

---

This data has been gathered at two solar power plants over 34 days(From 2020-05-15 to 2020-06-17). It has two pairs of files - each pair has one power energy generation dataset and one weather sensor readings dataset.

## ✓ Exploratory Data Analysis

### ✓ Import needed libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid')
import warnings
import datetime as dt
import matplotlib.dates as mdates
warnings.filterwarnings('ignore')
```

### ✓ Load data of Plant 1

```
#load Plant_1_Generation_Data.csv file in dataframe
plant_gen_1=pd.read_csv('/content/Plant_1_Generation_Data.csv')
#load Plant_1_Weather_Sensor_Data.csv file in dataframe
plant_sens_1= pd.read_csv('/content/Plant_1_Weather_Sensor_Data.csv')

#format datetime
plant_gen_1['DATE_TIME']= pd.to_datetime(plant_gen_1['DATE_TIME'],format='%d-%m-%Y %H:%M')
plant_sens_1['DATE_TIME']= pd.to_datetime(plant_sens_1['DATE_TIME'],format='%Y-%m-%d %H:%M:%S')
```

```
#check number of columns and rows for plant 1 generation dataframe
plant_gen_1.shape

(68778, 7)
```

```
#check number of columns and rows for plant 1 weather dataframe
plant_sens_1.shape

(3182, 6)
```

## ✓ Load data of Plant 2

```
#load Plant_2_Generation_Data.csv file in dataframe
plant_gen_2=pd.read_csv('/content/Plant_2_Generation_Data.csv')
#load Plant_2_Weather_Sensor_Data.csv file in dataframe
plant_sens_2= pd.read_csv('/content/Plant_2_Weather_Sensor_Data.csv')

#format datetime
plant_gen_2['DATE_TIME']= pd.to_datetime(plant_gen_2['DATE_TIME'],format='%Y-%m-%d %H:%M:%S')
plant_sens_2['DATE_TIME']= pd.to_datetime(plant_sens_2['DATE_TIME'],format='%Y-%m-%d %H:%M:%S')

#check number of columns and rows for plant 2 generation dataframe
plant_gen_2.shape

(67698, 7)

#check number of columns and rows for plant 1 weather dataframe
plant_sens_2.shape

(3259, 6)
```

## ✓ Data cleaning

## ✓ Plant\_1\_Generation\_Data

### ✓ Display last 5 rows of Plant\_1\_Generation\_Data

```
#display last rows of Plant 1 generation dataframe  
plant_gen_1.tail()
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
<b>68773</b>	2020-06-17 23:45:00	4135001	uHbuxQJl8IW7ozc	0.0	0.0	5967.000	7287002.0
<b>68774</b>	2020-06-17 23:45:00	4135001	wCURE6d3bPkepu2	0.0	0.0	5147.625	7028601.0
<b>68775</b>	2020-06-17 23:45:00	4135001	z9Y9gH1T5YWrNuG	0.0	0.0	5819.000	7251204.0
<b>68776</b>	2020-06-17 23:45:00	4135001	zBlq5rxdHJRwDNY	0.0	0.0	5817.000	6583369.0
<b>68777</b>	2020-06-17 23:45:00	4135001	zVJPv84UY57bAof	0.0	0.0	5910.000	7363272.0

```
#check information like data type and missing values for plant 1 generation dataframe  
plant_gen_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 68778 entries, 0 to 68777  
Data columns (total 7 columns):  
#   Column          Non-Null Count  Dtype  
---  -  
0   DATE_TIME       68778 non-null  datetime64[ns]  
1   PLANT_ID        68778 non-null  int64  
2   SOURCE_KEY      68778 non-null  object  
3   DC_POWER        68778 non-null  float64  
4   AC_POWER        68778 non-null  float64  
5   DAILY_YIELD     68778 non-null  float64  
6   TOTAL_YIELD     68778 non-null  float64  
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)  
memory usage: 3.7+ MB
```

```
# Count the number of solar panel at plant 1
num_solar_panel = plant_gen_1['SOURCE_KEY'].nunique()
print("Number of Solar Panel at Plant 1 was : ", num_solar_panel)
```

Number of Solar Panel at Plant 1 was : 22

```
# Add column of plant 1 Efficiency to dataframe
plant_gen_1['Efficiency']= plant_gen_1.AC_POWER * 10 / plant_gen_1.DC_POWER
plant_gen_1.head()
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	Efficiency
0	2020-05-15	4135001	1BY6WEcLGh8j5v7	0.0	0.0	0.0	6259559.0	NaN
1	2020-05-15	4135001	1IF53ai7Xc0U56Y	0.0	0.0	0.0	6183645.0	NaN
2	2020-05-15	4135001	3PZuoBAID5Wc2HD	0.0	0.0	0.0	6987759.0	NaN
3	2020-05-15	4135001	7JYdWkrLSPkdwr4	0.0	0.0	0.0	7602960.0	NaN
4	2020-05-15	4135001	McdE0feGgRqW7Ca	0.0	0.0	0.0	7158964.0	NaN

```
#Check for statical description of plant 1 generation dataframe
plant_gen_1.describe()
```

	PLANT_ID	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	Efficiency
<b>count</b>	68778.0	68778.000000	68778.000000	68778.000000	6.877800e+04	36827.000000
<b>mean</b>	4135001.0	3147.426211	307.802752	3295.968737	6.978712e+06	0.977189
<b>std</b>	0.0	4036.457169	394.396439	3145.178309	4.162720e+05	0.004583
<b>min</b>	4135001.0	0.000000	0.000000	0.000000	6.183645e+06	0.955518
<b>25%</b>	4135001.0	0.000000	0.000000	0.000000	6.512003e+06	0.975790
<b>50%</b>	4135001.0	429.000000	41.493750	2658.714286	7.146685e+06	0.978451
<b>75%</b>	4135001.0	6366.964286	623.618750	6274.000000	7.268706e+06	0.980144
<b>max</b>	4135001.0	14471.125000	1410.950000	9163.000000	7.846821e+06	1.065922

we can observe the maximum generated DC\_Power was 14471.125kwh and the maximum AC\_Power were 9163kwh, with mean efficiency of 0.977

```
#Check the time when Plant 1 generated the maximum DC_Power
DC_power_counts = plant_gen_1['DC_POWER'].groupby(plant_gen_1['DATE_TIME']).size()
max_DC_Power_time = DC_power_counts.idxmax()
print("The maximum DC_Power generated by Plant 1 was on : ",max_DC_Power_time)
```

```
The maximum DC_Power generated by Plant 1 was on : 2020-05-15 01:00:00
```

**Observation:** Plant 1 with ID 4135001 has data that have 7 columns and 68778 rows which has no null values, and it has 22 solar panels which generates energy, where the maximum DC\_Power is 14471.125 KWh generated on 2020-05-15 at 1:00 PM sharp and the average converted AC\_Power is 307.8 KWh with mean efficiency of 97.7%

✓ Separate DATE\_TIME into date and time columns for Plant 1 dataset, then only keep necessary columns

```
#Aggregate the data recorded by date and time with interval of 15 minutes, Then separate the DATE_TIME into time and date c
plant1_energy=plant_gen_1
plant1_energy = plant1_energy.groupby('DATE_TIME')[['DC_POWER','AC_POWER','DAILY_YIELD','TOTAL_YIELD', 'Efficiency']].agg('
plant1_energy = plant1_energy.reset_index()
plant1_energy['DATE_TIME'] = pd.to_datetime(plant1_energy['DATE_TIME'], errors='coerce')
plant1_energy['time'] = plant1_energy['DATE_TIME'].dt.time
plant1_energy['date'] = pd.to_datetime(plant1_energy['DATE_TIME']).dt.date)

#Display the last 5 rows of new dataframe of Plant 1 data with 15min interval
plant1_energy.tail()
```

	DATE_TIME	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	Efficiency	time	date
<b>3153</b>	2020-06-17 22:45:00	0.0	0.0	129571.000000	156142755.0	0.0	22:45:00	2020-06-17
<b>3154</b>	2020-06-17 23:00:00	0.0	0.0	129571.000000	156142755.0	0.0	23:00:00	2020-06-17
<b>3155</b>	2020-06-17 23:15:00	0.0	0.0	129571.000000	156142755.0	0.0	23:15:00	2020-06-17
<b>3156</b>	2020-06-17 23:30:00	0.0	0.0	129571.000000	156142755.0	0.0	23:30:00	2020-06-17
<b>3157</b>	2020-06-17 23:45:00	0.0	0.0	127962.767857	156142755.0	0.0	23:45:00	2020-06-17

## ✓ Plant\_1\_Weather\_Sensor\_Data

## ✓ Display 5 last rows of Plant\_1\_Weather\_Sensor\_Data

```
#show last rows of weather data for Plant 1
plant_sens_1.tail()
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
<b>3177</b>	2020-06-17 22:45:00	4135001	HmiyD2TTLFNqkNe	22.150570	21.480377	0.0
<b>3178</b>	2020-06-17 23:00:00	4135001	HmiyD2TTLFNqkNe	22.129816	21.389024	0.0
<b>3179</b>	2020-06-17 23:15:00	4135001	HmiyD2TTLFNqkNe	22.008275	20.709211	0.0
<b>3180</b>	2020-06-17 23:30:00	4135001	HmiyD2TTLFNqkNe	21.969495	20.734963	0.0
<b>3181</b>	2020-06-17 23:45:00	4135001	HmiyD2TTLFNqkNe	21.909288	20.427972	0.0

```
#check information regarding null values, columns, rows and data type for plant 1 weather data
plant_sens_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3182 entries, 0 to 3181
Data columns (total 6 columns):
#   Column                Non-Null Count  Dtype
---  -
0   DATE_TIME              3182 non-null   datetime64[ns]
1   PLANT_ID               3182 non-null   int64
2   SOURCE_KEY             3182 non-null   object
3   AMBIENT_TEMPERATURE    3182 non-null   float64
4   MODULE_TEMPERATURE     3182 non-null   float64
5   IRRADIATION            3182 non-null   float64
dtypes: datetime64[ns](1), float64(3), int64(1), object(1)
memory usage: 149.3+ KB
```

Plant 1 weather data has 6 columns, 3182 rows with non-null values

```
#check for statistical description of the dataframe
plant_sens_1.describe()
```

	PLANT_ID	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
<b>count</b>	3182.0	3182.000000	3182.000000	3182.000000
<b>mean</b>	4135001.0	25.531606	31.091015	0.228313
<b>std</b>	0.0	3.354856	12.261222	0.300836
<b>min</b>	4135001.0	20.398505	18.140415	0.000000
<b>25%</b>	4135001.0	22.705182	21.090553	0.000000
<b>50%</b>	4135001.0	24.613814	24.618060	0.024653
<b>75%</b>	4135001.0	27.920532	41.307840	0.449588
<b>max</b>	4135001.0	35.252486	65.545714	1.221652

**Observation:** Plant 1 with ID 4135001 had one weather sensor which recorded data with 6 columns and 3182 rows with no null values, the maximum ambient temperature recorded was 35.25C and module temperature were 65.54C

- ✓ Separate DATE\_TIME into date and time columns, then delete Source\_key as it is the same for all rows of Plant 1 weather data

```
#split the DATE_TIME column into time and date columns for the weater dataframe
plant1_weather=plant_sens_1
plant1_weather['DATE_TIME'] = pd.to_datetime(plant1_weather['DATE_TIME'], errors='coerce')
plant1_weather['date'] = pd.to_datetime(pd.to_datetime(plant1_weather['DATE_TIME']).dt.date)
plant1_weather['time'] = pd.to_datetime(plant1_weather['DATE_TIME']).dt.time
# drop the source_key column
del plant1_weather['SOURCE_KEY']
```

```
#display the last rows of the new Plant 1 weather dataframe
plant1_weather.tail()
```



	DATE_TIME	PLANT_ID	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	date	time
<b>3177</b>	2020-06-17 22:45:00	4135001	22.150570	21.480377	0.0	2020-06-17	22:45:00
<b>3178</b>	2020-06-17 23:00:00	4135001	22.129816	21.389024	0.0	2020-06-17	23:00:00
<b>3179</b>	2020-06-17 23:15:00	4135001	22.008275	20.709211	0.0	2020-06-17	23:15:00
<b>3180</b>	2020-06-17 23:30:00	4135001	21.969495	20.734963	0.0	2020-06-17	23:30:00
<b>3181</b>	2020-06-17 23:45:00	4135001	21.909288	20.427972	0.0	2020-06-17	23:45:00

## ✓ Merge both datasets of Plant\_1\_Generation\_Data and Plant\_1\_Weather\_Sensor\_Data

```
#Concatnate columns data of weather dataframe to generated dataframe
plant1_data = plant1_weather.merge(plant1_energy, left_on='DATE_TIME', right_on='DATE_TIME')
#drop the date and time columns as there are duplicate
del plant1_data['date_x']
del plant1_data['time_x']
```

## ✓ Diplay 5 first rows of plant 1 data combined power generation and weather sensor

```
#display first rows of plant 1 dataframe(Generation and Weather data combined)
plant1_data.head()
```

	DATE_TIME	PLANT_ID	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YI
0	2020-05-15 00:00:00	4135001	25.184316	22.857507	0.0	0.0	0.0	0.0	14358167
1	2020-05-15 00:15:00	4135001	25.084589	22.761668	0.0	0.0	0.0	0.0	14358167



## ✓ Plant\_2\_Generation\_Data

### ✓ Display 5 first rows of Plant 2 Generation Data

```
#display first rows of Plant 2 generation data
plant_gen_2.head()
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	2020-05-15	4136001	4UPUqMRk7TRMgml	0.0	0.0	9425.000000	2.429011e+06
1	2020-05-15	4136001	81aHJ1q11NBPMrL	0.0	0.0	0.000000	1.215279e+09
2	2020-05-15	4136001	9kRcWv60rDACzjR	0.0	0.0	3075.333333	2.247720e+09
3	2020-05-15	4136001	Et9kgGMDI729KT4	0.0	0.0	269.933333	1.704250e+06
4	2020-05-15	4136001	IQ2d7wF4YD8zU1Q	0.0	0.0	3177.000000	1.994153e+07

```
#check infoormation of the dataframe
plant_gen_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 67698 entries, 0 to 67697
```

```
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   DATE_TIME    67698 non-null    datetime64[ns]
1   PLANT_ID      67698 non-null    int64
2   SOURCE_KEY    67698 non-null    object
3   DC_POWER      67698 non-null    float64
4   AC_POWER      67698 non-null    float64
5   DAILY_YIELD   67698 non-null    float64
6   TOTAL_YIELD   67698 non-null    float64
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 3.6+ MB
```

```
# Count the number of solar panel at plant 2
num_solar_panel2 = plant_gen_2['SOURCE_KEY'].nunique()
print("Number of Solar Panel at Plant 2 was : ", num_solar_panel2)
```

Number of Solar Panel at Plant 2 was : 22

```
#Add Efficiency column to Plant 2 generation dataframe
plant_gen_2['Efficiency']= plant_gen_2.AC_POWER / plant_gen_2.DC_POWER
plant_gen_2.head()
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	Efficiency
0	2020-05-15	4136001	4UPUqMRk7TRMgml	0.0	0.0	9425.000000	2.429011e+06	NaN
1	2020-05-15	4136001	81aHJ1q11NBPMrL	0.0	0.0	0.000000	1.215279e+09	NaN
2	2020-05-15	4136001	9kRcWv60rDACzjR	0.0	0.0	3075.333333	2.247720e+09	NaN
3	2020-05-15	4136001	Et9kgGMDI729KT4	0.0	0.0	269.933333	1.704250e+06	NaN
4	2020-05-15	4136001	IQ2d7wF4YD8zU1Q	0.0	0.0	3177.000000	1.994153e+07	NaN

```
#check for statistical description of the dataframe
plant_gen_2.describe()
```

	PLANT_ID	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	Efficiency
<b>count</b>	67698.0	67698.000000	67698.000000	67698.000000	6.769800e+04	32036.000000
<b>mean</b>	4136001.0	246.701961	241.277825	3294.890295	6.589448e+08	0.976806
<b>std</b>	0.0	370.569597	362.112118	2919.448386	7.296678e+08	0.005020
<b>min</b>	4136001.0	0.000000	0.000000	0.000000	0.000000e+00	0.912790
<b>25%</b>	4136001.0	0.000000	0.000000	272.750000	1.996494e+07	0.975014
<b>50%</b>	4136001.0	0.000000	0.000000	2911.000000	2.826276e+08	0.978432
<b>75%</b>	4136001.0	446.591667	438.215000	5534.000000	1.348495e+09	0.980247
<b>max</b>	4136001.0	1420.933333	1385.420000	9873.000000	2.247916e+09	1.008320

**Observation:** Plant 2 has ID 4136001 with data of 7 columns and 67698 rows with no null values, generated maximum DC\_Power was 1420.93Kwh with AC\_Power of 1385.42Kwh with mean efficiency of 97.6%

✓ Separate DATE\_TIME into date and time columns, then delete Source\_key as it is the same for all rows

```
#split DATE_TIME column into time and date
plant2_energy=plant_gen_2
plant2_energy = plant2_energy.groupby('DATE_TIME')[['DC_POWER', 'AC_POWER', 'DAILY_YIELD', 'TOTAL_YIELD', 'Efficiency']].agg('
plant2_energy = plant2_energy.reset_index()
plant2_energy['DATE_TIME'] = pd.to_datetime(plant2_energy['DATE_TIME'], errors='coerce')
plant2_energy['time'] = plant2_energy['DATE_TIME'].dt.time
plant2_energy['date'] = pd.to_datetime(plant2_energy['DATE_TIME'].dt.date)

#display last rows of Plant 2 generation dataframe with time and data columns
plant2_energy.tail()
```

	DATE_TIME	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	Efficiency	time	date
<b>3254</b>	2020-06-17 22:45:00	0.0	0.0	93040.0	1.419408e+10	0.0	22:45:00	2020-06-17
<b>3255</b>	2020-06-17 23:00:00	0.0	0.0	93040.0	1.419408e+10	0.0	23:00:00	2020-06-17
<b>3256</b>	2020-06-17 23:15:00	0.0	0.0	93040.0	1.419408e+10	0.0	23:15:00	2020-06-17
<b>3257</b>	2020-06-17 23:30:00	0.0	0.0	93040.0	1.419408e+10	0.0	23:30:00	2020-06-17
<b>3258</b>	2020-06-17 23:45:00	0.0	0.0	93040.0	1.419408e+10	0.0	23:45:00	2020-06-17

## ✓ Plant\_2\_Weather\_Sensor\_Data

## ✓ Display 5 last rows of Plant\_2\_Weather\_Sensor\_Data

```
#display last rows of weather data for Plant 2
plant_sens_2.tail()
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
<b>3254</b>	2020-06-17 22:45:00	4136001	iq8k7ZNt4Mwm3w0	23.511703	22.856201	0.0
<b>3255</b>	2020-06-17 23:00:00	4136001	iq8k7ZNt4Mwm3w0	23.482282	22.744190	0.0
<b>3256</b>	2020-06-17 23:15:00	4136001	iq8k7ZNt4Mwm3w0	23.354743	22.492245	0.0
<b>3257</b>	2020-06-17 23:30:00	4136001	iq8k7ZNt4Mwm3w0	23.291048	22.373909	0.0
<b>3258</b>	2020-06-17 23:45:00	4136001	iq8k7ZNt4Mwm3w0	23.202871	22.535908	0.0

```
#check information about plant 2 weather dataframe
plant_sens_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3259 entries, 0 to 3258
```

```
Data columns (total 6 columns):
#   Column                Non-Null Count  Dtype
---  -
0   DATE_TIME              3259 non-null  datetime64[ns]
1   PLANT_ID                3259 non-null  int64
2   SOURCE_KEY              3259 non-null  object
3   AMBIENT_TEMPERATURE     3259 non-null  float64
4   MODULE_TEMPERATURE      3259 non-null  float64
5   IRRADIATION             3259 non-null  float64
dtypes: datetime64[ns](1), float64(3), int64(1), object(1)
memory usage: 152.9+ KB
```

```
#check statistical description of Plant 2 weather dataframe
plant_sens_2.describe()
```

	PLANT_ID	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
<b>count</b>	3259.0	3259.000000	3259.000000	3259.000000
<b>mean</b>	4136001.0	28.069400	32.772408	0.232737
<b>std</b>	0.0	4.061556	11.344034	0.312693
<b>min</b>	4136001.0	20.942385	20.265123	0.000000
<b>25%</b>	4136001.0	24.602135	23.716881	0.000000
<b>50%</b>	4136001.0	26.981263	27.534606	0.019040
<b>75%</b>	4136001.0	31.056757	40.480653	0.438717
<b>max</b>	4136001.0	39.181638	66.635953	1.098766

**Observation:** Plant 2 with ID 41365001 had one weather sensor which recorded data with 6 columns and 3259 rows with no null values, the maximum ambient temperature recrded was 39.18C and module temperature were 66.63C

✓ Separate DATE\_TIME into date and time columns, then delete Source\_key as it is the same for all rows

```
#split DATE_TIME column into time and date
plant2_weather=plant_sens_2
plant2_weather['DATE_TIME'] = pd.to_datetime(plant2_weather['DATE_TIME'], errors='coerce')
plant2_weather['date'] = pd.to_datetime(pd.to_datetime(plant2_weather['DATE_TIME']).dt.date)
plant2_weather['time'] = pd.to_datetime(plant2_weather['DATE_TIME']).dt.time
# drop the source key column
del plant2_weather['SOURCE_KEY']

#display last rows of new Plant 2 weather dataframe
plant2_weather.tail()
```

	DATE_TIME	PLANT_ID	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	date	time
<b>3254</b>	2020-06-17 22:45:00	4136001	23.511703	22.856201	0.0	2020-06-17	22:45:00
<b>3255</b>	2020-06-17 23:00:00	4136001	23.482282	22.744190	0.0	2020-06-17	23:00:00
<b>3256</b>	2020-06-17 23:15:00	4136001	23.354743	22.492245	0.0	2020-06-17	23:15:00
<b>3257</b>	2020-06-17 23:30:00	4136001	23.291048	22.373909	0.0	2020-06-17	23:30:00
<b>3258</b>	2020-06-17 23:45:00	4136001	23.202871	22.535908	0.0	2020-06-17	23:45:00


## ✓ Merge both datasets of Plant\_1\_Generation\_Data and Plant\_1\_Weather\_Sensor\_Data

```
#Concatenate both dataset of Plant 2 generation and weather and drop duplicated columns(time and date)
plant2_data = plant2_weather.merge(plant2_energy, left_on='DATE_TIME', right_on='DATE_TIME')
del plant2_data['date_x']
del plant2_data['time_x']
```

## ✓ Display 5 first rows of plant 2 data combined power generation and weather sensor

```
#display the first rows of Plant 2 combined data
plant2_data.head()
```

	DATE_TIME	PLANT_ID	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_Y
0	2020-05-15 00:00:00	4136001	27.004764	25.060789	0.0	0.0	0.0	48899.938095	1.418960
1	2020-05-15 00:15:00	4136001	26.880811	24.421869	0.0	0.0	0.0	28401.000000	1.418960



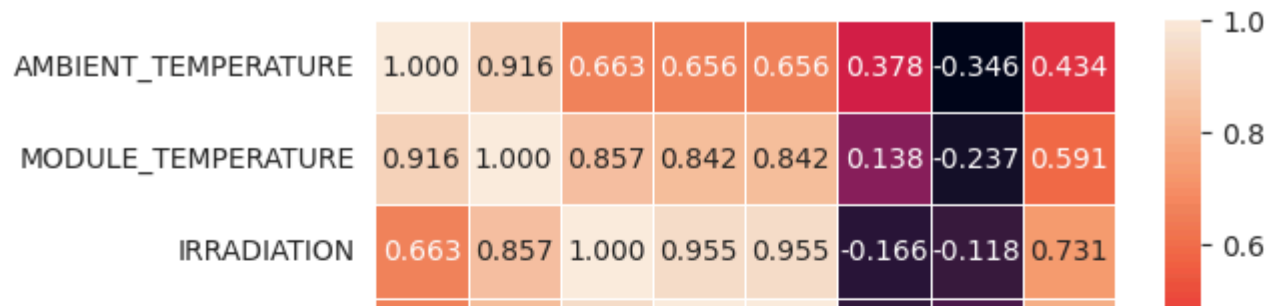
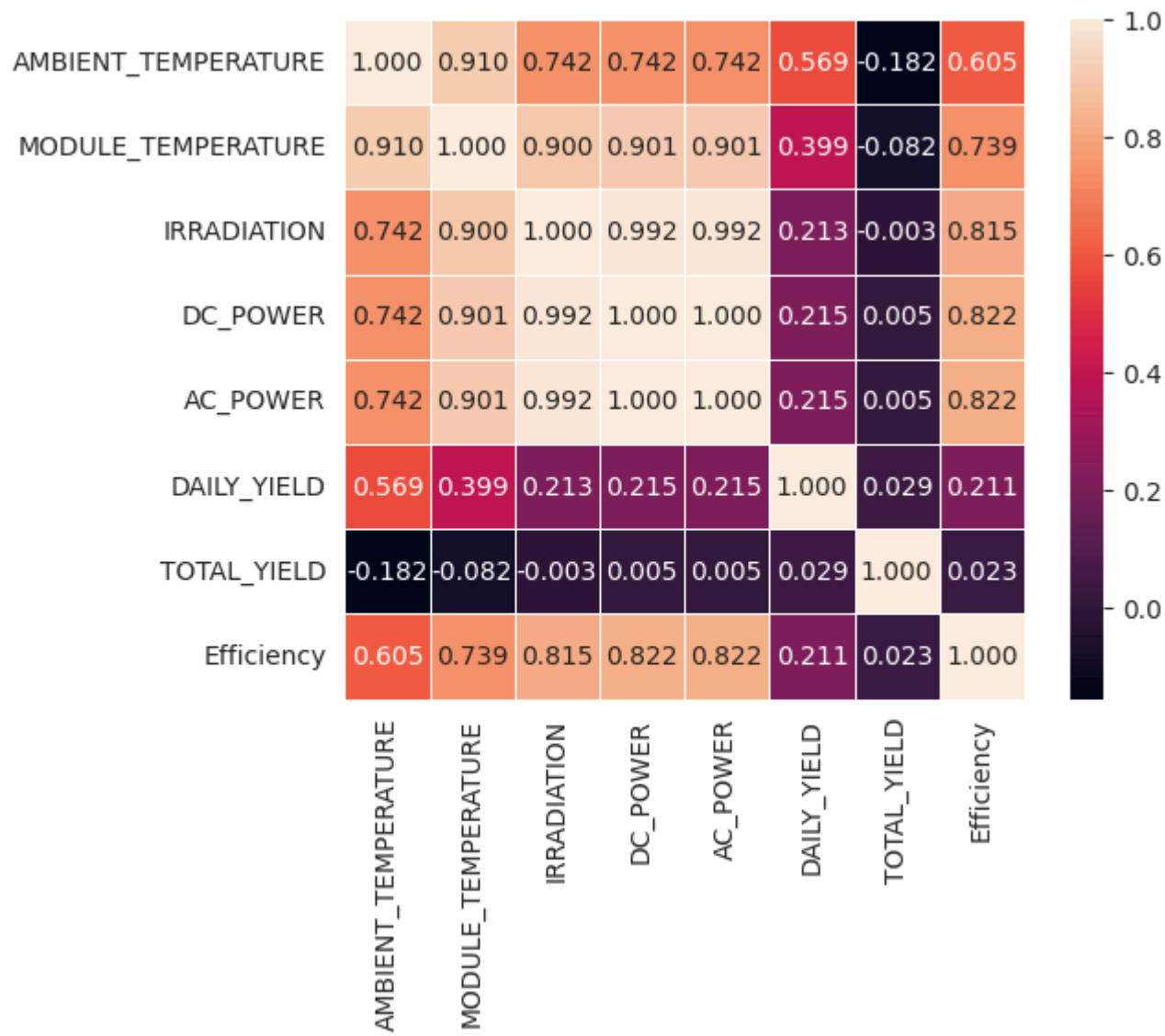
## ✓ Plant 1 and Plant 2 Comparisons of power generation

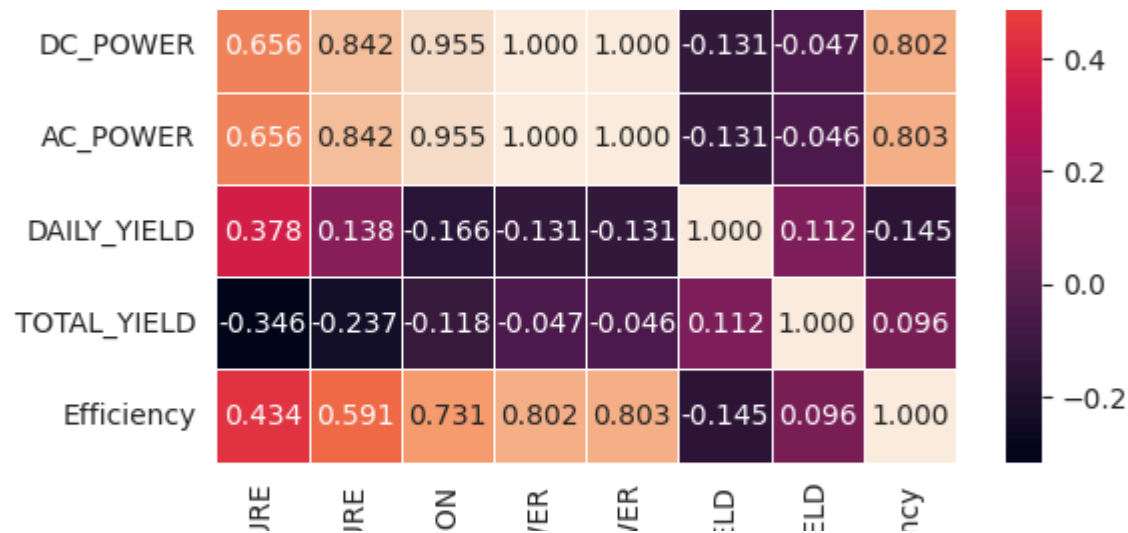
### ✓ Let's assess the correlation of our Plant 1 and 2 data

```
#Plant1 correlation of features
corr = plant1_data.drop(columns=['DATE_TIME', 'PLANT_ID', 'time_y', 'date_y']).corr(method = 'spearman')
plt.figure(dpi=100)
sns.heatmap(corr, robust=True, annot=True, fmt='0.3f', linewidths=.5, square=True)
plt.show()
```

```
#Plant2 correlation of features
corr = plant2_data.drop(columns=['DATE_TIME', 'PLANT_ID', 'time_y', 'date_y']).corr(method = 'spearman')
plt.figure(dpi=100)
sns.heatmap(corr, robust=True, annot=True, fmt='0.3f', linewidths=.5, square=True)
plt.show()
```





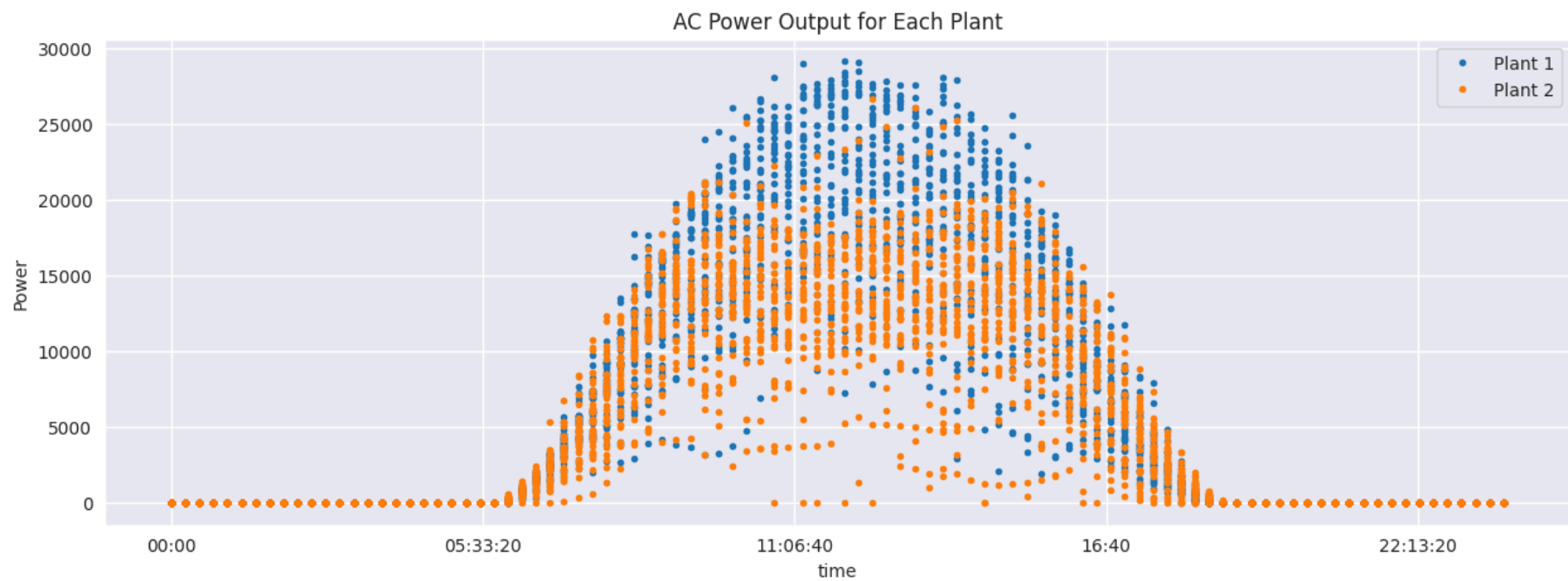
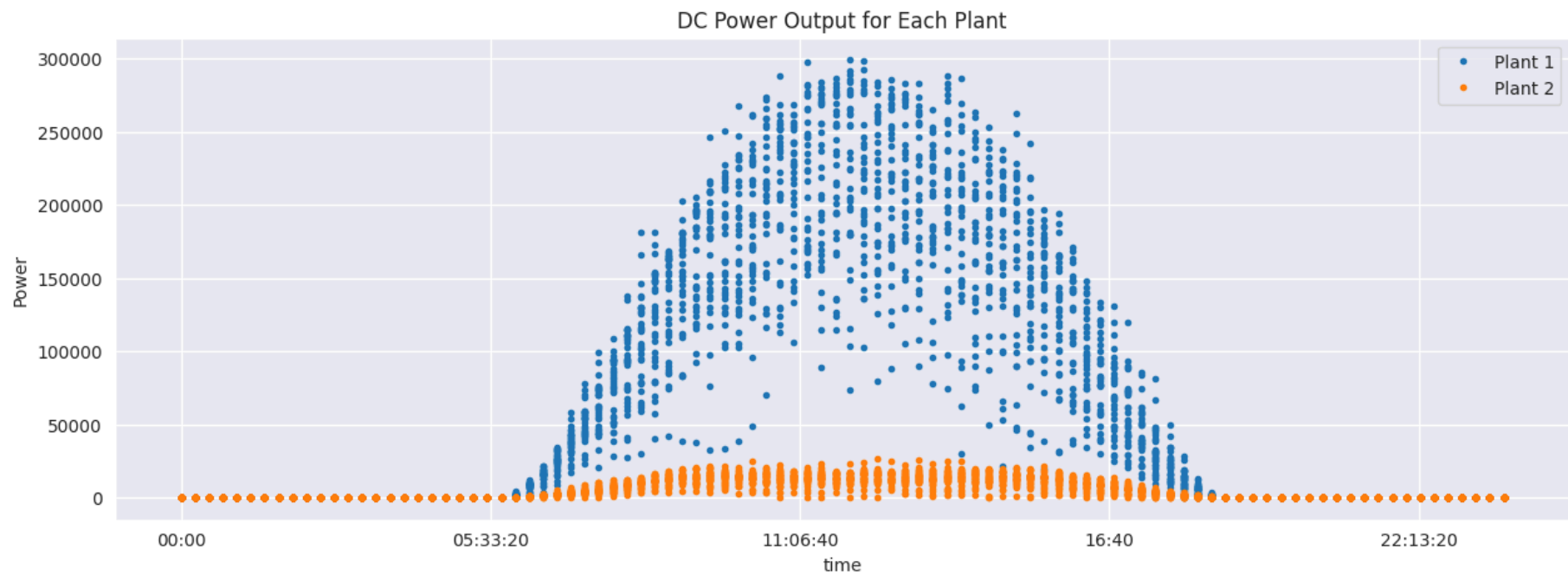


We can observe DC\_Power is highly correlated to AC\_Power at 100% which means the generated DC\_Power goes through the inverter to be converted into AC\_Power with less energy loss, and also there is high correlation of Irradiation at 99.2%(Plant1) & 95.5%(Plant2) and Module\_Temperature at 90%(Plant1) & 84.2%(Plant2), which means the higher the sunlight intensity increases the Module\_temperature and result in gradually increase in solar panel DC\_Power production.

✓ Let's compare Plant1 and 2 power distribution

```
# DC output from solar module
DCcompare = plant1_energy.plot(x='time', y='DC_POWER', figsize=(15,5), legend=True, style='.', label='Plant 1')
plant2_energy.plot(x='time', y='DC_POWER', legend=True, style='.', label='Plant 2', ax=DCcompare)
plt.title('DC Power Output for Each Plant')
plt.ylabel('Power')
plt.show()

# AC output from inverter
ACcompare = plant1_energy.plot(x='time', y='AC_POWER', figsize=(15,5), legend=True, style='.', label='Plant 1')
plant2_energy.plot(x='time', y='DC_POWER', legend=True, style='.', label='Plant 2', ax=ACcompare)
plt.title('AC Power Output for Each Plant')
plt.ylabel('Power')
plt.show()
```



There is no recorded instance of ac\_power at 5:33; which means that there is no production before 5:33. For the same reason can be concluded that there are no production at or after 18:45.

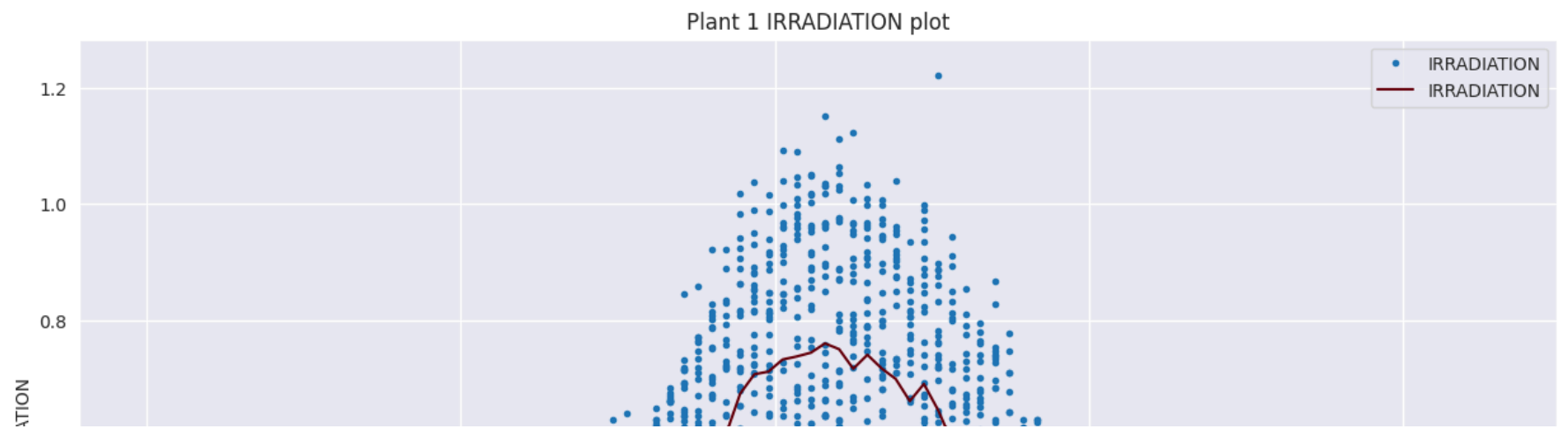
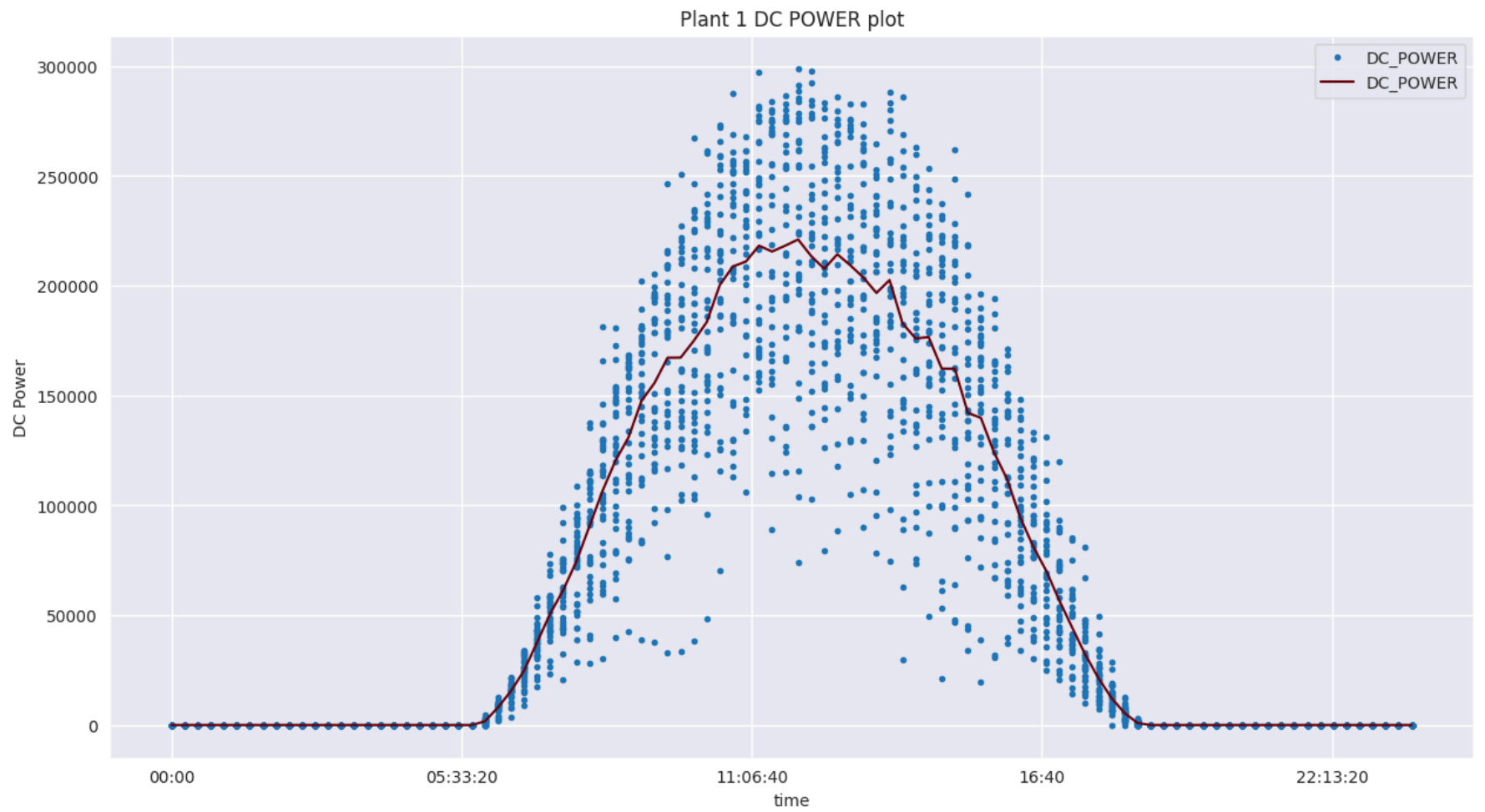
**Observation:**

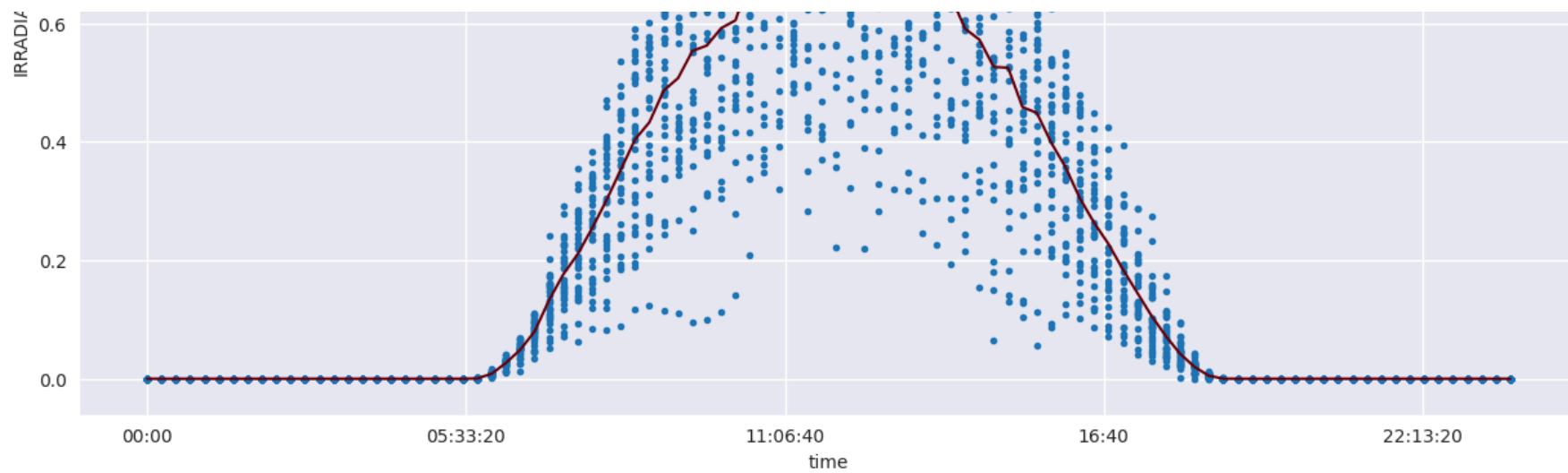
Power output is generated with the presence of sunlight, which starts at around 05:33hrs and ends at around 16:40hrs. But we can observe the DC\_Power generated by plant 2 is low compared to plant 1, then when it has gone through the inverter of Plant2 the AC\_Power get boosted to be higher to the level of the one generated by plant 1. This might be due to the plant 2 were located in rainy or cloudy area compared to plant 1 area which might be sunny

- a. Use the data to analyze how solar panel efficiency behaves as environmental
- ✓ conditions (like sunlight intensity) approach certain operational limits. Examine if there are points of discontinuity or leveling off in efficiency.
- ✓ Distribution of Plant 1 DC\_Power and IRRADIATION(sunlight intensity) across daily time

```
#Plant 1 DC_Power data distribution
plant1_energy.plot(x= 'time', y='DC_POWER', style='.', figsize = (15, 8))
plant1_energy.groupby('time')['DC_POWER'].agg('mean').plot(legend=True, colormap='Reds_r')
plt.ylabel('DC Power')
plt.title('Plant 1 DC POWER plot')
plt.show()

#Plant 1 Irradiation data distribution
plant1_weather.plot(x= 'time', y='IRRADIATION', style='.', figsize = (15, 8))
plant1_weather.groupby('time')['IRRADIATION'].agg('mean').plot(legend=True, colormap='Reds_r')
plt.ylabel('IRRADIATION')
plt.title('Plant 1 IRRADIATION plot')
plt.show()
```







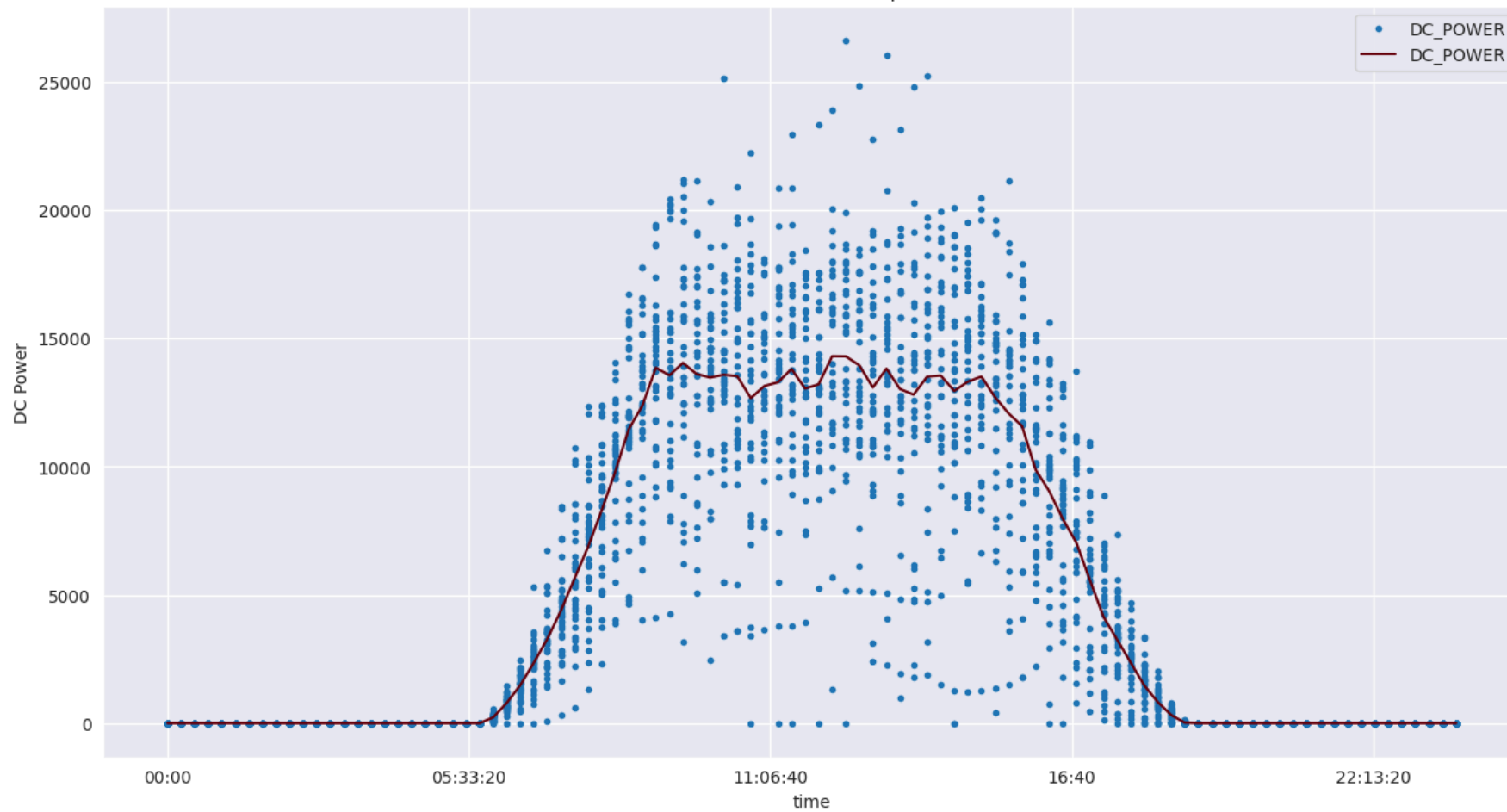
In the graph above is clear that irradiation is zero between 18:45 and 5:45 but in a closer inspection there are values other than zero before 6:00 and also after 18:30. Those values might correspond to high sensitivity of the sensor, but irradiation too low to be usable by the solar panels.

## ✓ Distribution of Plant 2 DC\_Power and IRRADIATION(sunlight intensity) accross daily time

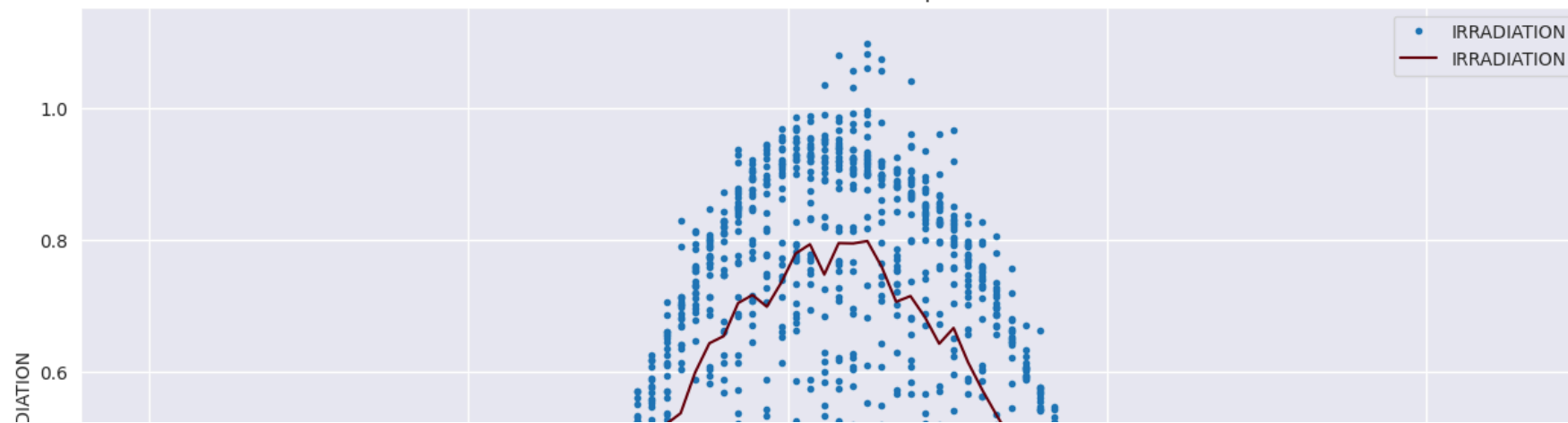
```
# Plant2 DC_Power data distribution
plant2_energy.plot(x= 'time', y='DC_POWER', style='.', figsize = (15, 8))
plant2_energy.groupby('time')['DC_POWER'].agg('mean').plot(legend=True, colormap='Reds_r')
plt.ylabel('DC Power')
plt.title('Plant 2 DC POWER plot')
plt.show()

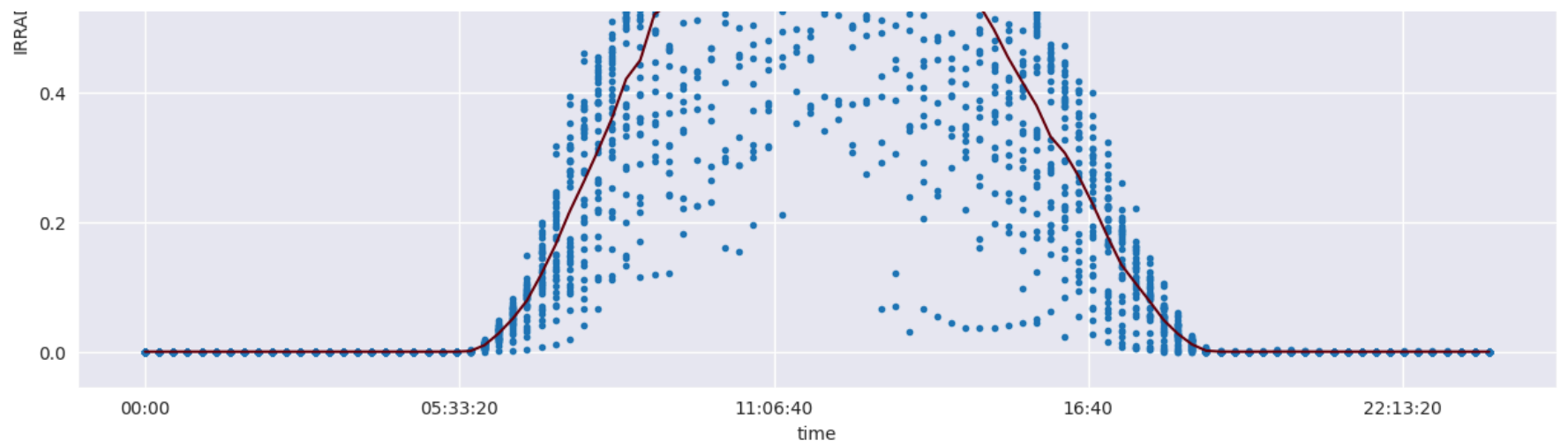
#Plant 2 Irradiation data distribution
plant2_weather.plot(x= 'time', y='IRRADIATION', style='.', figsize = (15, 8))
plant2_weather.groupby('time')['IRRADIATION'].agg('mean').plot(legend=True, colormap='Reds_r')
plt.ylabel('IRRADIATION')
plt.title('Plant 2 IRRADIATION plot')
plt.show()
```

Plant 2 DC POWER plot



Plant 2 IRRADIATION plot





Observation:

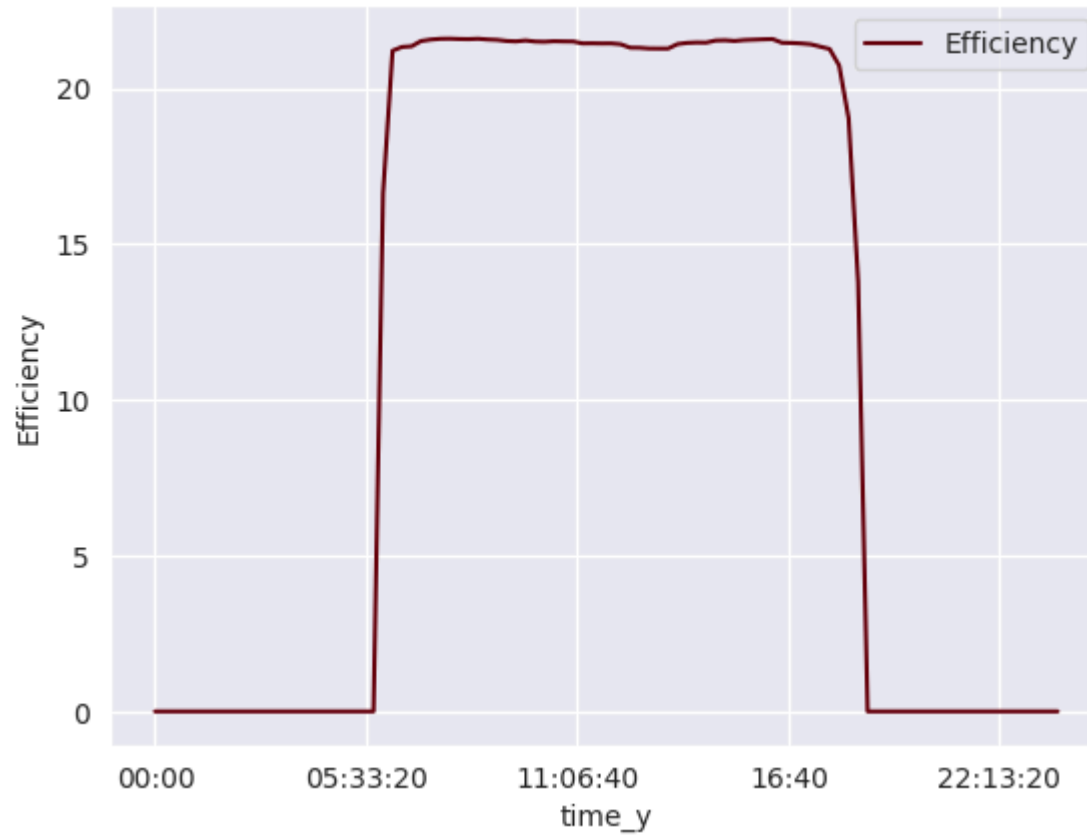
Power output is generated with the presence of sunlight, which starts at around 05:33 hrs and ends at around 18:00hrs. We can confirm the correlation between power generation and the sunlight of the solar panel efficiency, we see on the above graphs either DC\_Power and Irradiation start rising after 5:33 when the Sun Set On at the beginning of the day and it starts fall after 16:40 as the Sun Set Off at the end of the day

## ✓ Plant 1 and 2 Efficiency comparison

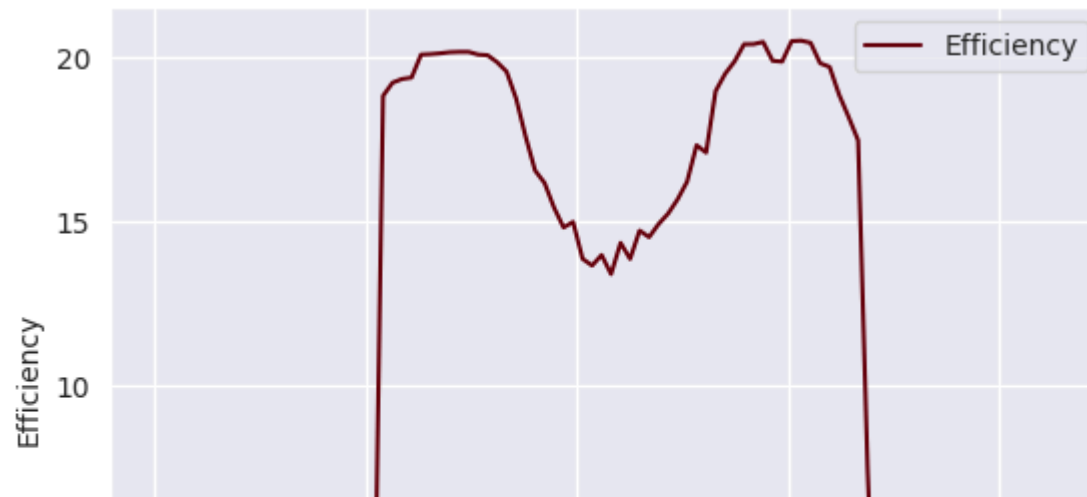
```
#Plant 1 efficiency
plant1_data.groupby('time_y')['Efficiency'].agg('mean').plot(legend=True, colormap='Reds_r')
plt.ylabel('Efficiency')
plt.title('Plant 1 Efficiency plot')
plt.show()

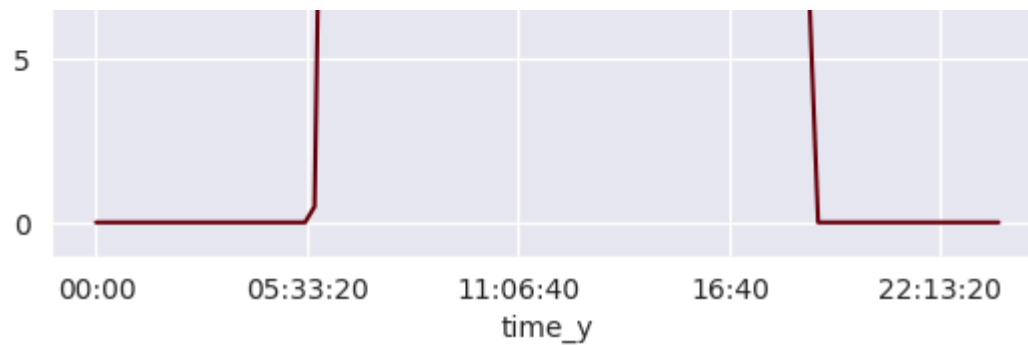
#Plant 2 efficiency
plant2_data.groupby('time_y')['Efficiency'].agg('mean').plot(legend=True, colormap='Reds_r')
plt.ylabel('Efficiency')
plt.title('Plant 2 Efficiency plot')
plt.show()
```

Plant 1 Efficiency plot



Plant 2 Efficiency plot





**Insight:**

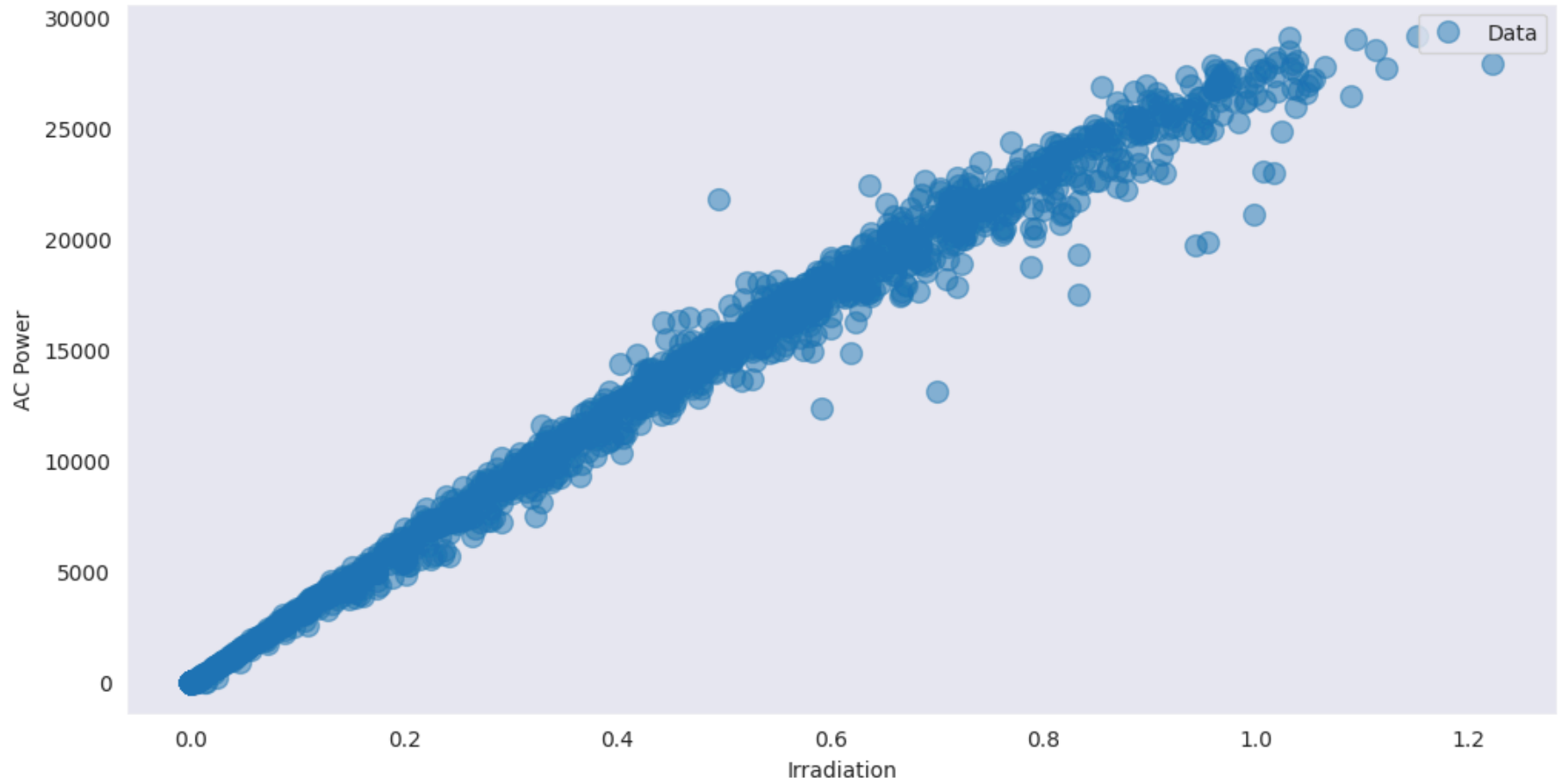
We observe for the efficiency of Plant 1 rose to the highest when the Sun set On at the beginning of the day(05:33) and it stays high untill the Sun set Off after 16:40, where we observe the drop of efficiency at around 18:00. While for Plant 2 during the day they were a drop of efficiency at 11:06 which might be due to the rain that causes less irradiation(Sunlight intensity)

- b. Utilize the dataset to model how changes in factors like sunlight intensity or
- ✓ panel orientation impact the rate of energy production. This involves calculating the derivative of the energy output with respect to these variables.

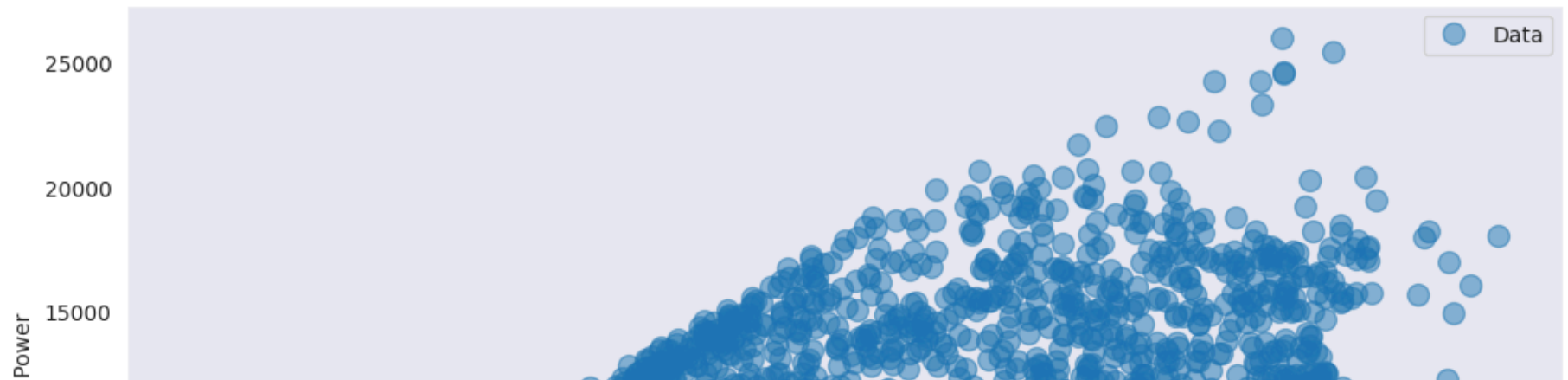
```
# for plant 1 plot AC_Power vs Irradiation
_, ax = plt.subplots(1, 1, figsize=(12,6))
ax.plot(plant1_data['IRRADIATION'], plant1_data['AC_POWER'], marker='o', linestyle='', alpha=0.5, ms=10, label='Data')
ax.grid()
ax.margins(0.05)
ax.legend()
plt.title('Plant 1 AC Power vs. Irradiation')
plt.xlabel('Irradiation')
plt.ylabel('AC Power')
plt.show()

# for plant 2 plot AC_Power vs Irradiation
_, ax = plt.subplots(1, 1, figsize=(12,6))
ax.plot(plant2_data['IRRADIATION'], plant2_data['AC_POWER'], marker='o', linestyle='', alpha=0.5, ms=10, label='Data')
ax.grid()
ax.margins(0.05)
ax.legend()
plt.title('Plant 2 AC Power vs. Irradiation')
plt.xlabel('Irradiation')
plt.ylabel('AC Power')
plt.show()
```

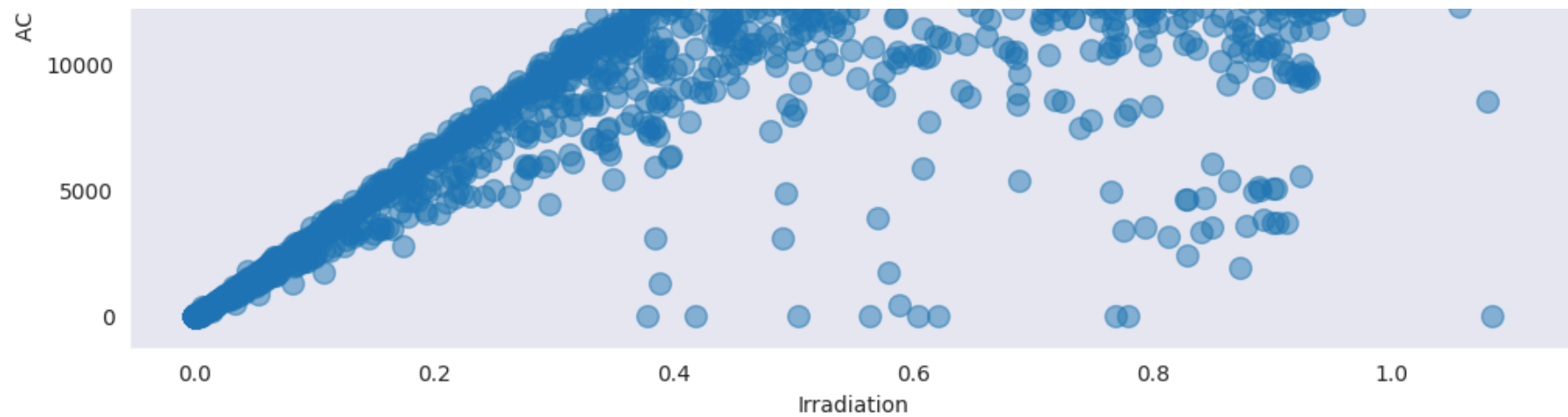
Plant 1 AC Power vs. Irradiation



Plant 2 AC Power vs. Irradiation







```
from scipy.interpolate import interp1d

# Calculate energy output (assuming AC_POWER represents energy output)
Plant1_energy = plant1_data['AC_POWER']

# Choose variable: sunlight intensity (irradiation)
Plant1_sunlight_intensity = plant1_data['IRRADIATION']

# Remove NaN and infinite values from sunlight_intensity and energy_output
Plant1_sunlight_intensity_cleaned = Plant1_sunlight_intensity.dropna()
Plant1_energy_cleaned = Plant1_energy.dropna()

# Interpolate energy output to smooth the data
Plant1_energy_interp = interp1d(Plant1_sunlight_intensity_cleaned, Plant1_energy_cleaned, kind='linear')

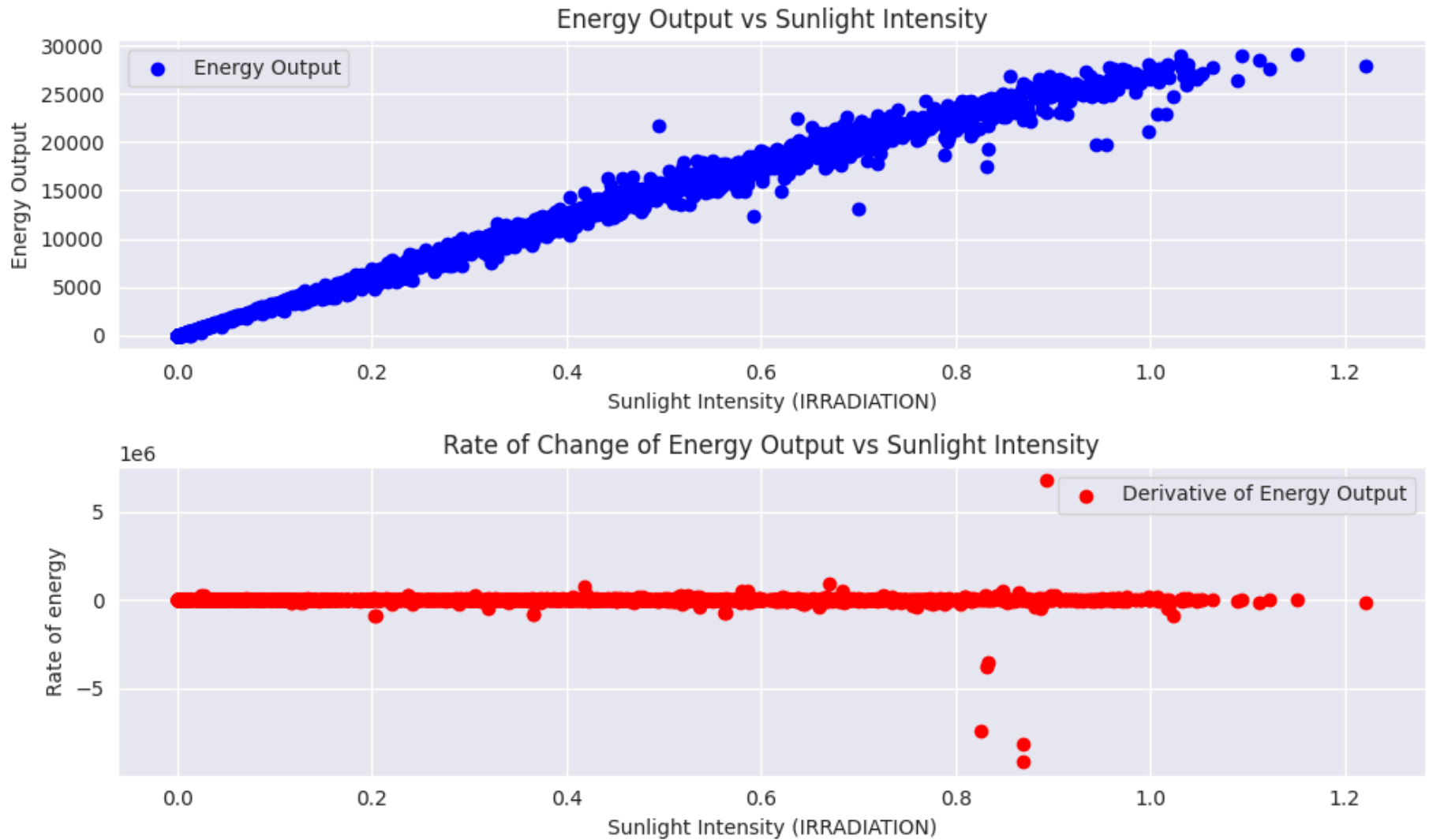
# Calculate the derivative of energy output with respect to sunlight intensity
Rate_of_energy = np.gradient(Plant1_energy_interp(Plant1_sunlight_intensity_cleaned), Plant1_sunlight_intensity_cleaned)

# Plotting
plt.figure(figsize=(10, 6))

plt.subplot(2, 1, 1)
plt.scatter(Plant1_sunlight_intensity_cleaned, Plant1_energy_cleaned, color='blue', label='Energy Output')
plt.xlabel('Sunlight Intensity (IRRADIATION)')
plt.ylabel('Energy Output')
plt.title('Energy Output vs Sunlight Intensity')
plt.legend()

plt.subplot(2, 1, 2)
plt.scatter(Plant1_sunlight_intensity_cleaned, Rate_of_energy, color='red', label='Derivative of Energy Output')
plt.xlabel('Sunlight Intensity (IRRADIATION)')
plt.ylabel('Rate of energy')
plt.title('Rate of Change of Energy Output vs Sunlight Intensity')
plt.legend()

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



We can see Energy output points are distributed in liner model as the Sunlight Intensity increases, which means if the solar panel is positioned to maximize the sunlight intensity it will result in high rate of energy production

```
from scipy.interpolate import interp1d

# Calculate energy output (assuming AC_POWER represents energy output)
Plant2_energy = plant2_data['AC_POWER']

# Choose variable: sunlight intensity (irradiation)
Plant2_sunlight_intensity = plant2_data['IRRADIATION']

# Remove NaN and infinite values from sunlight_intensity and energy_output
Plant2_sunlight_intensity_cleaned = Plant2_sunlight_intensity.dropna()
Plant2_energy_cleaned = Plant2_energy.dropna()

# Interpolate energy output to smooth the data
Plant2_energy_interp = interp1d(Plant2_sunlight_intensity_cleaned, Plant2_energy_cleaned, kind='linear')

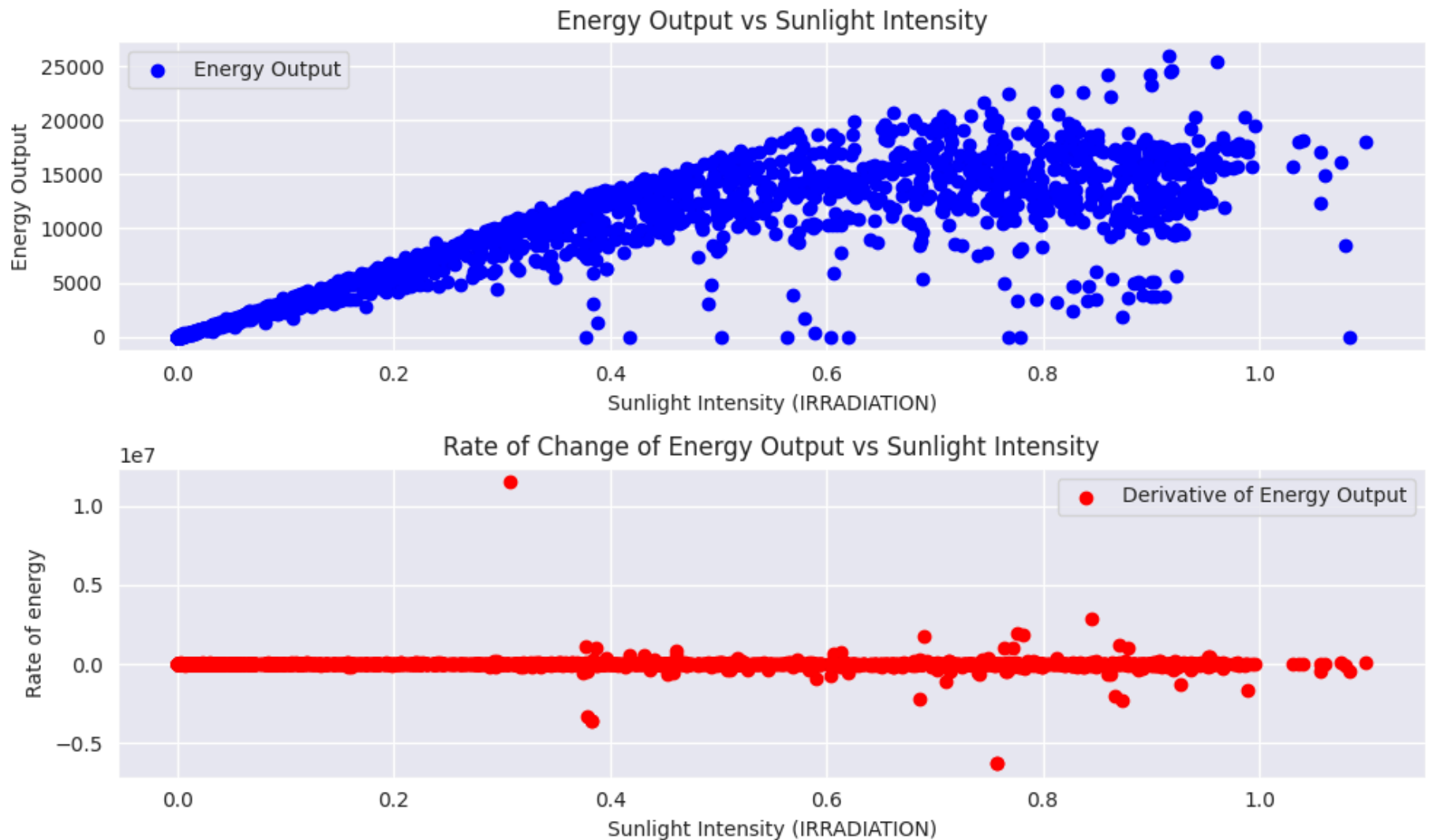
# Calculate the derivative of energy output with respect to sunlight intensity
Rate_of_energy = np.gradient(Plant2_energy_interp(Plant2_sunlight_intensity_cleaned), Plant2_sunlight_intensity_cleaned)

# Plotting
plt.figure(figsize=(10, 6))

plt.subplot(2, 1, 1)
plt.scatter(Plant2_sunlight_intensity_cleaned, Plant2_energy_cleaned, color='blue', label='Energy Output')
plt.xlabel('Sunlight Intensity (IRRADIATION)')
plt.ylabel('Energy Output')
plt.title('Energy Output vs Sunlight Intensity')
plt.legend()

plt.subplot(2, 1, 2)
plt.scatter(Plant2_sunlight_intensity_cleaned, Rate_of_energy, color='red', label='Derivative of Energy Output')
plt.xlabel('Sunlight Intensity (IRRADIATION)')
plt.ylabel('Rate of energy')
plt.title('Rate of Change of Energy Output vs Sunlight Intensity')
plt.legend()

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



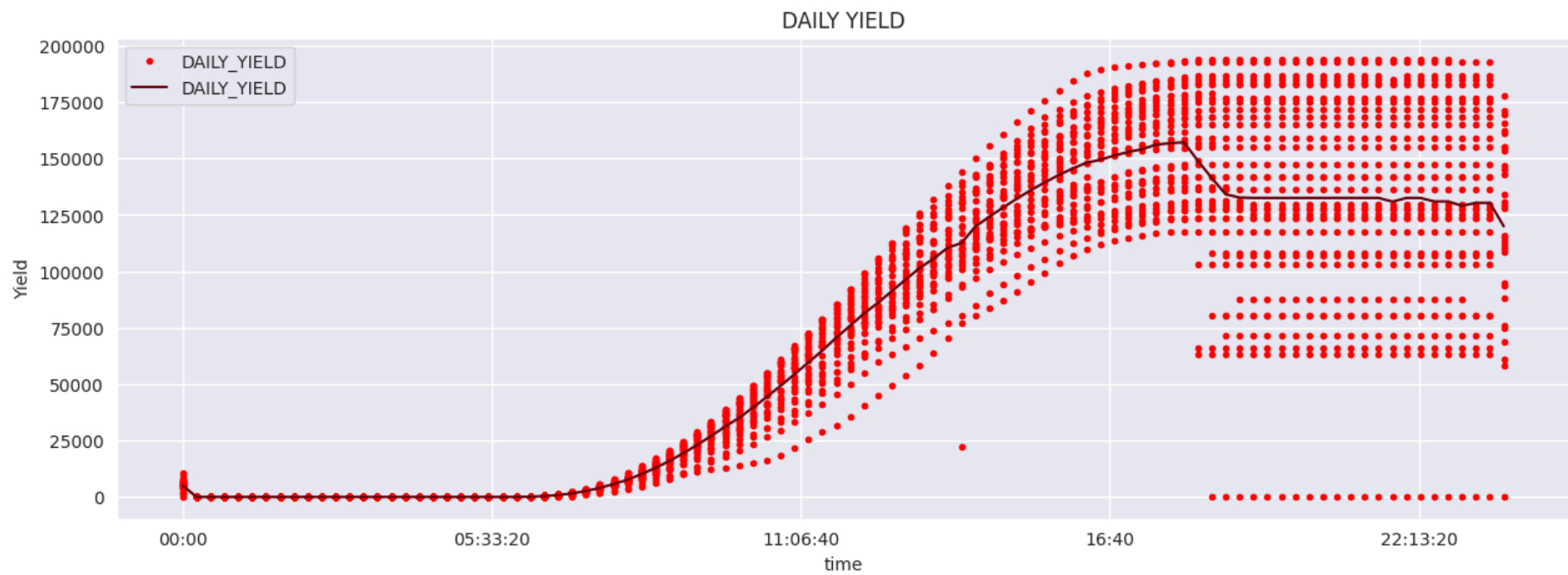
### Insight:

This graph shows how AC\_POWER varies with irradiation. There are certain points where even though there is some irradiation, the value of AC\_POWER is zero. This means that at that point, an inverter was not working. But there are some values for which, even though there is exact same irradiation, the AC\_POWER is varying. This is because there is a single irradiation sensor for the whole

plant. Due to various reasons, same quality panels may not give same output(eg.one panel might be cleaner than the other). So there seems to be streaking.

- c. Analyze cumulative solar energy production over time by integrating the
- ✓ power output data. This can help in understanding total energy yield under different conditions over a specific period.

```
#plot the cummulative daily produced energy of Plant 1
plant1_energy.plot(x='time', y='DAILY_YIELD', style='r.', figsize=(15,5))
plant1_energy.groupby('time')['DAILY_YIELD'].agg('mean').plot(legend=True, colormap='Reds_r')
plt.title('DAILY YIELD')
plt.ylabel('Yield')
plt.show()
```



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

# Set figure size
plt.figure(figsize=(15, 8))

# Daily Ambient Temperature
ambtemp_compare = sns.lineplot(x='DATE_TIME', y='DAILY_YIELD', data=plant1_energy, err_style='band', label='Plant 1')
sns.lineplot(x='DATE_TIME', y='DAILY_YIELD', data=plant2_energy, err_style='band', label='Plant 2', ax=ambtemp_compare)
plt.ylabel('DAILY_YIELD')
plt.xlabel('Date')
plt.title('DAILY_YIELD for Both Plants')
plt.xticks(rotation=45)
plt.show()
```