV12 input voltage(V)',
'PV13 Eingangsspannung(V)',:'PV13 input voltage(V)',
'PV14 Eingangsspannung(V)',:'PV14 input voltage(V)',
'PV15 Eingangsspannung(V)',:'PV15 input voltage(V)',
'PV16 Eingangsspannung(V)',:'PV16 input voltage(V)',
'PV17 Eingangsspannung(V)',:'PV17 input voltage(V)',
'PV18 Eingangsspannung(V)',:'PV18 input voltage(V)',
'PV19 Eingangsspannung(V)',:'PV19 input voltage(V)',
'PV20 Eingangsspannung(V)',:'PV20 input voltage(V)',
'PV21 Eingangsspannung(V)',:'PV21 input voltage(V)',
'PV22 Eingangsspannung(V)',:'PV22 input voltage(V)',
'PV23 Eingangsspannung(V)',:'PV23 input voltage(V)',
'PV24 Eingangsspannung(V)',:'PV24 input voltage(V)',
'PV25 Eingangsspannung(V)',:'PV25 input voltage(V)',
'PV26 Eingangsspannung(V)',:'PV26 input voltage(V)',
'PV27 Eingangsspannung(V)',:'PV27 input voltage(V)',
'PV28 Eingangsspannung(V)',:'PV28 input voltage(V)',
    'PV9 Eingangsstrom(A)',:'PV9 input current(A)',
   'PV10 Eingangsstrom(A)',:'PV10 input current(A)',
   'PV11 Eingangsstrom(A)',:'PV11 input current(A)',
   'PV12 Eingangsstrom(A)',:'PV12 input current(A)',
```

```
                                'PV13 Eingangsstrom(A)',:'PV13 input current(A)',
                                'PV14 Eingangsstrom(A)',:'PV14 input current(A)',
                                'PV15 Eingangsstrom(A)',:'PV15 input current(A)',
                                'PV16 Eingangsstrom(A)',:'PV16 input current(A)',
                                'PV17 Eingangsstrom(A)',:'PV17 input current(A)',
                                'PV18 Eingangsstrom(A)',:'PV18 input current(A)',
                                'PV19 Eingangsstrom(A)',:'PV19 input current(A)',
                                'PV20 Eingangsstrom(A)',:'PV20 input current(A)',
                                'PV21 Eingangsstrom(A)',:'PV21 input current(A)',
                                'PV22 Eingangsstrom(A)',:'PV22 input current(A)',
                                'PV23 Eingangsstrom(A)',:'PV23 input current(A)',
                                'PV24 Eingangsstrom(A)',:'PV24 input current(A)',
                                'PV25 Eingangsstrom(A)',:'PV25 input current(A)',
                                'PV26 Eingangsstrom(A)',:'PV26 input current(A)',
                                'PV27 Eingangsstrom(A)',:'PV27 input current(A)',
                                'PV28 Eingangsstrom(A)',:'PV28 input current(A)',
                             'Der heutige Ertrag(kWh)',:'Today's yield (kWh)',
                  'MPPT5 DC kumulierte Produktion(kWh)',:'MPPT5 DC cumulative production(kWh)
                  'MPPT6 DC kumulierte Produktion(kWh)',:'MPPT6 DC cumulative production(kWh)
                  'MPPT7 DC kumulierte Produktion(kWh)',:'MPPT7 DC cumulative production(kWh)
                  'MPPT8 DC kumulierte Produktion(kWh)',:'MPPT8 DC cumulative production(kWh)
                  'MPPT9 DC kumulierte Produktion(kWh)',:'MPPT9 DC cumulative production(kWh)
                 'MPPT10 DC kumulierte Produktion(kWh)',:'MPPT10 DC cumulative production(kWh)
                    'Spannung zwischen PV- und Erde(V)',:'Voltage between PV-
and earth(V)',
```

```
'Kumulierte Menge absorbierter Elektrizität(kWh)',:'Cumulative amount of electricity abs
```

[4]:
```
# The columns contain many characters of German origin. To structure the␣
 ↪project, we translated them into English
translated_columns = [
    'Site name', 'management domain', 'device name', 'start time',
        'Inverter status',
        'Mains voltage/L1L2(AB) line voltage in the power grid(V)',
        'L2-L3(BC) voltage in the power grid(V)',
        'L3-L1(CA) voltage in the power grid(V)', 'L1(A) phase voltage(V)',
        'L2(B)-Phase-Voltage(V)', 'L3(C)-Phase-Voltage(V)',
        'Mains current/L1(A) phase current in the power grid(A)',
        'L2(B) phase electricity in the power grid(A)',
        'L3(C) phase current in the grid(A)', 'Inverter efficiency(%)',
        'Indoor temperature(℃)', 'Power factor', 'Power grid frequency(Hz)',
        'Active power AC(kW)', 'Reactive power(kvar)',
        'Production of this day(kWh)', 'Total DC input power(kW)',
        'PV1 input voltage(V)', 'PV2 input voltage(V)',
        'PV3 input voltage(V)', 'PV4 input voltage(V)',
        'PV5 input voltage(V)', 'PV6 input voltage(V)',
        'PV7 input voltage(V)', 'PV8 input voltage(V)',
        'PV9 input voltage(V)', 'PV10 input voltage(V)',
        'PV11 input voltage(V)', 'PV12 input voltage(V)',
        'PV13 input voltage(V)', 'PV14 input voltage(V)',
        'PV15 input voltage(V)', 'PV16 input voltage(V)',
        'PV17 input voltage(V)', 'PV18 input voltage(V)',
        'PV19 input voltage(V)', 'PV20 input voltage(V)',
        'PV21 input voltage(V)', 'PV22 input voltage(V)',
        'PV23 input voltage(V)', 'PV24 input voltage(V)',
        'PV25 input voltage(V)', 'PV26 input voltage(V)',
        'PV27 input voltage(V)', 'PV28 input voltage(V)',
        'PV1 input current(A)', 'PV2 input current(A)', 'PV3 input current(A)',
        'PV4 input current(A)', 'PV5 input current(A)', 'PV6 input current(A)',
        'PV7 input current(A)', 'PV8 input current(A)', 'PV9 input current(A)',
        'PV10 input current(A)', 'PV11 input current(A)',
        'PV12 input current(A)', 'PV13 input current(A)',
        'PV14 input current(A)', 'PV15 input current(A)',
        'PV16 input current(A)', 'PV17 input current(A)',
        'PV18 input current(A)', 'PV19 input current(A)',
        'PV20 input current(A)', 'PV21 input current(A)',
        'PV22 input current(A)', 'PV23 input current(A)',
        'PV24 input current(A)', 'PV25 input current(A)',
        'PV26 input current(A)', 'PV27 input current(A)',
        'PV28 input current(A)', "Today's yield(kWh)",
        'Total production(kWh)', 'MPPT1 DC cumulative production(kWh)',
        'MPPT2 DC cumulative production(kWh)',
```

```
        'MPPT3 DC cumulative production(kWh)',
        'MPPT4 DC cumulative production(kWh)',
        'MPPT5 DC cumulative production(kWh)',
        'MPPT6 DC cumulative production(kWh)',
        'MPPT7 DC cumulative production(kWh)',
        'MPPT8 DC cumulative production(kWh)',
        'MPPT9 DC cumulative production(kWh)',
        'MPPT10 DC cumulative production(kWh)',
        'Production of this month(kWh)', 'Production of this year(kWh)',
        'Inverter start time', 'Inverter switch-off time',
        'Voltage between PV and earth(V)', 'Total yield(kWh)',
        'Cumulative amount of electricity absorbed(kWh)',
        'Irradiation (Satellite) Wh/m²'
]
# Replace the existing column names with the translated names
df.columns = translated_columns

# Display the first few rows of the DataFrame to verify the columns translated
df.head()
```

[4]:

	Site name	management domain	device name	start time	Inverter status	Mains voltage/L1L2(AB) line voltage in the power grid(V)	L2-L3(BC) voltage in the power grid(V)
0	59368 Werne Becklohhof 1-7	/Solarimo GmbH	102140090004/6T2199004131	2022-11-01 00:00:00	-	-	-
1	59368 Werne Becklohhof 1-7	/Solarimo GmbH	102140090004/6T2199004131	2022-11-01 07:05:00	-	-	-
2	59368 Werne Becklohhof 1-7	/Solarimo GmbH	102140090004/6T2199004131	2022-11-01 07:10:00	-	415.5	414.2
3	59368 Werne Becklohhof 1-7	/Solarimo GmbH	102140090004/6T2199004131	2022-11-01 07:15:00	-	-	-
4	59368 Werne Becklohhof 1-7	/Solarimo GmbH	102140090004/6T2199004131	2022-11-01 07:20:00	-	-	-

	L3-L1(CA) voltage in the power grid(V)	L1(A) phase voltage(V)	\
0	-	0.0	
1	-	0.0	
2	414.1	239.2	
3	-	239.1	
4	-	239.1	

	L2(B)-Phase-Voltage(V)	…	MPPT9 DC cumulative production(kWh)	\
0	0.0	…	-	
1	0.0	…	-	
2	238.3	…	0	
3	238.3	…	-	
4	238.3	…	-	

	MPPT10 DC cumulative production(kWh)	Production of this month(kWh)	\
0	-	-	
1	-	-	
2	0	0	
3	-	-	
4	-	-	

	Production of this year(kWh)	Inverter start time	\
0	-	-	
1	-	-	
2	55643.09	-	
3	-	-	
4	-	-	

	Inverter switch-off time	Voltage between PV and earth(V)	\
0	-	-0.1	
1	-	-0.1	
2	-	-0.1	
3	-	-0.1	
4	-	-0.1	

	Total yield(kWh)	Cumulative amount of electricity absorbed(kWh)	\
0	56883.34	-	
1	56883.34	-	
2	56883.34	-	
3	56883.34	-	
4	56883.34	-	

	Irradiation (Satellite) Wh/m²
0	NaN
1	NaN
2	NaN

```
3                              NaN
4                              NaN

[5 rows x 98 columns]
```

[5]:
```
df.tail(5)
```

[5]:
```
                        Site name management domain  \
62018  59368 Werne Becklohhof 1-7    /Solarimo GmbH
62019  59368 Werne Becklohhof 1-7    /Solarimo GmbH
62020  59368 Werne Becklohhof 1-7    /Solarimo GmbH
62021  59368 Werne Becklohhof 1-7    /Solarimo GmbH
62022  59368 Werne Becklohhof 1-7    /Solarimo GmbH

                     device name           start time  \
62018  102140090004/6T2199004131  2023-11-30 16:50:00
62019  102140090004/6T2199004131  2023-11-30 16:55:00
62020  102140090004/6T2199004131  2023-11-30 17:00:00
62021  102140090004/6T2199004131  2023-11-30 17:05:00
62022  102140090004/6T2199004131  2023-11-30 17:10:00

                 Inverter status  \
62018  Standby : Initialisierung
62019  Standby : Initialisierung
62020                          -
62021                          -
62022                          -

      Mains voltage/L1L2(AB) line voltage in the power grid(V)  \
62018                                              404.7
62019                                              404.7
62020                                                  -
62021                                                  -
62022                                                  -

      L2-L3(BC) voltage in the power grid(V)  \
62018                                  405.8
62019                                  405.8
62020                                      -
62021                                      -
62022                                      -

      L3-L1(CA) voltage in the power grid(V)  L1(A) phase voltage(V)  \
62018                                  405.1                    231.5
62019                                  405.1                    231.5
62020                                      -                    231.5
62021                                      -                    231.5
```

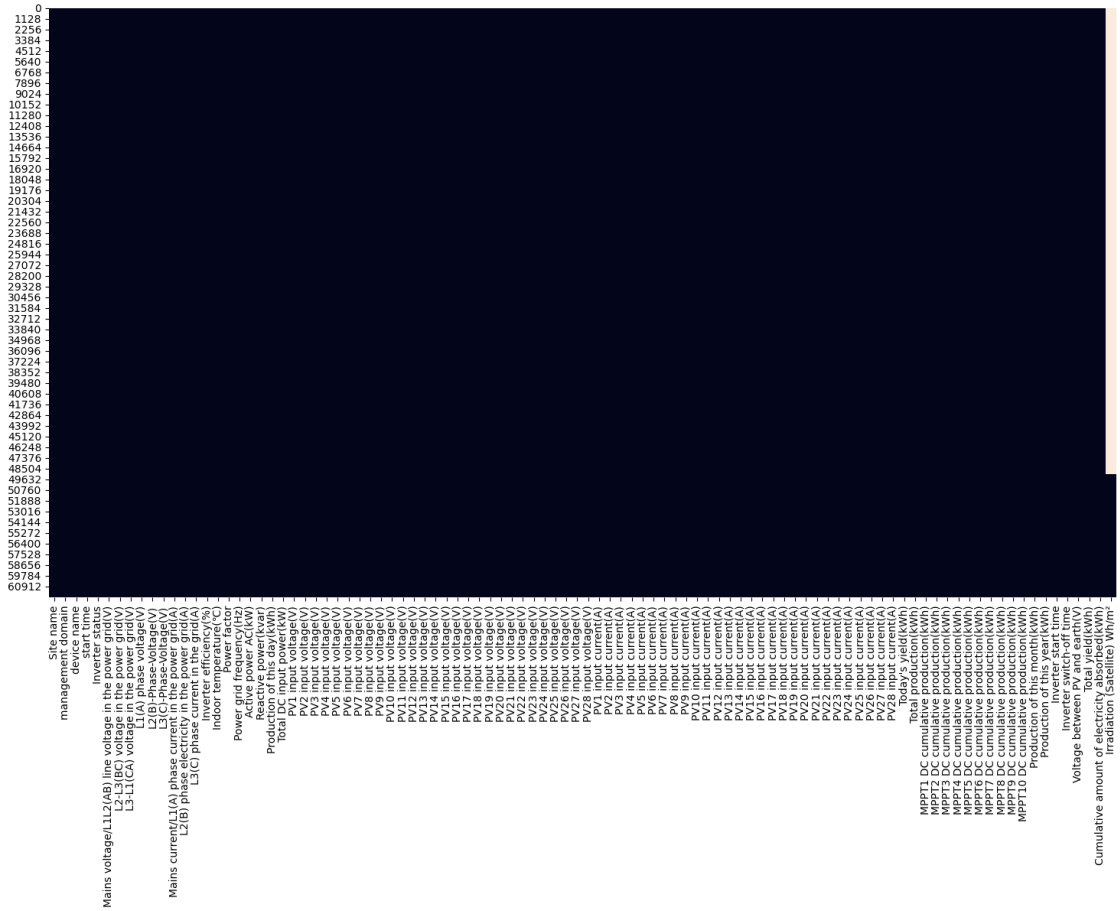
```
62022                                    -                    0.0

       L2(B)-Phase-Voltage(V)  …  MPPT9 DC cumulative production(kWh)  \
62018                   233.8  …                                    0
62019                   233.8  …                                    0
62020                   233.8  …                                    -
62021                   233.8  …                                    -
62022                     0.0  …                                    -

       MPPT10 DC cumulative production(kWh)  Production of this month(kWh)  \
62018                                     0                         1287.98
62019                                     0                         1287.98
62020                                     -                               -
62021                                     -                               -
62022                                     -                               -

       Production of this year(kWh)  Inverter start time  \
62018                      59805.45  2023/11/30 08:16:26
62019                      59805.45  2023/11/30 08:16:26
62020                             -                    -
62021                             -                    -
62022                             -                    -

       Inverter switch-off time  Voltage between PV and earth(V)  \
62018      2023/11/30 16:35:54                              0.0
62019      2023/11/30 16:35:54                              0.0
62020                        -                             -0.1
62021                        -                             -0.1
62022                        -                             -0.1

       Total yield(kWh)  Cumulative amount of electricity absorbed(kWh)  \
62018         120230.01                                                -
62019         120230.01                                                -
62020         120230.01                                                -
62021         120230.01                                                -
62022         120230.01                                                -

       Irradiation (Satellite) Wh/m²
62018                            0.0
62019                            0.0
62020                            0.0
62021                            0.0
62022                            0.0

[5 rows x 98 columns]
```

[6]:
```
df['Site name'].value_counts()
```

[6]:
```
Site name
59368 Werne Becklohhof 1-7    62023
Name: count, dtype: int64
```

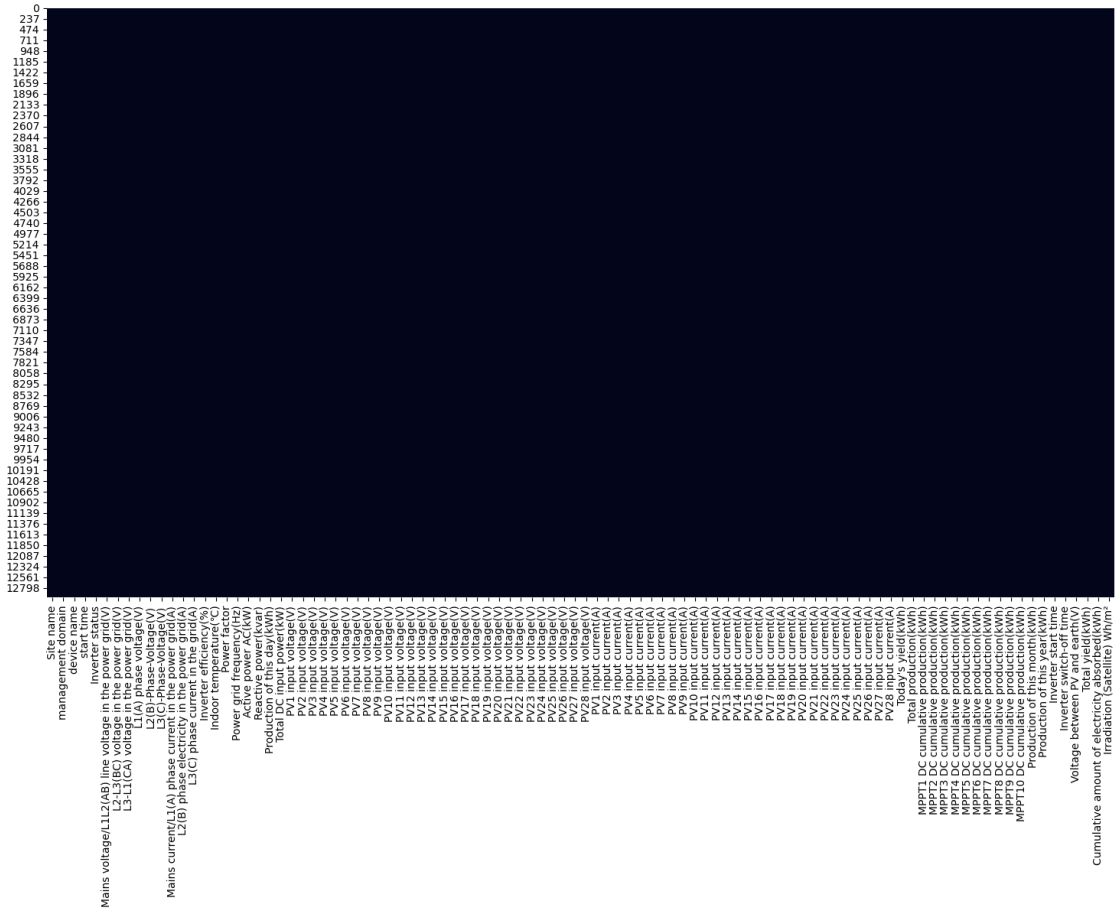
Missing visualization

[7]:
```
# Plotting Packages
import matplotlib.pyplot as plt
import seaborn as sns

# Missing visualization
plt.figure(figsize=(18,10))
sns.heatmap(df.isna(), cbar=False)
plt.show()
```



[8]:
```
# Remove rows where 'Irradiation' column is empty
df.dropna(subset=['Irradiation (Satellite) Wh/m²'], inplace=True)

# Plotting Packages
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Resetting the index to start from 0
df.reset_index(drop=True, inplace=True)

# Missing visualization
plt.figure(figsize=(18,10))
sns.heatmap(df.isna(), cbar=False)
plt.show()
```

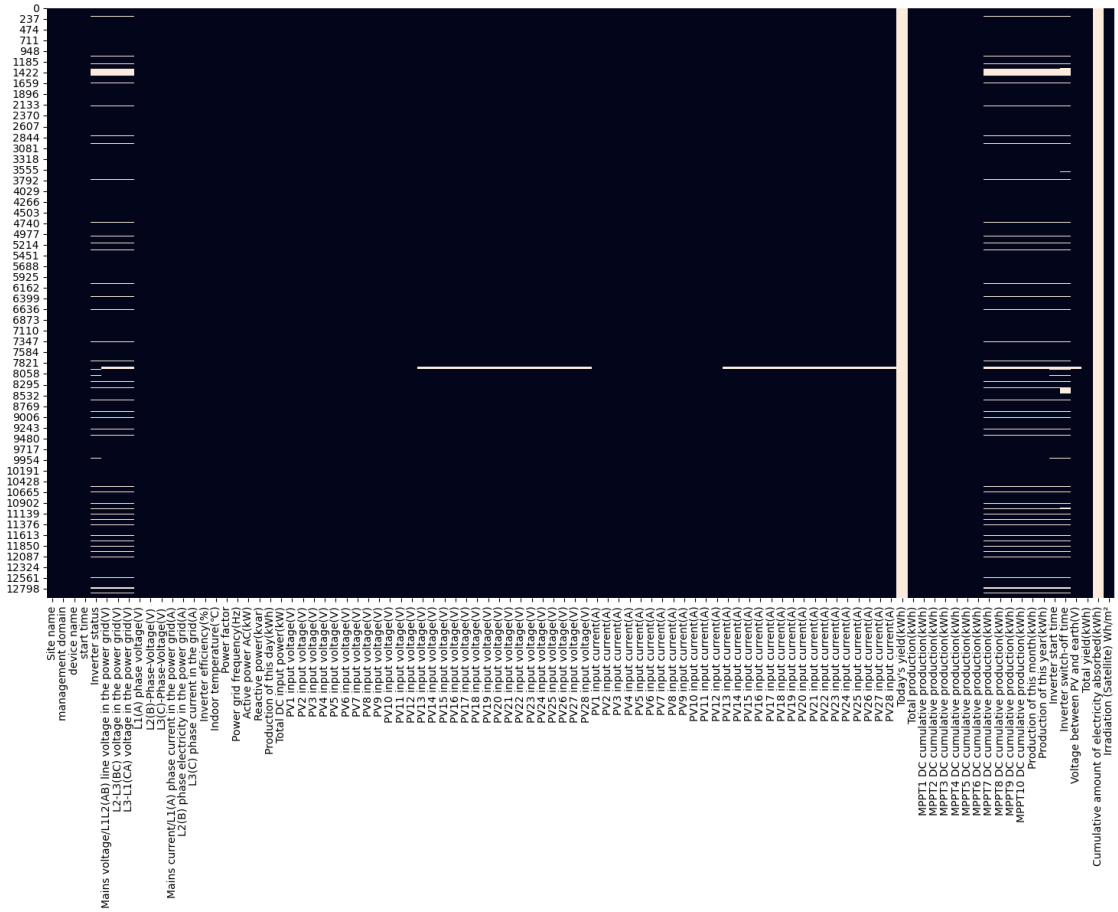


I replace "-" with " " to find NaN in the cells

[9]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Function to replace hyphen with space if cell contains only a hyphen
def replace_hyphen_with_space(cell):
```

```
    if cell.strip() == '-':
        return ' '
    return cell

# Apply the function to each cell in the DataFrame
df = df.applymap(lambda x: replace_hyphen_with_space(x) if isinstance(x, str)␣
 ↪else x)

# Replace spaces back to NaN
df = df.replace(' ', np.nan)

# Plotting missing values using heatmap
plt.figure(figsize=(18,10))
sns.heatmap(df.isna(), cbar=False)
plt.show()
```

```
/tmp/ipykernel_191436/1139617246.py:12: FutureWarning: DataFrame.applymap has
been deprecated. Use DataFrame.map instead.
  df = df.applymap(lambda x: replace_hyphen_with_space(x) if isinstance(x, str)
else x)
```



[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12996 entries, 0 to 12995
Data columns (total 98 columns):
 #   Column                                                  Non-Null Count
Dtype
---  ------                                                  --------------
-----
 0   Site name                                               12996 non-null
object
 1   management domain                                       12996 non-null
object
 2   device name                                             12996 non-null
object
 3   start time                                              12996 non-null
object
 4   Inverter status                                         12183 non-null
object
 5   Mains voltage/L1L2(AB) line voltage in the power grid(V)  12158 non-null
object
 6   L2-L3(BC) voltage in the power grid(V)                  12158 non-null
object
 7   L3-L1(CA) voltage in the power grid(V)                  12158 non-null
object
 8   L1(A) phase voltage(V)                                  12996 non-null
float64
 9   L2(B)-Phase-Voltage(V)                                  12996 non-null
float64
 10  L3(C)-Phase-Voltage(V)                                  12996 non-null
float64
 11  Mains current/L1(A) phase current in the power grid(A)  12996 non-null
float64
 12  L2(B) phase electricity in the power grid(A)            12996 non-null
float64
 13  L3(C) phase current in the grid(A)                      12996 non-null
float64
 14  Inverter efficiency(%)                                  12996 non-null
float64
 15  Indoor temperature(℃)                                   12996 non-null
float64
 16  Power factor                                            12996 non-null
float64
 17  Power grid frequency(Hz)                                12996 non-null
float64
```

```
 18  Active power AC(kW)                                  12996 non-null
float64
 19  Reactive power(kvar)                                 12996 non-null
float64
 20  Production of this day(kWh)                          12996 non-null
float64
 21  Total DC input power(kW)                             12996 non-null
float64
 22  PV1 input voltage(V)                                 12996 non-null
float64
 23  PV2 input voltage(V)                                 12996 non-null
float64
 24  PV3 input voltage(V)                                 12996 non-null
float64
 25  PV4 input voltage(V)                                 12996 non-null
float64
 26  PV5 input voltage(V)                                 12996 non-null
float64
 27  PV6 input voltage(V)                                 12996 non-null
float64
 28  PV7 input voltage(V)                                 12996 non-null
float64
 29  PV8 input voltage(V)                                 12996 non-null
float64
 30  PV9 input voltage(V)                                 12996 non-null
float64
 31  PV10 input voltage(V)                                12996 non-null
float64
 32  PV11 input voltage(V)                                12996 non-null
float64
 33  PV12 input voltage(V)                                12996 non-null
float64
 34  PV13 input voltage(V)                                12940 non-null
object
 35  PV14 input voltage(V)                                12940 non-null
object
 36  PV15 input voltage(V)                                12940 non-null
object
 37  PV16 input voltage(V)                                12940 non-null
object
 38  PV17 input voltage(V)                                12940 non-null
object
 39  PV18 input voltage(V)                                12940 non-null
object
 40  PV19 input voltage(V)                                12940 non-null
object
 41  PV20 input voltage(V)                                12940 non-null
object
```

```
 42  PV21 input voltage(V)                                  12940 non-null
object
 43  PV22 input voltage(V)                                  12940 non-null
object
 44  PV23 input voltage(V)                                  12940 non-null
object
 45  PV24 input voltage(V)                                  12940 non-null
object
 46  PV25 input voltage(V)                                  12940 non-null
object
 47  PV26 input voltage(V)                                  12940 non-null
object
 48  PV27 input voltage(V)                                  12940 non-null
object
 49  PV28 input voltage(V)                                  12940 non-null
object
 50  PV1 input current(A)                                   12996 non-null
float64
 51  PV2 input current(A)                                   12996 non-null
float64
 52  PV3 input current(A)                                   12996 non-null
float64
 53  PV4 input current(A)                                   12996 non-null
float64
 54  PV5 input current(A)                                   12996 non-null
float64
 55  PV6 input current(A)                                   12996 non-null
float64
 56  PV7 input current(A)                                   12996 non-null
float64
 57  PV8 input current(A)                                   12996 non-null
float64
 58  PV9 input current(A)                                   12996 non-null
float64
 59  PV10 input current(A)                                  12996 non-null
float64
 60  PV11 input current(A)                                  12996 non-null
float64
 61  PV12 input current(A)                                  12996 non-null
float64
 62  PV13 input current(A)                                  12940 non-null
object
 63  PV14 input current(A)                                  12940 non-null
object
 64  PV15 input current(A)                                  12940 non-null
object
 65  PV16 input current(A)                                  12940 non-null
object
```

```
 66  PV17 input current(A)                                  12940 non-null
object
 67  PV18 input current(A)                                  12940 non-null
object
 68  PV19 input current(A)                                  12940 non-null
object
 69  PV20 input current(A)                                  12940 non-null
object
 70  PV21 input current(A)                                  12940 non-null
object
 71  PV22 input current(A)                                  12940 non-null
object
 72  PV23 input current(A)                                  12940 non-null
object
 73  PV24 input current(A)                                  12940 non-null
object
 74  PV25 input current(A)                                  12940 non-null
object
 75  PV26 input current(A)                                  12940 non-null
object
 76  PV27 input current(A)                                  12940 non-null
object
 77  PV28 input current(A)                                  12940 non-null
object
 78  Today's yield(kWh)                                     0 non-null
float64
 79  Total production(kWh)                                  12996 non-null
float64
 80  MPPT1 DC cumulative production(kWh)                    12996 non-null
float64
 81  MPPT2 DC cumulative production(kWh)                    12996 non-null
float64
 82  MPPT3 DC cumulative production(kWh)                    12996 non-null
float64
 83  MPPT4 DC cumulative production(kWh)                    12996 non-null
float64
 84  MPPT5 DC cumulative production(kWh)                    12996 non-null
float64
 85  MPPT6 DC cumulative production(kWh)                    12996 non-null
float64
 86  MPPT7 DC cumulative production(kWh)                    12158 non-null
object
 87  MPPT8 DC cumulative production(kWh)                    12158 non-null
object
 88  MPPT9 DC cumulative production(kWh)                    12158 non-null
object
 89  MPPT10 DC cumulative production(kWh)                   12158 non-null
object
```

```
 90  Production of this month(kWh)                          12158 non-null
object
 91  Production of this year(kWh)                           12158 non-null
object
 92  Inverter start time                                    12127 non-null
object
 93  Inverter switch-off time                               11980 non-null
object
 94  Voltage between PV and earth(V)                        12940 non-null
object
 95  Total yield(kWh)                                       12996 non-null
float64
 96  Cumulative amount of electricity absorbed(kWh)         0 non-null
float64
 97  Irradiation (Satellite) Wh/m²                          12996 non-null
float64
dtypes: float64(49), object(49)
memory usage: 9.7+ MB
```

The data frame information shows that the columns ('Cumulative amount of electricity absorbed(kWh)', 'Today's yield (kWh)') are completely empty. That's why I delete them.

[11]:
```
df.drop(["Today's yield(kWh)", "Cumulative amount of electricity␣
↪absorbed(kWh)"], axis=1, inplace=True)
```

4.1 For the Werne location, line 6 is missing. Line PV6 are not connected for the Werne input current PV(A) = 0, which means that the string is not connected to the installation. Where PV(A) = 0 means that we can delete this data. Data analysis allows us to determine which lines are connected and which are not. We also checked it with the drawing and it turns out that there are 11 lines in the drawing, but only 10 are connected to the current. The version is for PV12, above PV12 we can delete the data.

For the PV13-28, we perform a value calculation and remove the columns with values of current(A) = 0.

The "PV13-PV28 input current(V)"

[12]:
```
df['PV13 input voltage(V)'].value_counts()
```

[12]:
```
PV13 input voltage(V)
0       7657
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[13]:
```
df['PV14 input voltage(V)'].value_counts()
```

[13]:
```
PV14 input voltage(V)
0       7657
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[14]:
```
df['PV15 input voltage(V)'].value_counts()
```

[14]:
```
PV15 input voltage(V)
0       7657
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[15]:
```
df['PV16 input voltage(V)'].value_counts()
```

[15]:
```
PV16 input voltage(V)
0       7657
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[16]:
```
df['PV17 input voltage(V)'].value_counts()
```

[16]:
```
PV17 input voltage(V)
0       7657
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[17]:
```
df['PV18 input voltage(V)'].value_counts()
```

[17]:
```
PV18 input voltage(V)
0       7657
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[18]:
```
df['PV19 input voltage(V)'].value_counts()
```

[18]:
```
PV19 input voltage(V)
0       7657
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[19]:
```
df['PV20 input voltage(V)'].value_counts()
```

[19]:
```
PV20 input voltage(V)
0       7657
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[20]:
```
df['PV21 input voltage(V)'].value_counts()
```

[20]:
```
PV21 input voltage(V)
0       7657
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[21]:
```
df['PV22 input voltage(V)'].value_counts()
```

[21]:
```
PV22 input voltage(V)
0       7657
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[22]:
```
df['PV23 input voltage(V)'].value_counts()
```

[22]:
```
PV23 input voltage(V)
0       7657
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[23]:
```
df['PV24 input voltage(V)'].value_counts()
```

[23]:
```
PV24 input voltage(V)
0       7657
```

```
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[24]:
```
df['PV25 input voltage(V)'].value_counts()
```

[24]:
```
PV25 input voltage(V)
-0.1    8136
-0.1    4804
Name: count, dtype: int64
```

[25]:
```
df['PV26 input voltage(V)'].value_counts()
```

[25]:
```
PV26 input voltage(V)
-0.1    8136
-0.1    4804
Name: count, dtype: int64
```

[26]:
```
df['PV27 input voltage(V)'].value_counts()
```

[26]:
```
PV27 input voltage(V)
-0.1    8136
-0.1    4804
Name: count, dtype: int64
```

[27]:
```
df['PV28 input voltage(V)'].value_counts()
```

[27]:
```
PV28 input voltage(V)
-0.1    8136
-0.1    4804
Name: count, dtype: int64
```

The "PV13-PV28 input current(A)"

[28]:
```
df['PV13 input current(A)'].value_counts()
```

[28]:
```
PV13 input current(A)
0        7657
0.0      4363
-0.01     479
-0.01     441
Name: count, dtype: int64
```

[29]:
```
df['PV14 input current(A)'].value_counts()
```

[29]:
```
PV14 input current(A)
0        7657
0.0      4363
-0.01     479
-0.01     441
Name: count, dtype: int64
```

[30]:
```
df['PV15 input current(A)'].value_counts()
```

[30]:
```
PV15 input current(A)
0        7657
0.0      4363
-0.01     479
-0.01     441
Name: count, dtype: int64
```

[31]:
```
df['PV16 input current(A)'].value_counts()
```

[31]:
```
PV16 input current(A)
0        7657
0.0      4363
-0.01     479
-0.01     441
Name: count, dtype: int64
```

[32]:
```
df['PV17 input current(A)'].value_counts()
```

[32]:
```
PV17 input current(A)
0        7657
0.0      4363
-0.01     479
-0.01     441
Name: count, dtype: int64
```

[33]:
```
df['PV18 input current(A)'].value_counts()
```

[33]:
```
PV18 input current(A)
0        7657
0.0      4363
-0.01     479
-0.01     441
Name: count, dtype: int64
```

[34]:
```
df['PV19 input current(A)'].value_counts()
```

[34]:
```
PV19 input current(A)
0        7657
```

```
0.0      4363
-0.01     479
-0.01     441
Name: count, dtype: int64
```

[35]:
```
df['PV20 input current(A)'].value_counts()
```

[35]:
```
PV20 input current(A)
0        7657
0.0      4363
-0.01     479
-0.01     441
Name: count, dtype: int64
```

[36]:
```
df['PV21 input current(A)'].value_counts()
```

[36]:
```
PV21 input current(A)
0        7657
0.0      4363
-0.01     479
-0.01     441
Name: count, dtype: int64
```

[37]:
```
df['PV22 input current(A)'].value_counts()
```

[37]:
```
PV22 input current(A)
0        7657
0.0      4363
-0.01     479
-0.01     441
Name: count, dtype: int64
```

[38]:
```
df['PV23 input current(A)'].value_counts()
```

[38]:
```
PV23 input current(A)
0        7657
0.0      4363
-0.01     479
-0.01     441
Name: count, dtype: int64
```

[39]:
```
df['PV24 input current(A)'].value_counts()
```

[39]:
```
PV24 input current(A)
0        7657
0.0      4363
-0.01     479
```

```
-0.01     441
Name: count, dtype: int64
```

[40]:
```
df['PV25 input current(A)'].value_counts()
```

[40]:
```
PV25 input current(A)
-0.01    8136
-0.01    4804
Name: count, dtype: int64
```

[41]:
```
df['PV26 input current(A)'].value_counts()
```

[41]:
```
PV26 input current(A)
-0.01    8136
-0.01    4804
Name: count, dtype: int64
```

[42]:
```
df['PV27 input current(A)'].value_counts()
```

[42]:
```
PV27 input current(A)
-0.01    8136
-0.01    4804
Name: count, dtype: int64
```

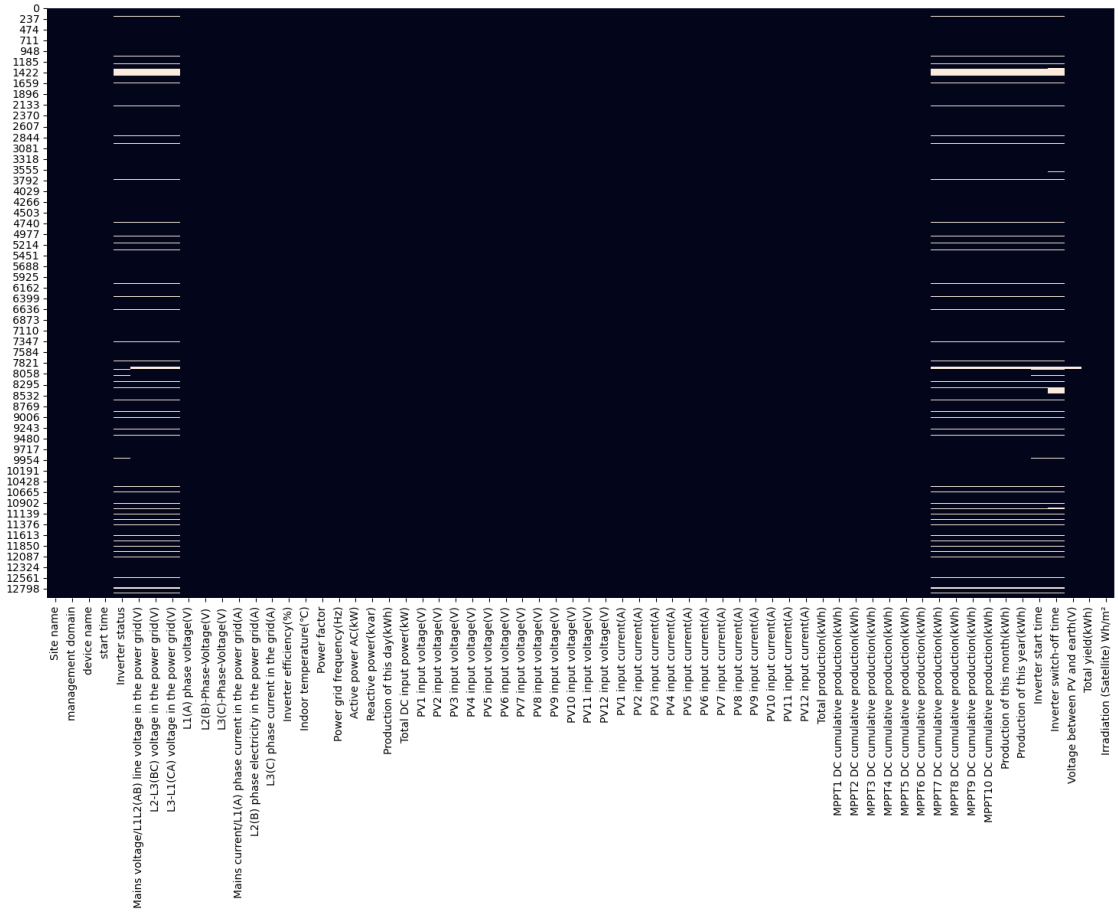
[43]:
```
df['PV28 input current(A)'].value_counts()
```

[43]:
```
PV28 input current(A)
-0.01    8136
-0.01    4804
Name: count, dtype: int64
```

[44]:
```
# I remove those columns that have a value close to zero. These speakers do not␣
 ↪add anything new, apart from the input voltage PV13 (V), which is very small␣
 ↪and can be ignored.
df.drop(['PV13 input voltage(V)', 'PV14 input voltage(V)', 'PV15 input␣
 ↪voltage(V)', 'PV16 input voltage(V)',
'PV17 input voltage(V)', 'PV18 input voltage(V)', 'PV19 input voltage(V)',␣
 ↪'PV20 input voltage(V)',
'PV21 input voltage(V)', 'PV22 input voltage(V)', 'PV23 input voltage(V)',␣
 ↪'PV24 input voltage(V)',
'PV25 input voltage(V)', 'PV26 input voltage(V)', 'PV27 input voltage(V)',␣
 ↪'PV28 input voltage(V)',

        'PV13 input current(A)', 'PV14 input current(A)', 'PV15 input␣
 ↪current(A)', 'PV16 input current(A)',
'PV17 input current(A)', 'PV18 input current(A)', 'PV19 input current(A)',␣
 ↪'PV20 input current(A)',
```

```
'PV21 input current(A)', 'PV22 input current(A)', 'PV23 input current(A)',␣
 ↪'PV24 input current(A)',
'PV25 input current(A)', 'PV26 input current(A)', 'PV27 input current(A)',␣
 ↪'PV28 input current(A)'
], axis=1, inplace=True)
```

[45]:
```
# Plotting Packages
import matplotlib.pyplot as plt
import seaborn as sns

# Missing visualization
plt.figure(figsize=(18,10))
sns.heatmap(df.isna(), cbar=False)
plt.show()
```



I am removing the columns (“management domain”, “MPPT7 DC cumulative production(kWh)”, “MPPT8 DC cumulative production(kWh)”, “MPPT9 DC cumulative production(kWh)”, “MPPT10 DC cumulative production(kWh)”, “Voltage between PV and earth(V)”) - mostly the most common value inside this columns is zero or similar to zero

[46]:
```
df['management domain'].value_counts()
```

[46]:
```
management domain
/Solarimo GmbH    12996
Name: count, dtype: int64
```

[47]:
```
df['MPPT7 DC cumulative production(kWh)'].value_counts()
```

[47]:
```
MPPT7 DC cumulative production(kWh)
0    12158
Name: count, dtype: int64
```

[48]:
```
df['MPPT8 DC cumulative production(kWh)'].value_counts()
```

[48]:
```
MPPT8 DC cumulative production(kWh)
0    12158
Name: count, dtype: int64
```

[49]:
```
df['MPPT9 DC cumulative production(kWh)'].value_counts()
```

[49]:
```
MPPT9 DC cumulative production(kWh)
0    12158
Name: count, dtype: int64
```

[50]:
```
df['MPPT10 DC cumulative production(kWh)'].value_counts()
```

[50]:
```
MPPT10 DC cumulative production(kWh)
0    12158
Name: count, dtype: int64
```

[51]:
```
df['Voltage between PV and earth(V)'].value_counts()
```

[51]:
```
Voltage between PV and earth(V)
0       7657
0.0     4363
-0.1     479
-0.1     441
Name: count, dtype: int64
```

[52]:
```
df.drop(['management domain', 'MPPT7 DC cumulative production(kWh)', 'MPPT8 DC␣
↪cumulative production(kWh)',
       'MPPT9 DC cumulative production(kWh)', 'MPPT10 DC cumulative␣
↪production(kWh)', 'Voltage between PV and earth(V)'
       ], axis=1, inplace=True)
```

[53]:
```
# Replacing '59368 Werne Becklohhof 1-7' with "Werne" in the 'Site name' column
```

```
df['Site name'] = df['Site name'].str.replace('59368 Werne Becklohhof 1-7',␣
 ↪'Werne')

df.head()
```

[53]:

	Site name	device name	start time	Inverter status	Mains voltage/L1L2(AB) line voltage in the power grid(V)	L2-L3(BC) voltage in the power grid(V)	L3-L1(CA) voltage in the power grid(V)	L1(A) phase voltage(V)	L2(B)-Phase-Voltage(V)	L3(C)-Phase-Voltage(V)	…
0	Werne	102140090004/6T2199004131	2023-09-01 00:00:00 DST	NaN	NaN	NaN	NaN	0.0	0.0	0.0	…
1	Werne	102140090004/6T2199004131	2023-09-01 06:20:00 DST	NaN	NaN	NaN	NaN	0.0	0.0	0.0	…
2	Werne	102140090004/6T2199004131	2023-09-01 06:25:00 DST	NaN	410	408.6	409.4	236.7	235.5	237.9	…
3	Werne	102140090004/6T2199004131	2023-09-01 06:30:00 DST	NaN	NaN	NaN	NaN	235.7	235.3	237.9	…
4	Werne	102140090004/6T2199004131	2023-09-01 06:35:00 DST	Standby : Isolierwiderst.-Erkennung	410.4	408.8	409.1	235.0	235.5	237.4	…

```
   MPPT3 DC cumulative production(kWh)  MPPT4 DC cumulative production(kWh)  \
0                              9230.56                              25056.02
1                              9230.56                              25056.02
2                              9230.56                              25056.02
3                              9230.56                              25056.02
4                              9230.56                              25056.02

   MPPT5 DC cumulative production(kWh)  MPPT6 DC cumulative production(kWh)  \
0                             12385.89                              19730.8
1                             12385.89                              19730.8
2                             12385.89                              19730.8
3                             12385.89                              19730.8
4                             12385.89                              19730.8

   Production of this month(kWh)  Production of this year(kWh)  \
0                            NaN                           NaN
1                            NaN                           NaN
2                              0                       49296.99
3                            NaN                           NaN
4                              0                       49296.99

   Inverter start time  Inverter switch-off time  Total yield(kWh)  \
0                  NaN                       NaN         109475.74
1                  NaN                       NaN         109475.74
2                  NaN                       NaN         109475.74
3                  NaN                       NaN         109475.74
4  2023/08/31 06:36:57       2023/08/31 20:22:24         109475.74

   Irradiation (Satellite) Wh/m²
0                            0.0
1                            0.0
2                            0.0
3                            0.0
4                            0.0

[5 rows x 58 columns]
```

[54]:
```
df = pd.DataFrame(df)

# Convert the column to string explicitly and then replace ' DST' with an empty␣
 ↪string
df['start time'] = df['start time'].astype(str).str.replace(' DST', '')

# Convert the updated column to datetime
df['start time'] = pd.to_datetime(df['start time'])

# Display the updated DataFrame
```

```
df.head()
```

[54]:

```
  Site name                device name          start time  \
0     Werne  102140090004/6T2199004131 2023-09-01 00:00:00
1     Werne  102140090004/6T2199004131 2023-09-01 06:20:00
2     Werne  102140090004/6T2199004131 2023-09-01 06:25:00
3     Werne  102140090004/6T2199004131 2023-09-01 06:30:00
4     Werne  102140090004/6T2199004131 2023-09-01 06:35:00

                      Inverter status  \
0                                 NaN
1                                 NaN
2                                 NaN
3                                 NaN
4  Standby : Isolierwiderst.-Erkennung

  Mains voltage/L1L2(AB) line voltage in the power grid(V)  \
0                                                      NaN
1                                                      NaN
2                                                      410
3                                                      NaN
4                                                    410.4

  L2-L3(BC) voltage in the power grid(V)  \
0                                    NaN
1                                    NaN
2                                  408.6
3                                    NaN
4                                  408.8

  L3-L1(CA) voltage in the power grid(V)  L1(A) phase voltage(V)  \
0                                    NaN                     0.0
1                                    NaN                     0.0
2                                  409.4                   236.7
3                                    NaN                   235.7
4                                  409.1                   235.0

   L2(B)-Phase-Voltage(V)  L3(C)-Phase-Voltage(V)  …  \
0                     0.0                     0.0  …
1                     0.0                     0.0  …
2                   235.5                   237.9  …
3                   235.3                   237.9  …
4                   235.5                   237.4  …

   MPPT3 DC cumulative production(kWh)  MPPT4 DC cumulative production(kWh)  \
0                              9230.56                              25056.02
1                              9230.56                              25056.02
```

```
2                                9230.56                                25056.02
3                                9230.56                                25056.02
4                                9230.56                                25056.02

   MPPT5 DC cumulative production(kWh)  MPPT6 DC cumulative production(kWh)  \
0                              12385.89                                19730.8
1                              12385.89                                19730.8
2                              12385.89                                19730.8
3                              12385.89                                19730.8
4                              12385.89                                19730.8

   Production of this month(kWh)  Production of this year(kWh)  \
0                            NaN                           NaN
1                            NaN                           NaN
2                              0                       49296.99
3                            NaN                           NaN
4                              0                       49296.99

   Inverter start time  Inverter switch-off time  Total yield(kWh)  \
0                  NaN                       NaN         109475.74
1                  NaN                       NaN         109475.74
2                  NaN                       NaN         109475.74
3                  NaN                       NaN         109475.74
4  2023/08/31 06:36:57       2023/08/31 20:22:24         109475.74

   Irradiation (Satellite) Wh/m²
0                            0.0
1                            0.0
2                            0.0
3                            0.0
4                            0.0

[5 rows x 58 columns]
```

[55]:
```
# Convert 'Inverter start time' column to datetime
df['Inverter start time'] = pd.to_datetime(df['Inverter start time'])

# Display the updated DataFrame
print(df)
```

```
      Site name                device name          start time  \
0         Werne  102140090004/6T2199004131 2023-09-01 00:00:00
1         Werne  102140090004/6T2199004131 2023-09-01 06:20:00
2         Werne  102140090004/6T2199004131 2023-09-01 06:25:00
3         Werne  102140090004/6T2199004131 2023-09-01 06:30:00
4         Werne  102140090004/6T2199004131 2023-09-01 06:35:00
…         …                         …                   …
```

```
12991     Werne  102140090004/6T2199004131 2023-11-30 16:50:00
12992     Werne  102140090004/6T2199004131 2023-11-30 16:55:00
12993     Werne  102140090004/6T2199004131 2023-11-30 17:00:00
12994     Werne  102140090004/6T2199004131 2023-11-30 17:05:00
12995     Werne  102140090004/6T2199004131 2023-11-30 17:10:00

                          Inverter status  \
0                                     NaN
1                                     NaN
2                                     NaN
3                                     NaN
4      Standby : Isolierwiderst.-Erkennung
…                                     …
12991            Standby : Initialisierung
12992            Standby : Initialisierung
12993                                 NaN
12994                                 NaN
12995                                 NaN

      Mains voltage/L1L2(AB) line voltage in the power grid(V)  \
0                                                       NaN
1                                                       NaN
2                                                       410
3                                                       NaN
4                                                     410.4
…                                                     …
12991                                                 404.7
12992                                                 404.7
12993                                                   NaN
12994                                                   NaN
12995                                                   NaN

      L2-L3(BC) voltage in the power grid(V)  \
0                                        NaN
1                                        NaN
2                                      408.6
3                                        NaN
4                                      408.8
…                                      …
12991                                  405.8
12992                                  405.8
12993                                    NaN
12994                                    NaN
12995                                    NaN

      L3-L1(CA) voltage in the power grid(V)  L1(A) phase voltage(V)  \
0                                        NaN                     0.0
1                                        NaN                     0.0
```

```
2                                    409.4                  236.7
3                                      NaN                  235.7
4                                    409.1                  235.0
…                                        …                      …
12991                                405.1                  231.5
12992                                405.1                  231.5
12993                                  NaN                  231.5
12994                                  NaN                  231.5
12995                                  NaN                    0.0

       L2(B)-Phase-Voltage(V)  L3(C)-Phase-Voltage(V)  …  \
0                         0.0                     0.0  …
1                         0.0                     0.0  …
2                       235.5                   237.9  …
3                       235.3                   237.9  …
4                       235.5                   237.4  …
…                           …                       …  …
12991                   233.8                   235.9  …
12992                   233.8                   235.9  …
12993                   233.8                   235.9  …
12994                   233.8                   235.9  …
12995                     0.0                     0.0  …

       MPPT3 DC cumulative production(kWh)  \
0                                  9230.56
1                                  9230.56
2                                  9230.56
3                                  9230.56
4                                  9230.56
…                                      …
12991                             10003.35
12992                             10003.35
12993                             10003.35
12994                             10003.35
12995                             10003.35

       MPPT4 DC cumulative production(kWh)  \
0                                 25056.02
1                                 25056.02
2                                 25056.02
3                                 25056.02
4                                 25056.02
…                                      …
12991                             27418.22
12992                             27418.22
12993                             27418.22
12994                             27418.22
12995                             27418.22
```

```
       MPPT5 DC cumulative production(kWh)  \
0                                 12385.89
1                                 12385.89
2                                 12385.89
3                                 12385.89
4                                 12385.89
…                                    …
12991                             13560.67
12992                             13560.67
12993                             13560.67
12994                             13560.67
12995                             13560.67

       MPPT6 DC cumulative production(kWh)  Production of this month(kWh)  \
0                                 19730.80                            NaN
1                                 19730.80                            NaN
2                                 19730.80                              0
3                                 19730.80                            NaN
4                                 19730.80                              0
…                                    …                              …
12991                             21609.82                         1287.98
12992                             21609.82                         1287.98
12993                             21609.82                            NaN
12994                             21609.82                            NaN
12995                             21609.82                            NaN

       Production of this year(kWh)  Inverter start time  \
0                               NaN                  NaT
1                               NaN                  NaT
2                          49296.99                  NaT
3                               NaN                  NaT
4                          49296.99  2023-08-31 06:36:57
…                             …                   …
12991                      59805.45  2023-11-30 08:16:26
12992                      59805.45  2023-11-30 08:16:26
12993                           NaN                  NaT
12994                           NaN                  NaT
12995                           NaN                  NaT

       Inverter switch-off time  Total yield(kWh)  \
0                           NaN         109475.74
1                           NaN         109475.74
2                           NaN         109475.74
3                           NaN         109475.74
4           2023/08/31 20:22:24         109475.74
…                          …              …
12991       2023/11/30 16:35:54         120230.01
```

```
12992       2023/11/30 16:35:54         120230.01
12993                       NaN         120230.01
12994                       NaN         120230.01
12995                       NaN         120230.01

       Irradiation (Satellite) Wh/m²
0                              0.0
1                              0.0
2                              0.0
3                              0.0
4                              0.0
…                              …
12991                          0.0
12992                          0.0
12993                          0.0
12994                          0.0
12995                          0.0

[12996 rows x 58 columns]
```

[56]:
```
# Convert 'Inverter start time' column to datetime
df['Inverter switch-off time'] = pd.to_datetime(df['Inverter switch-off time'])

# Display the updated DataFrame
print(df)
```

```
      Site name                 device name          start time  \
0         Werne  102140090004/6T2199004131 2023-09-01 00:00:00
1         Werne  102140090004/6T2199004131 2023-09-01 06:20:00
2         Werne  102140090004/6T2199004131 2023-09-01 06:25:00
3         Werne  102140090004/6T2199004131 2023-09-01 06:30:00
4         Werne  102140090004/6T2199004131 2023-09-01 06:35:00
…         …                         …                 …
12991     Werne  102140090004/6T2199004131 2023-11-30 16:50:00
12992     Werne  102140090004/6T2199004131 2023-11-30 16:55:00
12993     Werne  102140090004/6T2199004131 2023-11-30 17:00:00
12994     Werne  102140090004/6T2199004131 2023-11-30 17:05:00
12995     Werne  102140090004/6T2199004131 2023-11-30 17:10:00

                          Inverter status  \
0                                     NaN
1                                     NaN
2                                     NaN
3                                     NaN
4      Standby : Isolierwiderst.-Erkennung
…                                     …
12991            Standby : Initialisierung
12992            Standby : Initialisierung
```

```
12993                                  NaN
12994                                  NaN
12995                                  NaN

      Mains voltage/L1L2(AB) line voltage in the power grid(V)  \
0                                                       NaN
1                                                       NaN
2                                                       410
3                                                       NaN
4                                                     410.4
…                                                     …
12991                                                 404.7
12992                                                 404.7
12993                                                   NaN
12994                                                   NaN
12995                                                   NaN

      L2-L3(BC) voltage in the power grid(V)  \
0                                        NaN
1                                        NaN
2                                      408.6
3                                        NaN
4                                      408.8
…                                      …
12991                                  405.8
12992                                  405.8
12993                                    NaN
12994                                    NaN
12995                                    NaN

      L3-L1(CA) voltage in the power grid(V)  L1(A) phase voltage(V)  \
0                                        NaN                     0.0
1                                        NaN                     0.0
2                                      409.4                   236.7
3                                        NaN                   235.7
4                                      409.1                   235.0
…                                      …                     …
12991                                  405.1                   231.5
12992                                  405.1                   231.5
12993                                    NaN                   231.5
12994                                    NaN                   231.5
12995                                    NaN                     0.0

       L2(B)-Phase-Voltage(V)  L3(C)-Phase-Voltage(V)  …  \
0                         0.0                     0.0  …
1                         0.0                     0.0  …
2                       235.5                   237.9  …
3                       235.3                   237.9  …
```

```
4                      235.5                    237.4  …
…                      …                      …  …
12991                  233.8                    235.9  …
12992                  233.8                    235.9  …
12993                  233.8                    235.9  …
12994                  233.8                    235.9  …
12995                    0.0                      0.0  …

       MPPT3 DC cumulative production(kWh)  \
0                                   9230.56
1                                   9230.56
2                                   9230.56
3                                   9230.56
4                                   9230.56
…                                     …
12991                              10003.35
12992                              10003.35
12993                              10003.35
12994                              10003.35
12995                              10003.35

       MPPT4 DC cumulative production(kWh)  \
0                                  25056.02
1                                  25056.02
2                                  25056.02
3                                  25056.02
4                                  25056.02
…                                     …
12991                              27418.22
12992                              27418.22
12993                              27418.22
12994                              27418.22
12995                              27418.22

       MPPT5 DC cumulative production(kWh)  \
0                                  12385.89
1                                  12385.89
2                                  12385.89
3                                  12385.89
4                                  12385.89
…                                     …
12991                              13560.67
12992                              13560.67
12993                              13560.67
12994                              13560.67
12995                              13560.67

       MPPT6 DC cumulative production(kWh)  Production of this month(kWh)  \
```

```
0                                19730.80                              NaN
1                                19730.80                              NaN
2                                19730.80                                0
3                                19730.80                              NaN
4                                19730.80                                0
…                                   …                                  …
12991                            21609.82                           1287.98
12992                            21609.82                           1287.98
12993                            21609.82                              NaN
12994                            21609.82                              NaN
12995                            21609.82                              NaN

       Production of this year(kWh)  Inverter start time  \
0                               NaN                  NaT
1                               NaN                  NaT
2                          49296.99                  NaT
3                               NaN                  NaT
4                          49296.99  2023-08-31 06:36:57
…                             …                    …
12991                      59805.45  2023-11-30 08:16:26
12992                      59805.45  2023-11-30 08:16:26
12993                           NaN                  NaT
12994                           NaN                  NaT
12995                           NaN                  NaT

       Inverter switch-off time  Total yield(kWh)  \
0                           NaT         109475.74
1                           NaT         109475.74
2                           NaT         109475.74
3                           NaT         109475.74
4           2023-08-31 20:22:24         109475.74
…                          …              …
12991       2023-11-30 16:35:54         120230.01
12992       2023-11-30 16:35:54         120230.01
12993                       NaT         120230.01
12994                       NaT         120230.01
12995                       NaT         120230.01

       Irradiation (Satellite) Wh/m²
0                                0.0
1                                0.0
2                                0.0
3                                0.0
4                                0.0
…                               …
12991                            0.0
12992                            0.0
12993                            0.0
```

```
12994                              0.0
12995                              0.0

[12996 rows x 58 columns]
```

We can see that the system has built-in sunlight detection sensors and activates when sunlight appears. To prepare data for further process. I'm cutting out this data.

[57]:
```
df['Inverter status'].value_counts()
```

[57]:
```
Inverter status
Stromnetzplanung: Q-U-Kennlinie        11655
Standby : Sonnenlichterkennung           261
Standby : Initialisierung                152
Standby : kein Sonnenlicht                60
Standby : Isolierwiderst.-Erkennung       40
Wird gestartet                            10
AUS : unerwartetes AUS                     5
Name: count, dtype: int64
```

[58]:
```
df.iloc[170:185]
```

[58]:
```
    Site name                 device name          start time  \
170     Werne  102140090004/6T2199004131 2023-09-01 20:25:00
171     Werne  102140090004/6T2199004131 2023-09-01 20:30:00
172     Werne  102140090004/6T2199004131 2023-09-01 20:35:00
173     Werne  102140090004/6T2199004131 2023-09-01 20:40:00
174     Werne  102140090004/6T2199004131 2023-09-01 20:45:00
175     Werne  102140090004/6T2199004131 2023-09-01 20:50:00
176     Werne  102140090004/6T2199004131 2023-09-01 20:55:00
177     Werne  102140090004/6T2199004131 2023-09-02 00:00:00
178     Werne  102140090004/6T2199004131 2023-09-02 06:25:00
179     Werne  102140090004/6T2199004131 2023-09-02 06:30:00
180     Werne  102140090004/6T2199004131 2023-09-02 06:35:00
181     Werne  102140090004/6T2199004131 2023-09-02 06:40:00
182     Werne  102140090004/6T2199004131 2023-09-02 06:45:00
183     Werne  102140090004/6T2199004131 2023-09-02 06:50:00
184     Werne  102140090004/6T2199004131 2023-09-02 06:55:00

                         Inverter status  \
170       Standby : Sonnenlichterkennung
171       Standby : Sonnenlichterkennung
172            Standby : Initialisierung
173            Standby : Initialisierung
174                                  NaN
175                                  NaN
176                                  NaN
177                                  NaN
```

```
178                                  NaN
179                                  NaN
180                                  NaN
181                                  NaN
182  Standby : Isolierwiderst.-Erkennung
183      Stromnetzplanung: Q-U-Kennlinie
184      Stromnetzplanung: Q-U-Kennlinie

    Mains voltage/L1L2(AB) line voltage in the power grid(V)  \
170                                                  413.8
171                                                    413
172                                                  414.1
173                                                  414.1
174                                                    NaN
175                                                    NaN
176                                                    NaN
177                                                    NaN
178                                                    NaN
179                                                    412
180                                                    NaN
181                                                    NaN
182                                                  413.1
183                                                  411.3
184                                                  410.5

    L2-L3(BC) voltage in the power grid(V)  \
170                                   413.6
171                                   414.1
172                                   413.1
173                                   413.1
174                                     NaN
175                                     NaN
176                                     NaN
177                                     NaN
178                                     NaN
179                                   411.8
180                                     NaN
181                                     NaN
182                                   411.7
183                                   411.7
184                                   412.1

    L3-L1(CA) voltage in the power grid(V)  L1(A) phase voltage(V)  \
170                                   414.5                   238.0
171                                   414.6                   236.4
172                                   414.6                   237.4
173                                   414.6                   237.4
```

```
174                                  NaN                   237.4
175                                  NaN                   237.4
176                                  NaN                     0.0
177                                  NaN                     0.0
178                                  NaN                     0.0
179                                411.7                   237.8
180                                  NaN                   237.4
181                                  NaN                   237.4
182                                411.5                   237.4
183                                410.4                   234.8
184                                410.2                   233.7

     L2(B)-Phase-Voltage(V)  L3(C)-Phase-Voltage(V)  …  \
170                   237.4                   241.3  …
171                   238.4                   241.3  …
172                   237.9                   241.3  …
173                   237.9                   241.3  …
174                   237.9                   241.3  …
175                   237.9                   241.3  …
176                     0.0                     0.0  …
177                     0.0                     0.0  …
178                     0.0                     0.0  …
179                   236.7                   240.5  …
180                   237.3                   240.6  …
181                   237.1                   239.4  …
182                   237.2                   240.4  …
183                   237.2                   239.2  …
184                   237.0                   239.4  …

     MPPT3 DC cumulative production(kWh)  MPPT4 DC cumulative production(kWh)  \
170                              9241.04                              25086.11
171                              9241.04                              25086.11
172                              9241.04                              25086.11
173                              9241.04                              25086.11
174                              9241.04                              25086.11
175                              9241.04                              25086.11
176                              9241.04                              25086.11
177                              9241.04                              25086.11
178                              9241.04                              25086.11
179                              9241.04                              25086.11
180                              9241.04                              25086.11
181                              9241.04                              25086.11
182                              9241.04                              25086.11
183                              9241.04                              25086.11
184                              9241.04                              25086.12

     MPPT5 DC cumulative production(kWh)  MPPT6 DC cumulative production(kWh)  \
```

170	12401.35	19753.88
171	12401.35	19753.88
172	12401.35	19753.88
173	12401.35	19753.88
174	12401.35	19753.88
175	12401.35	19753.88
176	12401.35	19753.88
177	12401.35	19753.88
178	12401.35	19753.88
179	12401.35	19753.88
180	12401.35	19753.88
181	12401.35	19753.88
182	12401.35	19753.88
183	12401.35	19753.89
184	12401.35	19753.89

	Production of this month(kWh)	Production of this year(kWh)	\
170	128.03	49425.02	
171	128.03	49425.02	
172	128.03	49425.02	
173	128.03	49425.02	
174	NaN	NaN	
175	NaN	NaN	
176	NaN	NaN	
177	NaN	NaN	
178	NaN	NaN	
179	128.03	49425.02	
180	NaN	NaN	
181	NaN	NaN	
182	128.03	49425.02	
183	128.04	49425.03	
184	128.07	49425.06	

	Inverter start time	Inverter switch-off time	Total yield(kWh)	\
170	2023-09-01 06:36:56	2023-09-01 20:20:24	109606.67	
171	2023-09-01 06:36:56	2023-09-01 20:20:24	109606.67	
172	2023-09-01 06:36:56	2023-09-01 20:20:24	109606.67	
173	2023-09-01 06:36:56	2023-09-01 20:20:24	109606.67	
174	NaT	NaT	109606.67	
175	NaT	NaT	109606.67	
176	NaT	NaT	109606.67	
177	NaT	NaT	109606.67	
178	NaT	NaT	109606.67	
179	NaT	NaT	109606.67	
180	NaT	NaT	109606.67	
181	NaT	NaT	109606.67	
182	2023-09-01 06:36:56	2023-09-01 20:20:24	109606.67	

```
183  2023-09-02 06:46:27       2023-09-01 20:20:24         109606.69
184  2023-09-02 06:46:27       2023-09-01 20:20:24         109606.71

     Irradiation (Satellite) Wh/m²
170                            0.0
171                            0.0
172                            0.0
173                            0.0
174                            0.0
175                            0.0
176                            0.0
177                            0.0
178                            0.0
179                            0.0
180                            0.0
181                            0.0
182                            0.0
183                            0.0
184                            0.0

[15 rows x 58 columns]
```

[59]:
```
# Remove rows where 'Inverter status' column is blank
df = df.dropna(subset=['Inverter status'])

# Resetting the index
df = df.reset_index(drop=True)

df.head()
```

[59]:
```
  Site name                device name          start time  \
0     Werne  102140090004/6T2199004131 2023-09-01 06:35:00
1     Werne  102140090004/6T2199004131 2023-09-01 06:40:00
2     Werne  102140090004/6T2199004131 2023-09-01 06:45:00
3     Werne  102140090004/6T2199004131 2023-09-01 06:50:00
4     Werne  102140090004/6T2199004131 2023-09-01 06:55:00

                       Inverter status  \
0  Standby : Isolierwiderst.-Erkennung
1      Stromnetzplanung: Q-U-Kennlinie
2      Stromnetzplanung: Q-U-Kennlinie
3      Stromnetzplanung: Q-U-Kennlinie
4      Stromnetzplanung: Q-U-Kennlinie

  Mains voltage/L1L2(AB) line voltage in the power grid(V)  \
0                                              410.4
1                                              409.2
```

```
2                                          412.4
3                                          410.6
4                                          409.9

  L2-L3(BC) voltage in the power grid(V)  \
0                                  408.8
1                                  406.8
2                                  410.2
3                                  410.5
4                                  409.5

  L3-L1(CA) voltage in the power grid(V)  L1(A) phase voltage(V)  \
0                                  409.1                   235.0
1                                    408                   236.6
2                                  411.4                   237.0
3                                  411.3                   236.3
4                                  408.2                   237.1

   L2(B)-Phase-Voltage(V)  L3(C)-Phase-Voltage(V)  …  \
0                   235.5                   237.4  …
1                   232.4                   238.4  …
2                   235.3                   238.4  …
3                   234.9                   239.4  …
4                   235.9                   235.8  …

   MPPT3 DC cumulative production(kWh)  MPPT4 DC cumulative production(kWh)  \
0                              9230.56                              25056.02
1                              9230.56                              25056.02
2                              9230.56                              25056.02
3                              9230.56                              25056.03
4                              9230.58                              25056.04

   MPPT5 DC cumulative production(kWh)  MPPT6 DC cumulative production(kWh)  \
0                             12385.89                             19730.80
1                             12385.89                             19730.80
2                             12385.89                             19730.81
3                             12385.90                             19730.81
4                             12385.91                             19730.83

   Production of this month(kWh)  Production of this year(kWh)  \
0                              0                       49296.99
1                           0.01                          49297
2                           0.03                       49297.02
3                           0.05                       49297.04
4                            0.1                       49297.09

   Inverter start time  Inverter switch-off time  Total yield(kWh)  \
```

```
0  2023-08-31 06:36:57       2023-08-31 20:22:24         109475.74
1  2023-09-01 06:36:56       2023-08-31 20:22:24         109475.74
2  2023-09-01 06:36:56       2023-08-31 20:22:24         109475.76
3  2023-09-01 06:36:56       2023-08-31 20:22:24         109475.80
4  2023-09-01 06:36:56       2023-08-31 20:22:24         109475.87

   Irradiation (Satellite) Wh/m²
0                           0.0
1                           0.0
2                           0.0
3                           0.0
4                           0.0

[5 rows x 58 columns]
```

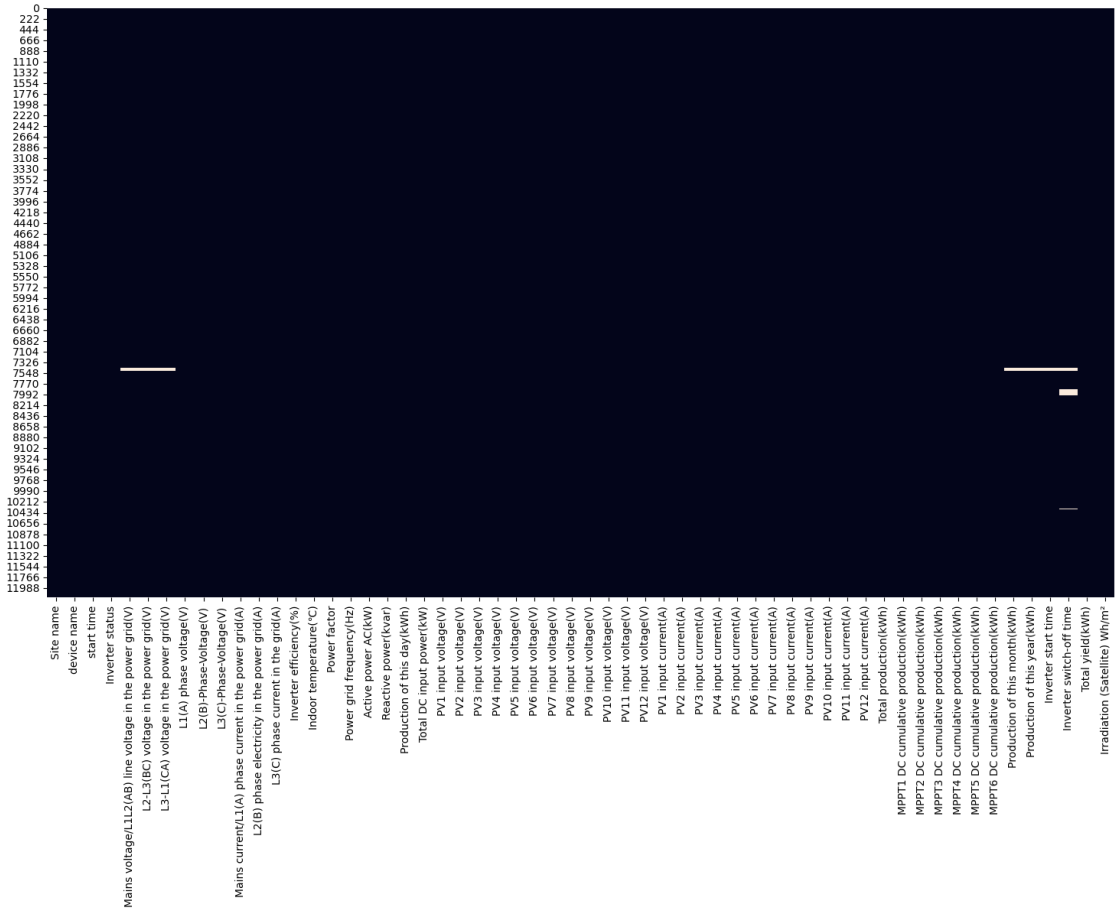
[60]:
```
# To check for any missing values
df.isna().sum()
```

[60]:
```
Site name                                                    0
device name                                                  0
start time                                                   0
Inverter status                                              0
Mains voltage/L1L2(AB) line voltage in the power grid(V)    63
L2-L3(BC) voltage in the power grid(V)                      63
L3-L1(CA) voltage in the power grid(V)                      63
L1(A) phase voltage(V)                                       0
L2(B)-Phase-Voltage(V)                                       0
L3(C)-Phase-Voltage(V)                                       0
Mains current/L1(A) phase current in the power grid(A)       0
L2(B) phase electricity in the power grid(A)                 0
L3(C) phase current in the grid(A)                           0
Inverter efficiency(%)                                       0
Indoor temperature(℃)                                        0
Power factor                                                 0
Power grid frequency(Hz)                                     0
Active power AC(kW)                                          0
Reactive power(kvar)                                         0
Production of this day(kWh)                                  0
Total DC input power(kW)                                     0
PV1 input voltage(V)                                         0
PV2 input voltage(V)                                         0
PV3 input voltage(V)                                         0
PV4 input voltage(V)                                         0
PV5 input voltage(V)                                         0
PV6 input voltage(V)                                         0
PV7 input voltage(V)                                         0
PV8 input voltage(V)                                         0
```

```
PV9 input voltage(V)                                0
PV10 input voltage(V)                               0
PV11 input voltage(V)                               0
PV12 input voltage(V)                               0
PV1 input current(A)                                0
PV2 input current(A)                                0
PV3 input current(A)                                0
PV4 input current(A)                                0
PV5 input current(A)                                0
PV6 input current(A)                                0
PV7 input current(A)                                0
PV8 input current(A)                                0
PV9 input current(A)                                0
PV10 input current(A)                               0
PV11 input current(A)                               0
PV12 input current(A)                               0
Total production(kWh)                               0
MPPT1 DC cumulative production(kWh)                 0
MPPT2 DC cumulative production(kWh)                 0
MPPT3 DC cumulative production(kWh)                 0
MPPT4 DC cumulative production(kWh)                 0
MPPT5 DC cumulative production(kWh)                 0
MPPT6 DC cumulative production(kWh)                 0
Production of this month(kWh)                      63
Production of this year(kWh)                       63
Inverter start time                                56
Inverter switch-off time                          203
Total yield(kWh)                                    0
Irradiation (Satellite) Wh/m²                       0
dtype: int64
```

Check for missing values for each column -> returns the sum of missing values in a given column

[61]:
```
# Plotting Packages
import matplotlib.pyplot as plt
import seaborn as sns

# Missing visualization
plt.figure(figsize=(18,10))
sns.heatmap(df.isna(), cbar=False)
plt.show()
```



[62]:
```
# Convert the column to numeric (if necessary)
df['Mains voltage/L1L2(AB) line voltage in the power grid(V)'] = pd.
 ↪to_numeric(df['Mains voltage/L1L2(AB) line voltage in the power grid(V)'],␣
 ↪errors='coerce')
df['L2-L3(BC) voltage in the power grid(V)'] = pd.to_numeric(df['L2-L3(BC)␣
 ↪voltage in the power grid(V)'], errors='coerce')
df['L3-L1(CA) voltage in the power grid(V)'] = pd.to_numeric(df['L3-L1(CA)␣
 ↪voltage in the power grid(V)'], errors='coerce')
df['Production of this year(kWh)'] = pd.to_numeric(df['Production of this␣
 ↪year(kWh)'], errors='coerce')
df['Production of this month(kWh)'] = pd.to_numeric(df['Production of this␣
 ↪month(kWh)'], errors='coerce')

# Interpolate missing values in the specified column (linear interpolation)
df['Mains voltage/L1L2(AB) line voltage in the power grid(V)'] = df['Mains␣
 ↪voltage/L1L2(AB) line voltage in the power grid(V)'].
 ↪interpolate(method='linear')
```

```
df['L2-L3(BC) voltage in the power grid(V)'] = df['L2-L3(BC) voltage in the␣
 ↪power grid(V)'].interpolate(method='linear')
df['L3-L1(CA) voltage in the power grid(V)'] = df['L3-L1(CA) voltage in the␣
 ↪power grid(V)'].interpolate(method='linear')
df['Production of this year(kWh)'] = df['Production of this year(kWh)'].
 ↪interpolate(method='linear')
df['Production of this month(kWh)'] = df['Production of this month(kWh)'].
 ↪interpolate(method='linear')
```

Forward/Backward Fill:

Fill the missing values with the last known value (backward fill) or the next known value (forward fill). This method assumes the missing values should be the same as the previous or subsequent recorded values.

I perform forward fill and backward fill to fill in missing values:

[63]:
```
# Fill missing values using forward fill and backward fill
df['Inverter start time'].fillna(method='ffill', inplace=True)  # Forward fill
# Fill missing values using forward fill
df['Inverter switch-off time'] = df['Inverter switch-off time'].ffill()
```

```
/tmp/ipykernel_191436/1836685099.py:2: FutureWarning: Series.fillna with
'method' is deprecated and will raise in a future version. Use obj.ffill() or
obj.bfill() instead.
  df['Inverter start time'].fillna(method='ffill', inplace=True)  # Forward fill
```

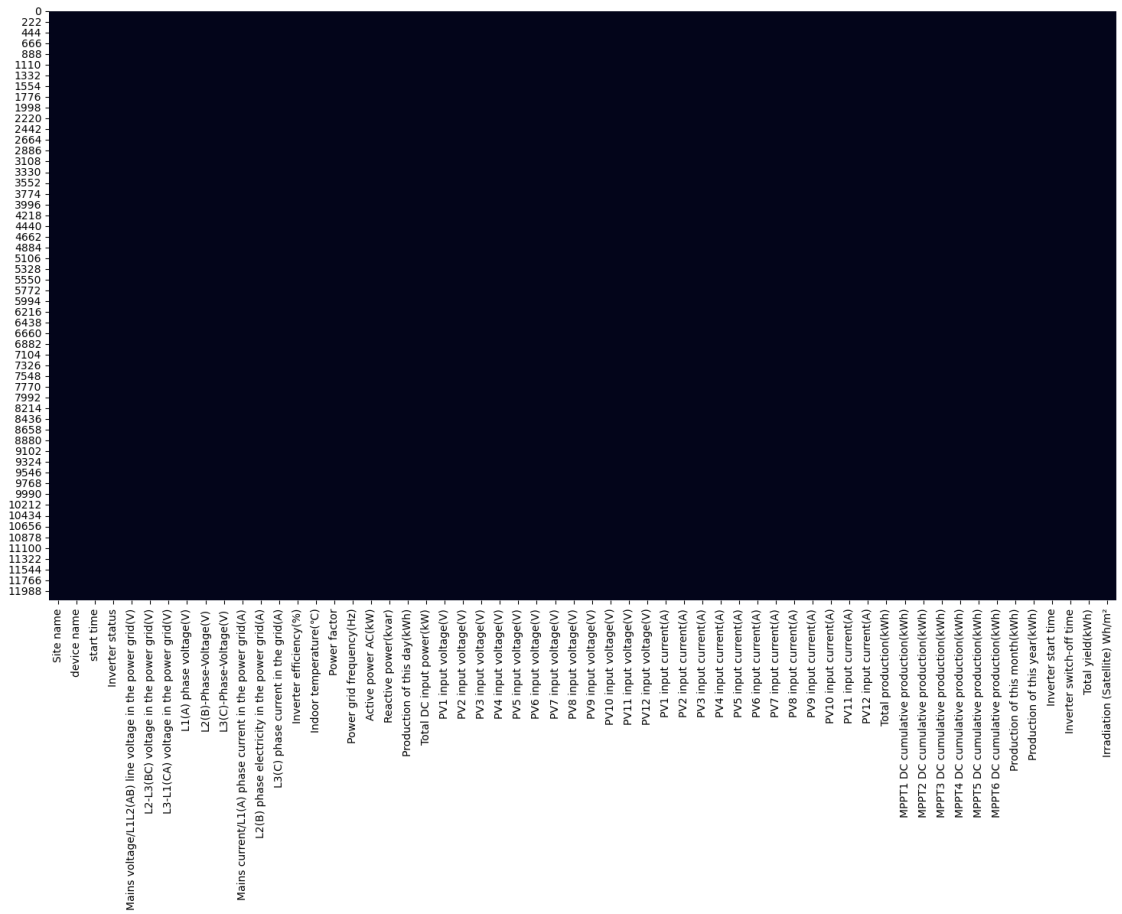
[64]:
```
# To check for any missing values
df.isna().sum()
```

[64]:
```
Site name                                                    0
device name                                                  0
start time                                                   0
Inverter status                                              0
Mains voltage/L1L2(AB) line voltage in the power grid(V)     0
L2-L3(BC) voltage in the power grid(V)                       0
L3-L1(CA) voltage in the power grid(V)                       0
L1(A) phase voltage(V)                                       0
L2(B)-Phase-Voltage(V)                                       0
L3(C)-Phase-Voltage(V)                                       0
Mains current/L1(A) phase current in the power grid(A)       0
L2(B) phase electricity in the power grid(A)                 0
L3(C) phase current in the grid(A)                           0
Inverter efficiency(%)                                       0
Indoor temperature(℃)                                        0
Power factor                                                 0
Power grid frequency(Hz)                                     0
Active power AC(kW)                                          0
```

```
Reactive power(kvar)                                0
Production of this day(kWh)                         0
Total DC input power(kW)                            0
PV1 input voltage(V)                                0
PV2 input voltage(V)                                0
PV3 input voltage(V)                                0
PV4 input voltage(V)                                0
PV5 input voltage(V)                                0
PV6 input voltage(V)                                0
PV7 input voltage(V)                                0
PV8 input voltage(V)                                0
PV9 input voltage(V)                                0
PV10 input voltage(V)                               0
PV11 input voltage(V)                               0
PV12 input voltage(V)                               0
PV1 input current(A)                                0
PV2 input current(A)                                0
PV3 input current(A)                                0
PV4 input current(A)                                0
PV5 input current(A)                                0
PV6 input current(A)                                0
PV7 input current(A)                                0
PV8 input current(A)                                0
PV9 input current(A)                                0
PV10 input current(A)                               0
PV11 input current(A)                               0
PV12 input current(A)                               0
Total production(kWh)                               0
MPPT1 DC cumulative production(kWh)                 0
MPPT2 DC cumulative production(kWh)                 0
MPPT3 DC cumulative production(kWh)                 0
MPPT4 DC cumulative production(kWh)                 0
MPPT5 DC cumulative production(kWh)                 0
MPPT6 DC cumulative production(kWh)                 0
Production of this month(kWh)                       0
Production of this year(kWh)                        0
Inverter start time                                 0
Inverter switch-off time                            0
Total yield(kWh)                                    0
Irradiation (Satellite) Wh/m²                       0
dtype: int64
```

[65]:
```
# Plotting Packages
import matplotlib.pyplot as plt
import seaborn as sns

# Missing visualization
```

```
plt.figure(figsize=(18,10))
sns.heatmap(df.isna(), cbar=False)
plt.show()
```



[66]:
```
# Displaying all column names
print(df.columns)
```

```
Index(['Site name', 'device name', 'start time', 'Inverter status',
       'Mains voltage/L1L2(AB) line voltage in the power grid(V)',
       'L2-L3(BC) voltage in the power grid(V)',
       'L3-L1(CA) voltage in the power grid(V)', 'L1(A) phase voltage(V)',
       'L2(B)-Phase-Voltage(V)', 'L3(C)-Phase-Voltage(V)',
       'Mains current/L1(A) phase current in the power grid(A)',
       'L2(B) phase electricity in the power grid(A)',
       'L3(C) phase current in the grid(A)', 'Inverter efficiency(%)',
       'Indoor temperature(℃)', 'Power factor', 'Power grid frequency(Hz)',
       'Active power AC(kW)', 'Reactive power(kvar)',
       'Production of this day(kWh)', 'Total DC input power(kW)',
       'PV1 input voltage(V)', 'PV2 input voltage(V)', 'PV3 input voltage(V)',
```

```
       'PV4 input voltage(V)', 'PV5 input voltage(V)', 'PV6 input voltage(V)',
       'PV7 input voltage(V)', 'PV8 input voltage(V)', 'PV9 input voltage(V)',
       'PV10 input voltage(V)', 'PV11 input voltage(V)',
       'PV12 input voltage(V)', 'PV1 input current(A)', 'PV2 input current(A)',
       'PV3 input current(A)', 'PV4 input current(A)', 'PV5 input current(A)',
       'PV6 input current(A)', 'PV7 input current(A)', 'PV8 input current(A)',
       'PV9 input current(A)', 'PV10 input current(A)',
       'PV11 input current(A)', 'PV12 input current(A)',
       'Total production(kWh)', 'MPPT1 DC cumulative production(kWh)',
       'MPPT2 DC cumulative production(kWh)',
       'MPPT3 DC cumulative production(kWh)',
       'MPPT4 DC cumulative production(kWh)',
       'MPPT5 DC cumulative production(kWh)',
       'MPPT6 DC cumulative production(kWh)', 'Production of this month(kWh)',
       'Production of this year(kWh)', 'Inverter start time',
       'Inverter switch-off time', 'Total yield(kWh)',
       'Irradiation (Satellite) Wh/m²'],
      dtype='object')
```

[67]:
```
df.drop(['Inverter status',
       'Mains voltage/L1L2(AB) line voltage in the power grid(V)',
       'L2-L3(BC) voltage in the power grid(V)',
       'L3-L1(CA) voltage in the power grid(V)', 'L1(A) phase voltage(V)',
       'L2(B)-Phase-Voltage(V)', 'L3(C)-Phase-Voltage(V)',
       'Mains current/L1(A) phase current in the power grid(A)',
       'L2(B) phase electricity in the power grid(A)',
       'L3(C) phase current in the grid(A)', 'Inverter efficiency(%)',
       'Indoor temperature(℃)', 'Power factor', 'Power grid frequency(Hz)',␣
 ↪'Reactive power(kvar)',
       'PV1 input voltage(V)', 'PV2 input voltage(V)', 'PV3 input voltage(V)',
       'PV4 input voltage(V)', 'PV5 input voltage(V)', 'PV6 input voltage(V)',
       'PV7 input voltage(V)', 'PV8 input voltage(V)', 'PV9 input voltage(V)',
       'PV10 input voltage(V)', 'PV11 input voltage(V)',
       'PV12 input voltage(V)', 'PV1 input current(A)', 'PV2 input current(A)',
       'PV3 input current(A)', 'PV4 input current(A)', 'PV5 input current(A)',
       'PV6 input current(A)', 'PV7 input current(A)', 'PV8 input current(A)',
       'PV9 input current(A)', 'PV10 input current(A)',
       'PV11 input current(A)', 'PV12 input current(A)',
       'Total production(kWh)', 'MPPT1 DC cumulative production(kWh)',
       'MPPT2 DC cumulative production(kWh)',
       'MPPT3 DC cumulative production(kWh)',
       'MPPT4 DC cumulative production(kWh)',
       'MPPT5 DC cumulative production(kWh)',
       'MPPT6 DC cumulative production(kWh)', 'Production of this month(kWh)',
       'Production of this year(kWh)', 'Inverter start time',
       'Inverter switch-off time', 'Irradiation (Satellite) Wh/m²'], axis=1,␣
 ↪inplace=True)
```

[68]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12183 entries, 0 to 12182
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Site name                     12183 non-null  object
 1   device name                   12183 non-null  object
 2   start time                    12183 non-null  datetime64[ns]
 3   Active power AC(kW)           12183 non-null  float64
 4   Production of this day(kWh)   12183 non-null  float64
 5   Total DC input power(kW)      12183 non-null  float64
 6   Total yield(kWh)              12183 non-null  float64
dtypes: datetime64[ns](1), float64(4), object(2)
memory usage: 666.4+ KB
```

[69]:
```
# Data export to cdv file to check for missing data
df.to_csv("df_144444.csv")
```

5 Plant I (Werne): Solar Power Generation data

Plant contains 1 inverters where inverter is connected with several PV array. Every 5 min, inverter records his data. So, if we want to know how many the plant has produced a power in a hour, we just compute the contribution of 1 inverters.

[70]:
```
Werne1_data = df
```

[71]:
```
# Renaming the column 'old_column_name' to 'new_column_name'
Werne1_data = Werne1_data.rename(columns={'start time': 'DATE_TIME'})
```

[72]:
```
Werne1_data.tail()
```

[72]:

	Site name	device name	DATE_TIME	Active power AC(kW)	Production of this day(kWh)
12178	Werne	102140090004/6T2199004131	2023-11-30 16:35:00	0.0	37.27
12179	Werne	102140090004/6T2199004131	2023-11-30 16:40:00	0.0	37.27
12180	Werne	102140090004/6T2199004131	2023-11-30 16:45:00	0.0	37.27
12181	Werne	102140090004/6T2199004131	2023-11-30 16:50:00	0.0	37.27
12182	Werne	102140090004/6T2199004131	2023-11-30 16:55:00	0.0	37.27

```
       Total DC input power(kW)  Total yield(kWh)
12178                       0.0         120230.00
12179                       0.0         120230.01
12180                       0.0         120230.01
12181                       0.0         120230.01
12182                       0.0         120230.01
```

[73]:
```
#we compute a sum of 22 inverters
Werne1_data = Werne1_data.groupby('DATE_TIME')[['Total DC input␣
 ↪power(kW)','Active power AC(kW)', 'Production of this day(kWh)','Total␣
 ↪yield(kWh)']].agg('sum')
```

[74]:
```
Werne1_data = Werne1_data.reset_index()
```

[75]:
```
Werne1_data.head()
```

[75]:
```
             DATE_TIME  Total DC input power(kW)  Active power AC(kW)  \
0 2023-09-01 06:35:00                     0.000                0.000
1 2023-09-01 06:40:00                     0.214                0.164
2 2023-09-01 06:45:00                     0.319                0.244
3 2023-09-01 06:50:00                     0.499                0.409
4 2023-09-01 06:55:00                     0.751                0.693

   Production of this day(kWh)  Total yield(kWh)
0                         0.00         109475.74
1                         0.01         109475.74
2                         0.03         109475.76
3                         0.05         109475.80
4                         0.10         109475.87
```

Cleaning data

I convert DATE_TIME object type to datetime type. After I separate DATE_TIME to date and time

[76]:
```
Werne1_data['DATE_TIME'] = pd.to_datetime(Werne1_data['DATE_TIME'],␣
 ↪errors='coerce')
```

[77]:
```
Werne1_data['time'] = Werne1_data['DATE_TIME'].dt.time
Werne1_data['date'] = pd.to_datetime(Werne1_data['DATE_TIME'].dt.date)
```

[78]:
```
Werne1_data.shape # our data reduced very well
```

[78]:
```
(12183, 7)
```

[79]:
```
#we check
Werne1_data.head()
```

[79]:
```
             DATE_TIME  Total DC input power(kW)  Active power AC(kW)  \
0 2023-09-01 06:35:00                     0.000                0.000
1 2023-09-01 06:40:00                     0.214                0.164
2 2023-09-01 06:45:00                     0.319                0.244
3 2023-09-01 06:50:00                     0.499                0.409
4 2023-09-01 06:55:00                     0.751                0.693

   Production of this day(kWh)  Total yield(kWh)      time       date
0                         0.00         109475.74  06:35:00 2023-09-01
1                         0.01         109475.74  06:40:00 2023-09-01
2                         0.03         109475.76  06:45:00 2023-09-01
3                         0.05         109475.80  06:50:00 2023-09-01
4                         0.10         109475.87  06:55:00 2023-09-01
```

[80]:
```
Werne1_data.info()
```

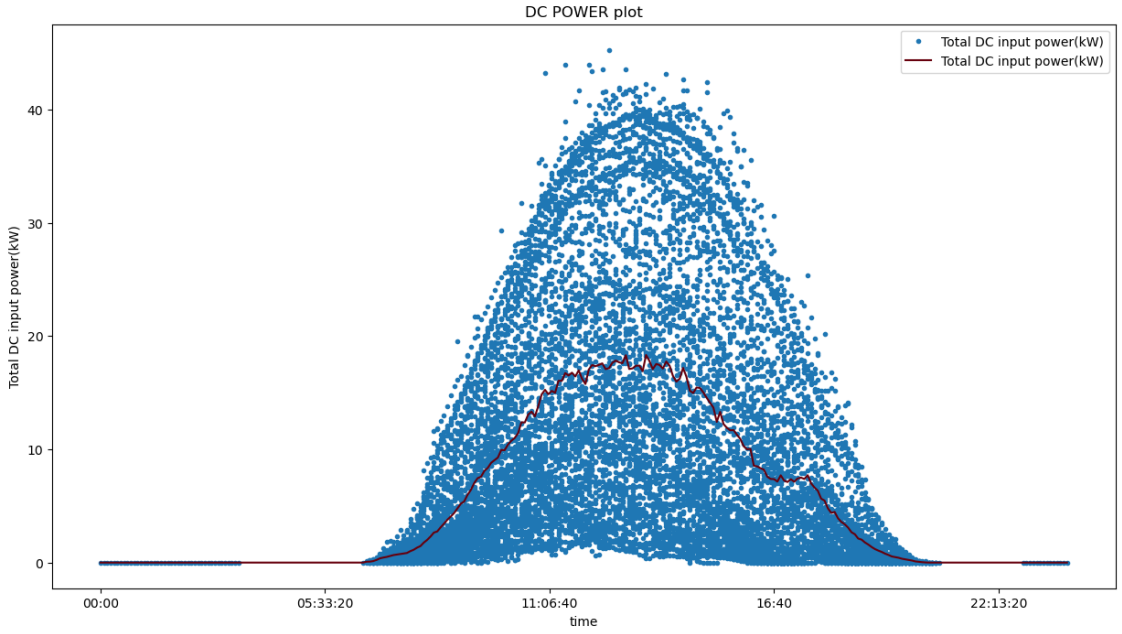
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12183 entries, 0 to 12182
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   DATE_TIME                     12183 non-null  datetime64[ns]
 1   Total DC input power(kW)      12183 non-null  float64
 2   Active power AC(kW)           12183 non-null  float64
 3   Production of this day(kWh)   12183 non-null  float64
 4   Total yield(kWh)              12183 non-null  float64
 5   time                          12183 non-null  object
 6   date                          12183 non-null  datetime64[ns]
dtypes: datetime64[ns](2), float64(4), object(1)
memory usage: 666.4+ KB
```

EDA for DC power, AC power **and** Yield.

Here, i use 1. Line or scatter plot 2. change rate. 3. Box and Whisker plot 4. calendar plot 5. Bar chart.

5.0.1 DC Power

[81]:
```
#plant1_data.iplot(x= 'time', y='DC_POWER', xTitle='Time',  yTitle= 'DC Power',␣
 ↪title='DC POWER plot')
Werne1_data.plot(x= 'time', y='Total DC input power(kW)', style='.', figsize =␣
 ↪(15, 8))
Werne1_data.groupby('time')['Total DC input power(kW)'].agg('mean').
 ↪plot(legend=True, colormap='Reds_r')
plt.ylabel('Total DC input power(kW)')
plt.title('DC POWER plot')
plt.show()
```

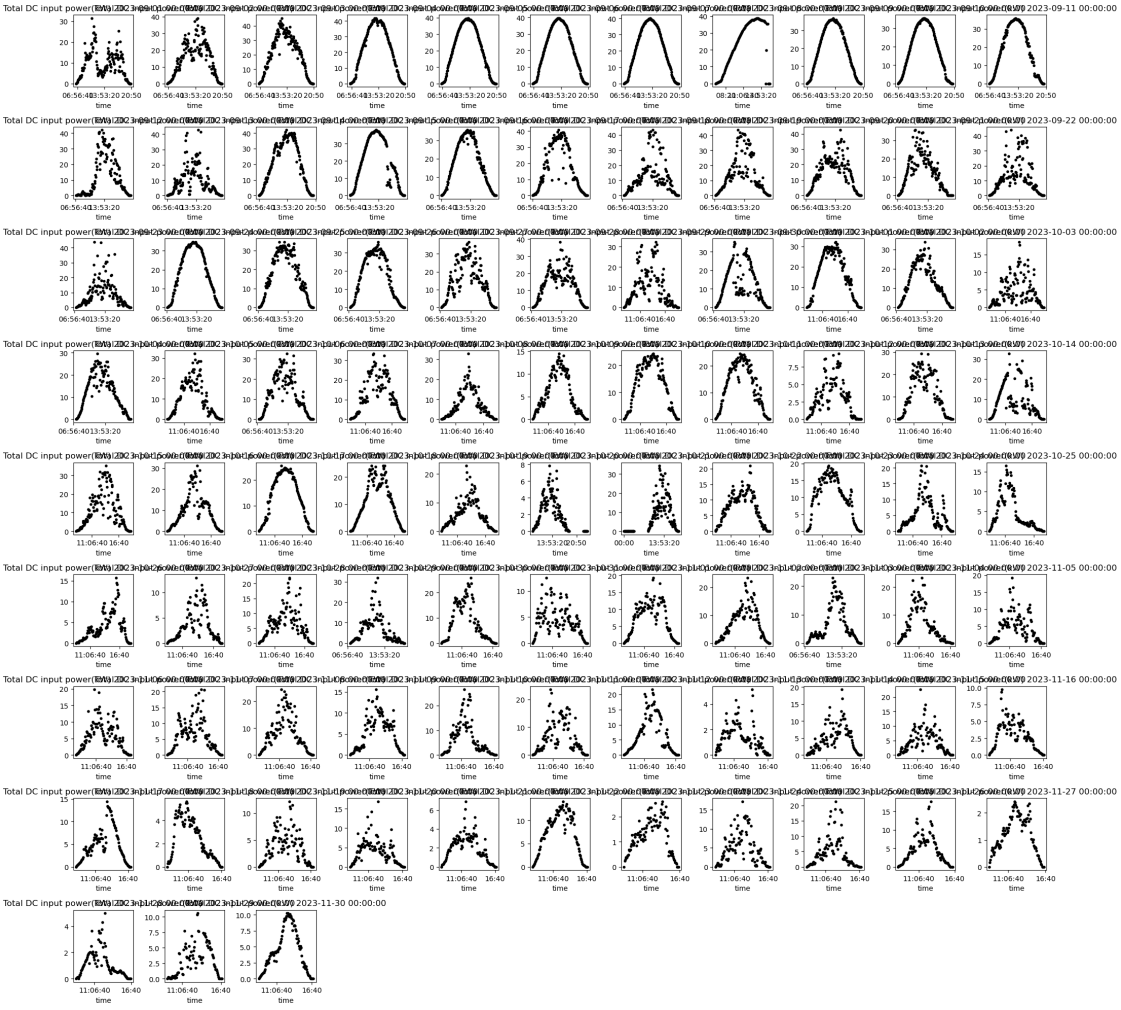


Between 06:00:00 and 18:00:00, the Plant produces a dc power but otherwise there is null. The reason is sunlight.

[82]:
```
# Okay, we are going to see dc power in each day produced by Plant.
# we create calendar_dc data how in each day Plant produce a dc power in each␣
 ↪time.

calendar_dc = Werne1_data.pivot_table(values='Total DC input power(kW)',␣
 ↪index='time', columns='date')
```

[83]:
```
def multi_plot(data=None, row=None, col=None, title='Total DC input power(kW)'):
    cols = data.columns
    total_plots = len(cols)

    # Adjust row and col if they are not provided or insufficient
    if row is None or col is None:
        row = int(total_plots ** 0.5)
        col = int(total_plots / row) + 1 if total_plots % row != 0 else␣
 ↪int(total_plots / row)

    gp = plt.figure(figsize=(20, 20))
    gp.subplots_adjust(wspace=0.2, hspace=0.8)

    for i in range(1, total_plots + 1):
        if i <= row * col:
            ax = gp.add_subplot(row, col, i)
```

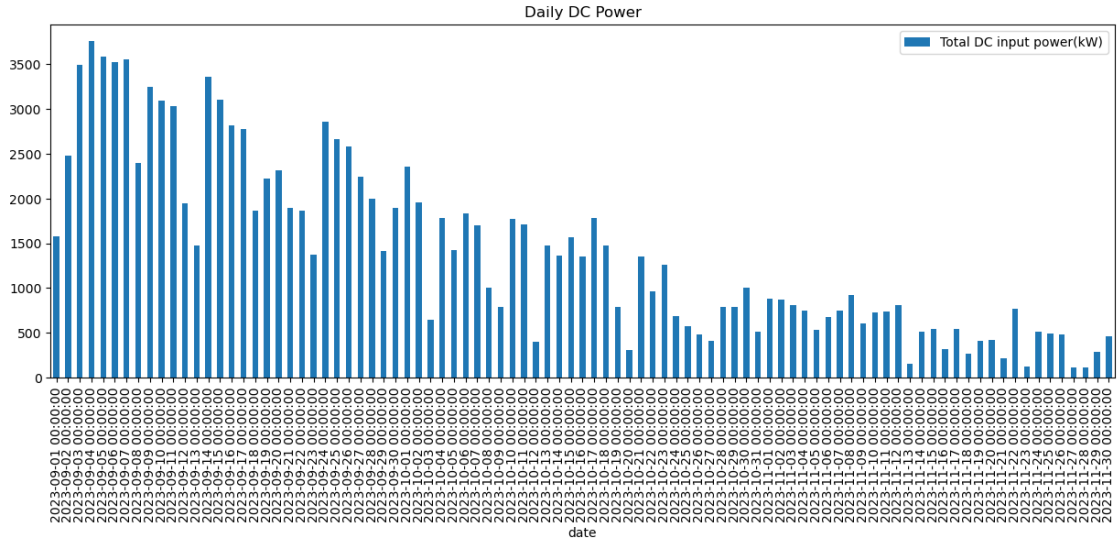
```
            data[cols[i - 1]].plot(ax=ax, style='k.')
            ax.set_title('{} {}'.format(title, cols[i - 1]))

# Call the function
multi_plot(data=calendar_dc)
plt.tight_layout()  # Adjust layout for better visualization
plt.show()
```



Almost all the curves are the same despite some fluctuation between 11 am and 4 pm. except the curve of September 5, 6 and 7 which gives a uniform shape.

[84]:
```
daily_dc = Werne1_data.groupby('date')['Total DC input power(kW)'].agg('sum')
```

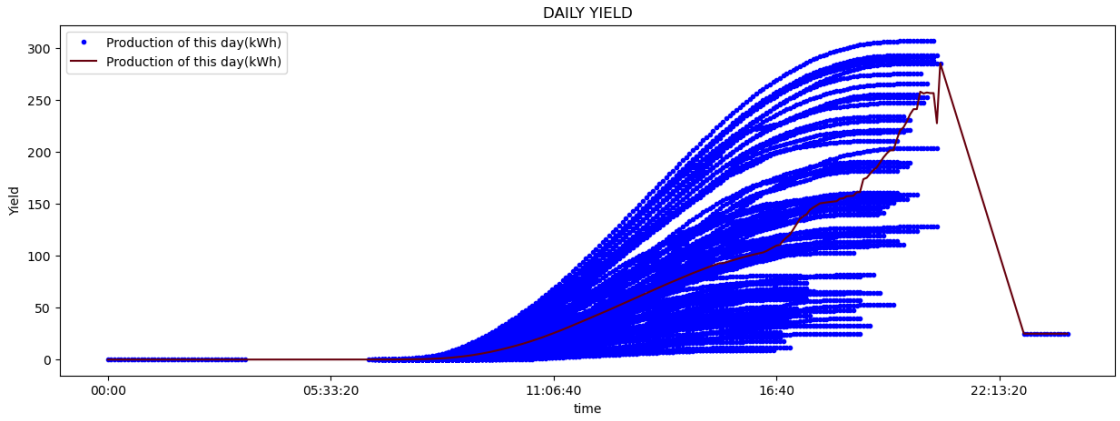
[85]:
```
daily_dc.plot.bar(figsize=(15,5), legend=True)
plt.title('Daily DC Power')
plt.show()
```



Only 2023-09-04 dc power is maximun.

Daily Yield

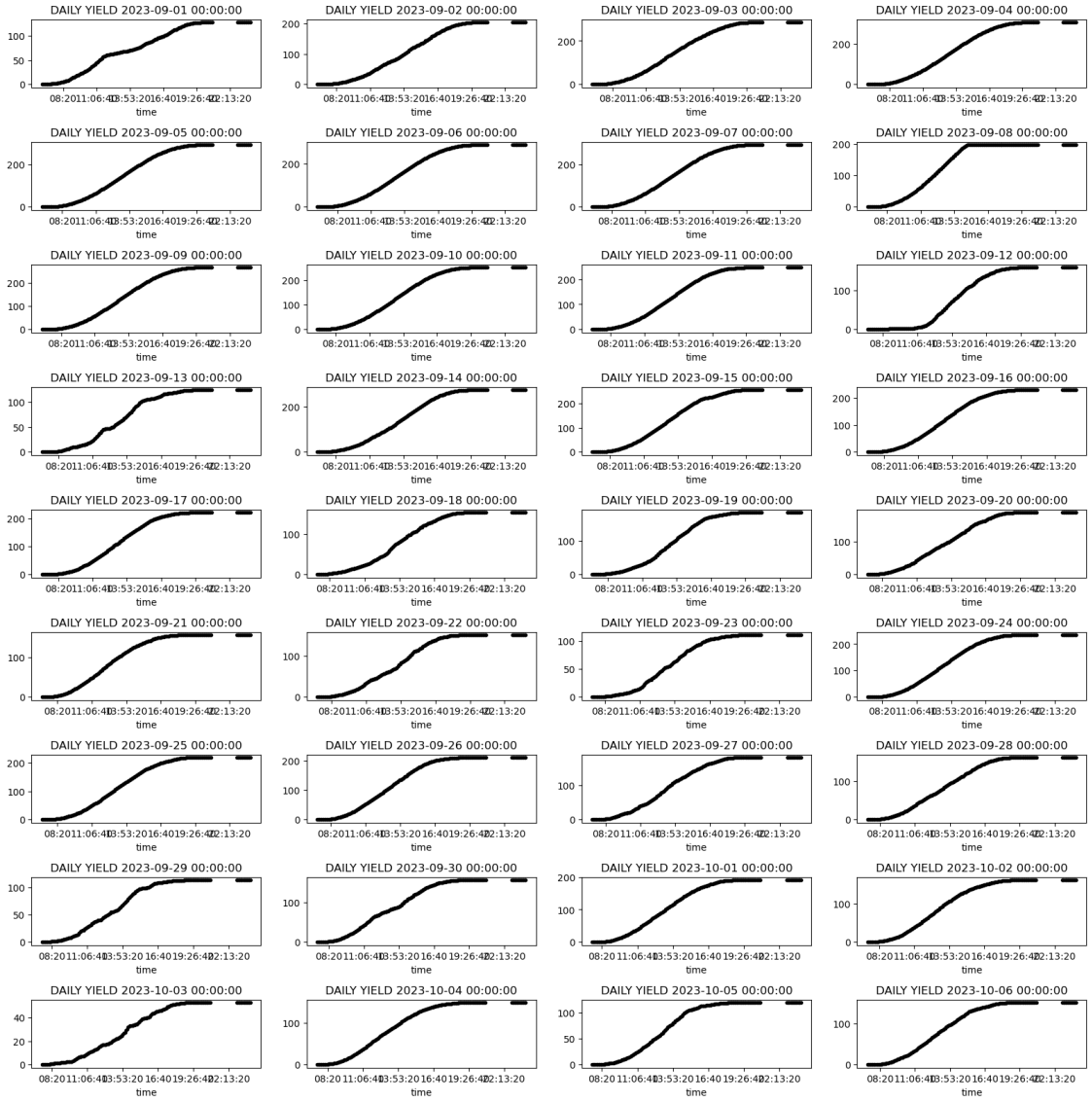
[86]:
```
Werne1_data.plot(x='time', y='Production of this day(kWh)', style='b.',␣
↪figsize=(15,5))
Werne1_data.groupby('time')['Production of this day(kWh)'].agg('mean').
↪plot(legend=True, colormap='Reds_r')
plt.title('DAILY YIELD')
plt.ylabel('Yield')
plt.show()
```



data gives us a logistics-like function but after 18:00 the energy decrease slowly; suddenly at 00:00 breakdown.

[87]:
```
#pivot table data
daily_yield = Werne1_data.pivot_table(values='Production of this day(kWh)',␣
 ↪index='time', columns='date')
```

[88]:
```
# we plot all daily yield
multi_plot(data=daily_yield.interpolate(), row=9, col=4, title='DAILY YIELD')
```



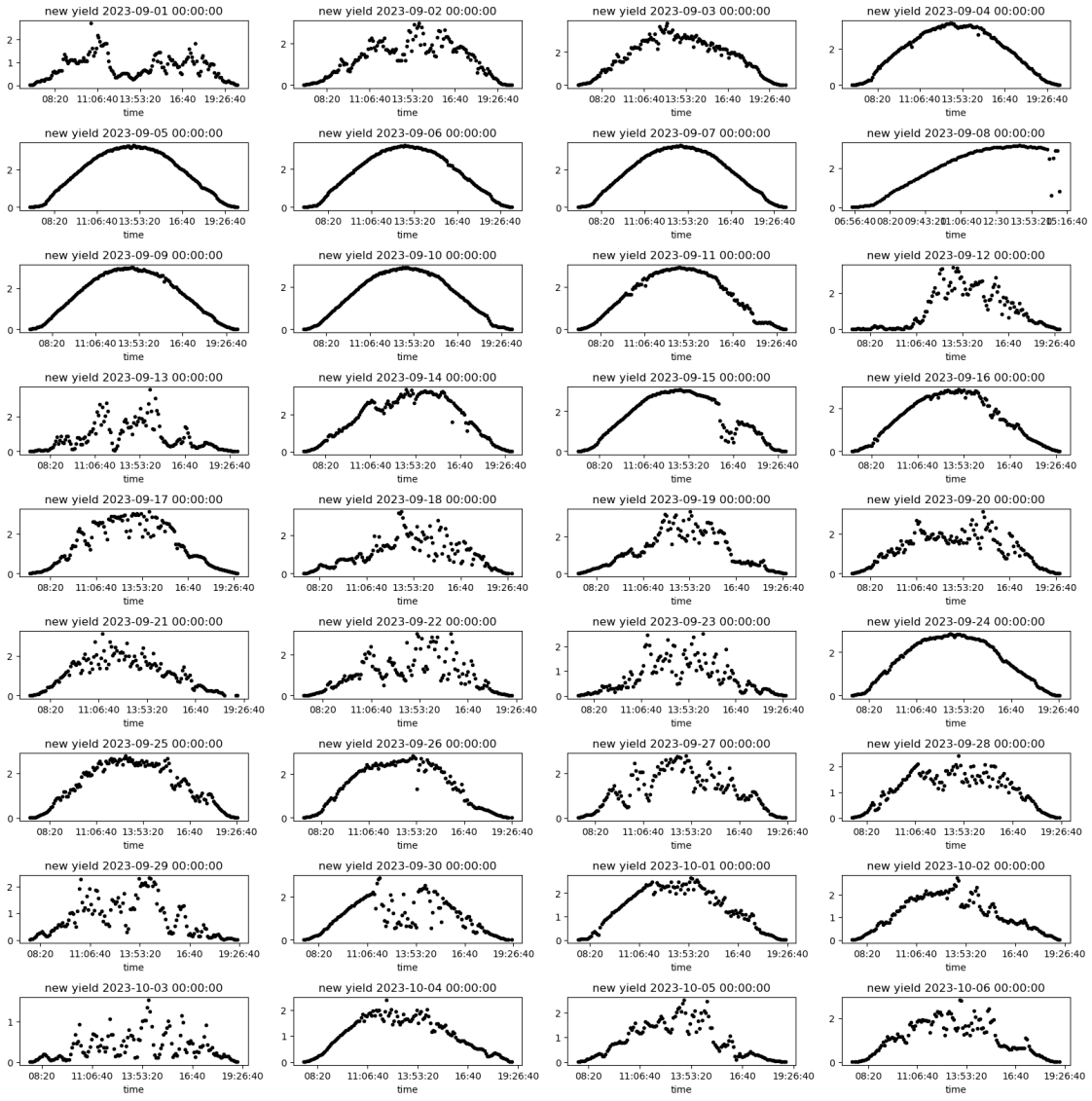
As we can see some daily_yield date (2023-09-01, 2023-10-30,...) have a logistic shape with missing values.

Every 5 min data is recorded. After 5 min, we get a new yield. To compute this new yield it is just this formula:

`new yield = next yield - previous yield`. It is a difference equation that `.diff()` pandas

function can help us to compute it.

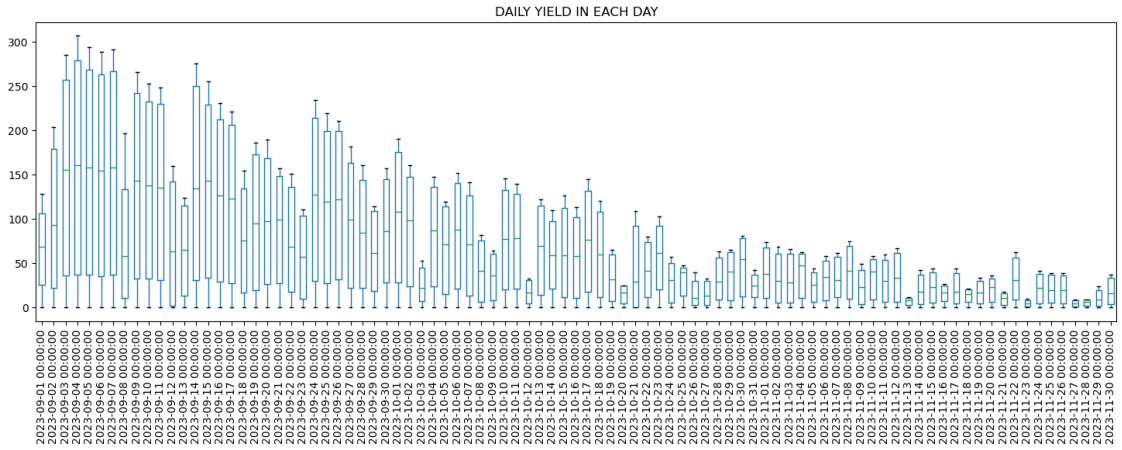
[89]:
```
#plotting a change rate daily yield over time
multi_plot(data=daily_yield.diff()[daily_yield.diff()>0], row=9, col=4,␣
 ↪title='new yield')
```



Between 08:20 and 16:40, we obtain each 5min, newyield>1 with fluctuation.

Daily Yield each day

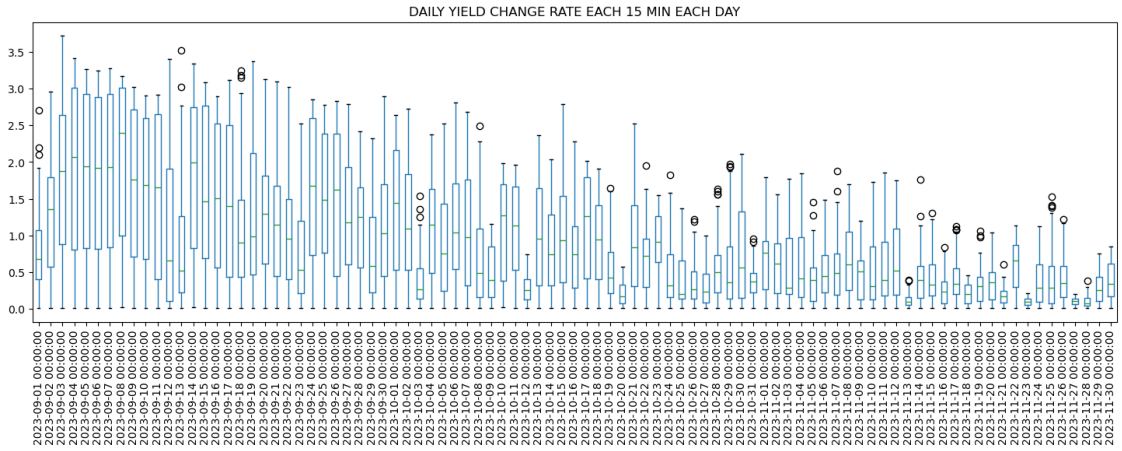
[90]:
```
daily_yield.boxplot(figsize=(18,5), rot=90, grid=False)
plt.title('DAILY YIELD IN EACH DAY')
plt.show()
```



For each day, the daily yield change. some day is high. The observation of all boxes is good, outlier does not exist.

For further details see Wikipedia's entry for `boxplot`.

[91]:
```
daily_yield.diff()[daily_yield.diff()>0].boxplot(figsize=(18,5), rot=90,␣
  ↪grid=False)
plt.title('DAILY YIELD CHANGE RATE EACH 15 MIN EACH DAY')
plt.show()
```

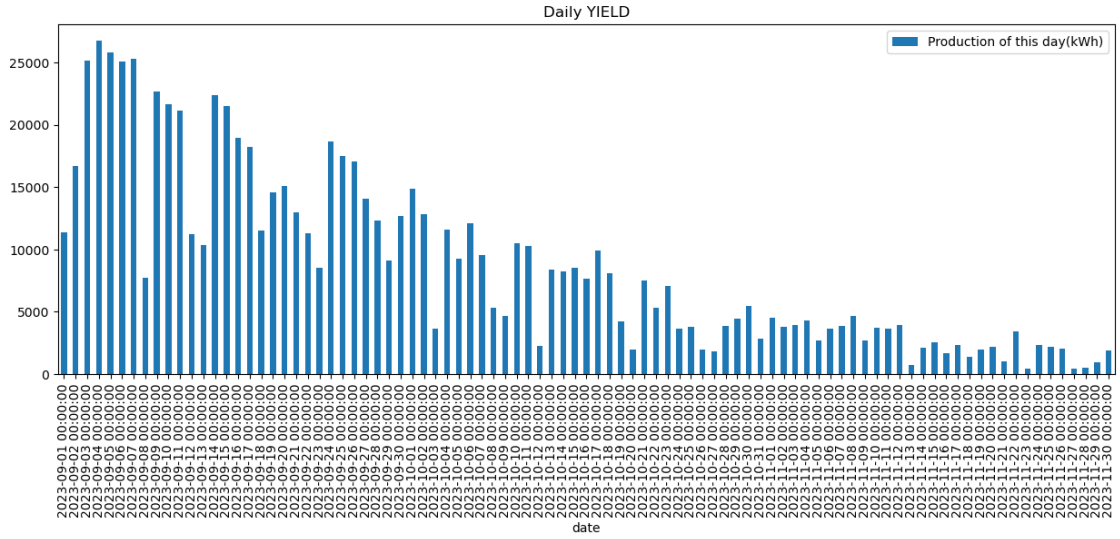


Some days have an outlier for example: 2023-09-01 and 2023-10-03.

[92]:
```
#we compute a daily yield for each date.
dyield = Werne1_data.groupby('date')['Production of this day(kWh)'].agg('sum')
```

[93]:
```
dyield.plot.bar(figsize=(15,5), legend=True)
plt.title('Daily YIELD')
```

```
plt.show()
```



6 Plant I (Werne): Weather Sensor Data

[94]:
```
file1 = 'Werne1_sensor.csv'
```

[95]:
```
Werne1_sensor = pd.read_csv(file1)
```

[96]:
```
Werne1_sensor.head()
```

[96]:

	DATE_TIME	Site name	device name	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	Irradiation (Satellite) Wh/m²
0	2023-09-01 06:35:00	Werne	102140090004/6T2199004131	17.356912	28.1	0.0
1	2023-09-01 06:40:00	Werne	102140090004/6T2199004131	17.493579	27.9	0.0
2	2023-09-01 06:45:00	Werne	102140090004/6T2199004131	17.630245	28.1	0.0
3	2023-09-01 06:50:00	Werne	102140090004/6T2199004131	17.766912	28.3	0.0
4	2023-09-01 06:55:00	Werne	102140090004/6T2199004131	17.903578	28.5	0.0

[97]:
```
Werne1_sensor.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12183 entries, 0 to 12182
```

```
Data columns (total 6 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   DATE_TIME                       12183 non-null  object
 1   Site name                       12183 non-null  object
 2   device name                     12183 non-null  object
 3   AMBIENT_TEMPERATURE             12183 non-null  float64
 4   MODULE_TEMPERATURE              12183 non-null  float64
 5   Irradiation (Satellite) Wh/m²   12183 non-null  float64
dtypes: float64(3), object(3)
memory usage: 571.2+ KB
```

Cleaning data

I convert DATE_TIME object type to datetime type. After I separate DATE_TIME to date and time

[98]:
```
Werne1_sensor['DATE_TIME'] = pd.to_datetime(Werne1_sensor['DATE_TIME'],␣
↪errors='coerce')
```

[99]:
```
Werne1_sensor['time'] = Werne1_sensor['DATE_TIME'].dt.time
Werne1_sensor['date'] = pd.to_datetime(Werne1_sensor['DATE_TIME'].dt.date)

del Werne1_sensor['Site name']
del Werne1_sensor['device name']
```

[100]:
```
Werne1_sensor.shape # our data reduced very well
```

[100]:
```
(12183, 6)
```

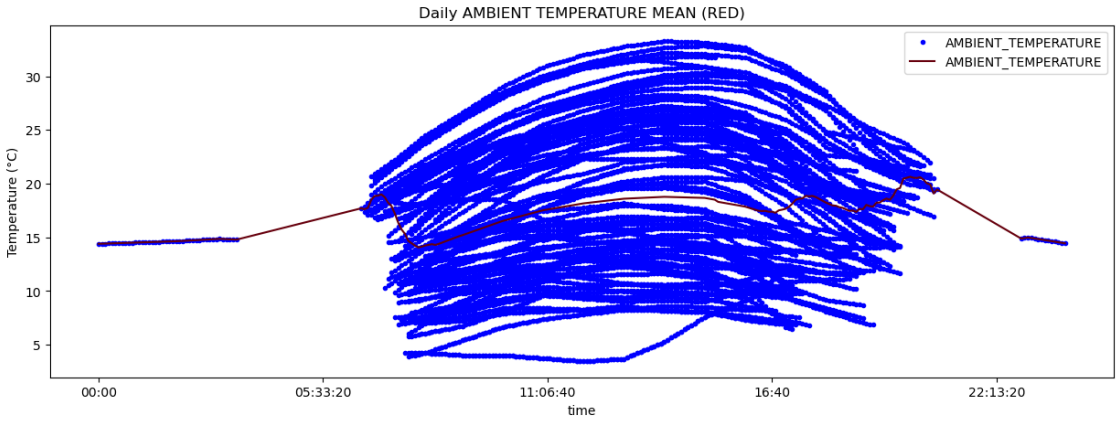
[101]:
```
Werne1_sensor.tail()
```

[101]:
```
                DATE_TIME  AMBIENT_TEMPERATURE  MODULE_TEMPERATURE  \
12178 2023-11-30 16:35:00            10.559412                24.2
12179 2023-11-30 16:40:00            10.523578                24.0
12180 2023-11-30 16:45:00            10.487745                23.8
12181 2023-11-30 16:50:00            10.451912                24.0
12182 2023-11-30 16:55:00            10.416078                24.6

       Irradiation (Satellite) Wh/m²      time       date
12178                            0.0  16:35:00 2023-11-30
12179                            0.0  16:40:00 2023-11-30
12180                            0.0  16:45:00 2023-11-30
12181                            0.0  16:50:00 2023-11-30
12182                            0.0  16:55:00 2023-11-30
```

EDA for Ambient Temperature, Module Temperature **and** Irradiation Here, we do 1. Line or scatter plot 2. %change. 3. Box and Whisker plot 4. calendar plot 5. Bar chart. 6. Lag plot

Ambient Temperature

[102]:
```
Werne1_sensor.plot(x='time', y = 'AMBIENT_TEMPERATURE' , style='b.',␣
 ↪figsize=(15,5))
Werne1_sensor.groupby('time')['AMBIENT_TEMPERATURE'].agg('mean').
 ↪plot(legend=True, colormap='Reds_r')
plt.title('Daily AMBIENT TEMPERATURE MEAN (RED)')
plt.ylabel('Temperature (°C)')
plt.show()
```



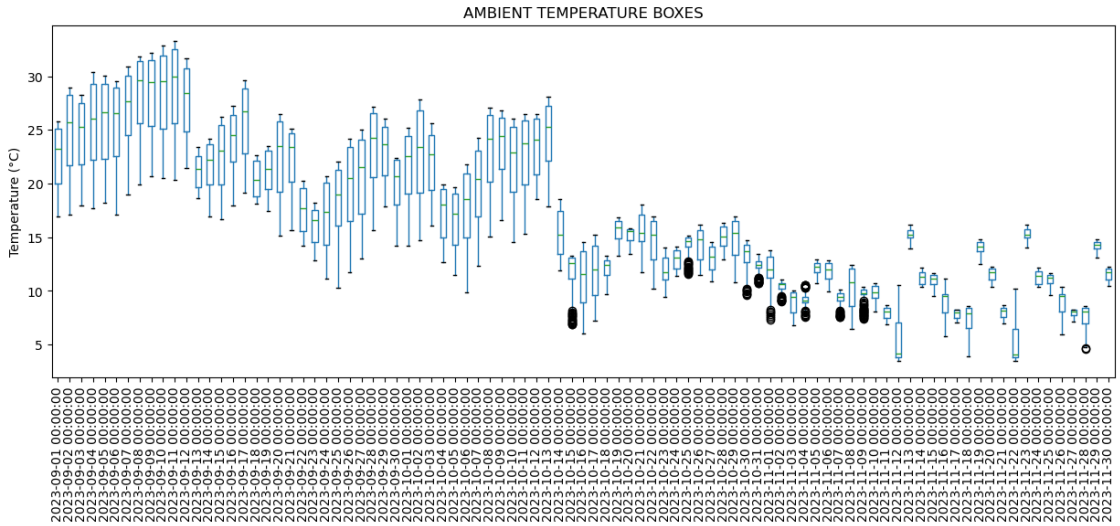
[103]:
```
ambient = Werne1_sensor.pivot_table(values='AMBIENT_TEMPERATURE', index='time',␣
 ↪columns='date')
```

[104]:
```
ambient.shape
```

[104]:
```
(228, 91)
```

[105]:
```
ambient.boxplot(figsize=(15,5), grid=False, rot=90)
plt.title('AMBIENT TEMPERATURE BOXES')
plt.ylabel('Temperature (°C)')
```

[105]:
```
Text(0, 0.5, 'Temperature (°C)')
```

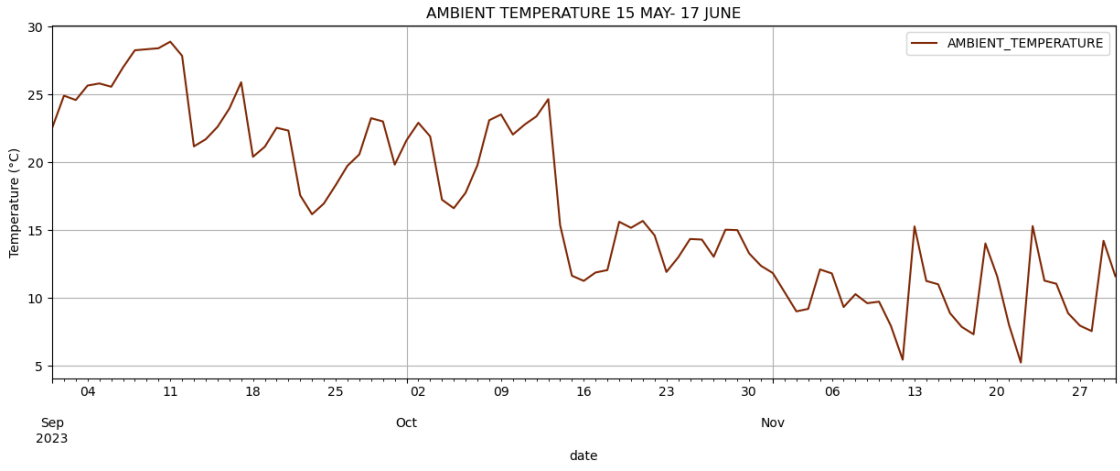


Which date ambient temperature mean is maximun?

[106]:
```
am_temp = Werne1_sensor.groupby('date')['AMBIENT_TEMPERATURE'].agg('mean')
```

[107]:
```
am_temp.plot(grid=True, figsize=(15,5), legend=True, colormap='Oranges_r')
plt.title('AMBIENT TEMPERATURE 15 MAY- 17 JUNE')
plt.ylabel('Temperature (°C)')
```

[107]:
```
Text(0, 0.5, 'Temperature (°C)')
```



Comment:

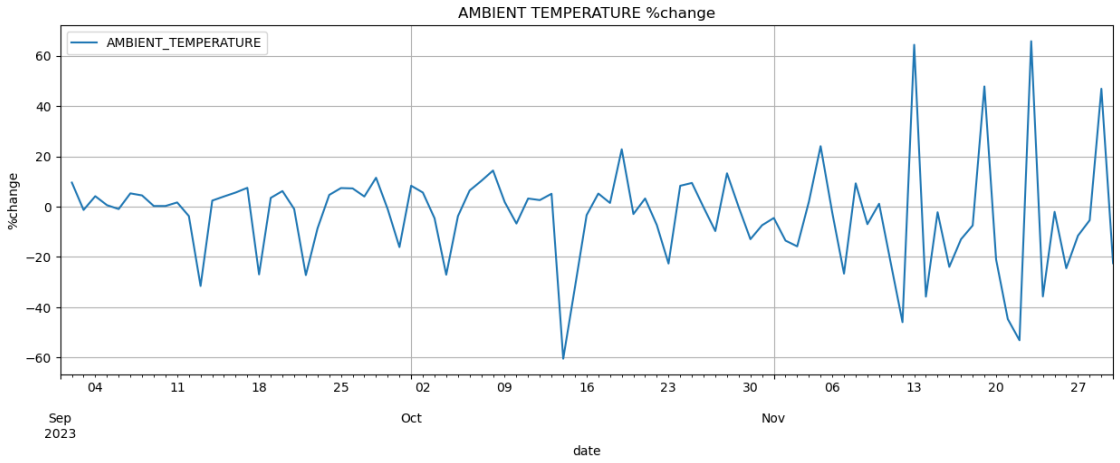
In September, ambiant Temperature in Werne was between 15 and 28°C, this means that September was very hot. But in October ambiant Temperature decreases considerately between 12 and 22°C.

In the next cell, we will seek how % change of ambient Temperature is.

[108]:
```
am_change_temp = (am_temp.diff()/am_temp)*100
```

[109]:
```
am_change_temp.plot(figsize=(15,5), grid=True, legend=True)
plt.ylabel('%change')
plt.title('AMBIENT TEMPERATURE %change')
```

[109]:
```
Text(0.5, 1.0, 'AMBIENT TEMPERATURE %change')
```



Comment

1. Since 11 September 2023 to 14 September 2023, the ambient Temperature decreases to -30%.
2. Since 24 September 2023 to 28 September 2023, the ambient Temperature increases to 15% and tomorrow decreases to -18%.
3. Since 15 October 2023 to 17 October 2023, the ambient Temperature decreases to -60% and tomorrow increases to 22%.
4. November month's, the ambiant Temperature %change stabilize between -50 and 50%.

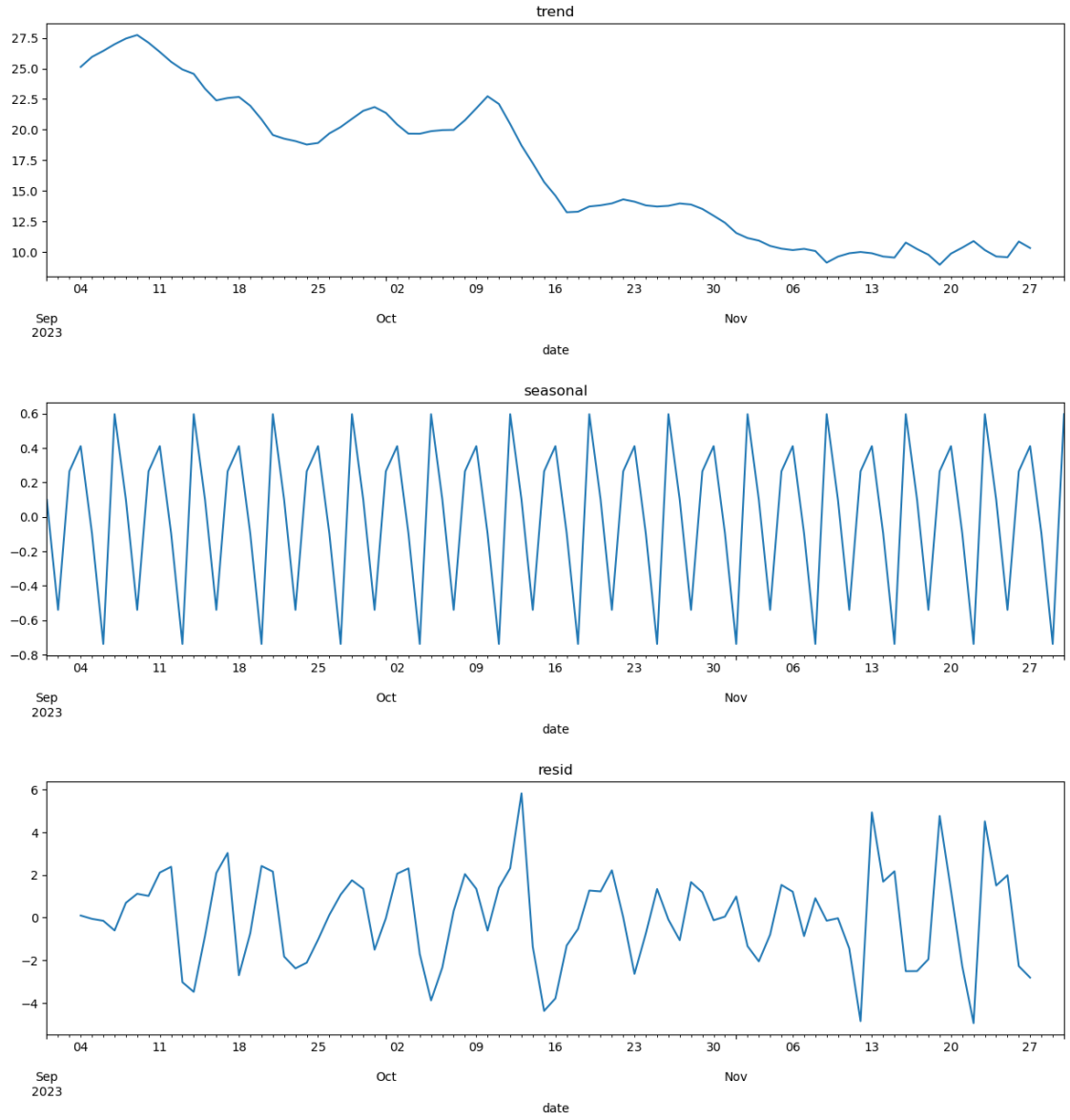
7 Ambient Temperature: seasonal, trend and residual.

[110]:
```
from scipy.signal import periodogram
import statsmodels.api as sm
```

[111]:
```
decomp = sm.tsa.seasonal_decompose(am_temp)
```

[112]:
```
cols = ['trend', 'seasonal', 'resid'] # take all column
data = [decomp.trend, decomp.seasonal, decomp.resid]
gp = plt.figure(figsize=(15,15))

gp.subplots_adjust(hspace=0.5)
```

```
for i in range(1, len(cols)+1):
    ax = gp.add_subplot(3,1, i)
    data[i-1].plot(ax=ax)
    ax.set_title('{}'.format(cols[i-1]))
```



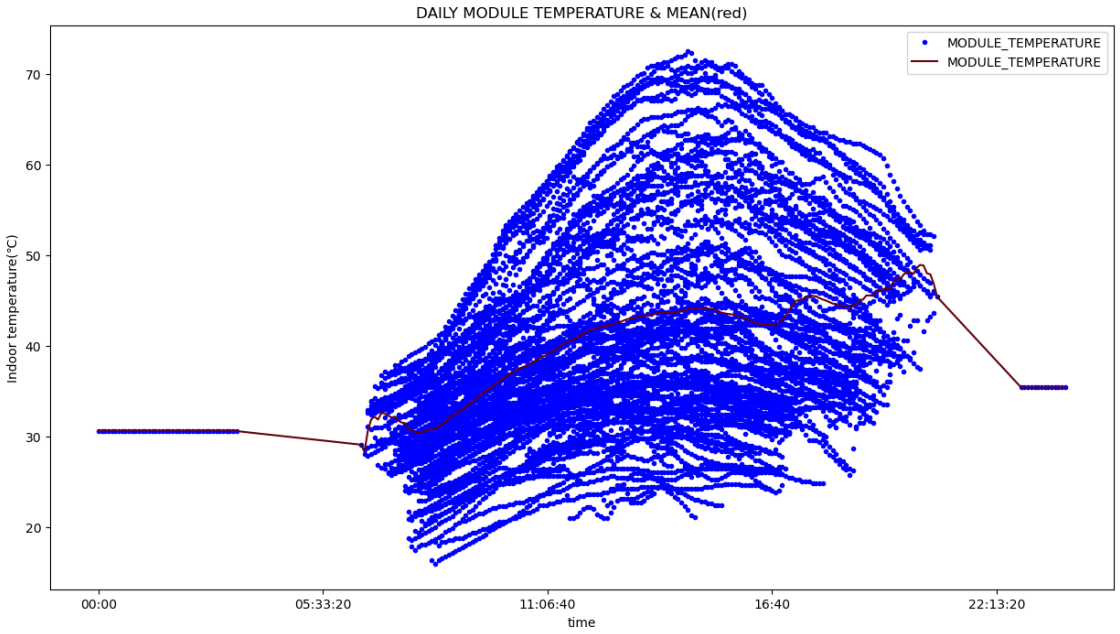
Comment

seasonality of ambient Temperature is the 7 days to see a maximun of temperature.

8 Module Temperature

[113]:
```
Werne1_sensor.plot(x='time', y='MODULE_TEMPERATURE', figsize=(15,8), style='b.')
Werne1_sensor.groupby('time')['MODULE_TEMPERATURE'].agg('mean').
 ↪plot(colormap='Reds_r', legend=True)
plt.title('DAILY MODULE TEMPERATURE & MEAN(red)')
plt.ylabel('Indoor temperature(℃)')
```

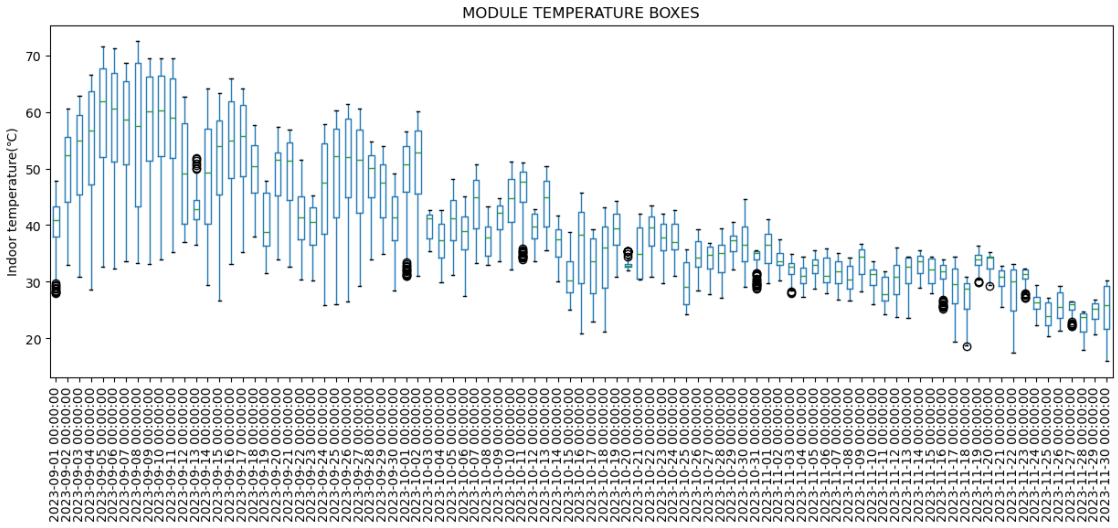
[113]: Text(0, 0.5, 'Indoor temperature(℃)')



[114]:
```
module_temp = Werne1_sensor.pivot_table(values='MODULE_TEMPERATURE',␣
 ↪index='time', columns='date')
```

[115]:
```
module_temp.boxplot(figsize=(15,5), grid=False, rot=90)
plt.title('MODULE TEMPERATURE BOXES')
plt.ylabel('Indoor temperature(℃)')
```

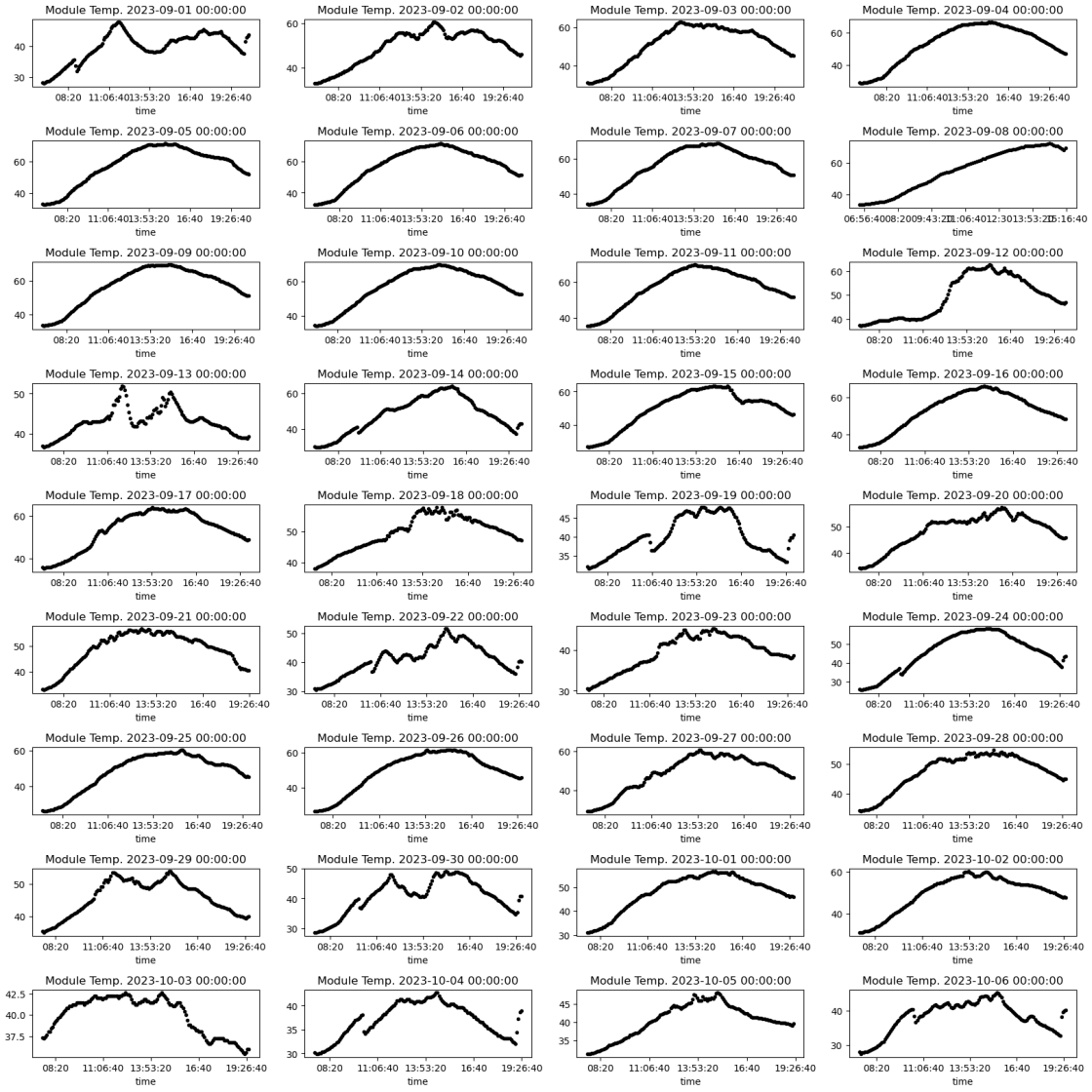
[115]: Text(0, 0.5, 'Indoor temperature(℃)')



Comment

Some dates contains outliers: 18-05-2020, 30-05-2020, 31-05-2020, 01-06-2020 etc.. The outlier of these dates occurs precisely at interval time [11:06:40,16:40].see

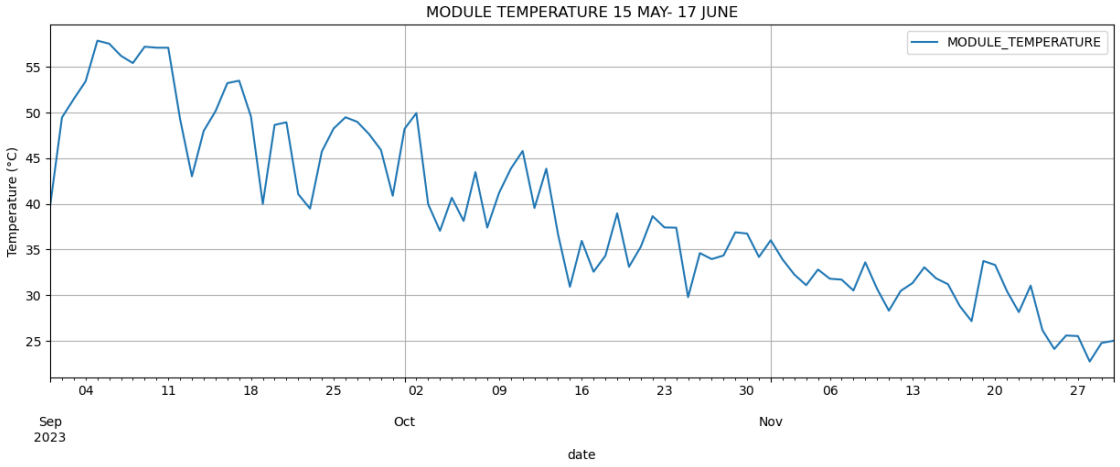
[116]:
```
multi_plot(module_temp, row=9,  col=4, title='Module Temp.')
```



[117]:
```
#we can also see also calendar plot
mod_temp = Werne1_sensor.groupby('date')['MODULE_TEMPERATURE'].agg('mean')
```

[118]:
```
mod_temp.plot(grid=True, figsize=(15,5), legend=True)
plt.title('MODULE TEMPERATURE 15 MAY- 17 JUNE')
plt.ylabel('Temperature (°C)')
```

[118]: Text(0, 0.5, 'Temperature (°C)')



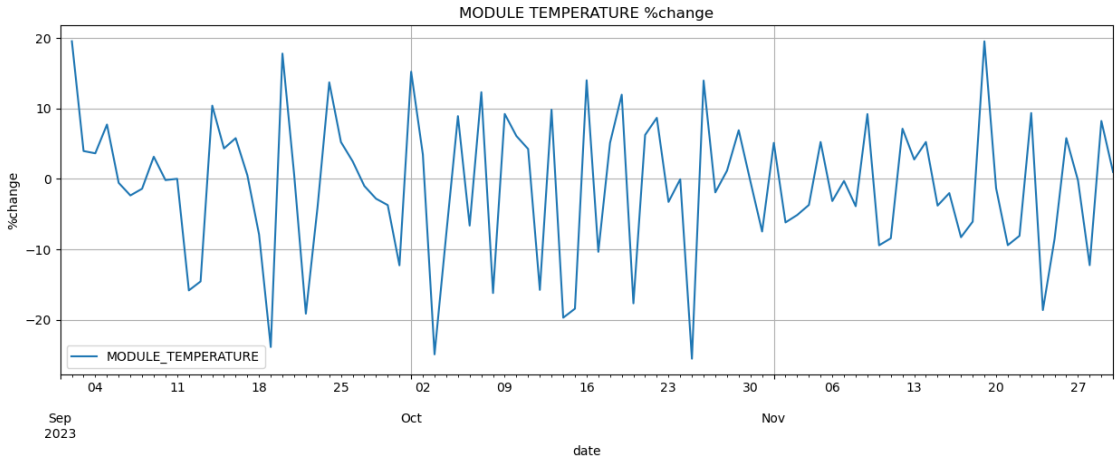
Comment

September months have: lots of hot dates from 6 to 11.

[119]:
```
#we plot a %change of MODULE TEMPERATURE.
chan_mod_temp = (mod_temp.diff()/mod_temp)*100
```

[120]:
```
chan_mod_temp.plot(grid=True, legend=True, figsize=(15,5))
plt.ylabel('%change')
plt.title('MODULE TEMPERATURE %change')
```

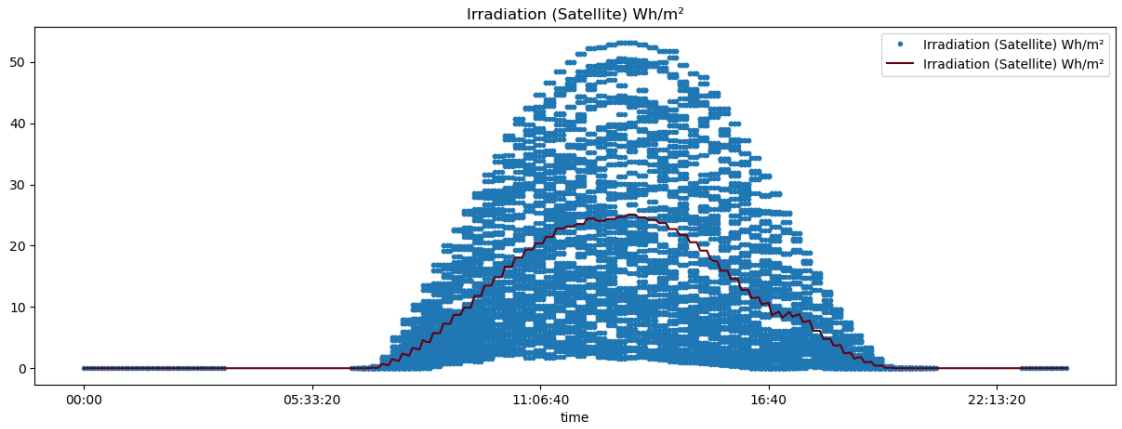
[120]:
```
Text(0.5, 1.0, 'MODULE TEMPERATURE %change')
```



9 Irradiation

[121]:
```
Werne1_sensor.plot(x='time', y = 'Irradiation (Satellite) Wh/m²', style='.',␣
↪legend=True, figsize=(15,5))
Werne1_sensor.groupby('time')['Irradiation (Satellite) Wh/m²'].agg('mean').
↪plot(legend=True, colormap='Reds_r')
plt.title('Irradiation (Satellite) Wh/m²')
```

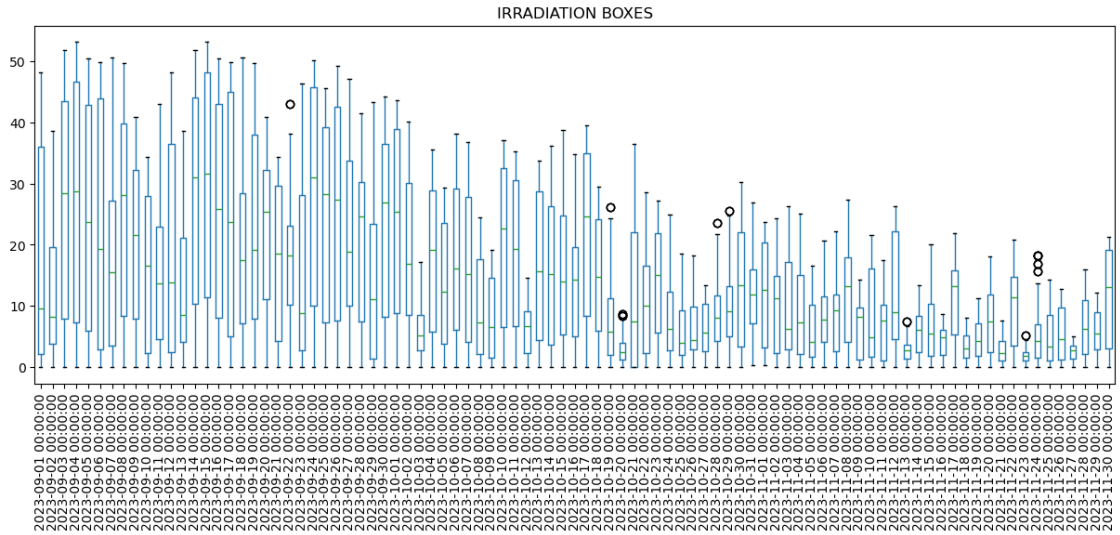
[121]: Text(0.5, 1.0, 'Irradiation (Satellite) Wh/m²')



[122]:
```
irra = Werne1_sensor.pivot_table(values='Irradiation (Satellite) Wh/m²',␣
↪index='time', columns='date')
```

[123]:
```
irra.boxplot(figsize=(15,5), rot = 90, grid=False)
plt.title('IRRADIATION BOXES')
```

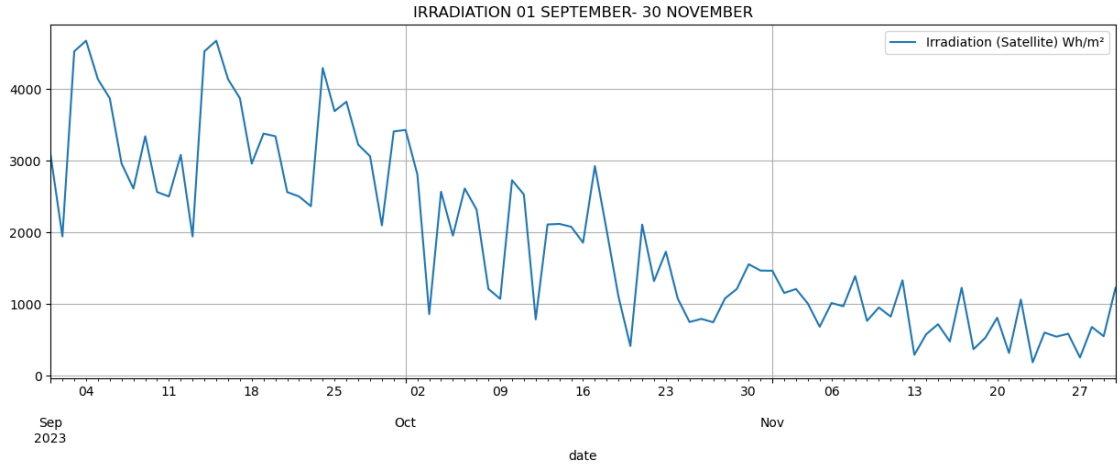
[123]: Text(0.5, 1.0, 'IRRADIATION BOXES')



[124]:
```
rad = Werne1_sensor.groupby('date')['Irradiation (Satellite) Wh/m²'].agg('sum')
```

[125]:
```
rad.plot(grid=True, figsize=(15,5), legend=True)
plt.title('IRRADIATION 01 SEPTEMBER- 30 NOVEMBER')
```

[125]: Text(0.5, 1.0, 'IRRADIATION 01 SEPTEMBER- 30 NOVEMBER')



N.B Since 03 September 2023 and 17 September are a dates where Werne are: 1. more produce dc power. 2. ambient temperature, module temperature are maximun.

This date is very special.

10 Correlation

In this part, we are making correlation between feature to see how some feature can explain another feature. or see relation between them.

[126]:
```
# we are merge our solar power generation data and weather sensor data
power_sensor = Werne1_sensor.merge(Werne1_data, left_on='DATE_TIME',␣
 ↪right_on='DATE_TIME')
```

[127]:
```
power_sensor.tail(3)
```

[127]:
```
                DATE_TIME  AMBIENT_TEMPERATURE  MODULE_TEMPERATURE  \
12180 2023-11-30 16:45:00            10.487745                23.8
12181 2023-11-30 16:50:00            10.451912                24.0
12182 2023-11-30 16:55:00            10.416078                24.6

       Irradiation (Satellite) Wh/m²    time_x     date_x  \
12180                            0.0  16:45:00 2023-11-30
12181                            0.0  16:50:00 2023-11-30
12182                            0.0  16:55:00 2023-11-30

       Total DC input power(kW)  Active power AC(kW)  \
12180                       0.0                  0.0
12181                       0.0                  0.0
12182                       0.0                  0.0

       Production of this day(kWh)  Total yield(kWh)    time_y     date_y
12180                        37.27         120230.01  16:45:00 2023-11-30
12181                        37.27         120230.01  16:50:00 2023-11-30
12182                        37.27         120230.01  16:55:00 2023-11-30
```

[128]:
```
#we remove the columns that we do not need
del power_sensor['date_x']
del power_sensor['date_y']
del power_sensor['time_x']
del power_sensor['time_y']
```

[129]:
```
power_sensor.tail(3)
```

[129]:
```
                DATE_TIME  AMBIENT_TEMPERATURE  MODULE_TEMPERATURE  \
12180 2023-11-30 16:45:00            10.487745                23.8
12181 2023-11-30 16:50:00            10.451912                24.0
12182 2023-11-30 16:55:00            10.416078                24.6

       Irradiation (Satellite) Wh/m²  Total DC input power(kW)  \
12180                            0.0                       0.0
12181                            0.0                       0.0
12182                            0.0                       0.0
```

```
       Active power AC(kW)  Production of this day(kWh)  Total yield(kWh)
12180                  0.0                        37.27         120230.01
12181                  0.0                        37.27         120230.01
12182                  0.0                        37.27         120230.01
```

[130]:
```
power_sensor.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12183 entries, 0 to 12182
Data columns (total 8 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   DATE_TIME                       12183 non-null  datetime64[ns]
 1   AMBIENT_TEMPERATURE             12183 non-null  float64
 2   MODULE_TEMPERATURE              12183 non-null  float64
 3   Irradiation (Satellite) Wh/m²   12183 non-null  float64
 4   Total DC input power(kW)        12183 non-null  float64
 5   Active power AC(kW)             12183 non-null  float64
 6   Production of this day(kWh)     12183 non-null  float64
 7   Total yield(kWh)                12183 non-null  float64
dtypes: datetime64[ns](1), float64(7)
memory usage: 761.6 KB
```

[131]:
```
#we start correlation
power_sensor.corr(method = 'spearman')
```

[131]:
```
                                DATE_TIME  AMBIENT_TEMPERATURE  \
DATE_TIME                        1.000000            -0.831410
AMBIENT_TEMPERATURE             -0.831410             1.000000
MODULE_TEMPERATURE              -0.768448             0.846828
Irradiation (Satellite) Wh/m²   -0.348208             0.495431
Total DC input power(kW)        -0.422273             0.550015
Active power AC(kW)             -0.421862             0.549579
Production of this day(kWh)     -0.440746             0.455717
Total yield(kWh)                 1.000000            -0.831409

                                MODULE_TEMPERATURE  \
DATE_TIME                                -0.768448
AMBIENT_TEMPERATURE                       0.846828
MODULE_TEMPERATURE                        1.000000
Irradiation (Satellite) Wh/m²             0.548117
Total DC input power(kW)                  0.610510
Active power AC(kW)                       0.609902
Production of this day(kWh)               0.723558
Total yield(kWh)                         -0.768434
```

```
                                Irradiation (Satellite) Wh/m²  \
DATE_TIME                                           -0.348208
AMBIENT_TEMPERATURE                                  0.495431
MODULE_TEMPERATURE                                   0.548117
Irradiation (Satellite) Wh/m²                        1.000000
Total DC input power(kW)                             0.917215
Active power AC(kW)                                  0.917262
Production of this day(kWh)                          0.322045
Total yield(kWh)                                    -0.348209

                                Total DC input power(kW)  Active power AC(kW)  \
DATE_TIME                                      -0.422273            -0.421862
AMBIENT_TEMPERATURE                             0.550015             0.549579
MODULE_TEMPERATURE                              0.610510             0.609902
Irradiation (Satellite) Wh/m²                   0.917215             0.917262
Total DC input power(kW)                        1.000000             0.999994
Active power AC(kW)                             0.999994             1.000000
Production of this day(kWh)                     0.383860             0.383128
Total yield(kWh)                               -0.422274            -0.421862

                                Production of this day(kWh)  Total yield(kWh)
DATE_TIME                                         -0.440746          1.000000
AMBIENT_TEMPERATURE                                0.455717         -0.831409
MODULE_TEMPERATURE                                 0.723558         -0.768434
Irradiation (Satellite) Wh/m²                      0.322045         -0.348209
Total DC input power(kW)                           0.383860         -0.422274
Active power AC(kW)                                0.383128         -0.421862
Production of this day(kWh)                        1.000000         -0.440716
Total yield(kWh)                                  -0.440716          1.000000
```

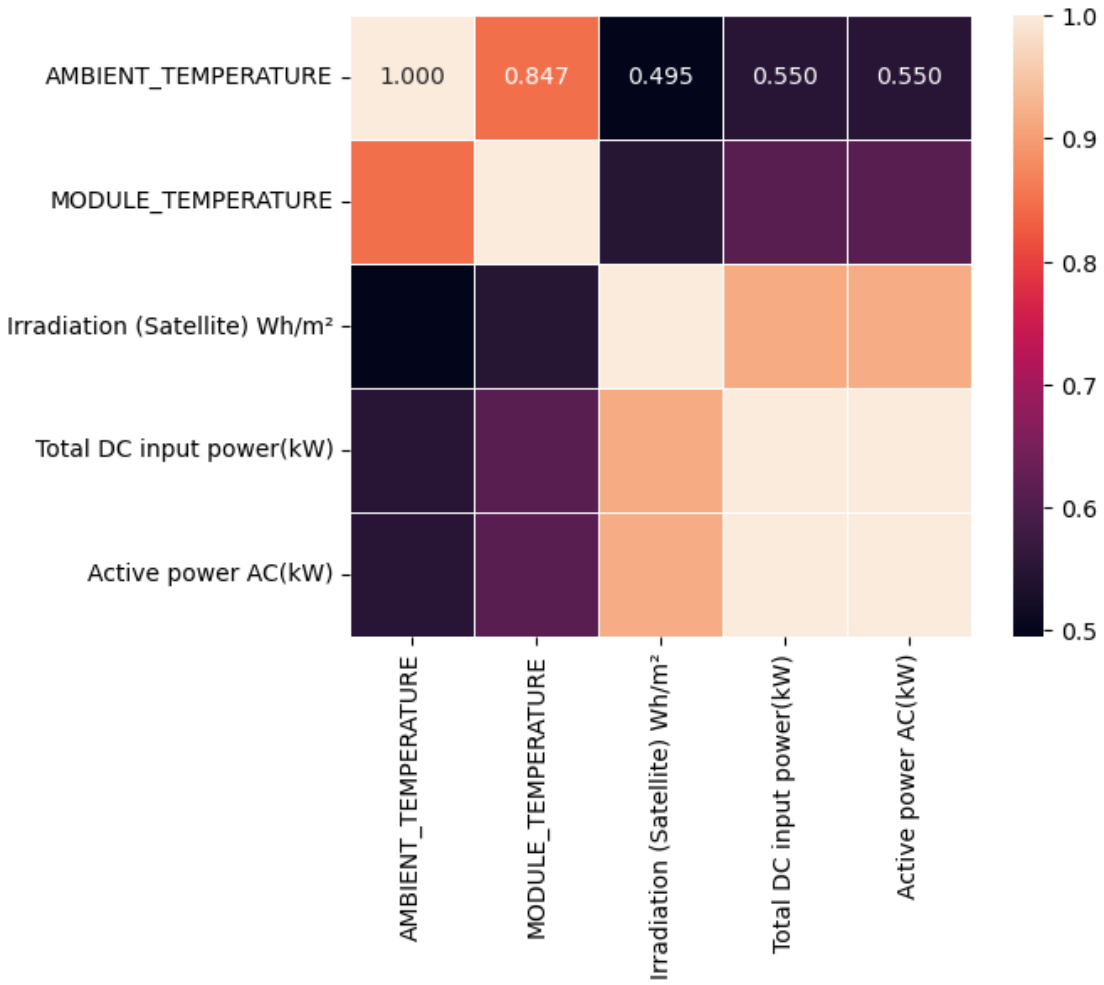
Comment

DAILY_YIELD is not correlated with all feature but AMBIENT_TEMPERATURE is moreless correlated.

TOTAL_YIELD is also not correlated with all feature. I remove it in the correlation matrix.

[132]:
```
del power_sensor['DATE_TIME']
corr = power_sensor.drop(columns=['Production of this day(kWh)', 'Total␣
 ↪yield(kWh)']).corr(method = 'spearman')
```

[133]:
```
plt.figure(dpi=100)
sns.heatmap(corr, robust=True, annot=True, fmt='0.3f', linewidths=.5,␣
 ↪square=True)
plt.show()
```



[134]:
```
# we make pairplot
sns.pairplot(power_sensor.drop(columns=['Production of this day(kWh)', 'Total␣
↪yield(kWh)']))
plt.show()
```

```
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
```

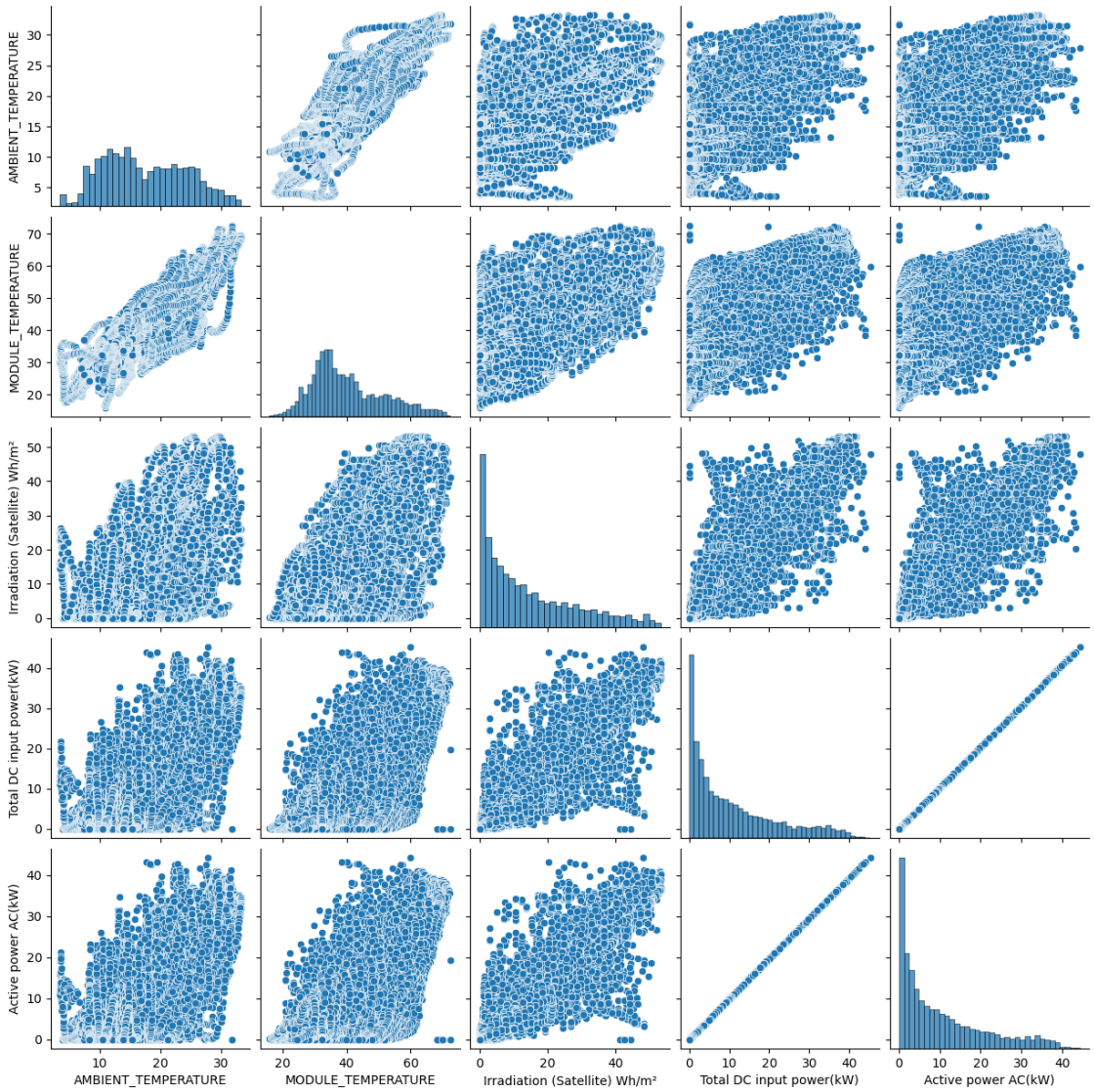
```
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
```

```
  with pd.option_context('mode.use_inf_as_na', True):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
```

```
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
```

```
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
```

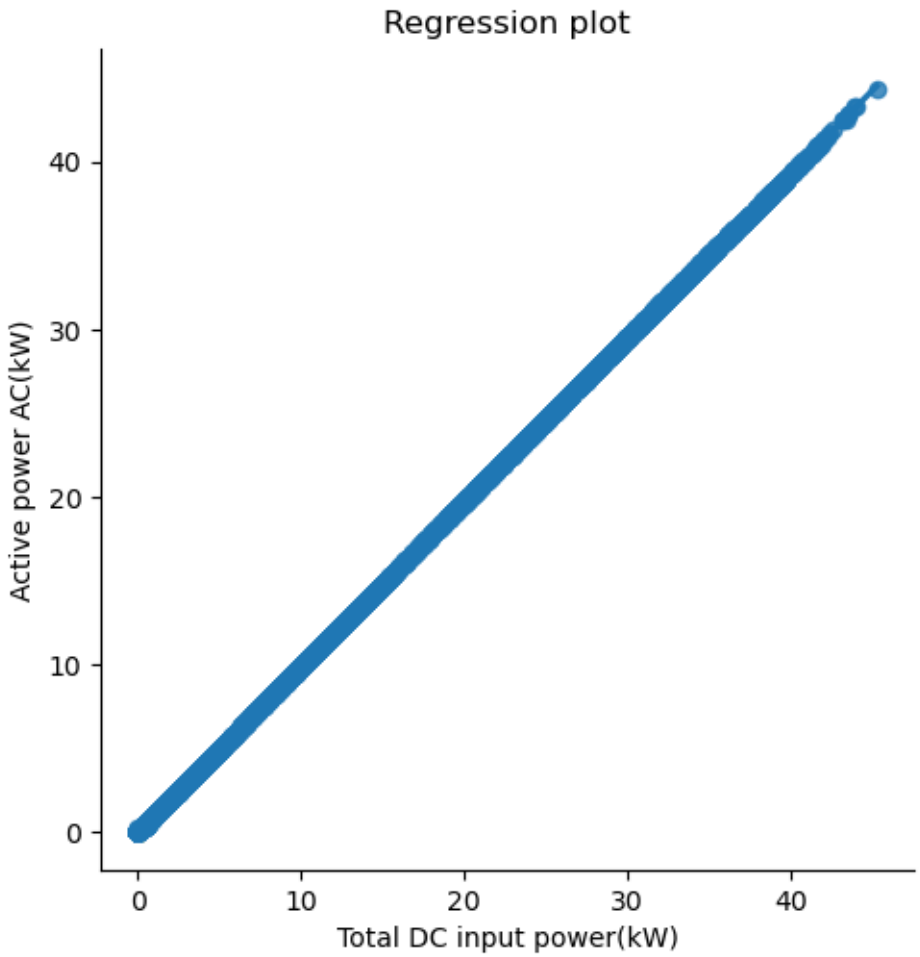
```
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/marek/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



[135]:
```
#we plot dc power vs ac power
```

[136]:
```
plt.figure(dpi=100)
sns.lmplot(x='Total DC input power(kW)', y='Active power AC(kW)',␣
 ↪data=power_sensor)
plt.title('Regression plot')
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

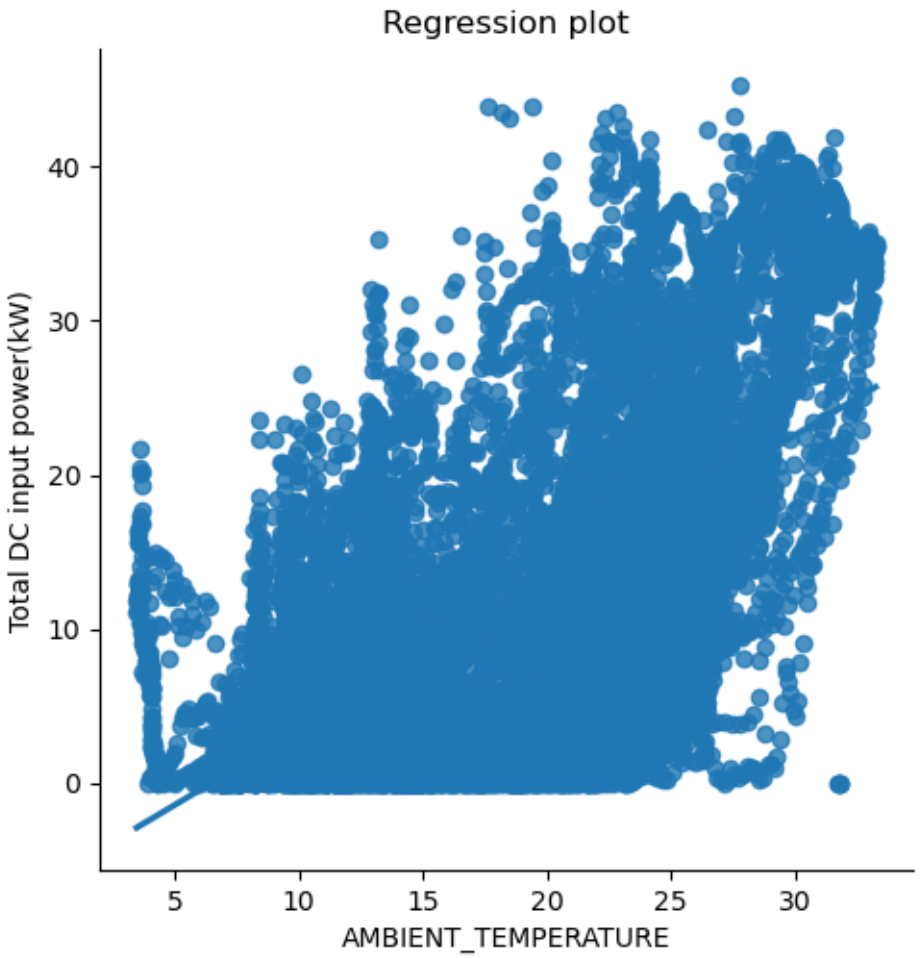


Comment

This graph said that inverter convert dc power to ac power linearly. dcpower=10∗acpower inverter lost 90% of their power when it convert.

[137]:
```
plt.figure(dpi=100)
sns.lmplot(x='AMBIENT_TEMPERATURE', y='Total DC input power(kW)',␣
  ↪data=power_sensor)
plt.title('Regression plot')
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

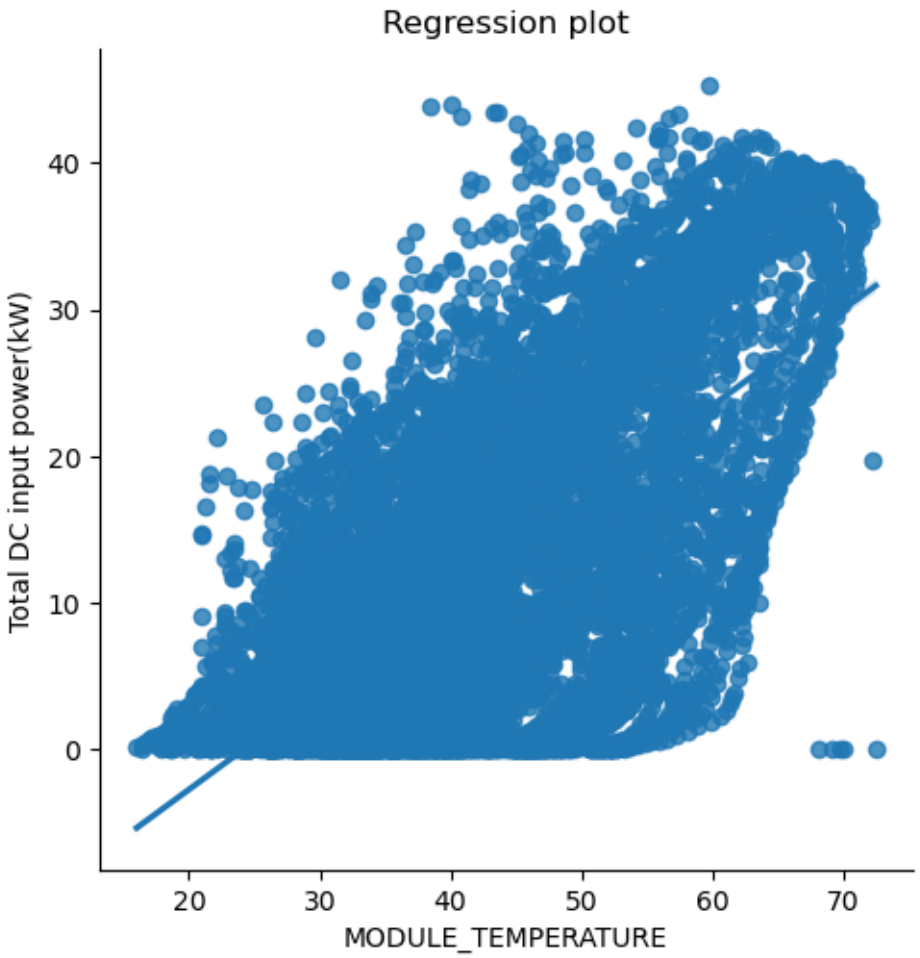


comment

DC_power increases non linearly with an Ambient_Temperature.

[138]:
```
plt.figure(dpi=100)
sns.lmplot(x='MODULE_TEMPERATURE', y='Total DC input power(kW)',␣
 ↪data=power_sensor)
plt.title('Regression plot')
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

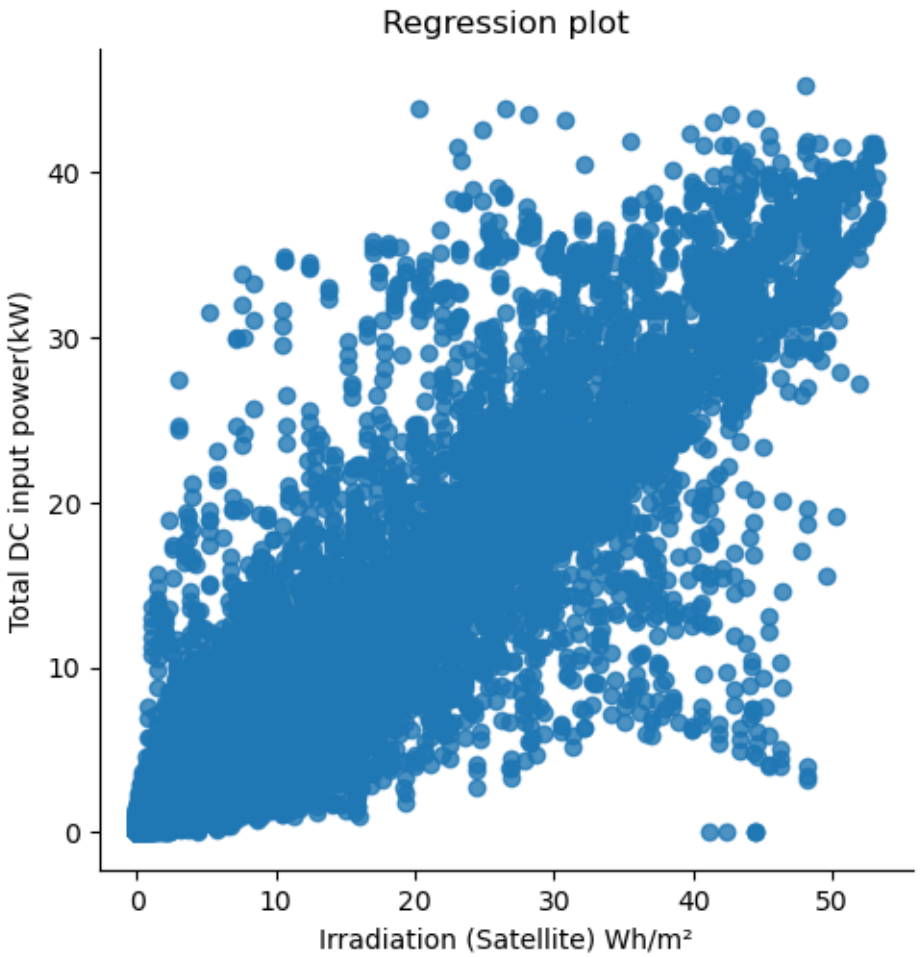


comment

DC_POWER is produced linearly by MODULE_TEMPERATURE with some variability.

[139]:
```
plt.figure(dpi=100)
sns.lmplot(x='Irradiation (Satellite) Wh/m²', y='Total DC input power(kW)',␣
↪data=power_sensor)
plt.title('Regression plot')
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



Comment

DC_Power increase with IRRADIATION.

What happens if I introuduce a difference Temperature between `AMBIENT_TEMPERATURE` AND `MODULE_TEMPERATURE`.

[140]:
```
# we introduce DELTA_TEMPERATURE
power_sensor['DELTA_TEMPERATURE'] = abs(power_sensor.AMBIENT_TEMPERATURE -␣
↪power_sensor.MODULE_TEMPERATURE)
```

[141]:
```
# we check if all is ok
power_sensor.tail(3)
```

[141]:

	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	Irradiation (Satellite) Wh/m²	\
12180	10.487745	23.8	0.0	
12181	10.451912	24.0	0.0	
12182	10.416078	24.6	0.0	

```
       Total DC input power(kW)  Active power AC(kW)  \
12180                       0.0                  0.0
12181                       0.0                  0.0
12182                       0.0                  0.0

       Production of this day(kWh)  Total yield(kWh)  DELTA_TEMPERATURE
12180                        37.27         120230.01          13.312255
12181                        37.27         120230.01          13.548088
12182                        37.27         120230.01          14.183922
```

[142]:
```
#now we use correlation
power_sensor.corr(method='spearman')['DELTA_TEMPERATURE']
```

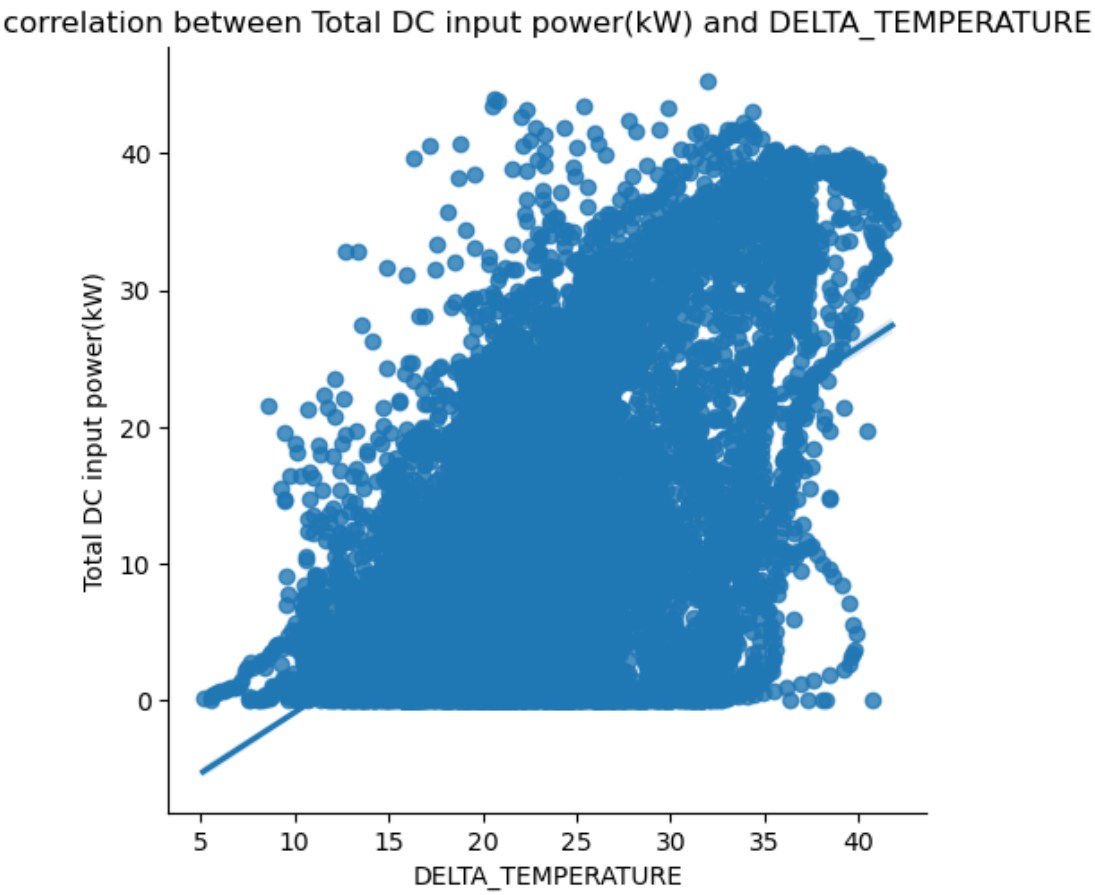
[142]:
```
AMBIENT_TEMPERATURE              0.337961
MODULE_TEMPERATURE               0.757974
Irradiation (Satellite) Wh/m²    0.409007
Total DC input power(kW)         0.466875
Active power AC(kW)              0.466267
Production of this day(kWh)      0.762579
Total yield(kWh)                -0.379689
DELTA_TEMPERATURE                1.000000
Name: DELTA_TEMPERATURE, dtype: float64
```

comment

we remark that YIELD does not depend on DELTA_TEMPERATURE also.

[143]:
```
sns.lmplot(x='DELTA_TEMPERATURE', y='Total DC input power(kW)',␣
 ↪data=power_sensor)
plt.title('correlation between Total DC input power(kW) and DELTA_TEMPERATURE')
```

[143]:
```
Text(0.5, 1.0, 'correlation between Total DC input power(kW) and
DELTA_TEMPERATURE')
```



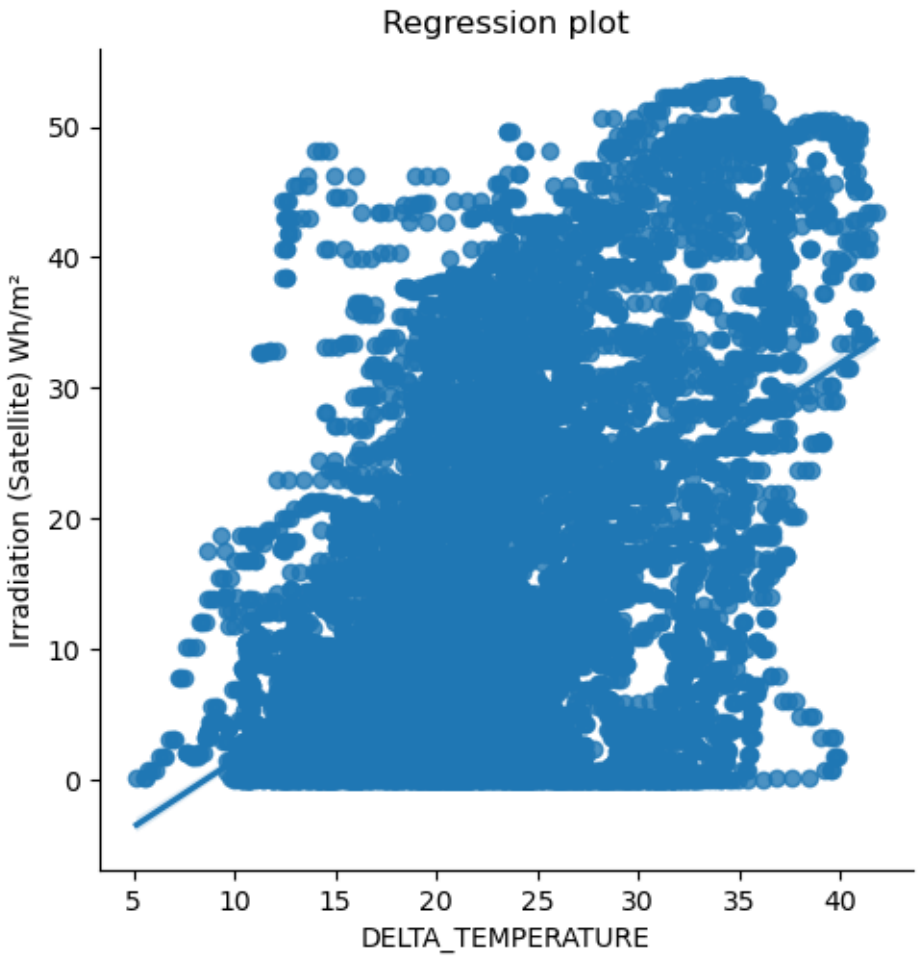
comment

We know that Q˙ ΔT .So, we could say that DC_POWER is influenced by heat transfer.

Heat transfer coefficient

[144]:
```
sns.lmplot(x='DELTA_TEMPERATURE', y='Irradiation (Satellite) Wh/m²',␣
 ↪data=power_sensor)
plt.title('Regression plot')
```

[144]:
```
Text(0.5, 1.0, 'Regression plot')
```



comment

`IRRADIATION` of Module and Heat Transfert between ambient air and Module are very well correlated.

short conclusion

In this section, I conclude that: 1. Yield does not depend on the Temperature, the dc/ac power and irradiation. 2. The transfert function between dc and ac power is linear. 3. DC power is indeed influenced by the ambient temperature, by the temperature of the module, by the irradiation and finally by the heat transfer between the module and the air. 4. Inverter of Werne lost 90% of their DC power when it convert.

11 Comparison of two power plants

11.1 Werne data vs Plant2 data

Rated power: - 01987 Schwarzheide Ruhlander Str. 66a: **Nennleistung 33,71 kWp** - 14728 Rhinow Straße der Jugend 7-10: **Nennleistung 64,6 kWp** - 14728 Rhinow Straße der Jugend 11-14:

Nennleistung 62,32 kWp - 14728 Rhinow Straße der Jugend 15-18: **Nennleistung 54,72 kWp** - 16831 Rheinsberg Mariefredstraße 7: **Nennleistung 39,6 kWp** - 32049 Herford_Halberstädter Str 25: **Nennleistung 67,32 kWp** - 39517 Tagerhütte Neustädter Ring 30: **Nennleistung 46,8 kWp** - 39517 Tangerhütte_Neustädter Ring 35: **Nennleistung 32,76 kWp** - 59368 Werne Becklohhof 1-7: **Nennleistung 69,3 kWp (Gesamt)** - 63303 Dreieich Hainer Chaussee 49: **Nennleistung 43,66 kWp**

[145]:
```
file2 = 'Plant2Data.csv'
```

[146]:
```
Herford2_data = pd.read_csv(file2)
```

[147]:
```
Herford2_data.head(3)
```

[147]:

	Site name	device name	start time	\
0	Herford	1020B0083371/2101073871ESL3000191	2023-09-01 06:25:00	
1	Herford	1020B0083371/2101073871ESL3000191	2023-09-01 06:30:00	
2	Herford	1020B0083371/2101073871ESL3000191	2023-09-01 06:35:00	

	Indoor temperature(℃)	Active power AC(kW)	Production of this day(kWh)	\
0	21.8	0.000	0.00	
1	21.7	0.031	0.00	
2	22.3	0.000	0.01	

	Total DC input power(kW)	Total yield(kWh)
0	0.000	74009.4
1	0.048	74009.4
2	0.000	74009.4

[148]:
```
Herford2_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11731 entries, 0 to 11730
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Site name                    11731 non-null  object
 1   device name                  11731 non-null  object
 2   start time                   11731 non-null  object
 3   Indoor temperature(℃)        11731 non-null  float64
 4   Active power AC(kW)          11731 non-null  float64
 5   Production of this day(kWh)  11731 non-null  float64
 6   Total DC input power(kW)     11731 non-null  float64
 7   Total yield(kWh)             11731 non-null  float64
dtypes: float64(5), object(3)
memory usage: 733.3+ KB
```

[149]:
```
#we compute a sum of 22 inverters
Herford2_data = Herford2_data.groupby('start time')[['Total DC input␣
↪power(kW)','Active power AC(kW)', 'Production of this day(kWh)','Total␣
↪yield(kWh)']].agg('sum').reset_index()
```

[150]:
```
Herford2_data['start time'] = pd.to_datetime(Herford2_data['start time'],␣
↪errors='coerce')
Herford2_data['time'] = Herford2_data['start time'].dt.time
Herford2_data['date'] = pd.to_datetime(Herford2_data['start time'].dt.date)

# To handle missing values, I use dropna() to remove rows with missing datetime␣
↪values:
Herford2_data.dropna(subset=['start time'], inplace=True)
```

[151]:
```
Herford2_data.head()
```

[151]:

	start time	Total DC input power(kW)	Active power AC(kW)	\
0	2023-09-01 06:25:00	0.000	0.000	
1	2023-09-01 06:30:00	0.048	0.031	
2	2023-09-01 06:35:00	0.000	0.000	
3	2023-09-01 06:40:00	0.183	0.152	
4	2023-09-01 06:45:00	0.309	0.252	

	Production of this day(kWh)	Total yield(kWh)	time	date
0	0.00	74009.40	06:25:00	2023-09-01
1	0.00	74009.40	06:30:00	2023-09-01
2	0.01	74009.40	06:35:00	2023-09-01
3	0.02	74009.44	06:40:00	2023-09-01
4	0.04	74009.45	06:45:00	2023-09-01

[152]:
```
Herford2_data.tail(15)
```

[152]:

	start time	Total DC input power(kW)	Active power AC(kW)	\
11716	2023-11-30 16:10:00	1.320	1.261	
11717	2023-11-30 16:15:00	1.313	1.241	
11718	2023-11-30 16:20:00	1.211	1.145	
11719	2023-11-30 16:25:00	0.760	0.724	
11720	2023-11-30 16:30:00	0.653	0.578	
11721	2023-11-30 16:35:00	0.630	0.570	
11722	2023-11-30 16:40:00	0.582	0.517	
11723	2023-11-30 16:45:00	0.389	0.319	
11724	2023-11-30 16:50:00	0.199	0.151	
11725	2023-11-30 16:55:00	0.071	0.049	
11726	2023-11-30 17:00:00	0.000	0.000	
11727	2023-11-30 17:05:00	0.000	0.000	
11728	2023-11-30 17:10:00	0.000	0.000	
11729	2023-11-30 17:15:00	0.000	0.000	

```
11730 2023-11-30 17:20:00                    0.000                0.000
```

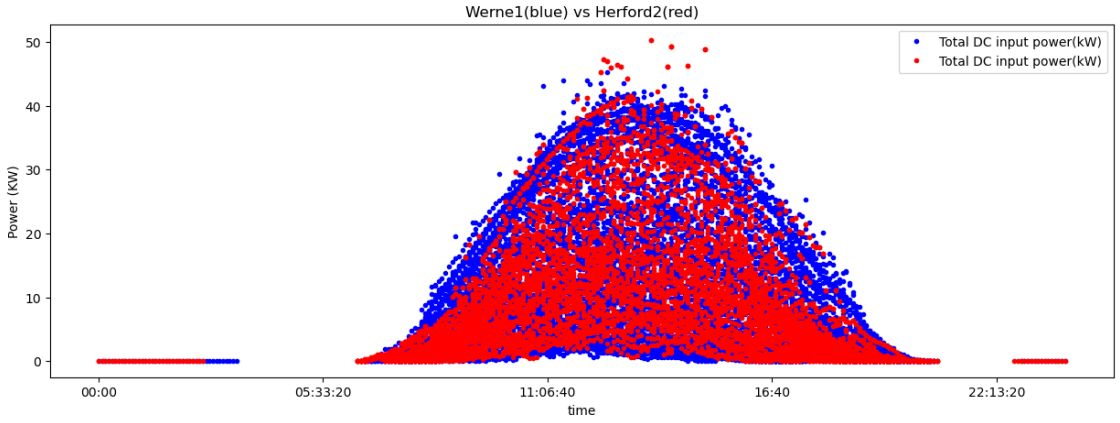
	Production of this day(kWh)	Total yield(kWh)	time	date
11716	86.75	82412.52	16:10:00	2023-11-30
11717	86.86	82412.60	16:15:00	2023-11-30
11718	86.95	82412.68	16:20:00	2023-11-30
11719	87.03	82412.77	16:25:00	2023-11-30
11720	87.08	82412.82	16:30:00	2023-11-30
11721	87.13	82412.89	16:35:00	2023-11-30
11722	87.18	82412.92	16:40:00	2023-11-30
11723	87.21	82412.95	16:45:00	2023-11-30
11724	87.23	82412.97	16:50:00	2023-11-30
11725	87.24	82412.98	16:55:00	2023-11-30
11726	87.24	82413.00	17:00:00	2023-11-30
11727	87.24	82413.01	17:05:00	2023-11-30
11728	87.24	82413.01	17:10:00	2023-11-30
11729	87.24	82413.01	17:15:00	2023-11-30
11730	87.24	82413.01	17:20:00	2023-11-30

[153]:
```
Herford2_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7944 entries, 0 to 11730
Data columns (total 7 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   start time                   7944 non-null   datetime64[ns]
 1   Total DC input power(kW)     7944 non-null   float64
 2   Active power AC(kW)          7944 non-null   float64
 3   Production of this day(kWh)  7944 non-null   float64
 4   Total yield(kWh)             7944 non-null   float64
 5   time                         7944 non-null   object
 6   date                         7944 non-null   datetime64[ns]
dtypes: datetime64[ns](2), float64(4), object(1)
memory usage: 496.5+ KB
```

[154]:
```
#we conpare a dc power of two plant
ax = Werne1_data.plot(x='time', y='Total DC input power(kW)', figsize=(15,5),␣
 ↪legend=True, style='b.')
Herford2_data.plot(x='time', y='Total DC input power(kW)', legend=True,␣
 ↪style='r.',
                   ax=ax
                  )
plt.title('Werne1(blue) vs Herford2(red)')
plt.ylabel('Power (KW)')
```

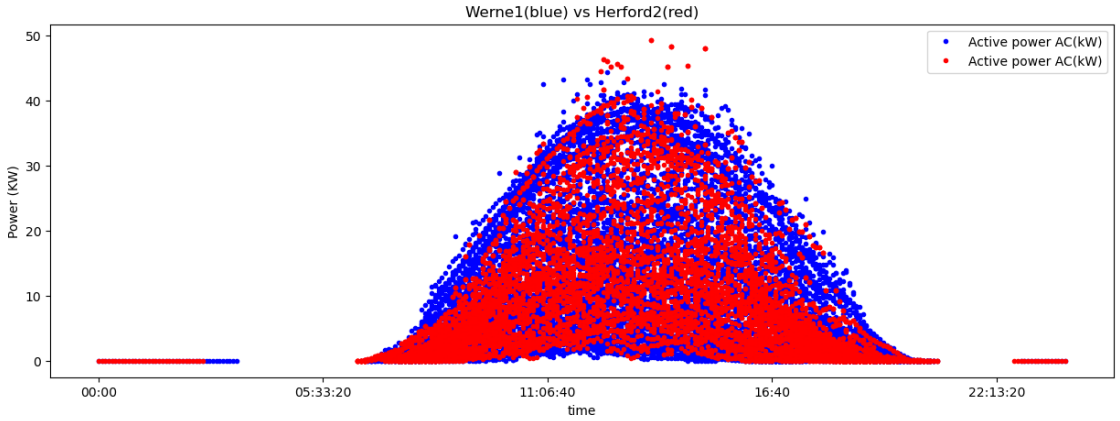
[154]:
```
Text(0, 0.5, 'Power (KW)')
```



Werne and Herford produce direct current. Werne produces approximately 12.5% more direct current per day compared to Herford

[155]:
```
#we conpare a dc power of two plant
ax1 = Werne1_data.plot(x='time', y='Active power AC(kW)', figsize=(15,5),␣
 ↪legend=True, style='b.', )
Herford2_data.plot(x='time', y='Active power AC(kW)', legend=True, style='r.',␣
 ↪ax=ax1)
plt.title('Werne1(blue) vs Herford2(red)')
plt.ylabel('Power (KW)')
```

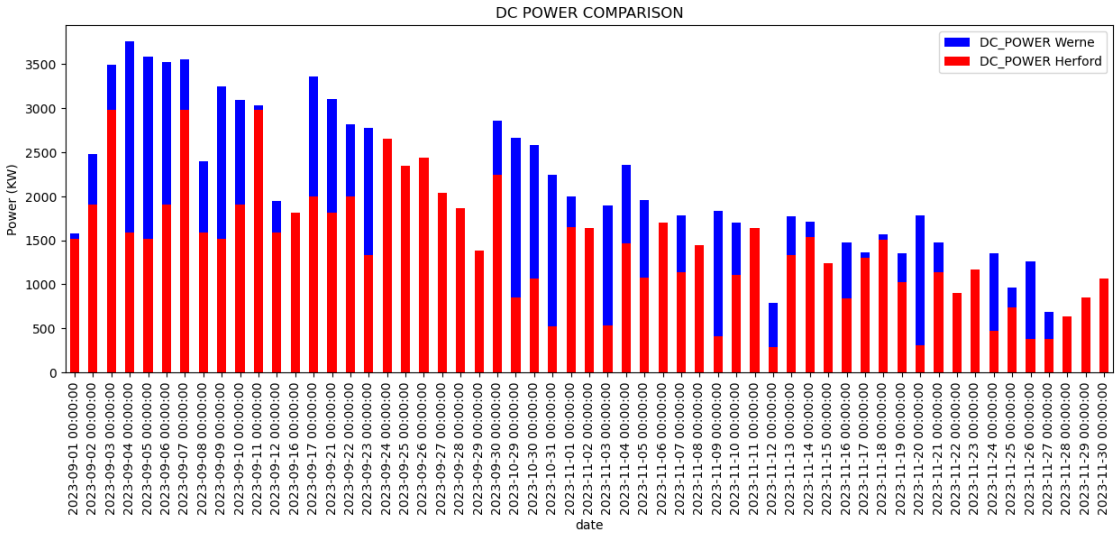
[155]:
```
Text(0, 0.5, 'Power (KW)')
```



Werne ma okolo 12,5% wieksza moc prądu przemiennego dziennego w szczegółowym z Herford

[156]:
```
p2_daily_dc = Herford2_data.groupby('date')['Total DC input power(kW)'].
 ↪agg('sum')
```

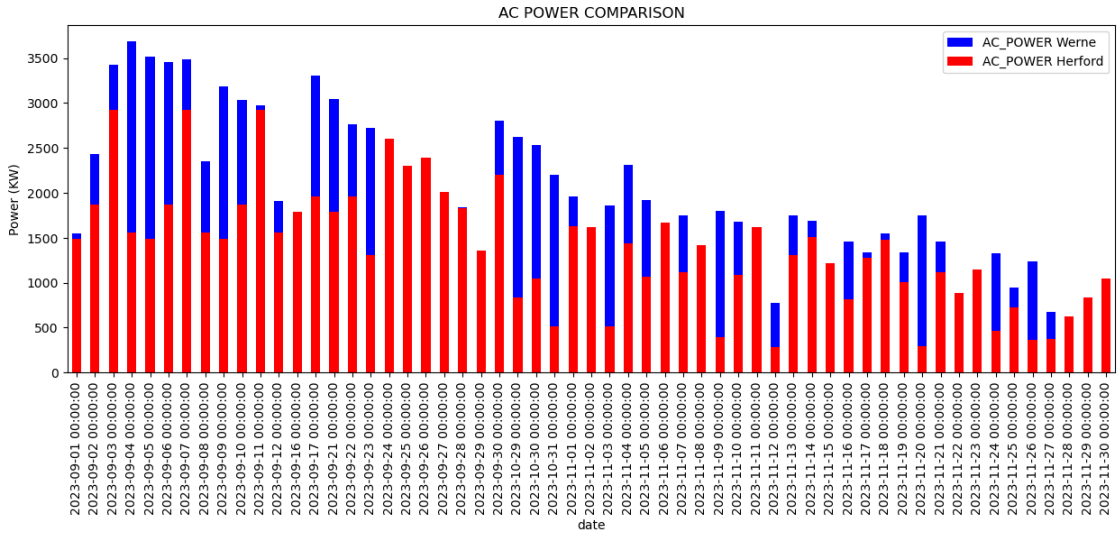
[157]:
```
axh = daily_dc.plot.bar(legend=True, figsize=(15,5), color='Blue',␣
↪label='DC_POWER Werne')
p2_daily_dc.plot.bar(legend=True, color='Red', label='DC_POWER Herford',␣
↪stacked=False)
plt.title('DC POWER COMPARISON')
plt.ylabel('Power (KW)')
plt.show()
```



Each location (Werne, Herford) has daily DC production capacity, but Werne has a higher capacity than Herford. For example, the maximum value of direct current for Werne is above 35000KW and for Herford max. 30000KW

[158]:
```
daily_ac = Werne1_data.groupby('date')['Active power AC(kW)'].agg('sum')
p2_daily_ac = Herford2_data.groupby('date')['Active power AC(kW)'].agg('sum')
```

[159]:
```
ac = daily_ac.plot.bar(legend=True, figsize=(15,5), color='Blue',␣
↪label='AC_POWER Werne')
p2_daily_ac.plot.bar(legend=True, color='Red', label='AC_POWER Herford')
plt.title('AC POWER COMPARISON')
plt.ylabel('Power (KW)')
plt.show()
```

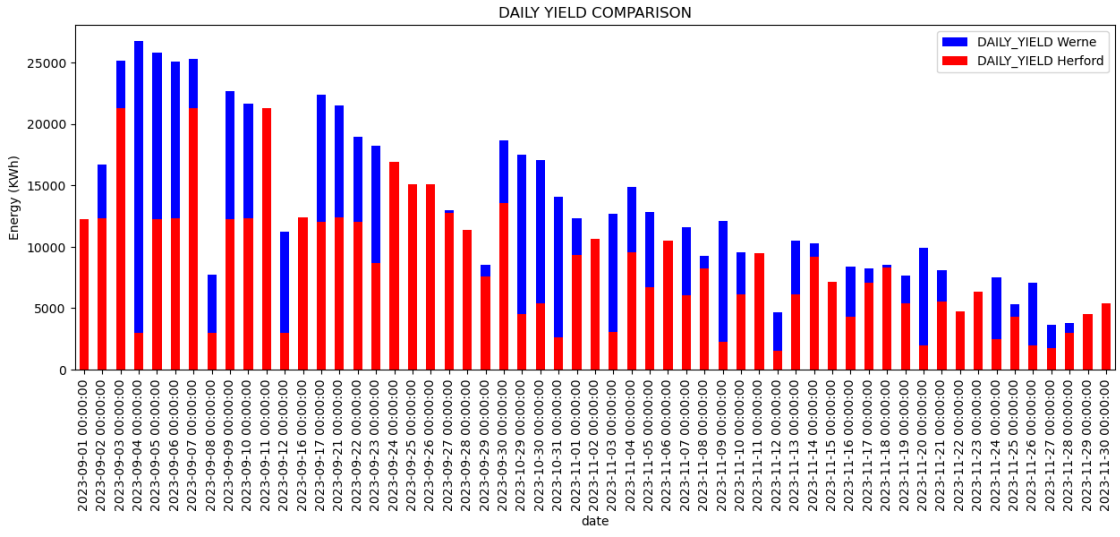


Werne has more capacity than Herford when it comes to generating alternating current for every day.

[160]:
```
#compute daily_yield for each date
p2_dyield = Herford2_data.groupby('date')['Production of this day(kWh)'].
 ↪agg('sum')
```

[161]:
```
dy = dyield.plot.bar(figsize=(15,5), legend=True, label='DAILY_YIELD Werne',␣
 ↪color='Blue')
p2_dyield.plot.bar(legend=True, label='DAILY_YIELD Herford', color='Red')
plt.ylabel('Energy (KWh)')
plt.title('DAILY YIELD COMPARISON')
```

[161]:
```
Text(0.5, 1.0, 'DAILY YIELD COMPARISON')
```

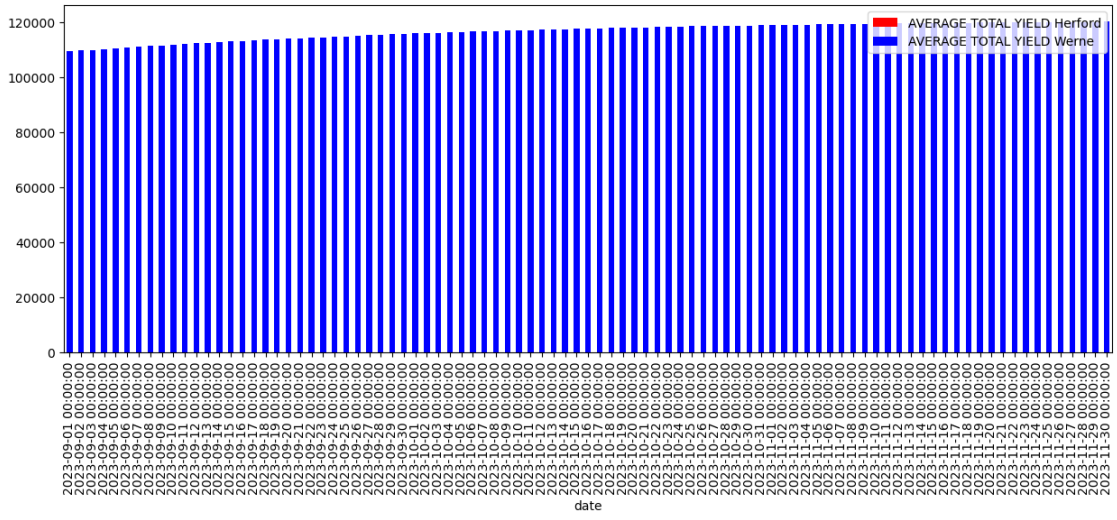


Werne from Hereford has higher daily yields but varies on some days

[162]:
```
#compute a average total_yield for plant I for each day
tyield = Werne1_data.groupby('date')['Total yield(kWh)'].agg('mean')

#compute a average total_yield for plant II for each day
p2_tyield = Herford2_data.groupby('date')['Total yield(kWh)'].agg('mean')
```

[163]:
```
aver = p2_tyield.plot.bar(figsize=(15,5), legend=True, label='AVERAGE TOTAL␣
 ↪YIELD Herford', color='Red')
tyield.plot.bar(legend=True, label='AVERAGE TOTAL YIELD Werne',␣
 ↪color='Blue',ax=aver)
```

[163]:
```
<Axes: xlabel='date'>
```



The gap between average total yield for Herford and average total yield for Werne for each date is very large.

12 Werne weather sensor vs Herford weather sensor

[164]:
```
file3 = 'String-Herford2_Weather_Sensor_Data_Full.csv'
```

[165]:
```
Herford2_sensor = pd.read_csv(file3)
```

[166]:
```
Herford2_sensor.tail()
```

[166]:
```
                start time Site name                         device name  \
11726  2023-11-30 17:00:00   Herford  1020B0083371/2101073871ESL3000191
11727  2023-11-30 17:05:00   Herford  1020B0083371/2101073871ESL3000191
11728  2023-11-30 17:10:00   Herford  1020B0083371/2101073871ESL3000191
11729  2023-11-30 17:15:00   Herford  1020B0083371/2101073871ESL3000191
11730  2023-11-30 17:20:00   Herford  1020B0083371/2101073871ESL3000191

       AMBIENT_TEMPERATURE  Irradiation (Satellite) Wh/m²  MODULE_TEMPERATURE
11726            10.416078                            0.0                34.7
11727            10.416078                            0.0                34.7
11728            10.416078                            0.0                34.1
11729            10.416078                            0.0                33.9
11730            10.416078                            0.0                33.9
```

[167]:
```
Herford2_sensor.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11731 entries, 0 to 11730
Data columns (total 6 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   start time                      11731 non-null  object
 1   Site name                       11731 non-null  object
 2   device name                     11731 non-null  object
 3   AMBIENT_TEMPERATURE             11731 non-null  float64
 4   Irradiation (Satellite) Wh/m²   11731 non-null  float64
 5   MODULE_TEMPERATURE              11731 non-null  float64
dtypes: float64(3), object(3)
memory usage: 550.0+ KB
```

[168]:
```
Herford2_sensor['start time'] = pd.to_datetime(Herford2_sensor['start time'],␣
↪errors='coerce')
```

[169]:
```
# same work cleaning data for plant II
Herford2_sensor['date'] = pd.to_datetime(pd.to_datetime(Herford2_sensor['start␣
↪time']).dt.date)
Herford2_sensor['time'] = pd.to_datetime(Herford2_sensor['start time']).dt.time


del Herford2_sensor['Site name']
del Herford2_sensor['device name']
```

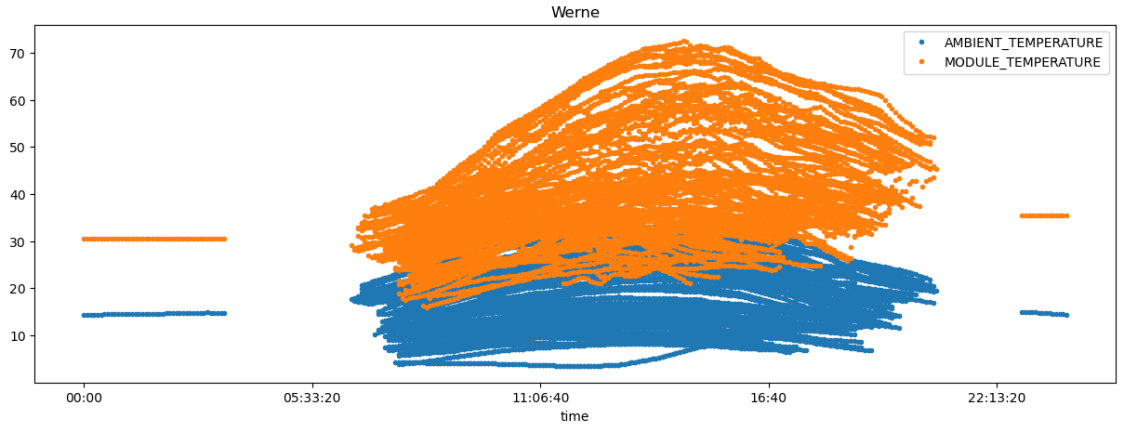
[170]:
```
Herford2_sensor.head()
```

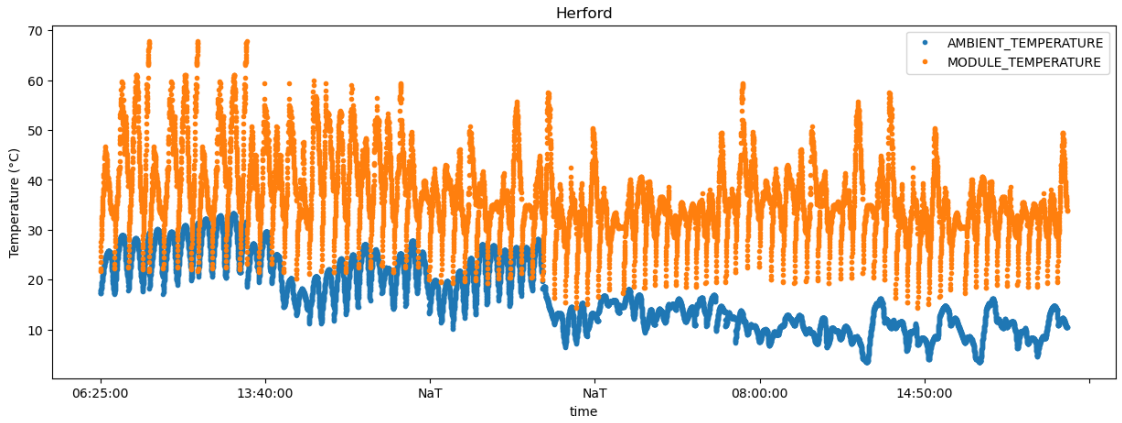
[170]:
```
           start time  AMBIENT_TEMPERATURE  Irradiation (Satellite) Wh/m²  \
0 2023-09-01 06:25:00            17.356912                            0.0
1 2023-09-01 06:30:00            17.356912                            0.0
```

```
2 2023-09-01 06:35:00            17.356912                          0.0
3 2023-09-01 06:40:00            17.493579                          0.0
4 2023-09-01 06:45:00            17.630245                          0.0

   MODULE_TEMPERATURE       date      time
0                21.8 2023-09-01  06:25:00
1                21.7 2023-09-01  06:30:00
2                22.3 2023-09-01  06:35:00
3                23.5 2023-09-01  06:40:00
4                24.7 2023-09-01  06:45:00
```

[171]:
```
Werne1_sensor[['AMBIENT_TEMPERATURE','MODULE_TEMPERATURE','time']].
 ↪plot(x='time', label='Werne', title='Werne', figsize=(15,5), style='.')
Herford2_sensor[['AMBIENT_TEMPERATURE','MODULE_TEMPERATURE','time']].
 ↪plot(x='time', label='Herford', title='Herford', figsize=(15,5), style='.')
plt.ylabel('Temperature (°C)')
```

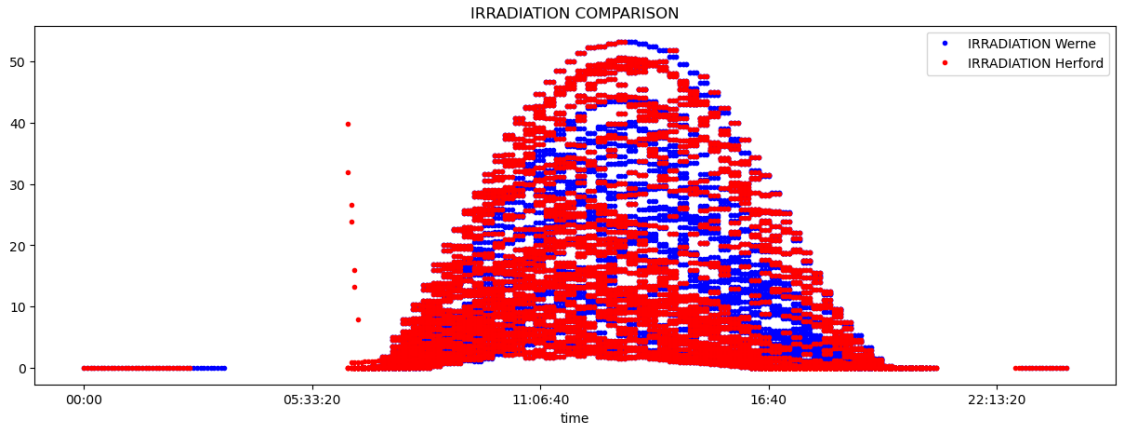
[171]:
```
Text(0, 0.5, 'Temperature (°C)')
```





[172]:
```
# To handle missing values, I use dropna() to remove rows with missing datetime␣
 ↪values:
Herford2_sensor.dropna(subset=['start time'], inplace=True)

#compare IRRADIATION PLANT I VS PLANT II
aq = Werne1_sensor.plot(x='time', y='Irradiation (Satellite) Wh/m²',␣
 ↪legend=True, label='IRRADIATION Werne', color='Blue', style='.',␣
 ↪figsize=(15,5))
Herford2_sensor.plot(x='time', y='Irradiation (Satellite) Wh/m²', legend=True,␣
 ↪label='IRRADIATION Herford',  color='Red', style='.', ax=aq)
plt.title('IRRADIATION COMPARISON')
```

[172]:
```
Text(0.5, 1.0, 'IRRADIATION COMPARISON')
```



Werne and Herford have same IRRADIATION distribution between 05:33:20 and 18:00:00

13 correlation for Herford

[173]:
```
# we are merging our solar power generation data and weather sensor data for␣
 ↪plant 2
sensorData = Herford2_sensor.merge(Herford2_data, left_on='start time',␣
 ↪right_on='start time')
sensorData.head()
```

[173]:

	start time	AMBIENT_TEMPERATURE	Irradiation (Satellite) Wh/m²	\
0	2023-09-01 06:25:00	17.356912	0.0	
1	2023-09-01 06:30:00	17.356912	0.0	
2	2023-09-01 06:35:00	17.356912	0.0	
3	2023-09-01 06:40:00	17.493579	0.0	
4	2023-09-01 06:45:00	17.630245	0.0	

```
   MODULE_TEMPERATURE     date_x    time_x  Total DC input power(kW)  \
0                21.8 2023-09-01  06:25:00                     0.000
1                21.7 2023-09-01  06:30:00                     0.048
2                22.3 2023-09-01  06:35:00                     0.000
3                23.5 2023-09-01  06:40:00                     0.183
4                24.7 2023-09-01  06:45:00                     0.309

   Active power AC(kW)  Production of this day(kWh)  Total yield(kWh)  \
0                0.000                         0.00          74009.40
1                0.031                         0.00          74009.40
2                0.000                         0.01          74009.40
3                0.152                         0.02          74009.44
4                0.252                         0.04          74009.45

     time_y     date_y
0  06:25:00 2023-09-01
1  06:30:00 2023-09-01
2  06:35:00 2023-09-01
3  06:40:00 2023-09-01
4  06:45:00 2023-09-01
```

[174]:
```
#we remove the columns that we do not need
del sensorData['date_x']
del sensorData['date_y']
del sensorData['time_x']
del sensorData['time_y']
```

[175]:
```
sensorData.tail()
```

[175]:
```
              start time  AMBIENT_TEMPERATURE  Irradiation (Satellite) Wh/m²  \
7939 2023-11-30 17:00:00            10.416078                             0.0
7940 2023-11-30 17:05:00            10.416078                             0.0
7941 2023-11-30 17:10:00            10.416078                             0.0
7942 2023-11-30 17:15:00            10.416078                             0.0
7943 2023-11-30 17:20:00            10.416078                             0.0

      MODULE_TEMPERATURE  Total DC input power(kW)  Active power AC(kW)  \
7939                34.7                       0.0                  0.0
7940                34.7                       0.0                  0.0
7941                34.1                       0.0                  0.0
7942                33.9                       0.0                  0.0
7943                33.9                       0.0                  0.0

      Production of this day(kWh)  Total yield(kWh)
7939                        87.24          82413.00
7940                        87.24          82413.01
```

```
7941                        87.24          82413.01
7942                        87.24          82413.01
7943                        87.24          82413.01
```

I create five new feature DELTA_TEMPERATURE, NEW_DAILY_YIELD, NEW_TOTAL_YIELD, NEW_AMBIENT_TEMPERATURE and NEW_MODULE_TEMPERATURE.

delta temperature = ambient temperature - module temperature. All other new variable is just the first derivative in time.

1. New daily yield is the next daily yield - previous daily yield.
2. new total yield is the next total yield - previous total yield
3. new ambient temperature is the next ambient temperature - previous ambient temperature.

and so on, do not forget that it is after 5 min of each daily

[176]:
```
# Rename the new column to 'start time' if needed
sensorData.rename(columns={'Production of this day(kWh)': 'DAILY_YIELD'},␣
↪inplace=True)
sensorData.rename(columns={'Total yield(kWh)': 'TOTAL_YIELD'}, inplace=True)
sensorData.rename(columns={'Active power AC(kW)': 'AC_POWER'}, inplace=True)
sensorData.tail()
```

[176]:
```
              start time  AMBIENT_TEMPERATURE  Irradiation (Satellite) Wh/m²  \
7939 2023-11-30 17:00:00            10.416078                            0.0
7940 2023-11-30 17:05:00            10.416078                            0.0
7941 2023-11-30 17:10:00            10.416078                            0.0
7942 2023-11-30 17:15:00            10.416078                            0.0
7943 2023-11-30 17:20:00            10.416078                            0.0

      MODULE_TEMPERATURE  Total DC input power(kW)  AC_POWER  DAILY_YIELD  \
7939                34.7                       0.0       0.0        87.24
7940                34.7                       0.0       0.0        87.24
7941                34.1                       0.0       0.0        87.24
7942                33.9                       0.0       0.0        87.24
7943                33.9                       0.0       0.0        87.24

      TOTAL_YIELD
7939     82413.00
7940     82413.01
7941     82413.01
7942     82413.01
7943     82413.01
```

[177]:
```
sensorData = sensorData.assign(DELTA_TEMPERATURE = abs(sensorData.
↪MODULE_TEMPERATURE - sensorData.AMBIENT_TEMPERATURE),
                               NEW_DAILY_YIELD = sensorData.DAILY_YIELD.diff(),
                               NEW_TOTAL_YIELD = sensorData.TOTAL_YIELD.diff(),
```

```
                              NEW_AMBIENT_TEMPERATURE = sensorData.
↪AMBIENT_TEMPERATURE.diff(),
                              NEW_MODULE_TEMPERATURE = sensorData.
↪MODULE_TEMPERATURE.diff(),
                              NEW_AC_POWER = sensorData.AC_POWER.diff())
```

[178]:
```
#see
sensorData.head()
```

[178]:

	start time	AMBIENT_TEMPERATURE	Irradiation (Satellite) Wh/m²	\
0	2023-09-01 06:25:00	17.356912	0.0	
1	2023-09-01 06:30:00	17.356912	0.0	
2	2023-09-01 06:35:00	17.356912	0.0	
3	2023-09-01 06:40:00	17.493579	0.0	
4	2023-09-01 06:45:00	17.630245	0.0	

	MODULE_TEMPERATURE	Total DC input power(kW)	AC_POWER	DAILY_YIELD	\
0	21.8	0.000	0.000	0.00	
1	21.7	0.048	0.031	0.00	
2	22.3	0.000	0.000	0.01	
3	23.5	0.183	0.152	0.02	
4	24.7	0.309	0.252	0.04	

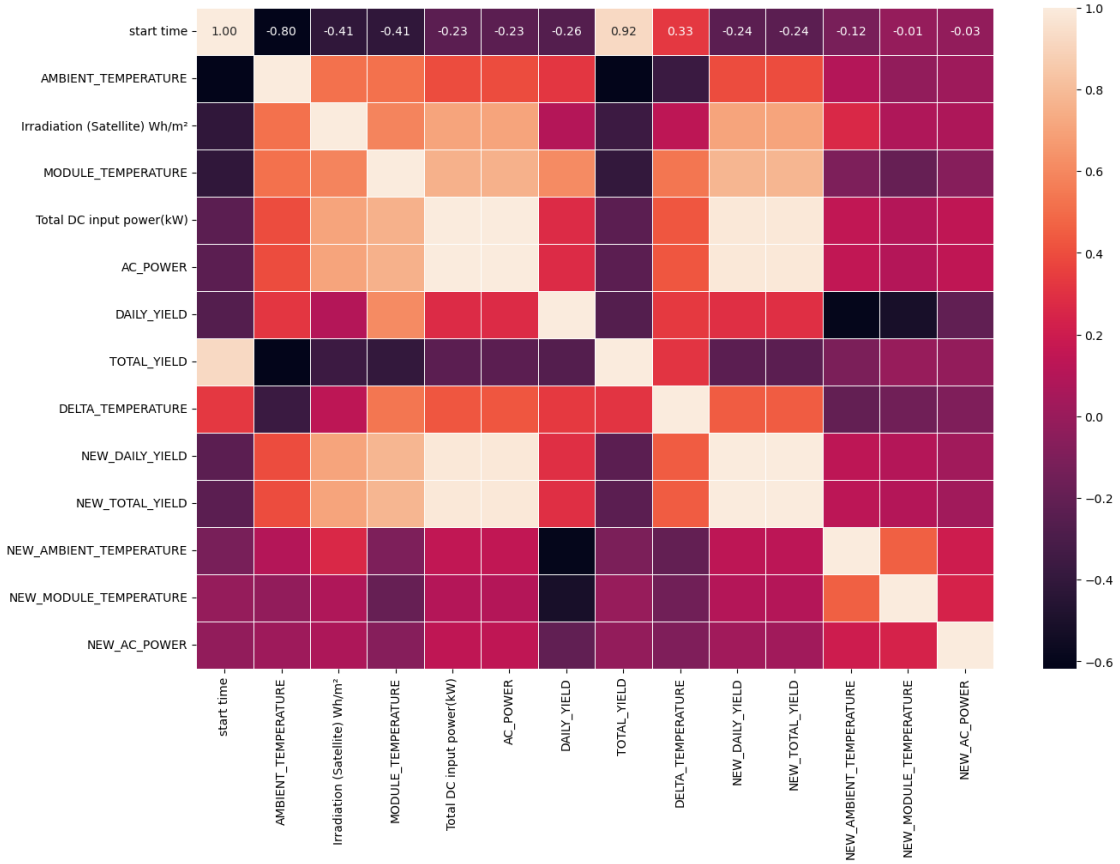
	TOTAL_YIELD	DELTA_TEMPERATURE	NEW_DAILY_YIELD	NEW_TOTAL_YIELD	\
0	74009.40	4.443088	NaN	NaN	
1	74009.40	4.343088	0.00	0.00	
2	74009.40	4.943088	0.01	0.00	
3	74009.44	6.006421	0.01	0.04	
4	74009.45	7.069755	0.02	0.01	

	NEW_AMBIENT_TEMPERATURE	NEW_MODULE_TEMPERATURE	NEW_AC_POWER
0	NaN	NaN	NaN
1	0.000000	-0.1	0.031
2	0.000000	0.6	-0.031
3	0.136667	1.2	0.152
4	0.136667	1.2	0.100

[179]:
```
sensorData.corr(method='spearman').style.background_gradient('viridis')
```

[179]:
```
<pandas.io.formats.style.Styler at 0x7f5c01108450>
```

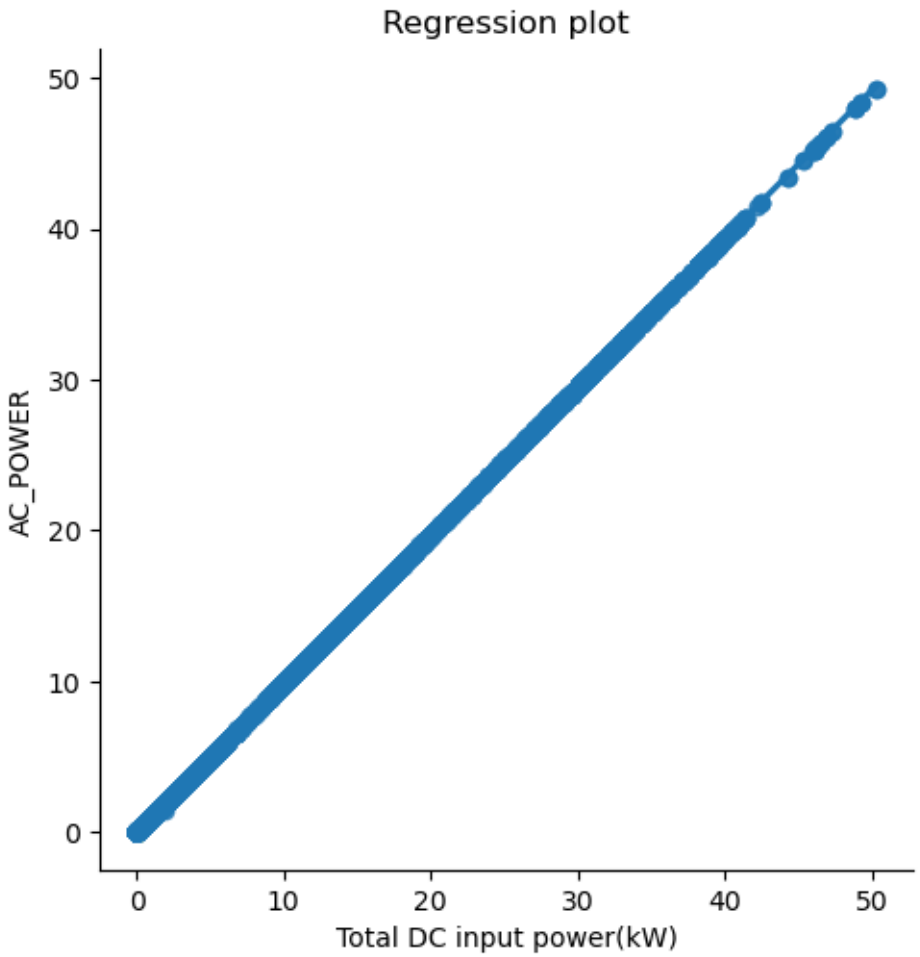
[180]:
```
plt.figure(dpi=100, figsize=(15,10))
sns.heatmap(sensorData.corr(method='spearman'), robust=True, annot=True, fmt='0.
↪2f', linewidths=.5, square=False)
plt.show()
```



In Herford, `TOTAL_YIELD` is opposite with all feature except `DAILY_YIELD`.

[181]:
```
#we plot ac vs dc power
sns.lmplot(x='Total DC input power(kW)', y='AC_POWER', data=sensorData)
plt.title('Regression plot')
```

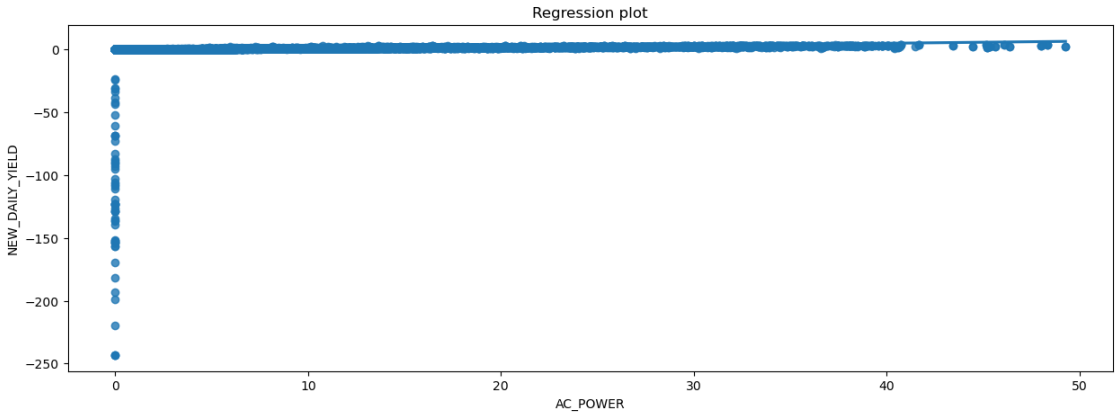
[181]:
```
Text(0.5, 1.0, 'Regression plot')
```



In Herford, dc power = ac power, Inverter lost 0% of the power.

[182]:
```
#we plot New DAILY YIELD vs ac power
plt.figure(dpi=(100), figsize=(15,5))
sns.regplot(x='AC_POWER', y='NEW_DAILY_YIELD', data=sensorData)
plt.title('Regression plot')
```

[182]:
```
Text(0.5, 1.0, 'Regression plot')
```

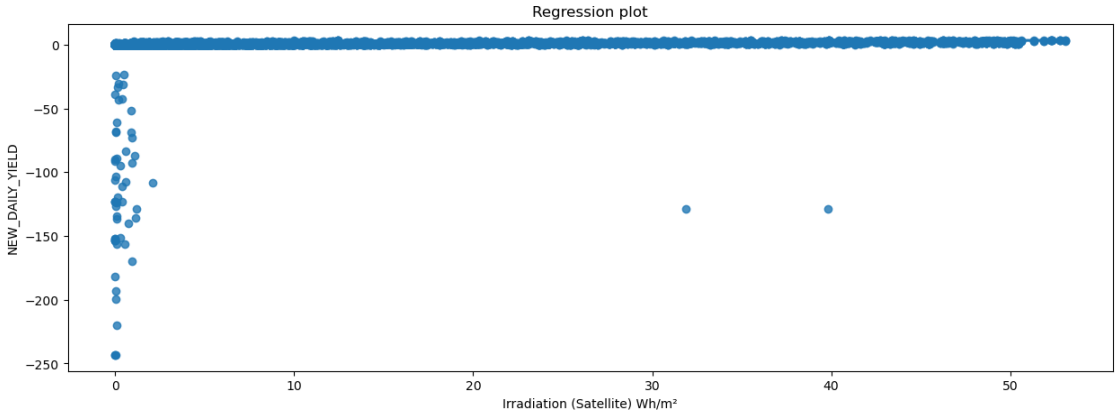


We learn

1. AC_POWER $<$ 10 KW, the NEW_DAILY_YIELD is negative.
2. AC_POWER between 10 KW and 40 KW, NEW_DAILY_YIELD is both positive and negative but more positive
3. AC_POWER $>$ 40 KW is positive.

[183]:
```
#we plot New DAILY YIELD vs IRRADIATION
plt.figure(dpi=(100), figsize=(15,5))
sns.regplot(x='Irradiation (Satellite) Wh/m²', y='NEW_DAILY_YIELD',␣
 ↪data=sensorData)
plt.title('Regression plot')
```

[183]:
```
Text(0.5, 1.0, 'Regression plot')
```



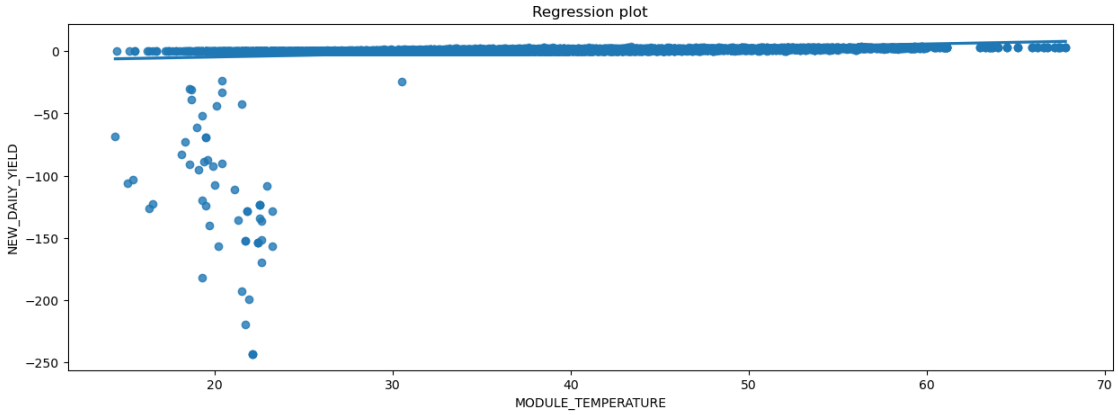
We learn

NEW_DAILY_YIELD are positive and negative along the variation of irradiation 1. Irradiation (Satellite) Wh/m² $<$ 10 KW, the NEW_DAILY_YIELD is negative. 2. Irradiation (Satellite)

Wh/m² between 10 KW and 40 KW, NEW_DAILY_YIELD is both positive and negative but more positive 3. Irradiation (Satellite) Wh/m² > 40 KW is positive.

[184]:
```
#we plot New DAILY YIELD vs ac power
plt.figure(dpi=(100), figsize=(15,5))
sns.regplot(x='MODULE_TEMPERATURE', y='NEW_DAILY_YIELD', data=sensorData)
plt.title('Regression plot')
```

[184]: Text(0.5, 1.0, 'Regression plot')

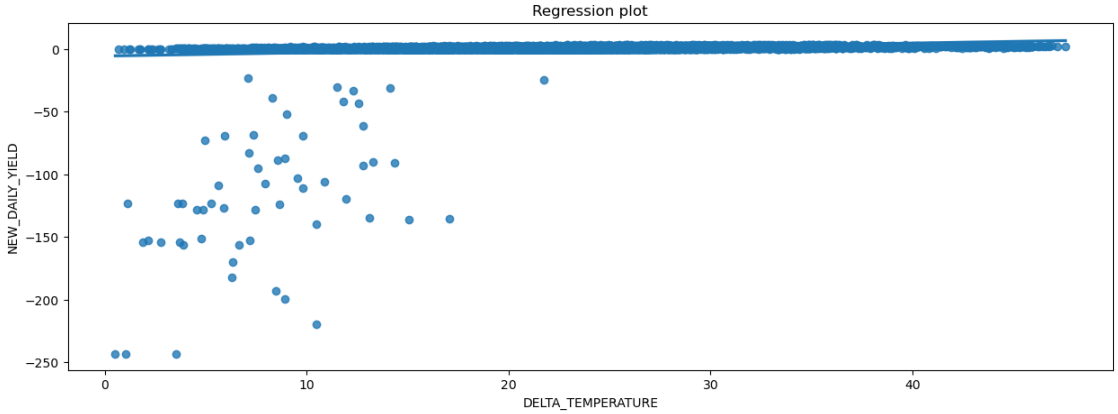


We learn

for MODULE_TEMPERATURE < 30°C, NEW_DAILY_TEMPERATURE is negative. This means that PV panel product the energy if temperature is around 35°C.

[185]:
```
#we plot New DAILY YIELD vs DELTA TEMPERATURE
plt.figure(dpi=(100), figsize=(15,5))
sns.regplot(x='DELTA_TEMPERATURE', y='NEW_DAILY_YIELD', data=sensorData)
plt.title('Regression plot')
```

[185]: Text(0.5, 1.0, 'Regression plot')

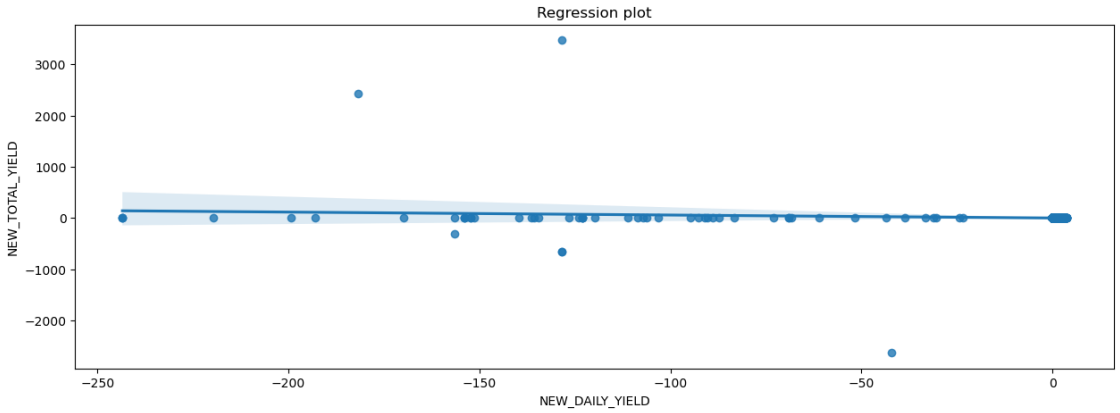


We learn

NEW_DAILY_YIELD is only negative if DELTA_TEMPERATURE < 20°C. This means that daily yield decrease every 5min if the difference temperature between ambient and module temperature is less than 20°C.

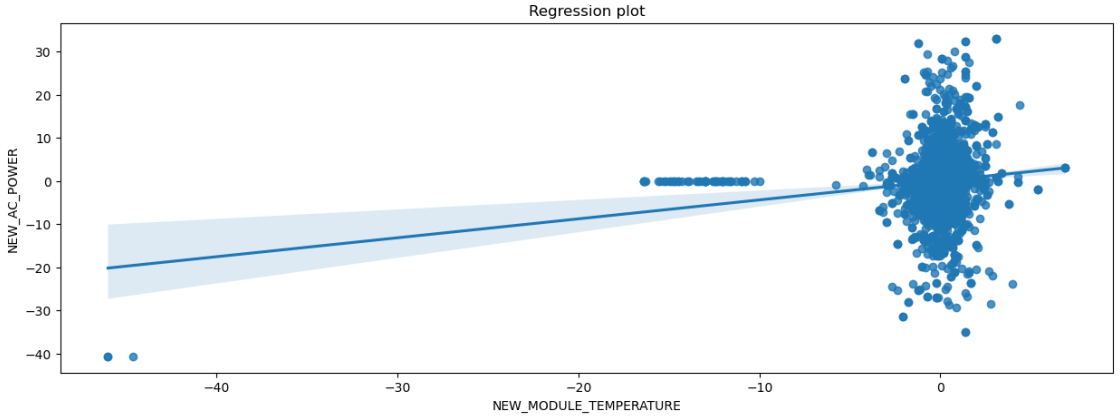
[186]:
```
#we plot New TOTAL YIELD vs New daily yield
plt.figure(dpi=(100), figsize=(15,5))
sns.regplot(y='NEW_TOTAL_YIELD', x='NEW_DAILY_YIELD', data=sensorData)
plt.title('Regression plot')
```

[186]: Text(0.5, 1.0, 'Regression plot')



[187]:
```
#we plot New TOTAL YIELD vs New daily yield
plt.figure(dpi=(100), figsize=(15,5))
sns.regplot(y='NEW_AC_POWER', x='NEW_MODULE_TEMPERATURE', data=sensorData)
plt.title('Regression plot')
```

[187]: Text(0.5, 1.0, 'Regression plot')



New AC Power is the change of previous and next AC Power produced in the time. We have more AC Power only if New Module Temperature is between -5 and 5.

We learn

1. New daily yield decrease total yield decrease. new daily yield increase, new total yield increase.
2. new daily yield is zeros, new total yield is zeros

14 General Conclusion

We compared 2 locations with similar rated capacity: - 32049 Herford_Halberstädter Str 25: **Nennleistung 67,32 kWp** - 59368 Werne Becklohhof 1-7: **Nennleistung 69,3 kWp (Gesamt)**

Throughout this notebook, we can say that Herford is more aginge then Werne. 1. Werne produces 12,5% more DC power than Herford. 2. AC power output is higher for Werne then Herford. 3. The daily yield is bigger for Werne then Herford. 4. The gap between The average total yield for Werne and Herford is very large. To Werne's advantage.

Applies to both: - Both are loses the same percence (%) of it when converting to AC power. - Daily yield decrease if delta temperature is less than 5°C. - Daily yield decrease for some value of AC power.

END.

We provide simple ML analysis to detect location aging in the German market for SOLARIMO. We are simply comparing 2 locations, but we can check more locations to find aging solar modules. Source: Solar Power Machine Learning I