Import Library

```
from tqdm import tqdm # Library untuk bar progrss
import time
import warnings # Library untuk handle warning
warnings.filterwarnings('ignore') # Untuk mengignore beberapa warning yang kuran

# Library pemrosesan data
import pandas as pd
import numpy as np

# Library untuk menyimpan data
from google.colab import files

# Library untuk visualisasi data
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import seaborn as sns
try:
  import windrose
except ImportError:
  !pip install windrose
from windrose import WindroseAxes
import ipywidgets as widgets
from IPython.display import display
from statsmodels.tsa.seasonal import seasonal_decompose

# Library untuk metrik dan model selection
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import (
    KFold,
    cross_val_predict as cvp,
    cross_val_score as cvs
)
from sklearn.feature_selection import mutual_info_regression

# Library-library untuk model prediksi
!pip install catboost lightgbm
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, E
from sklearn.tree import DecisionTreeRegressor
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, mean_absolute_error
import xgboost as xgb
import lightgbm as lgb
from sklearn.neighbors import KNeighborsRegressor
import catboost as cb
```

```
# Library untuk melakukan Hyperparameter Tuning
!pip install optuna
import optuna

# Library untuk menyimpan model
import pickle
```

```
Requirement already satisfied: catboost in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist
Requirement already satisfied: numpy<2.0,>=1.16.0 in /usr/local/lib/python3
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/di
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/di
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10
Requirement already satisfied: optuna in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: alembic>=1.5.0 in /usr/local/lib/python3.10/
Requirement already satisfied: colorlog in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10
Requirement already satisfied: sqlalchemy>=1.4.2 in /usr/local/lib/python3.
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: Mako in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: typing-extensions>=4 in /usr/local/lib/pytho
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.
```

```
# Supaya dapat melihat seluruh kolom
pd.set_option('display.max_columns', None)
```

```
# Inisiasi random state
SEED = 42
```

Import Data

```
!gdown 1FRT9v2_At4xRuRI3dfPHmOBoCFLWH2sd
!gdown 16tY1OR4uveQDkd1DGDefz7SnN9yJwRWg
!gdown 1ZgXW9eVz54zi4zzsQ7yw-aEr8w7Te6Vh
!gdown 1C0lLbPvDCLGpfc9tyu0EdRxPDZa1mWgL
!gdown 1idMKdhkX5PaJZ2xkJIa58ExpCWCCepPj
```

Show hidden output

```
# Load datasets
daily_train = pd.read_csv('/content/daily_train.csv')
daily_test = pd.read_csv('/content/daily_test.csv')
hourly_train = pd.read_csv('/content/hourly_train.csv')
hourly_test = pd.read_csv('/content/hourly_test.csv')
loc = pd.read_csv('/content/location.csv')


# Inspect data
print(daily_train.info())
print(daily_test.info())
print(hourly_train.info())
print(hourly_test.info())
print(loc.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19704 entries, 0 to 19703
Data columns (total 19 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   ID                                 19704 non-null  object 
 1   location_id                        19704 non-null  int64  
 2   time                               19704 non-null  object 
 3   temperature_2m_max (°C)            19704 non-null  float64
 4   temperature_2m_min (°C)            19704 non-null  float64
 5   temperature_2m_mean (°C)           19704 non-null  float64
 6   apparent_temperature_max (°C)      19704 non-null  float64
 7   apparent_temperature_min (°C)      19704 non-null  float64
 8   apparent_temperature_mean (°C)     19704 non-null  float64
 9   sunrise (iso8601)                  19704 non-null  object 
 10  sunset (iso8601)                   19704 non-null  object 
 11  daylight_duration (s)              19704 non-null  float64
 12  sunshine_duration (s)              19704 non-null  float64
 13  rain_sum (mm)                      19704 non-null  float64
 14  wind_speed_10m_max (km/h)          19704 non-null  float64
 15  wind_gusts_10m_max (km/h)          19704 non-null  float64
 16  wind_direction_10m_dominant (°)    19704 non-null  float64
 17  shortwave_radiation_sum (MJ/m²)    19704 non-null  float64
 18  et0_fao_evapotranspiration (mm)    19704 non-null  float64
dtypes: float64(14), int64(1), object(4)
memory usage: 2.9+ MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5892 entries, 0 to 5891
```

```
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   ID                                5892 non-null   object
 1   location_id                       5892 non-null   int64
 2   time                              5892 non-null   object
 3   temperature_2m_max (°C)           5892 non-null   float64
 4   temperature_2m_min (°C)           5892 non-null   float64
 5   temperature_2m_mean (°C)          5892 non-null   float64
 6   apparent_temperature_max (°C)     5892 non-null   float64
 7   apparent_temperature_min (°C)     5892 non-null   float64
 8   apparent_temperature_mean (°C)    5892 non-null   float64
 9   sunrise (iso8601)                 5892 non-null   object
 10  sunset (iso8601)                  5892 non-null   object
 11  daylight_duration (s)             5892 non-null   float64
 12  sunshine_duration (s)             5890 non-null   float64
 13  wind_speed_10m_max (km/h)         5892 non-null   float64
 14  wind_gusts_10m_max (km/h)         5892 non-null   float64
 15  wind_direction_10m_dominant (°)   5892 non-null   float64
 16  shortwave_radiation_sum (MJ/m²)   5890 non-null   float64
 17  et0_fao_evapotranspiration (mm)   5890 non-null   float64
dtypes: float64(13), int64(1), object(4)
memory usage: 828.7+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 472896 entries, 0 to 472895
Data columns (total 32 columns):
 #   Column                                    Non-Null Count   Dtype
---  ------                                    --------------   -----
 0   location_id                               472896 non-null  int64
```

EDA & Pre-Processing

Train

```
daily_train.info()
```

Show hidden output

```
hourly_train.info()
```

Show hidden output

```
daily_train.head()
```

Show hidden output

```
hourly_train.head()
```

Show hidden output

Missing Value and Duplicate

```
# Check for duplicate rows and missing values
print("\nNumber of duplicate rows in daily_train:", daily_train.duplicated().su
print("Number of duplicate rows in hourly_train:", hourly_train.duplicated().su
print("\nMissing values in daily_train:\n", daily_train.isnull().sum())
print("\nMissing values in hourly_train:\n", hourly_train.isnull().sum())
```

Show hidden output

Tidak terdapat row duplikat ataupun missing value pada dataset daily_train dan hourly_train.

Statistics Summary

```
daily_train.describe().T
```

Show hidden output

```
hourly_train.describe().T
```

Show hidden output

Monthly Rain Sum by Location

```
# Copy untuk keperluan EDA
dtrain = daily_train.copy()
htrain = hourly_train.copy()

# Ekstrak tahun dan bulan kolom time
dtrain['time'] = pd.to_datetime(dtrain['time'])
dtrain['year'] = dtrain['time'].dt.year
dtrain['month'] = dtrain['time'].dt.month
dtrain['day'] = dtrain['time'].dt.day

htrain['time'] = pd.to_datetime(htrain['time'])
htrain['year'] = htrain['time'].dt.year
htrain['month'] = htrain['time'].dt.month
htrain['day'] = htrain['time'].dt.day

# Merge dengan informasi dari dataset lokasi
dtrain = pd.merge(dtrain, loc[['location_id',  'elevation']], on='location_id',
htrain = pd.merge(htrain, loc[['location_id',  'elevation']], on='location_id',


# Line Chart for Rain Sum (monthly) over different years

for location_id in dtrain['location_id'].unique():
    location_data = dtrain[dtrain['location_id'] == location_id] # Filter per l

    month_year_rain = location_data.groupby(['year', 'month'])['rain_sum (mm)']
    month_year_pivot = month_year_rain.pivot(index='month', columns='year', val

    # Plot data dengan matplotlib
    month_year_pivot.plot(marker='o', linewidth=2, figsize=(24, 12))
    plt.title(f"Monthly Rain Sum Trends Over Different Years for Location {loca
    plt.xlabel("Month", fontsize=14)
    plt.ylabel("Average Rain Sum (mm)", fontsize=14)
    plt.xticks(ticks=range(1, 13), labels=["Jan", "Feb", "Mar", "Apr", "May", "
    plt.yticks(fontsize=12)

    plt.legend(title="Year", bbox_to_anchor=(1.05, 1), loc='upper left', fontsi
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

Show hidden output

Terlihat bahwa setiap tahun pasti ada tren penurunan (May - Sep) dan peningkatan curah hujan pada bulan-bulan tertentu. Pada beberapa lokasi dapat ditemukan rata-rata curah hujan yang cukup tinggi dibandingkan lainnya, seperti pada lokasi 8 dan 9.

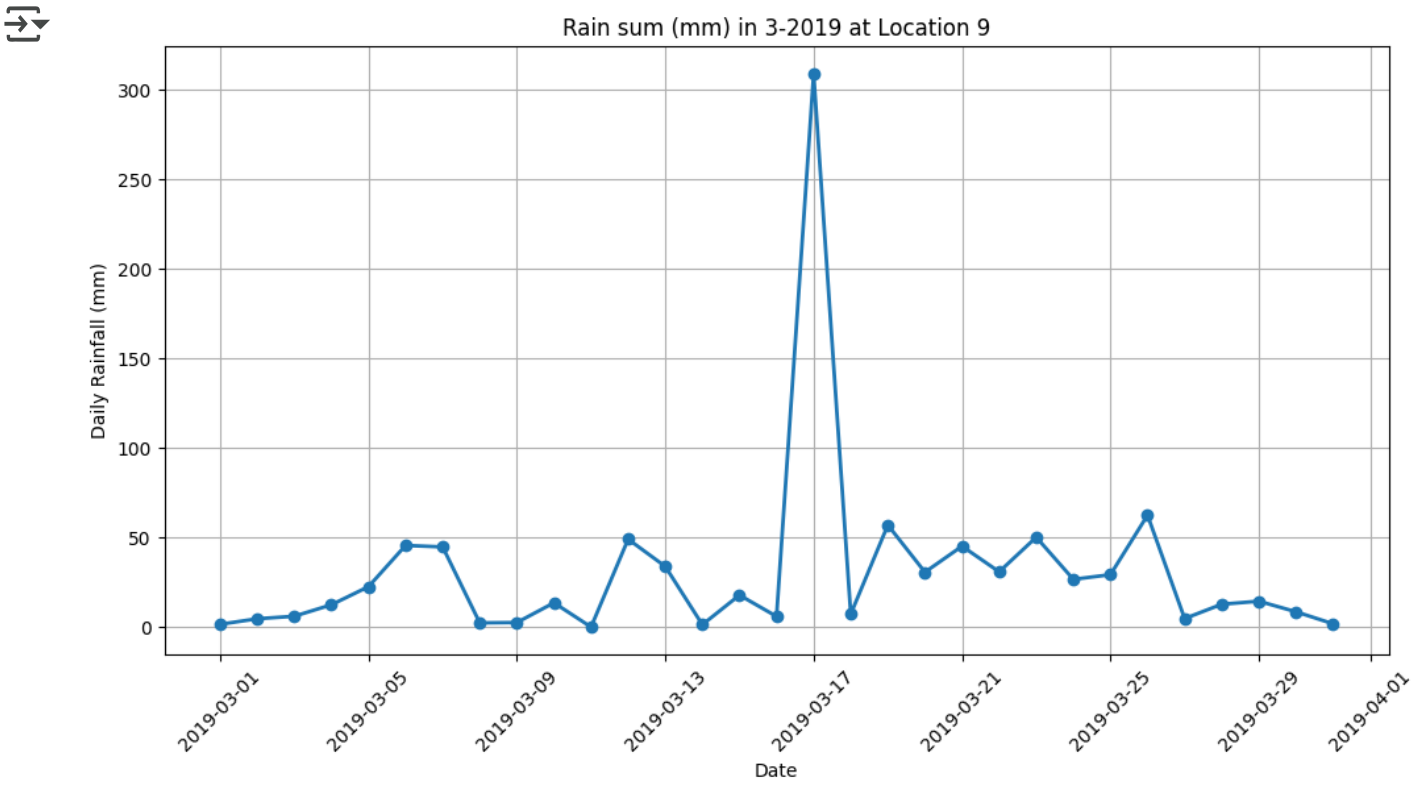
```
# Melihat curah hujan yang ekstem
dtrain[dtrain['rain_sum (mm)'] > 150]
```

	ID	location_id	time	temperature_2m_max (°C)	temperature_2m_mi (°C
754	loc1_20210124	1	2021-01-24	25.8	23
10217	loc7_20200101	7	2020-01-01	26.3	23
10591	loc7_20210109	7	2021-01-09	29.3	23
11799	loc8_20191102	8	2019-11-02	25.6	23
12157	loc8_20201025	8	2020-10-25	25.1	24
13211	loc9_20190317	9	2019-03-17	23.2	21

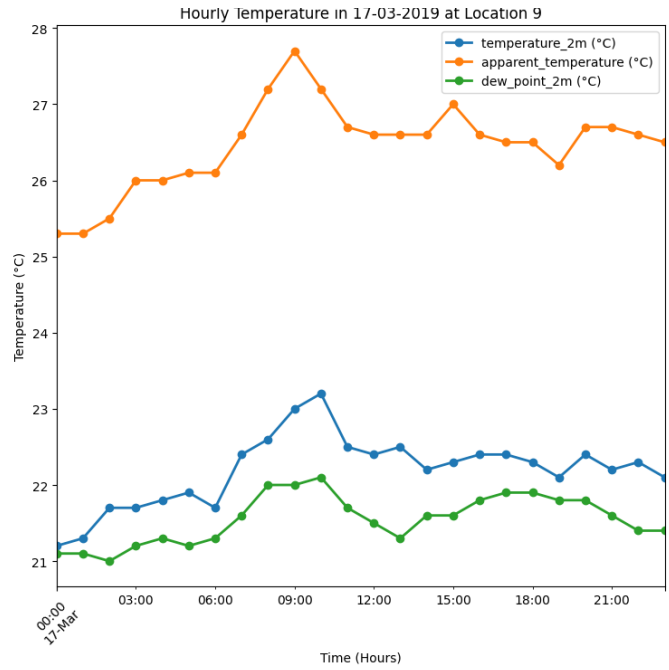
Menurut BMKG, curah hujan >150 mm/hari berada pada kategori hujan ekstrem. Ini berarti curah hujan yang lebih dari 150 mm/hari dapat disebut sebagai cuaca ekstrem atau kejadian yang jarang terjadi. Terdapat lokasi-lokasi seperti Lokasi 8 dan Lokasi 9 yang mencatatkan curah hujan harian mencapai lebih dari 300 mm. Kemungkinan pada hari itu terjadi fenomena alam yang tidak biasa, seperti badai tropis atau curah hujan lebat yang berlangsung dalam waktu singkat.

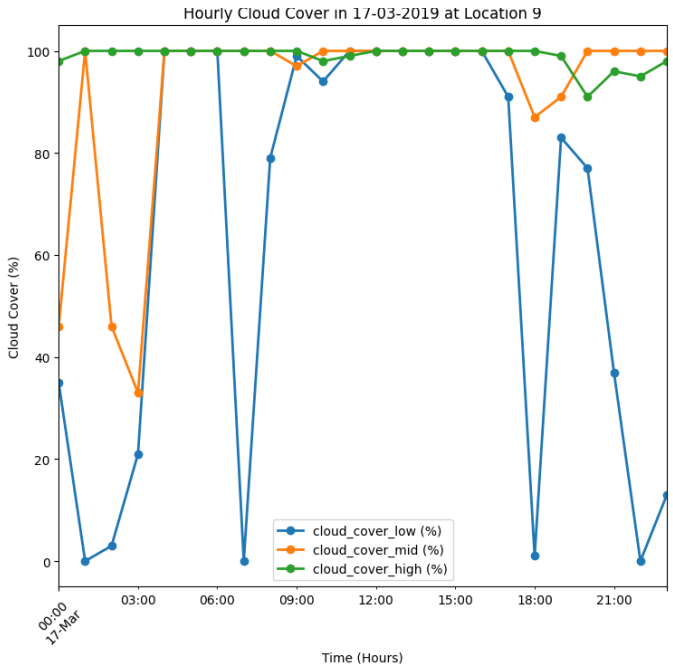
Selanjutnya, kita akan mengeksplorasi salah satu kejadian curah hujan ekstrem, yaitu pada tanggal 17 Maret 2019 di lokasi 9.

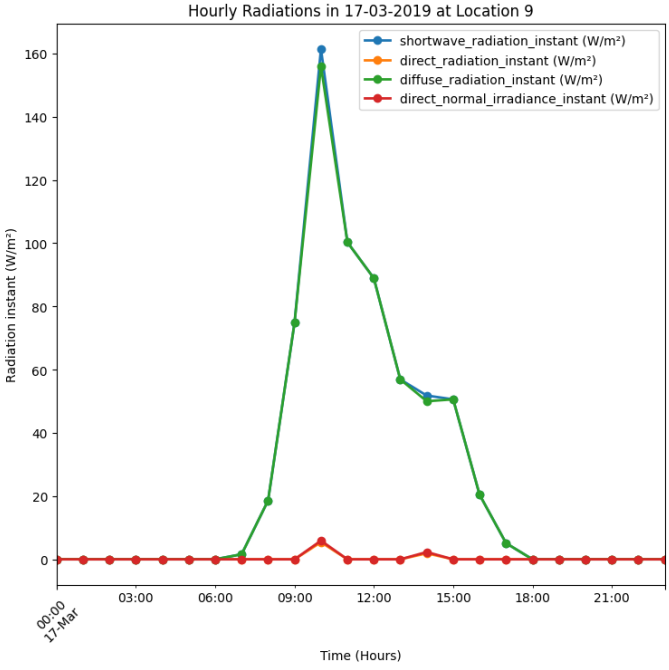
```
# Filter kembali untuk bulan dan lokasi yang diinginkan
dtrain_eks = dtrain[(dtrain['year'] == 2019) & (dtrain['month'] == 3) & (dtrain

# Extract unique values directly from df_feb_eda
year = dtrain_eks['year'].unique()[0]
month = dtrain_eks['month'].unique()[0]
location_id = dtrain_eks['location_id'].unique()[0]

# Plot data curah hujan harian
plt.figure(figsize=(12, 6))
plt.plot(dtrain_eks['time'], dtrain_eks['rain_sum (mm)'], marker='o', linestyle
plt.title(f"Rain sum (mm) in {month}-{year} at Location {location_id}")
plt.xlabel("Date")
plt.ylabel("Daily Rainfall (mm)")
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

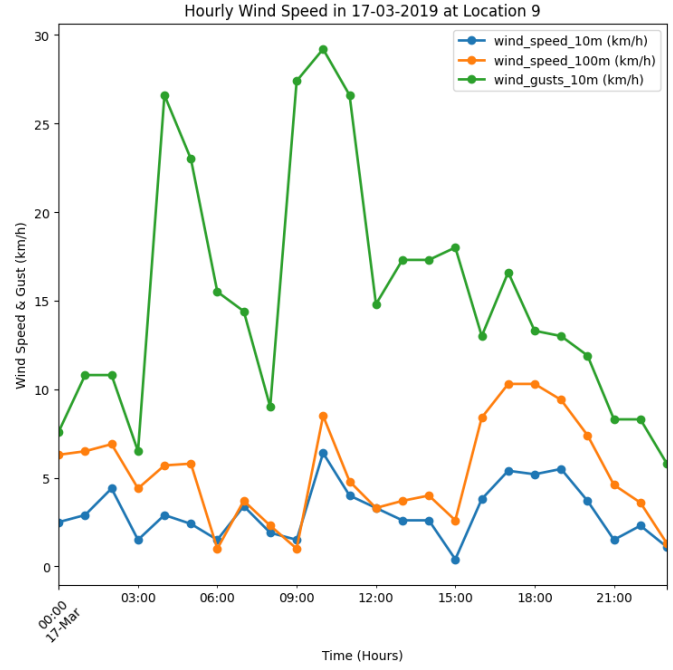


```
# Plot per Hour for Extreme Rain_sum
# Define the filter conditions
date_filter = (htrain['year'] == 2019) & (htrain['month'] == 3) & (htrain['day'
year = htrain[date_filter]['year'].unique()[0]
month = htrain[date_filter]['month'].unique()[0]
day = htrain[date_filter]['day'].unique()[0]
location_id = htrain[date_filter]['location_id'].unique()[0]

# Extract the data
hourly_temp = htrain[date_filter][['time', 'temperature_2m (°C)', 'apparent_tem
hourly_cloud = htrain[date_filter][['time', 'cloud_cover_low (%)', 'cloud_cover
hourly_rad = htrain[date_filter][['time', 'shortwave_radiation_instant (W/m²)',
                                  'diffuse_radiation_instant (W/m²)', 'direct_r
hourly_wind = htrain[date_filter][['time', 'wind_speed_10m (km/h)', 'wind_speed

hourly_temp.set_index('time', inplace=True)
hourly_cloud.set_index('time', inplace=True)
hourly_rad.set_index('time', inplace=True)
hourly_wind.set_index('time', inplace=True)

# Subplot
fig, axes = plt.subplots(2, 2, figsize=(15, 15))  # 1 row, 2 columns

# Plot temperature data
hourly_temp.plot(ax=axes[0,0], marker='o', linestyle='-', linewidth=2)
axes[0,0].set(title=f"Hourly Temperature in {day:02d}-{month:02d}-{year} at Loc
            xlabel="Time (Hours)", ylabel="Temperature (°C)")
axes[0, 0].tick_params(axis='x', rotation=45)

# Plot cloud cover data
hourly_cloud.plot(ax=axes[0, 1], marker='o', linestyle='-', linewidth=2)
axes[0, 1].set(title=f"Hourly Cloud Cover in {day:02d}-{month:02d}-{year} at Lo
               xlabel="Time (Hours)", ylabel="Cloud Cover (%)")
axes[0, 1].tick_params(axis='x', rotation=45)

# Plot radiation data
hourly_rad.plot(ax=axes[1,0], marker='o', linestyle='-', linewidth=2)
axes[1,0].set(title=f"Hourly Radiations in {day:02d}-{month:02d}-{year} at Loca
              xlabel="Time (Hours)", ylabel="Radiation instant (W/m²)")
axes[1,0].tick_params(axis='x', rotation=45)

# Plot wind data
hourly_wind.plot(ax=axes[1,1], marker='o', linestyle='-', linewidth=2)
axes[1,1].set(title=f"Hourly Wind Speed in {day:02d}-{month:02d}-{year} at Loca
              xlabel="Time (Hours)", ylabel="Wind Speed & Gust (km/h)")
axes[1,1].tick_params(axis='x', rotation=45)

# Show plot
plt.tight_layout()
plt.show()
```


Hourly Temperature in 17-03-2019 at Location 9
temperature_2m (°C)
apparent_temperature (°C)
dew_point_2m (°C)
Temperature (°C)
Time (Hours)
00:00
17-Mar
03:00
06:00
09:00
12:00
15:00
18:00
21:00


Hourly Cloud Cover in 17-03-2019 at Location 9
cloud_cover_low (%)
cloud_cover_mid (%)
cloud_cover_high (%)
Cloud Cover (%)
Time (Hours)
00:00
17-Mar
03:00
06:00
09:00
12:00
15:00
18:00
21:00


Hourly Radiations in 17-03-2019 at Location 9
shortwave_radiation_instant (W/m²)
direct_radiation_instant (W/m²)
diffuse_radiation_instant (W/m²)
direct_normal_irradiance_instant (W/m²)
Radiation instant (W/m²)
Time (Hours)
00:00
17-Mar
03:00
06:00
09:00
12:00
15:00
18:00
21:00


Hourly Wind Speed in 17-03-2019 at Location 9
wind_speed_10m (km/h)
wind_speed_100m (km/h)
wind_gusts_10m (km/h)
Wind Speed & Gust (km/h)
Time (Hours)
00:00
17-Mar
03:00
06:00
09:00
12:00
15:00
18:00
21:00

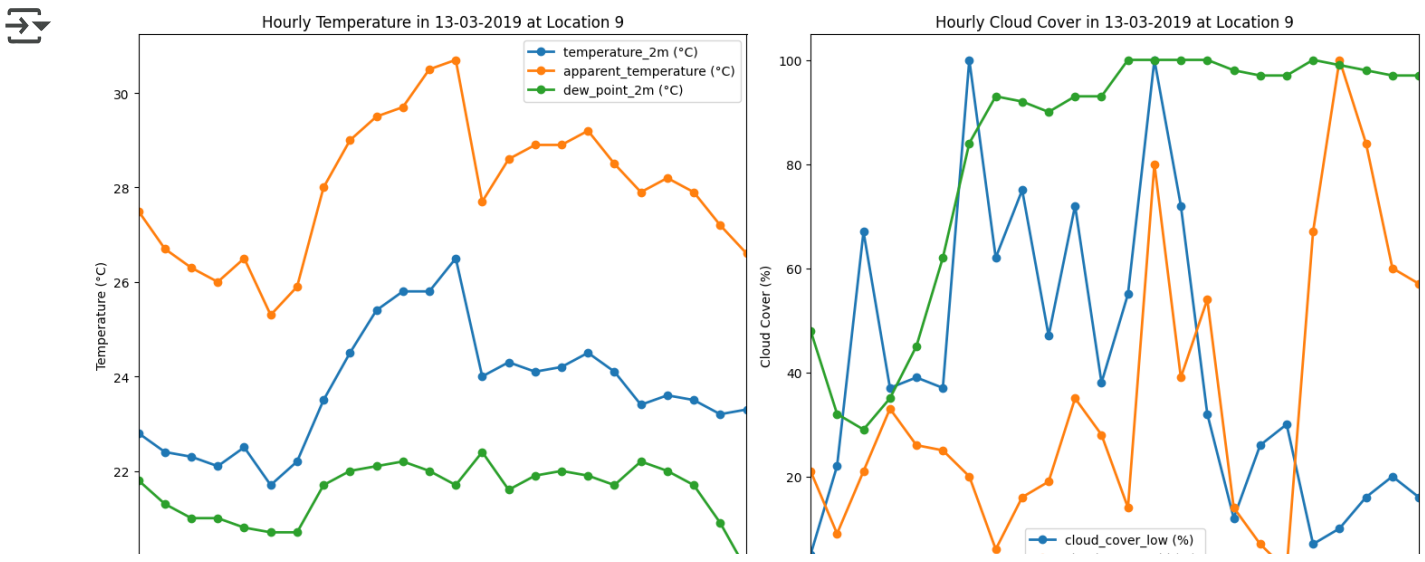
Plot di atas menunjukkan keadaan alam ketika terjadi hujan ekstrem pada lokasi 9 (308.8 mm).

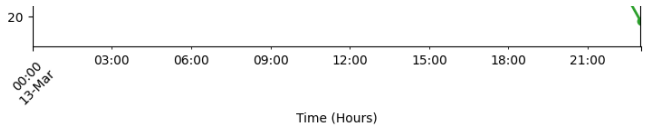
- Dew point (titik embun) stabil sepanjang hari (~21°C) dan hampir sama dengan suhu lingkungan, menunjukkan kadar **kelembapan yang relatif tinggi**. Ini mendukung pembentukan awan dan potensi hujan.[1]
- Cloud Cover (mid dan high) cenderung banyak sepanjang hari, menunjukkan adanya perkembangan awan konvektif. Awan konvektif adalah awan yang dihasilkan oleh proses konveksi akibat pemanasan radiasi surya.[2] Perkembangan **awan konvektif** sering dikaitkan dengan **cuaca buruk** seperti badai petir, hujan lebat, tornado, angin kencang, hujan es, dan fenomena lainnya.[3]
- Pada grafik radiasi, terlihat bahwa `direct radiation` tetap rendah sepanjang hari, sedangkan diffuse radiation meningkat terutama di pagi-sore hari. Hal ini menunjukkan adanya tutupan awan yang cukup signifikan (sesuai dengan grafik `cloud cover`), sehingga radiasi langsung dari matahari terhambat dan lebih banyak tersebar oleh awan maupun partikel di atmosfer. Selain itu, direct normal irradiance hampir nol, menandakan bahwa kondisi atmosfer tidak mendukung penerimaan energi maksimum dari radiasi matahari karena sudut optimal terganggu oleh tutupan awan tebal.
- Grafik angin menunjukkan kecepatan angin yang fluktuatif namun sering mengalami peningkatan signifikan. Ini menunjukkan adanya **aliran angin kuat** yang dapat **membawa massa udara lembap** ke lokasi tersebut, mendukung pembentukan awan hujan tebal.

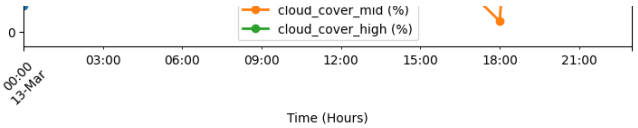
Mari bandingkan plot di atas dengan plot fitur-fitur ketika hujan tidak ekstrem pada bulan, tahun, dan lokasi yang sama.

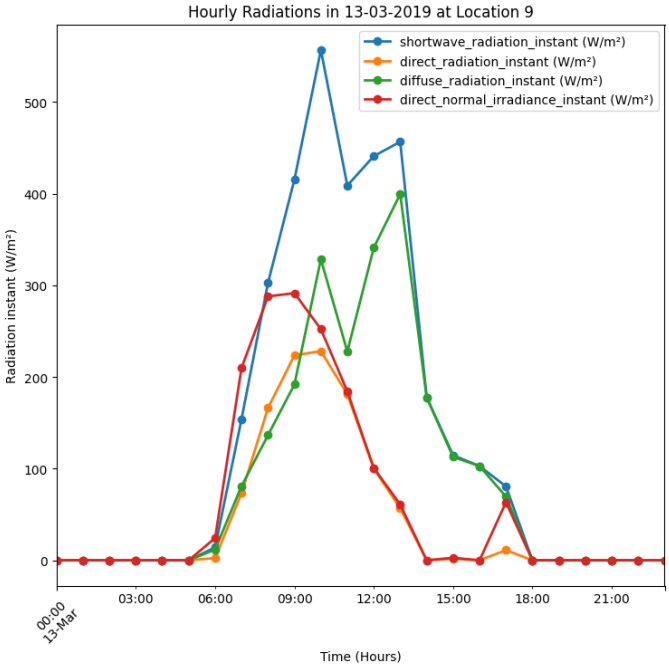
```
# Plot per Hour for Not Extreme Rain_sum (34 mm)
# Define the filter conditions
date_filter = (htrain['year'] == 2019) & (htrain['month'] == 3) & (htrain['day'
year = htrain[date_filter]['year'].unique()[0]
month = htrain[date_filter]['month'].unique()[0]
day = htrain[date_filter]['day'].unique()[0]
location_id = htrain[date_filter]['location_id'].unique()[0]

# Extract the data
hourly_temp = htrain[date_filter][['time', 'temperature_2m (°C)', 'apparent_tem
hourly_cloud = htrain[date_filter][['time', 'cloud_cover_low (%)', 'cloud_cover
hourly_rad = htrain[date_filter][['time', 'shortwave_radiation_instant (W/m²)',
                                  'diffuse_radiation_instant (W/m²)', 'direct_r
hourly_wind = htrain[date_filter][['time', 'wind_speed_10m (km/h)', 'wind_speed
```

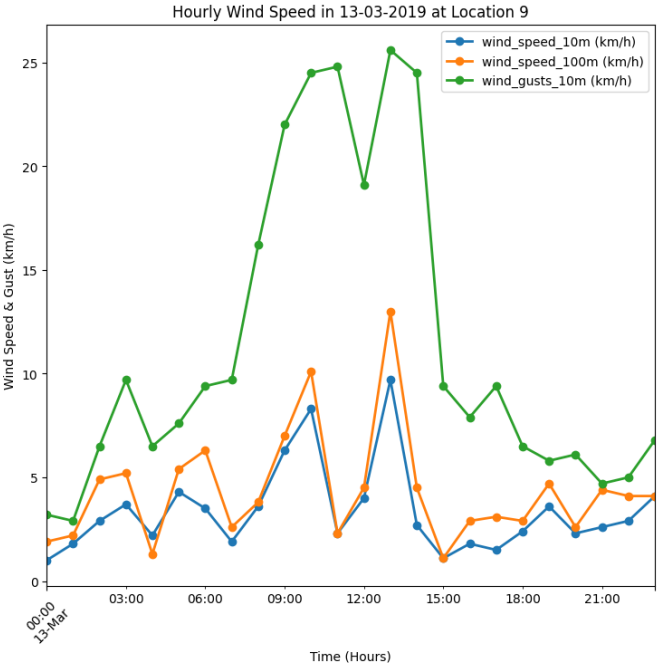
```
hourly_temp.set_index('time', inplace=True)
hourly_cloud.set_index('time', inplace=True)
hourly_rad.set_index('time', inplace=True)
hourly_wind.set_index('time', inplace=True)

# Subplot
fig, axes = plt.subplots(2, 2, figsize=(15, 15))  # 1 row, 2 columns

# Plot temperature data
hourly_temp.plot(ax=axes[0,0], marker='o', linestyle='-', linewidth=2)
axes[0,0].set(title=f"Hourly Temperature in {day:02d}-{month:02d}-{year} at Loc
            xlabel="Time (Hours)", ylabel="Temperature (°C)")
axes[0, 0].tick_params(axis='x', rotation=45)

# Plot cloud cover data
hourly_cloud.plot(ax=axes[0, 1], marker='o', linestyle='-', linewidth=2)
axes[0, 1].set(title=f"Hourly Cloud Cover in {day:02d}-{month:02d}-{year} at Lo
               xlabel="Time (Hours)", ylabel="Cloud Cover (%)")
axes[0, 1].tick_params(axis='x', rotation=45)

# Plot radiation data
hourly_rad.plot(ax=axes[1,0], marker='o', linestyle='-', linewidth=2)
axes[1,0].set(title=f"Hourly Radiations in {day:02d}-{month:02d}-{year} at Loca
              xlabel="Time (Hours)", ylabel="Radiation instant (W/m²)")
axes[1,0].tick_params(axis='x', rotation=45)

# Plot wind data
hourly_wind.plot(ax=axes[1,1], marker='o', linestyle='-', linewidth=2)
axes[1,1].set(title=f"Hourly Wind Speed in {day:02d}-{month:02d}-{year} at Loca
              xlabel="Time (Hours)", ylabel="Wind Speed & Gust (km/h)")
axes[1,1].tick_params(axis='x', rotation=45)

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()
```




20
00:00
13-Mar
03:00
06:00
09:00
12:00
15:00
18:00
21:00
Time (Hours)


cloud_cover_mid (%)
cloud_cover_high (%)
0
00:00
13-Mar
03:00
06:00
09:00
12:00
15:00
18:00
21:00
Time (Hours)

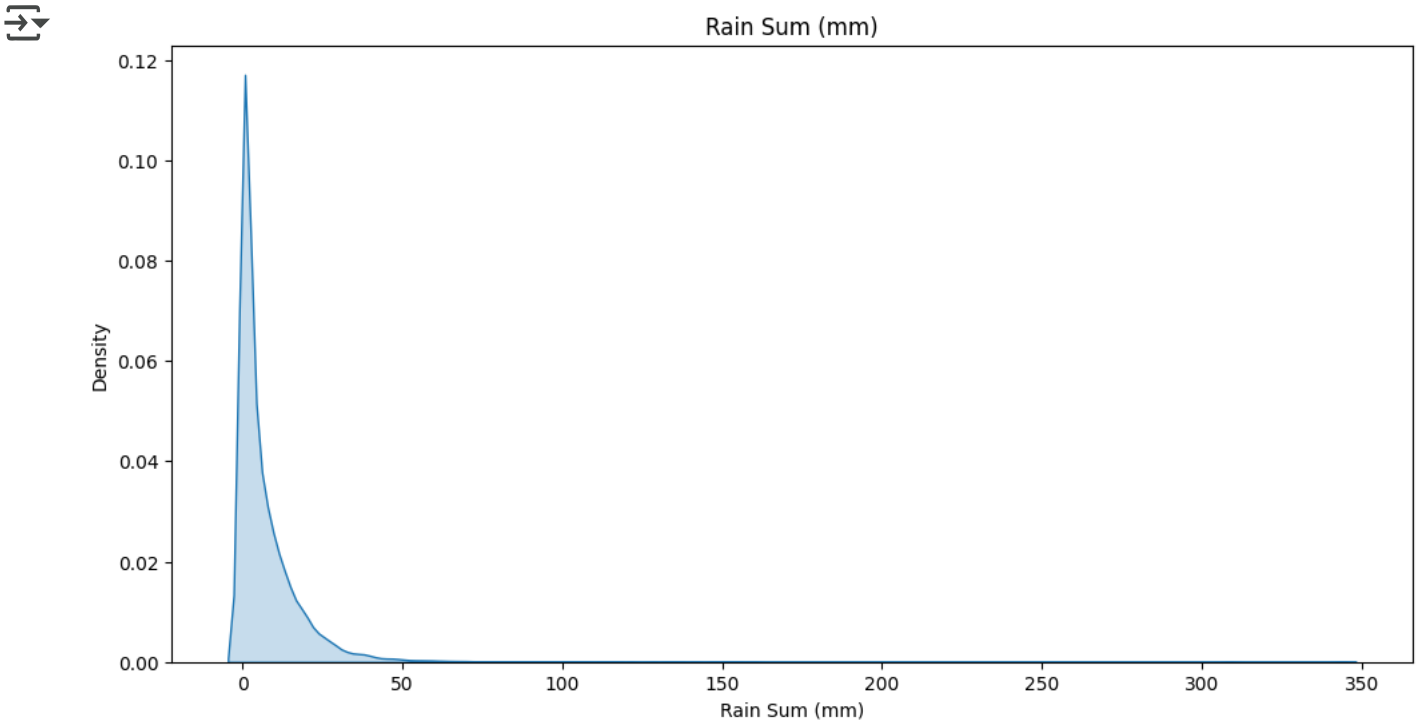

Hourly Radiations in 13-03-2019 at Location 9
shortwave_radiation_instant (W/m²)
direct_radiation_instant (W/m²)
diffuse_radiation_instant (W/m²)
direct_normal_irradiance_instant (W/m²)
Radiation instant (W/m²)
0
100
200
300
400
500
00:00
13-Mar
03:00
06:00
09:00
12:00
15:00
18:00
21:00
Time (Hours)


Hourly Wind Speed in 13-03-2019 at Location 9
wind_speed_10m (km/h)
wind_speed_100m (km/h)
wind_gusts_10m (km/h)
Wind Speed & Gust (km/h)
0
5
10
15
20
25
00:00
13-Mar
03:00
06:00
09:00
12:00
15:00
18:00
21:00
Time (Hours)

- Dew point cenderung lebih rendah dari suhu lingkungan. Kelembapan relatif lebih rendah sehingga pembentukan awan dan potensi hujan lebih kecil.
- Cloud cover low dan mid lebih bervariasi dan cenderung tidak banyak,menunjukkan kondisi langit yang lebih terbuka dan tidak tertutup awan.
- Secara keseluruhan, radiasi lebih tinggi dibandingkan dengan keadaan hujan ekstrem. Mencerminkan kondisi cuaca lebih sedikit hambatan awan dan agak cerah.
- Angin pada hujan tidak ekstrem tidak menunjukkan perbedaan yang signifikan dari hujan ekstrem. Kemungkinan angin tidak membawa massa udara lembap yang banyak untuk mendukung hujan lebat, namun hujan biasa.

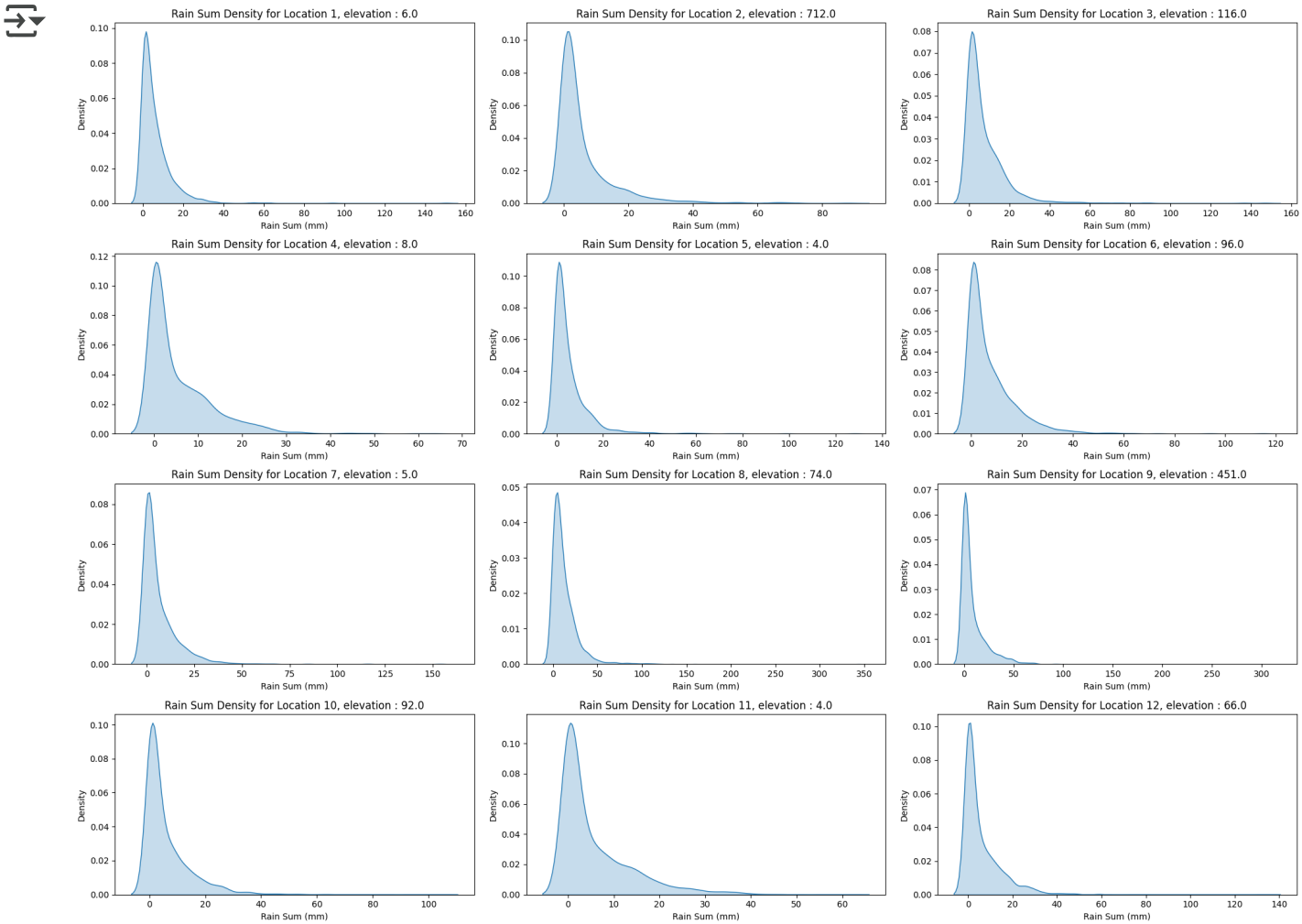
Rain Sum Distribution

```
plt.figure(figsize=(12, 6))
sns.kdeplot(dtrain['rain_sum (mm)'], shade=True)
plt.title('Rain Sum (mm)')
plt.xlabel('Rain Sum (mm)')
plt.ylabel('Density')
plt.show()
```




```
# Calculate the density of rain_sum for each location
location_density = {}
for location_id in dtrain['location_id'].unique():
    location_data = dtrain[dtrain['location_id'] == location_id]
    location_density[location_id] = location_data['rain_sum (mm)'].values

# Subplots
fig, axes = plt.subplots(4, 3, figsize=(20, 15))
axes = axes.flatten()

# Plot density for each location
for i, location_id in enumerate(location_density):
    ax = axes[i]
```

```
    sns.kdeplot(location_density[location_id], shade=True, ax=ax)
    ax.set_title(f'Rain Sum Density for Location {location_id}, elevation : {lo
    ax.set_xlabel("Rain Sum (mm)")
    ax.set_ylabel("Density")

# Remove any unused subplots
for j in range(i + 1, len(axes)):
  fig.delaxes(axes[j])

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


Rain Sum Density for Location 1, elevation : 6.0
Rain Sum Density for Location 2, elevation : 712.0
Rain Sum Density for Location 3, elevation : 116.0
Rain Sum Density for Location 4, elevation : 8.0
Rain Sum Density for Location 5, elevation : 4.0
Rain Sum Density for Location 6, elevation : 96.0
Rain Sum Density for Location 7, elevation : 5.0
Rain Sum Density for Location 8, elevation : 74.0
Rain Sum Density for Location 9, elevation : 451.0
Rain Sum Density for Location 10, elevation : 92.0
Rain Sum Density for Location 11, elevation : 4.0
Rain Sum Density for Location 12, elevation : 66.0
Density
Rain Sum (mm)

Secara keseluruhan, distribusi tampak miring ke kanan, yang menunjukkan bahwa sebagian besar curah hujan bernilai kecil dan curah hujan yang lebih tinggi jarang terjadi. Menurut jurnal, lokasi dengan elevasi lebih tinggi cenderung memiliki curah hujan yang relatif lebih tinggi pula. Namun, dapat dilihat bahwa ada juga lokasi dengan elevasi rendah (lokasi 1, 7) yang memiliki curah hujan cukup tinggi. Hal ini menunjukkan bahwa faktor lain selain elevasi mungkin memengaruhi curah hujan di lokasi-lokasi tersebut.

Sunshine Duration

[ ] ↳ 2 cells hidden

Relationship Between Elevation and Temperature

[ ] ↳ 2 cells hidden

Wind Direction

```
# Fungsi untuk mengkategorikan arah angin
def categorize_wind_direction(degree):
    if 337.5 <= degree or degree < 22.5:
        return 'N'
    elif 22.5 <= degree < 67.5:
        return 'NE'
    elif 67.5 <= degree < 112.5:
        return 'E'
    elif 112.5 <= degree < 157.5:
        return 'SE'
    elif 157.5 <= degree < 202.5:
        return 'S'
    elif 202.5 <= degree < 247.5:
        return 'SW'
    elif 247.5 <= degree < 292.5:
        return 'W'
    elif 292.5 <= degree < 337.5:
        return 'NW'
```

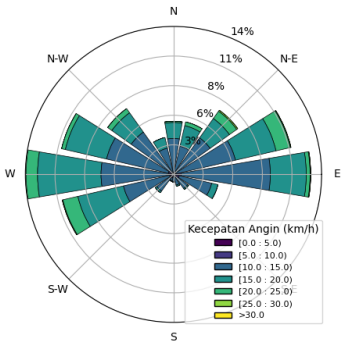
```
# Fungsi untuk plot Windrose
def plot_windrose(data, title):
    # Tambahkan kolom kategori arah angin
    data['wind_direction_category'] = data['wind_direction_10m_dominant (°)'].a
    locations = data['location_id'].unique()

    # Ukuran plot
    fig, axes = plt.subplots(4, 3, figsize=(20, 20), subplot_kw={'projection':
    axes = axes.flatten()  # Mengubah array axes menjadi list untuk iterasi

    # Iterasi setiap lokasi
    for i, location in enumerate(locations[:12]):
        # Filter data untuk lokasi tertentu
        location_data = data[data['location_id'] == location]
        elevation = location_data['elevation'].iloc[0]

        # Ambil kecepatan dan arah angin
        ws = location_data['wind_speed_10m_max (km/h)'].to_numpy()  # Kecepatan
        wd = location_data['wind_direction_10m_dominant (°)'].to_numpy()  # Ara

        # Tentukan lebar bin dan rentang kecepatan angin
        width = 5
        maxVal = 30  # Nilai maksimal kecepatan angin
        windRange = np.arange(0, maxVal + width, width)  # Rentang kecepatan an

        # WindroseAxes
        ax = axes[i]
        ax.bar(wd, ws, bins=windRange, normed=True, opening=0.85, linewidth=0.5

        # Title and Legend
        ax.set_title(f"Location {location}\nElevation: {elevation} m", fontsize
        ax.set_legend(title="Kecepatan Angin (km/h)", loc='lower right', fontsi

        # Format radius axis ke dalam persen
        fmt = '%.0f%%'
        yticks = mtick.FormatStrFormatter(fmt)
        ax.yaxis.set_major_formatter(yticks)

    # Sembunyikan sisa subplot jika jumlah lokasi < 12
    for j in range(len(locations), 12):
        fig.delaxes(axes[j])

    # Show Plot
    plt.suptitle(title, fontsize=14)
    plt.tight_layout()
    plt.show()


plot_windrose(dtrain, "Windrose\n")
```

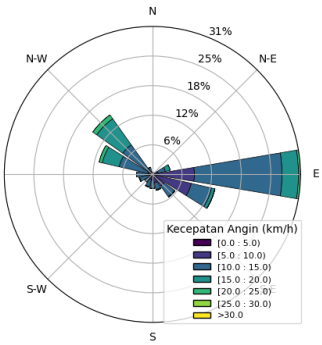
Windrose

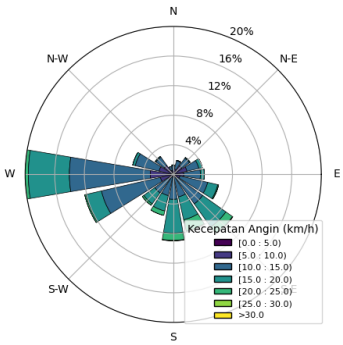
Location 1
Elevation: 6.0 m

Location 2
Elevation: 712.0 m

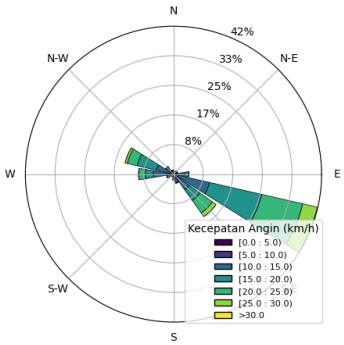
Location 3
Elevation: 116.0 m



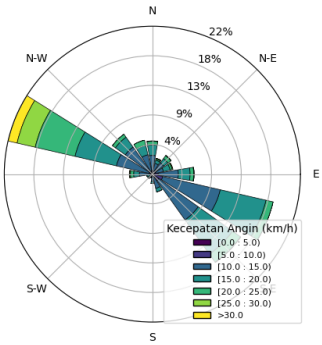




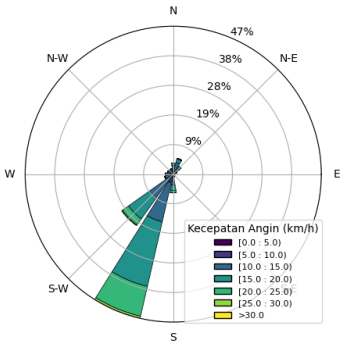
Location 4
Elevation: 8.0 m



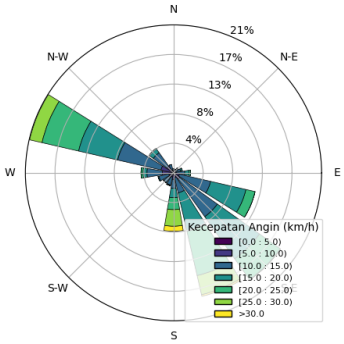
Location 5
Elevation: 4.0 m



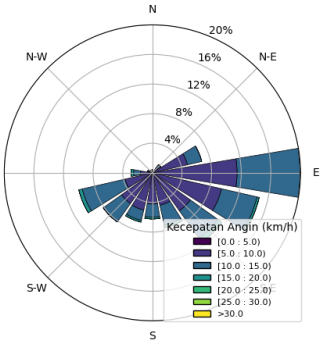
Location 6
Elevation: 96.0 m



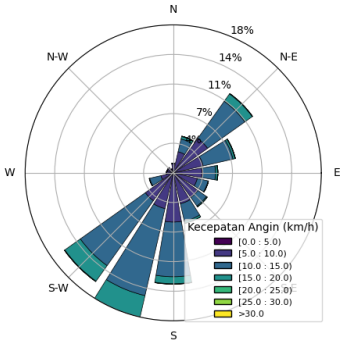
Location 7
Elevation: 5.0 m



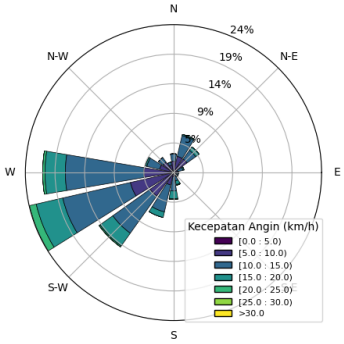
Location 8
Elevation: 74.0 m



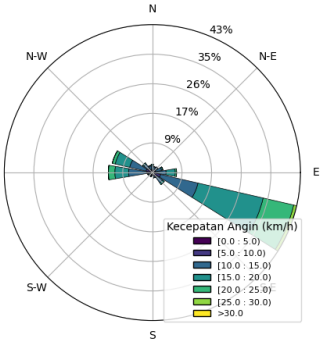
Location 9
Elevation: 451.0 m



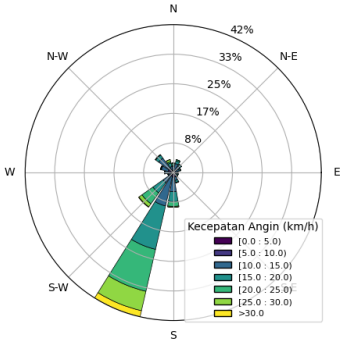
Location 10
Elevation: 92.0 m



Location 11
Elevation: 4.0 m



Location 12
Elevation: 66.0 m



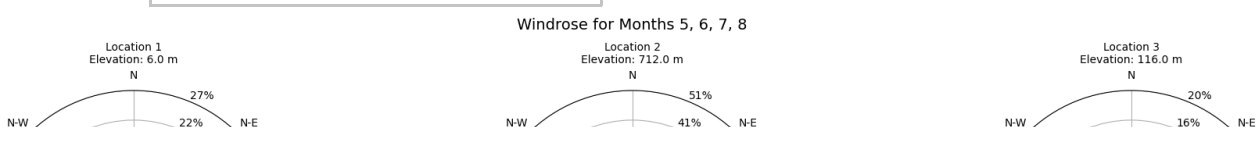
```
# Statistik Wind Speed per Lokasi
wind_stats = dtrain.groupby('location_id')['wind_speed_10m_max (km/h)'].agg(['m
wind_stats.T
```

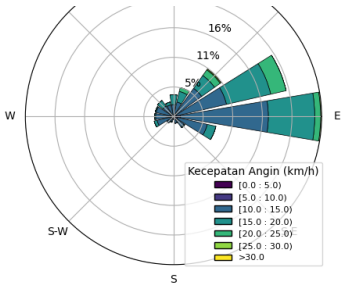
location_id	1	2	3	4	5	6	7	8	9	10	11	12
min	5.9	5.6	5.5	4.8	6.2	4.2	6.2	4.3	4.3	4.9	5.9	5.4
max	27.2	24.6	26.6	30.9	38.0	31.0	37.2	18.4	26.0	28.9	29.7	34.6
median	14.0	11.4	13.1	16.3	14.8	14.8	15.8	9.3	10.3	11.1	14.4	15.6

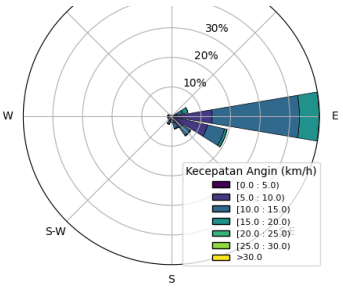
Berdasarkan plot windrose di atas, rentang kecepatan angin terlihat relatif serupa di berbagai elevasi. Tidak ada tren kuat yang mengindikasikan bahwa angin di lokasi tinggi selalu lebih cepat dibanding lokasi rendah. Lalu, beberapa lokasi memiliki distribusi pola angin yang tersebar ke banyak arah, sementara lokasi lainnya menunjukkan pola angin yang terpusat ke satu arah.

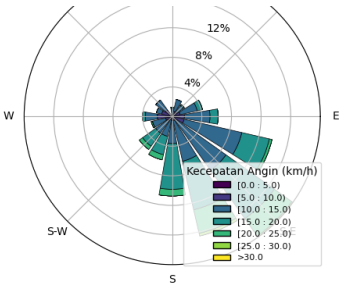
```
# Membandingkan Windrose untuk bulan-bulan tertentu

# Filter data
# Plot 1: Bulan 11, 12, 1, 2
plot1_data = dtrain[dtrain['month'].isin([11, 12, 1, 2])] # bulan dengan rain_s

# Plot 2: Bulan 5, 6, 7, 8
plot2_data = dtrain[dtrain['month'].isin([5, 6, 7, 8])] # bulan dengan rain_sum

# Fungsi untuk plot berdasarkan pilihan dropdown
def plot_based_on_selection(plot_selection):
    if plot_selection == 'Plot 1 (Nov-Feb)':
        plot_windrose(plot1_data, "Windrose for Months 11, 12, 1, 2\n")
    elif plot_selection == 'Plot 2 (May-Aug)':
        plot_windrose(plot2_data, "Windrose for Months 5, 6, 7, 8\n")

# Dropdown untuk memilih plot
dropdown = widgets.Dropdown(
    options=['Plot 1 (Nov-Feb)', 'Plot 2 (May-Aug)'],
    value='Plot 1 (Nov-Feb)',  # Default value
    description='Choose Plot:',
)

# Tampilkan dropdown dan Plot
widgets.interactive(plot_based_on_selection, plot_selection=dropdown)
```

Choose Plot: Plot 2 (May-Aug)

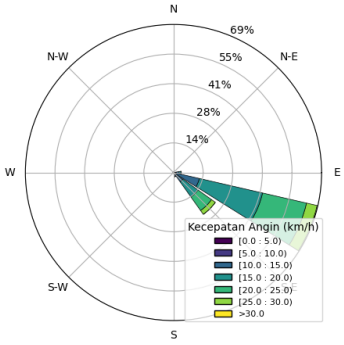




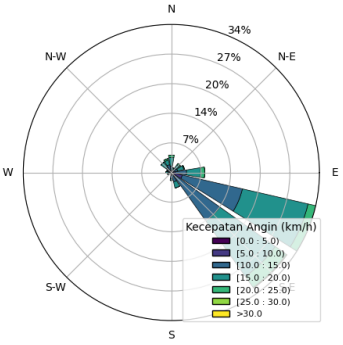




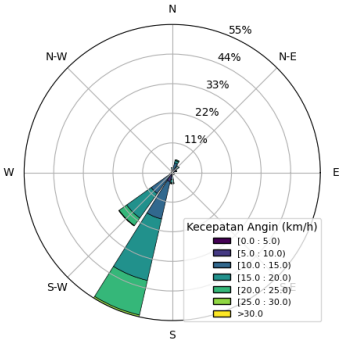
Location 4
Elevation: 8.0 m



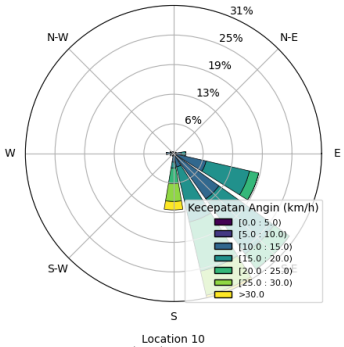
Location 5
Elevation: 4.0 m



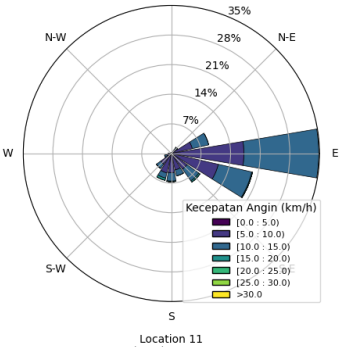
Location 6
Elevation: 96.0 m



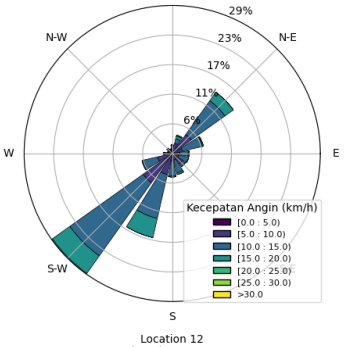
Location 7
Elevation: 5.0 m



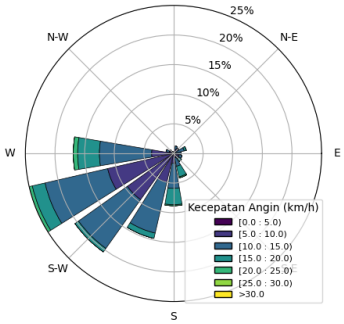
Location 8
Elevation: 74.0 m



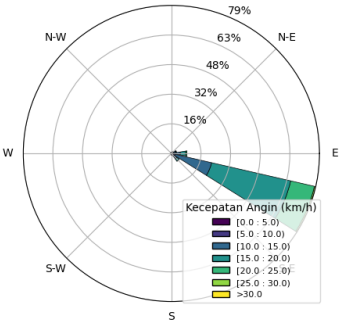
Location 9
Elevation: 451.0 m



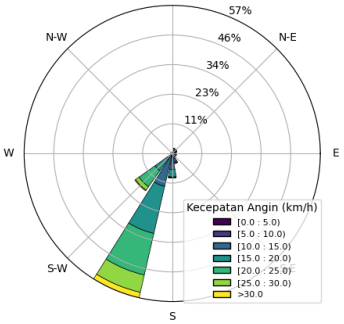
Location 10
Elevation: 92.0 m



Location 11
Elevation: 4.0 m



Location 12
Elevation: 66.0 m



Mengasumsikan `wind direction` atau arah angin yang dimaksud adalah "Dari mana angin bertiup", terlihat pola bahwa pada bulan basah (rain_sum cukup tinggi) angin cenderung bertiup dari arah Barat laut-Barat daya, kecuali untuk beberapa lokasi seperti lokasi 8 dan 9. Sementara itu, pada bulan kering (rain_sum cukup rendah) angin cenderung bertiup dari Timur laut-Tenggara, kecuali untuk beberapa lokasi.

Pola ini sesuai dengan pola **angin muson**. Secara umum, angin muson adalah fenomena atmosfer yang terjadi karena adanya perbedaan tekanan udara antara daratan dan lautan yang berganti arah setiap setengah tahun/6 bulan. Pada periode muson barat (Oktober–Maret), angin bergerak dari belahan bumi utara menuju selatan, membawa uap air dari lautan menuju daratan di wilayah tropis. Pola ini biasanya menghasilkan musim hujan, ditandai dengan curah hujan yang cukup tinggi.

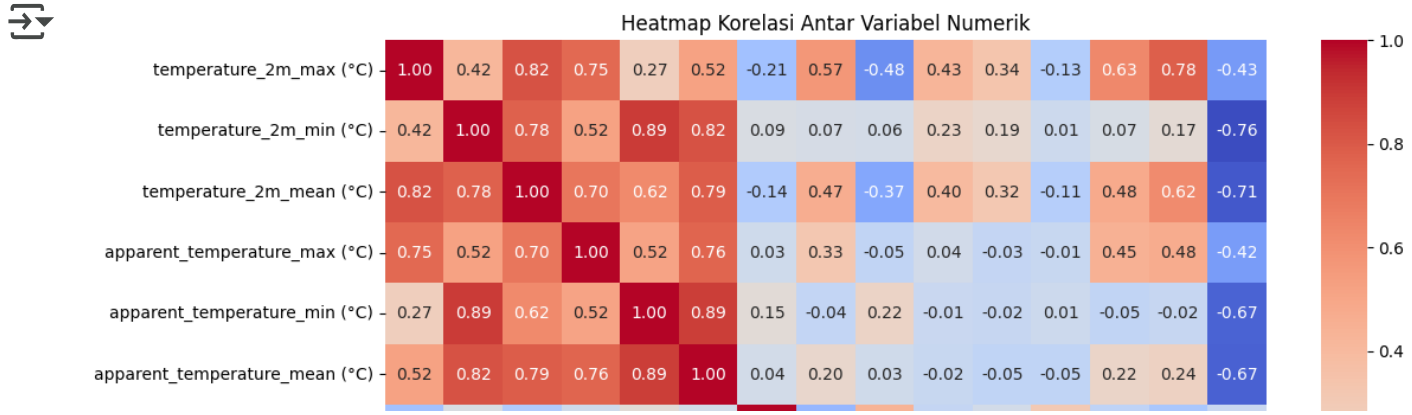
Sebaliknya, pada periode muson timur (April–September), arah angin membawa udara kering dari wilayah subtropis menuju tropis. Angin ini menyebabkan musim kemarau di beberapa wilayah tropis.

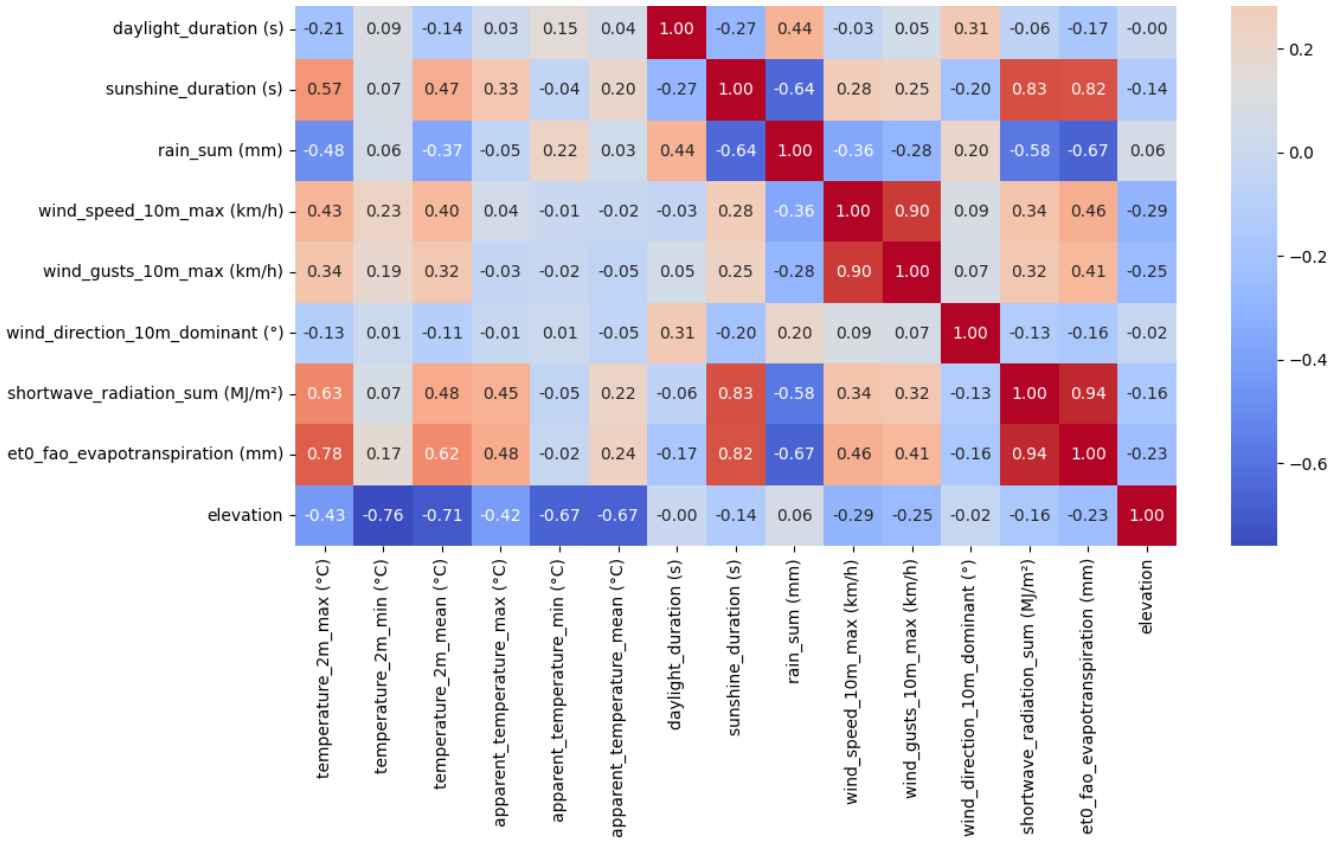
Pola angin pada wind rose di atas yang kurang sesuai dengan angin muson menandakan bahwa pada lokasi-lokasi tersebut angin juga dipengaruhi oleh pola angin lokal.

Sumber : [1] [2]

Numeric Feature Correlation

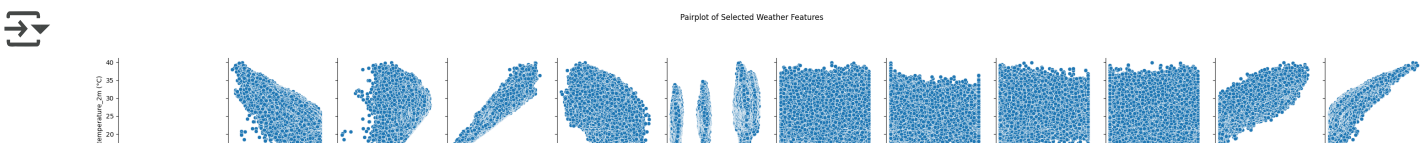
```
# Pilih kolom numerik saja
numeric_columns = dtrain.select_dtypes(include='float').columns

# Hitung matriks korelasi
correlation_matrix = dtrain[numeric_columns].corr(method='spearman')

# Plot heatmap korelasi
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm', cbar=Tr
plt.title('Heatmap Korelasi Antar Variabel Numerik')
plt.show()
```

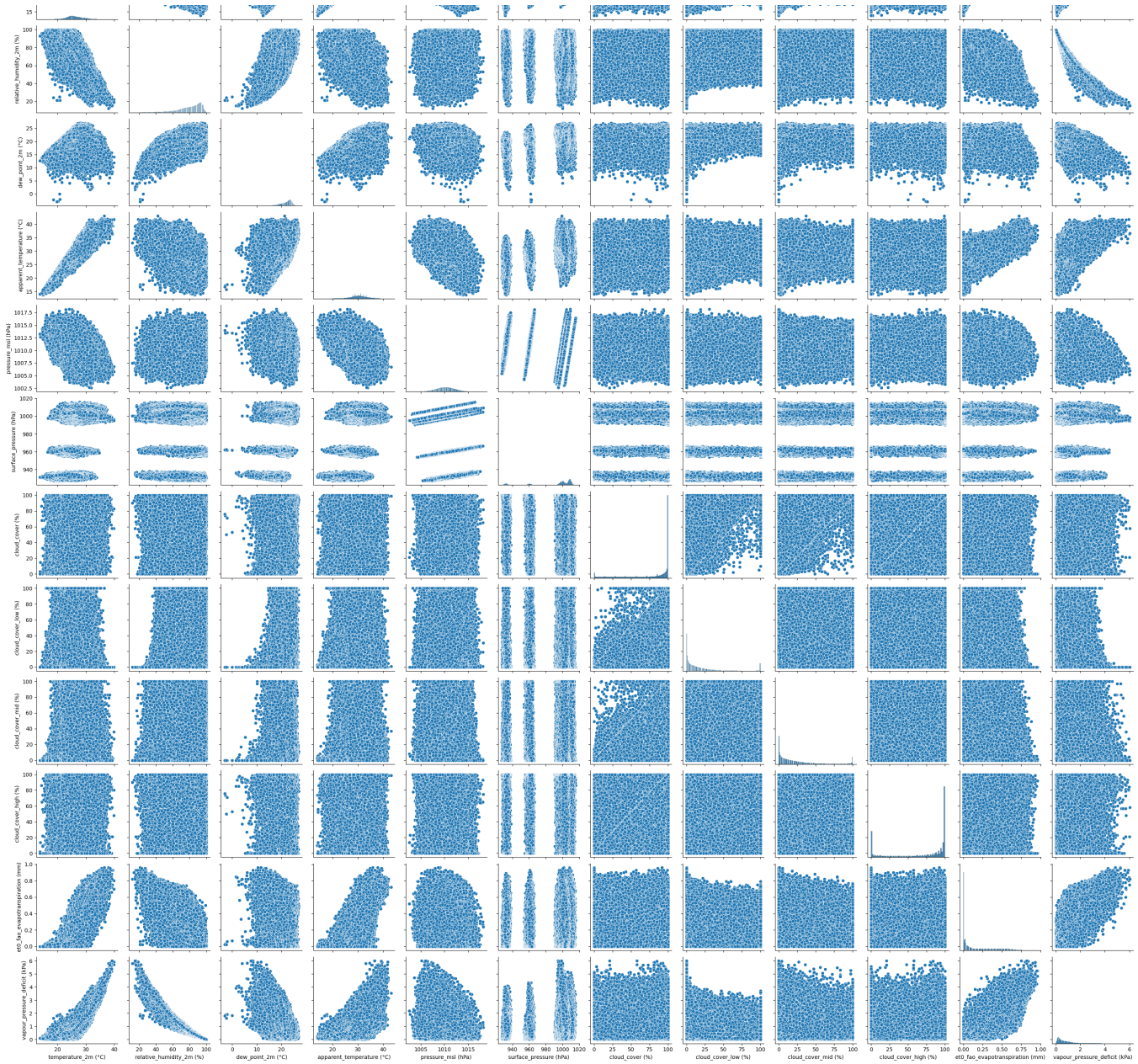



daylight_duration (s) -0.21 0.09 -0.14 0.03 0.15 0.04 1.00 -0.27 0.44 -0.03 0.05 0.31 -0.06 -0.17 -0.00
sunshine_duration (s) 0.57 0.07 0.47 0.33 -0.04 0.20 -0.27 1.00 -0.64 0.28 0.25 -0.20 0.83 0.82 -0.14
rain_sum (mm) -0.48 0.06 -0.37 -0.05 0.22 0.03 0.44 -0.64 1.00 -0.36 -0.28 0.20 -0.58 -0.67 0.06
wind_speed_10m_max (km/h) 0.43 0.23 0.40 0.04 -0.01 -0.02 -0.03 0.28 -0.36 1.00 0.90 0.09 0.34 0.46 -0.29
wind_gusts_10m_max (km/h) 0.34 0.19 0.32 -0.03 -0.02 -0.05 0.05 0.25 -0.28 0.90 1.00 0.07 0.32 0.41 -0.25
wind_direction_10m_dominant (°) -0.13 0.01 -0.11 -0.01 0.01 -0.05 0.31 -0.20 0.20 0.09 0.07 1.00 -0.13 -0.16 -0.02
shortwave_radiation_sum (MJ/m²) 0.63 0.07 0.48 0.45 -0.05 0.22 -0.06 0.83 -0.58 0.34 0.32 -0.13 1.00 0.94 -0.16
et0_fao_evapotranspiration (mm) 0.78 0.17 0.62 0.48 -0.02 0.24 -0.17 0.82 -0.67 0.46 0.41 -0.16 0.94 1.00 -0.23
elevation -0.43 -0.76 -0.71 -0.42 -0.67 -0.67 -0.00 -0.14 0.06 -0.29 -0.25 -0.02 -0.16 -0.23 1.00
temperature_2m_max (°C)
temperature_2m_min (°C)
temperature_2m_mean (°C)
apparent_temperature_max (°C)
apparent_temperature_min (°C)
apparent_temperature_mean (°C)
daylight_duration (s)
sunshine_duration (s)
rain_sum (mm)
wind_speed_10m_max (km/h)
wind_gusts_10m_max (km/h)
wind_direction_10m_dominant (°)
shortwave_radiation_sum (MJ/m²)
et0_fao_evapotranspiration (mm)
elevation
0.2
0.0
−0.2
−0.4
−0.6

- Suhu: Terdapat korelasi kuat (positif) antar variabel suhu, ini mengindikasikan bahwa suhu maksimum, minimum, dan rata-rata cenderung bergerak/berubah bersamaan.
- Radiasi Matahari & Evapotranspirasi: Korelasi positif yang sangat kuat antara radiasi matahari dan evapotranspirasi menandakan bahwa semakin banyak radiasi matahari yang diterima, semakin tinggi pula evapotranspirasi. Hal ini wajar karena radiasi matahari memang terdapat dalam rumus perhitungan et0 (FAO).
- Elevasi & Variabel Lainnya: Terdapat korelasi negatif antara elevasi dan beberapa variabel lainnya, terutama suhu, radiasi matahari, dan evapotranspirasi. Hal ini mengindikasikan bahwa semakin tinggi elevasi, semakin rendah suhu, radiasi matahari, dan evapotranspirasi, seperti yang telah dijelaskan pada plot sebelumnya.
- Durasi Siang : Terdapat korelasi positif cukup kuat antara durasi siang dan suhu, serta durasi siang dan radiasi matahari, menunjukkan bahwa suhu dan radiasi matahari cenderung lebih tinggi ketika durasi siang lebih panjang.
- Curah Hujan : Terdapat korelasi negatif antara curah hujan dengan suhu, `sunshine duration`, radiasi matahari, dan evapotranspirasi menunjukkan bahwa fitu-fitur tersebut cenderung lebih rendah ketika curah hujan lebih tinggi.

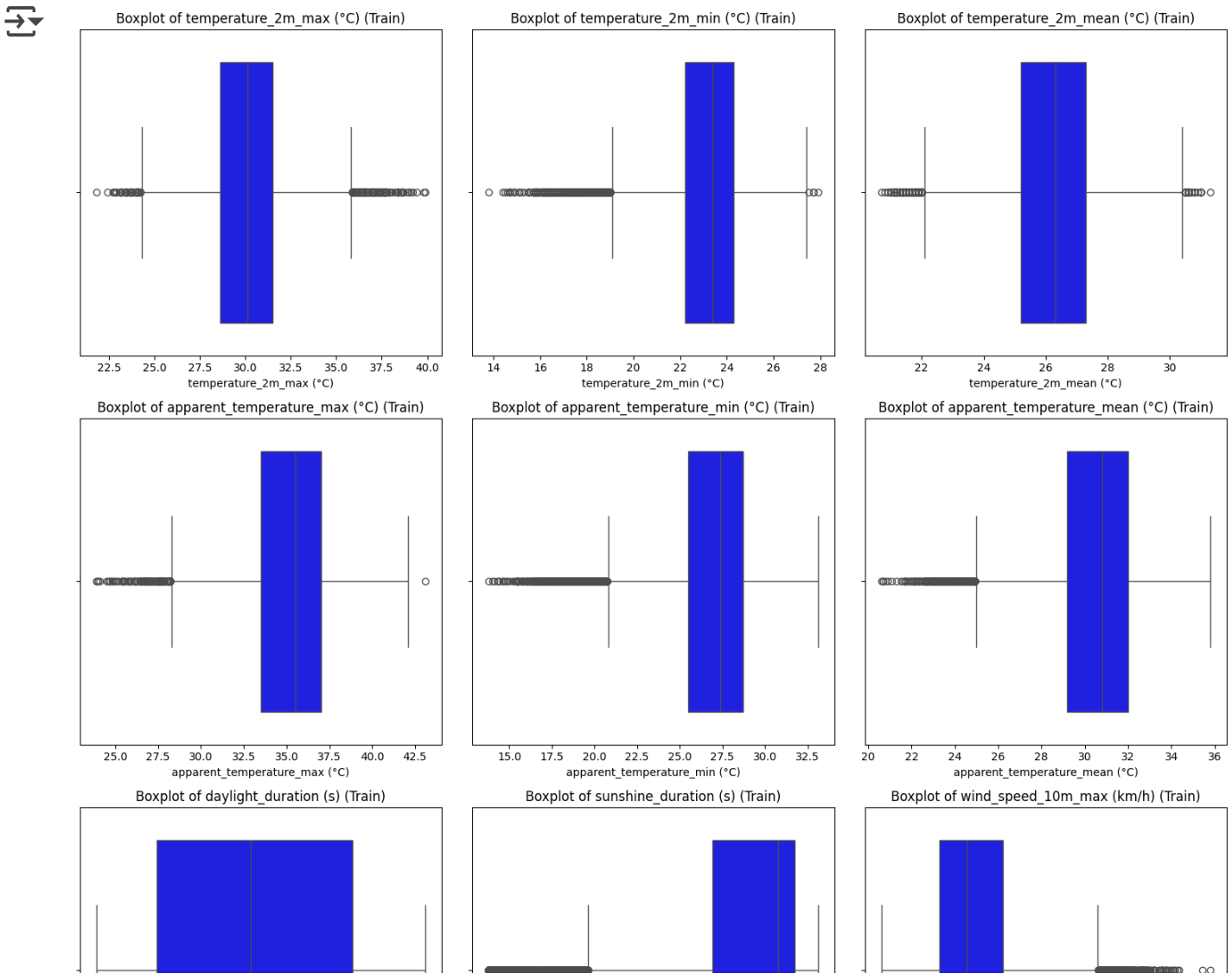
```
# Memilih kolom-kolom yang diinginkan untuk pairplot
cols_to_plot = [
    'temperature_2m (°C)',
    'relative_humidity_2m (%)',
    'dew_point_2m (°C)',
    'apparent_temperature (°C)',
    'pressure_msl (hPa)',
    'surface_pressure (hPa)',
    'cloud_cover (%)',
    'cloud_cover_low (%)',
    'cloud_cover_mid (%)',
    'cloud_cover_high (%)',
    'et0_fao_evapotranspiration (mm)',
    'vapour_pressure_deficit (kPa)'
]

# Membuat pairplot dengan kolom-kolom yang dipilih
sns.pairplot(hourly_train[cols_to_plot])

# Menambahkan judul
plt.suptitle('Pairplot of Selected Weather Features', y=1.02)

# Menampilkan plot
plt.show()
```

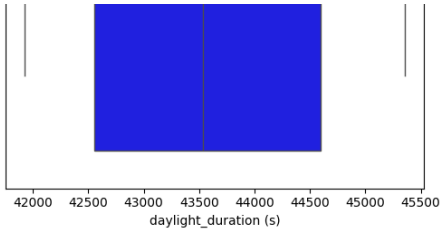


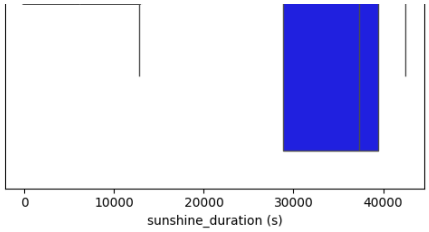

relative_humidity_2m (%)
dew_point_2m (°C)
apparent_temperature (°C)
pressure_msl (hPa)
surface_pressure (hPa)
cloud_cover (%)
cloud_cover_low (%)
cloud_cover_mid (%)
cloud_cover_high (%)
et0_fao_evapotranspiration (mm)
vapour_pressure_deficit (kPa)
temperature_2m (°C)
relative_humidity_2m (%)
dew_point_2m (°C)
apparent_temperature (°C)
pressure_msl (hPa)
surface_pressure (hPa)
cloud_cover (%)
cloud_cover_low (%)
cloud_cover_mid (%)
cloud_cover_high (%)
et0_fao_evapotranspiration (mm)
vapour_pressure_deficit (kPa)

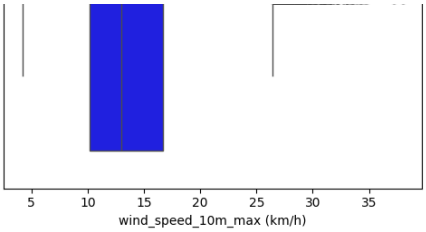
Outlier

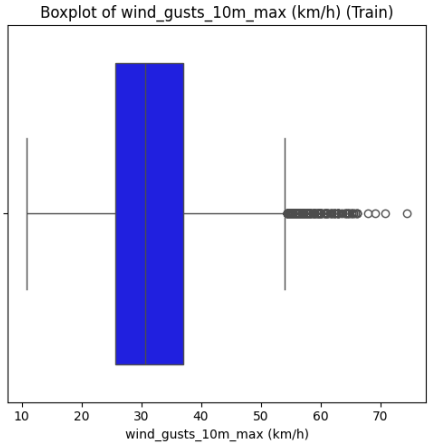
```
# Boxplot fitur pada data daily_train
# Pilih kolom numerik yang ada di kedua dataset
numeric_columns = [col for col in daily_train.select_dtypes(include='float').co

# Atur grid untuk visualisasi
num_cols = 3  # Jumlah kolom dalam grid
num_rows = (len(numeric_columns) + num_cols - 1) // num_cols  # Hitung jumlah b

# Boxplot visualization for train data only
plt.figure(figsize=(15, 5 * num_rows))  # Adjust the figure size for train data

for i, column in enumerate(numeric_columns, 1):
    plt.subplot(num_rows, num_cols, i)  # Subplot for each column
    sns.boxplot(data=daily_train, x=column, color='blue')
    plt.title(f'Boxplot of {column} (Train)')
    plt.xlabel(column)

plt.tight_layout()  # Avoid overlap between subplots
plt.show()
```

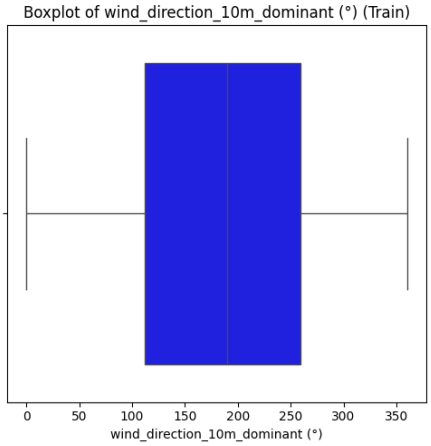


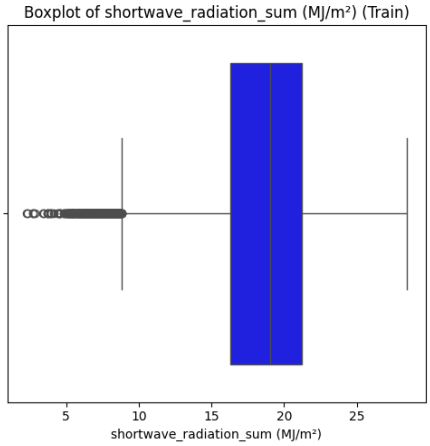


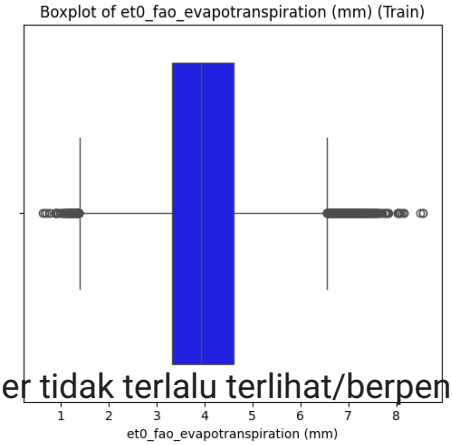












Outlier tidak terlalu terlihat/berpengaruh.

Merge

```
# Konversi ke satuan MJ/m²
sum_col = [
    'direct_radiation_instant (W/m²)', 'diffuse_radiation_instant (W/m²)',
    'direct_normal_irradiance_instant (W/m²)', 'terrestrial_radiation_instant (

for col in sum_col:
    new_col_name = col.replace(' (W/m²)', ' (MJ/m²)')  # Mengubah nama kolom
    hourly_train[new_col_name] = (hourly_train[col] * 3600) / 1_000_000

# Hitung sum per hari
sum_col2 = [
    'direct_radiation_instant (MJ/m²)', 'diffuse_radiation_instant (MJ/m²)',
    'direct_normal_irradiance_instant (MJ/m²)', 'terrestrial_radiation_instant

hourly_train['time'] = pd.to_datetime(hourly_train['time'])
hourly_train['date'] = hourly_train['time'].dt.date

hourly_sum = hourly_train.groupby(['location_id', 'date'])[sum_col2].sum()
hourly_sum.reset_index(inplace=True)


# Pengecualian kolom yang dijumlahkan
exclude_columns = [
    'direct_radiation_instant (MJ/m²)','diffuse_radiation_instant (MJ/m²)',
    'direct_normal_irradiance_instant (MJ/m²)','terrestrial_radiation_instant (
    'direct_radiation_instant (W/m²)', 'diffuse_radiation_instant (W/m²)',
    'direct_normal_irradiance_instant (W/m²)', 'terrestrial_radiation_instant (
    'shortwave_radiation_instant (W/m²)'
]

numeric_columns = hourly_train.select_dtypes(include=['float64']).columns
numeric_columns_to_include = [col for col in numeric_columns if col not in excl

# Definisikan fungsi agregasi
aggregation_functions = {col: ['mean', 'max', 'min', 'std'] for col in numeric_

# Agregasi hourly_train menjadi harian
hourly_aggregated = hourly_train.groupby(['location_id', 'date']).agg(aggregati

# Flatten MultiIndex columns
hourly_aggregated.columns = ['_'.join(col).strip() for col in hourly_aggregated
hourly_aggregated.reset_index(inplace=True)


# Merge hourly_sum and hourly_aggregated
hourly_agg = pd.merge(hourly_aggregated, hourly_sum, on=['location_id', 'date']

daily_train['time'] = pd.to_datetime(daily_train['time'])
hourly_agg['date'] = pd.to_datetime(hourly_agg['date'])

daily_combined = pd.merge(daily_train, hourly_agg,
```

```
                            how='left',
                            left_on=['location_id', 'time'],
                            right_on=['location_id', 'date'])

train = pd.merge(daily_combined, loc, how='left', on='location_id')

train = train.drop(columns=['date', # drop kolom redundant
                            'temperature_2m (°C)_mean', 'temperature_2m (°C)_ma
                            'apparent_temperature (°C)_mean',   'apparent_tempe
                            'wind_speed_10m (km/h)_max', 'wind_gusts_10m (km/h)
train
```

	ID	location_id	time	temperature_2m_max (°C)	temperature_2m_m (°
0	loc1_20190101	1	2019-01-01	26.8	2
1	loc1_20190102	1	2019-01-02	30.5	2
2	loc1_20190103	1	2019-01-03	30.1	2
3	loc1_20190104	1	2019-01-04	31.8	2
4	loc1_20190105	1	2019-01-05	30.7	2
...	...	...	...	...	
19699	loc12_20230626	12	2023-06-26	33.3	2
19700	loc12_20230627	12	2023-06-27	32.9	2
19701	loc12_20230628	12	2023-06-28	33.1	2
19702	loc12_20230629	12	2023-06-29	33.5	2
19703	loc12_20230630	12	2023-06-30	33.5	2

19704 rows × 118 columns

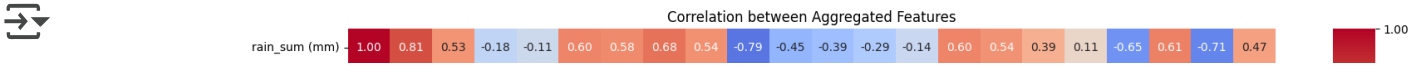
```
train.columns
```

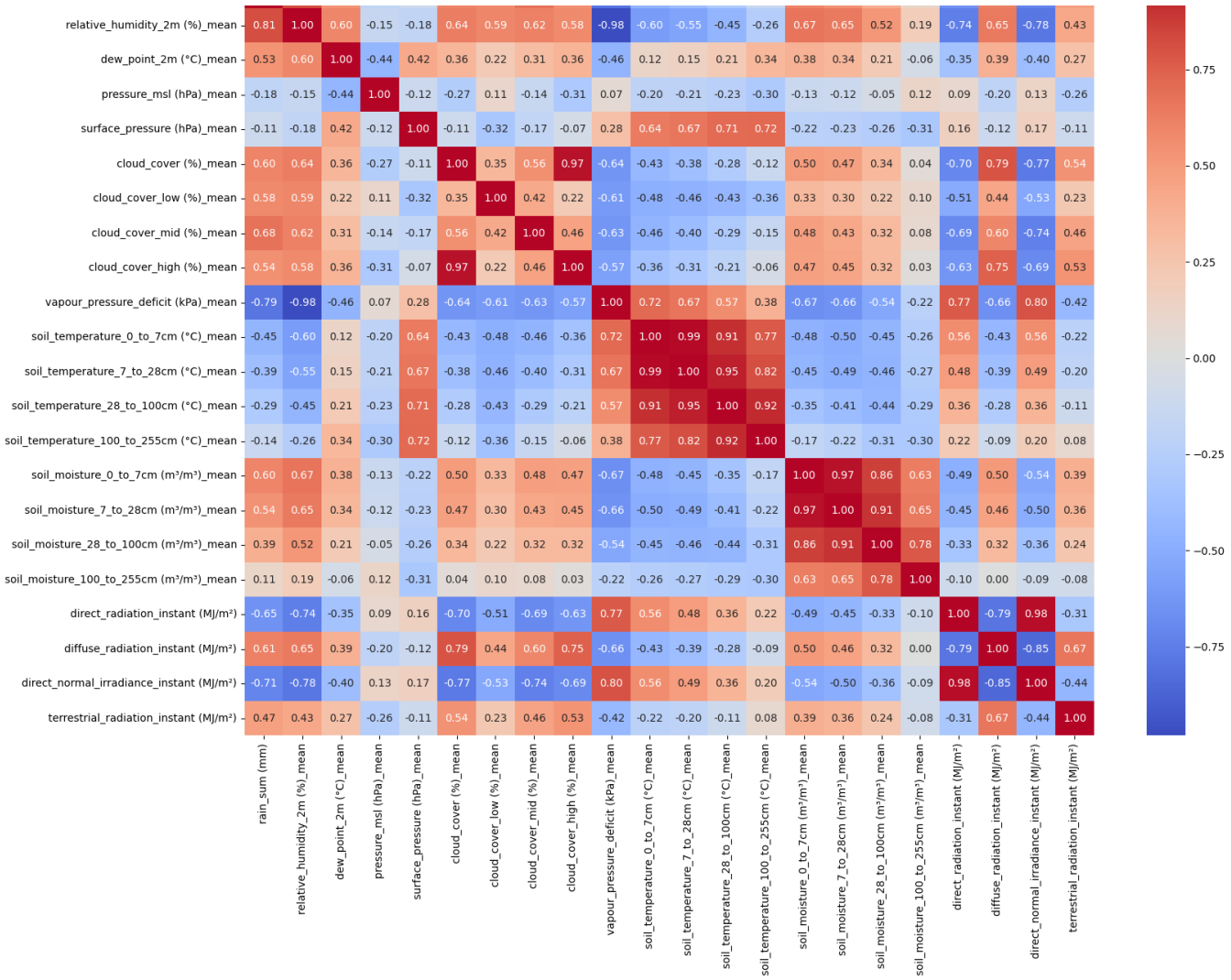
```
Index(['ID', 'location_id', 'time', 'temperature_2m_max (°C)',
       'temperature_2m_min (°C)', 'temperature_2m_mean (°C)',
       'apparent_temperature_max (°C)', 'apparent_temperature_min (°C)',
       'apparent_temperature_mean (°C)', 'sunrise (iso8601)',
       ...
       'soil_moisture_100_to_255cm (m³/m³)_max',
       'soil_moisture_100_to_255cm (m³/m³)_min',
       'soil_moisture_100_to_255cm (m³/m³)_std',
       'direct_radiation_instant (MJ/m²)', 'diffuse_radiation_instant
(MJ/m²)',
       'direct_normal_irradiance_instant (MJ/m²)',
       'terrestrial_radiation_instant (MJ/m²)', 'x', 'y', 'elevation'],
      dtype='object', length=118)
```

```
# Mengecek jumlah nilai kosong di setiap kolom
missing_values = train.isnull().sum()

# Menampilkan jumlah nilai kosong untuk semua kolom
print("Jumlah Nilai Kosong di Setiap Kolom:")
print(missing_values[missing_values > 0])  # Hanya menampilkan kolom yang memil

# Menampilkan jumlah total nilai kosong di seluruh dataset
total_missing = train.isnull().sum().sum()
print(f"\nTotal Nilai Kosong di Dataset: {total_missing}")
```

```
Jumlah Nilai Kosong di Setiap Kolom:
Series([], dtype: int64)

Total Nilai Kosong di Dataset: 0
```

```
# Matrix Korelasi rain_sum dengan fitur hasil agregasi
correlation_matrix = train[['rain_sum (mm)', 'relative_humidity_2m (%)_mean',
                            'dew_point_2m (°C)_mean', 'pressure_msl (hPa)_mean
                            'cloud_cover (%)_mean',    'cloud_cover_low (%)_me
                            'cloud_cover_high (%)_mean', 'vapour_pressure_defi
                            'soil_temperature_7_to_28cm (°C)_mean', 'soil_temp
                            'soil_temperature_100_to_255cm (°C)_mean', 'soil_m
                            'soil_moisture_7_to_28cm (m³/m³)_mean', 'soil_mois
                            'soil_moisture_100_to_255cm (m³/m³)_mean','direct_
                           'diffuse_radiation_instant (MJ/m²)', 'direct_normal
                           'terrestrial_radiation_instant (MJ/m²)']].corr(meth

# Heatmap
plt.figure(figsize=(18, 13))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation between Aggregated Features')
plt.show()
```




relative_humidity_2m (%)_mean 0.81 1.00 0.60 -0.15 -0.18 0.64 0.59 0.62 0.58 -0.98 -0.60 -0.55 -0.45 -0.26 0.67 0.65 0.52 0.19 -0.74 0.65 -0.78 0.43
dew_point_2m (°C)_mean 0.53 0.60 1.00 -0.44 0.42 0.36 0.22 0.31 0.36 -0.46 0.12 0.15 0.21 0.34 0.38 0.34 0.21 -0.06 -0.35 0.39 -0.40 0.27
pressure_msl (hPa)_mean -0.18 -0.15 -0.44 1.00 -0.12 -0.27 0.11 -0.14 -0.31 0.07 -0.20 -0.21 -0.23 -0.30 -0.13 -0.12 -0.05 0.12 0.09 -0.20 0.13 -0.26
surface_pressure (hPa)_mean -0.11 -0.18 0.42 -0.12 1.00 -0.11 -0.32 -0.17 -0.07 0.28 0.64 0.67 0.71 0.72 -0.22 -0.23 -0.26 -0.31 0.16 -0.12 0.17 -0.11
cloud_cover (%)_mean 0.60 0.64 0.36 -0.27 -0.11 1.00 0.35 0.56 0.97 -0.64 -0.43 -0.38 -0.28 -0.12 0.50 0.47 0.34 0.04 -0.70 0.79 -0.77 0.54
cloud_cover_low (%)_mean 0.58 0.59 0.22 0.11 -0.32 0.35 1.00 0.42 0.22 -0.61 -0.48 -0.46 -0.43 -0.36 0.33 0.30 0.22 0.10 -0.51 0.44 -0.53 0.23
cloud_cover_mid (%)_mean 0.68 0.62 0.31 -0.14 -0.17 0.56 0.42 1.00 0.46 -0.63 -0.46 -0.40 -0.29 -0.15 0.48 0.43 0.32 0.08 -0.69 0.60 -0.74 0.46
cloud_cover_high (%)_mean 0.54 0.58 0.36 -0.31 -0.07 0.97 0.22 0.46 1.00 -0.57 -0.36 -0.31 -0.21 -0.06 0.47 0.45 0.32 0.03 -0.63 0.75 -0.69 0.53
vapour_pressure_deficit (kPa)_mean -0.79 -0.98 -0.46 0.07 0.28 -0.64 -0.61 -0.63 -0.57 1.00 0.72 0.67 0.57 0.38 -0.67 -0.66 -0.54 -0.22 0.77 -0.66 0.80 -0.42
soil_temperature_0_to_7cm (°C)_mean -0.45 -0.60 0.12 -0.20 0.64 -0.43 -0.48 -0.46 -0.36 0.72 1.00 0.99 0.91 0.77 -0.48 -0.50 -0.45 -0.26 0.56 -0.43 0.56 -0.22
soil_temperature_7_to_28cm (°C)_mean -0.39 -0.55 0.15 -0.21 0.67 -0.38 -0.46 -0.40 -0.31 0.67 0.99 1.00 0.95 0.82 -0.45 -0.49 -0.46 -0.27 0.48 -0.39 0.49 -0.20
soil_temperature_28_to_100cm (°C)_mean -0.29 -0.45 0.21 -0.23 0.71 -0.28 -0.43 -0.29 -0.21 0.57 0.91 0.95 1.00 0.92 -0.35 -0.41 -0.44 -0.29 0.36 -0.28 0.36 -0.11
soil_temperature_100_to_255cm (°C)_mean -0.14 -0.26 0.34 -0.30 0.72 -0.12 -0.36 -0.15 -0.06 0.38 0.77 0.82 0.92 1.00 -0.17 -0.22 -0.31 -0.30 0.22 -0.09 0.20 0.08
soil_moisture_0_to_7cm (m³/m³)_mean 0.60 0.67 0.38 -0.13 -0.22 0.50 0.33 0.48 0.47 -0.67 -0.48 -0.45 -0.35 -0.17 1.00 0.97 0.86 0.63 -0.49 0.50 -0.54 0.39
soil_moisture_7_to_28cm (m³/m³)_mean 0.54 0.65 0.34 -0.12 -0.23 0.47 0.30 0.43 0.45 -0.66 -0.50 -0.49 -0.41 -0.22 0.97 1.00 0.91 0.65 -0.45 0.46 -0.50 0.36
soil_moisture_28_to_100cm (m³/m³)_mean 0.39 0.52 0.21 -0.05 -0.26 0.34 0.22 0.32 0.32 -0.54 -0.45 -0.46 -0.44 -0.31 0.86 0.91 1.00 0.78 -0.33 0.32 -0.36 0.24
soil_moisture_100_to_255cm (m³/m³)_mean 0.11 0.19 -0.06 0.12 -0.31 0.04 0.10 0.08 0.03 -0.22 -0.26 -0.27 -0.29 -0.30 0.63 0.65 0.78 1.00 -0.10 0.00 -0.09 -0.08
direct_radiation_instant (MJ/m²) -0.65 -0.74 -0.35 0.09 0.16 -0.70 -0.51 -0.69 -0.63 0.77 0.56 0.48 0.36 0.22 -0.49 -0.45 -0.33 -0.10 1.00 -0.79 0.98 -0.31
diffuse_radiation_instant (MJ/m²) 0.61 0.65 0.39 -0.20 -0.12 0.79 0.44 0.60 0.75 -0.66 -0.43 -0.39 -0.28 -0.09 0.50 0.46 0.32 0.00 -0.79 1.00 -0.85 0.67
direct_normal_irradiance_instant (MJ/m²) -0.71 -0.78 -0.40 0.13 0.17 -0.77 -0.53 -0.74 -0.69 0.80 0.56 0.49 0.36 0.20 -0.54 -0.50 -0.36 -0.09 0.98 -0.85 1.00 -0.44
terrestrial_radiation_instant (MJ/m²) 0.47 0.43 0.27 -0.26 -0.11 0.54 0.23 0.46 0.53 -0.42 -0.22 -0.20 -0.11 0.08 0.39 0.36 0.24 -0.08 -0.31 0.67 -0.44 1.00
rain_sum (mm)
relative_humidity_2m (%)_mean
dew_point_2m (°C)_mean
pressure_msl (hPa)_mean
surface_pressure (hPa)_mean
cloud_cover (%)_mean
cloud_cover_low (%)_mean
cloud_cover_mid (%)_mean
cloud_cover_high (%)_mean
vapour_pressure_deficit (kPa)_mean
soil_temperature_0_to_7cm (°C)_mean
soil_temperature_7_to_28cm (°C)_mean
soil_temperature_28_to_100cm (°C)_mean
soil_temperature_100_to_255cm (°C)_mean
soil_moisture_0_to_7cm (m³/m³)_mean
soil_moisture_7_to_28cm (m³/m³)_mean
soil_moisture_28_to_100cm (m³/m³)_mean
soil_moisture_100_to_255cm (m³/m³)_mean
direct_radiation_instant (MJ/m²)
diffuse_radiation_instant (MJ/m²)
direct_normal_irradiance_instant (MJ/m²)
terrestrial_radiation_instant (MJ/m²)
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Korelasi antar fitur :

- `vapour_pressure_deficit (kPa)_mean` dan `relative_humidity_2m (%)_mean` **Korelasi negatif yang sangat kuat** (-0.98) menunjukkan bahwa semakin rendah defisit tekanan uap, semakin tinggi kelembapan relatif. Hal ini masuk akal karena defisit tekanan uap adalah ukuran perbedaan antara jumlah uap air yang dapat ditampung udara dan jumlah uap air yang ada saat ini. Ketika kelembapan relatif mendekati 100% (udara jenuh), defisit tekanan uap akan mendekati nol.
- `soil_moisture_7_to_28cm_mean` dan `relative_humidity_2m (%)_mean` (r = 0.65)
  Kelembapan tanah yang tinggi berkaitan erat dengan kelembapan udara di atasnya. Ketika tanah memiliki kandungan air yang tinggi, terjadi proses evaporasi yang menghasilkan uap air di atmosfer, meningkatkan kelembapan relatif.
- Awan pada ketinggian menengah seringkali menunjukkan akumulasi uap air yang cukup untuk membentuk presipitasi. Awan jenis ini sering terlibat dalam proses hujan lebat karena mereka mengandung cukup banyak tetes air untuk bergabung dan jatuh sebagai hujan.
- `soil_temperature_7_to_28cm_mean` dan `soil_moisture_7_to_28cm_mean` (r = -0.66)
  Ketika suhu tanah meningkat, kandungan air tanah pada lapisan tersebut cenderung menurun karena terjadinya proses penguapan. Suhu yang tinggi menyebabkan air di tanah menguap lebih cepat, sehingga kelembapan tanah juga akan berkurang.
- `cloud_cover_low (%)_mean` dan `relative_humidity_2m (%)_mean` (r = 0.58)
  Kelembapan udara yang tinggi cenderung mendukung pembentukan awan pada lapisan rendah. Ketika udara berada di dekat permukaan jenuh dengan uap air, uap ini akan mengembun membentuk awan rendah seperti stratus atau kabut.

Korelasi Fitur dengan Target (`rain_sum (mm)`)

- `relative_humidity_2m (%)_mean` (r = 0.81)
  Semakin tinggi kelembapan relatif udara, semakin banyak hujan yang terjadi. Hal ini masuk akal karena kelembapan relatif yang tinggi menunjukkan bahwa udara sudah hampir jenuh dengan uap air, yang bisa mengarah pada kondensasi dan hujan jika ada faktor lain yang mendukung (seperti pergerakan udara atau kondisi cuaca tertentu). ambil dr docs
- `dew_point_2m (°C)_mean` (r = 0.53)
  semakin tinggi titik embun (dew point), semakin banyak hujan yang terjadi. Titik embun adalah suhu dimana udara harus didinginkan agar uap air mengembun menjadi cairan. Ketika titik embun tinggi, berarti udara lebih banyak mengandung uap air yang bisa menghasilkan hujan jika ada faktor kondensasi lainnya.
- `cloud_cover_mid (%)_mean` (r = 0.68)
  semakin banyak awan dengan ketinggian menengah (cloud cover mid), semakin tinggi jumlah hujan yang terjadi. Awan menengah (seperti altostratus) sering mengandung lebih banyak uap air yang dapat mengarah pada hujan lebat atau hujan sedang.
- `vapour_pressure_deficit (kPa)_mean` (r = -0.79)
  ketika defisit tekanan uap tinggi (udara lebih kering), curah hujan cenderung lebih rendah. Defisit tekanan uap adalah selisih antara kapasitas udara untuk menahan uap air dan jumlah uap air yang sebenarnya ada di udara. Ketika defisit tekanan uap tinggi, itu menunjukkan bahwa udara lebih kering dan kurang kondusif untuk hujan.

Kelembapan udara, kondisi awan, dan defisit tekanan uap memiliki hubungan yang kuat dengan jumlah curah hujan yang tercatat. Umumnya, semakin banyak uap air yang ada di udara (terlihat dari kelembapan relatif yang tinggi atau titik embun yang tinggi), semakin tinggi kemungkinan terjadinya hujan. Selain itu, keberadaan awan, baik rendah, menengah, atau tinggi, juga memainkan peran penting dalam mengindikasikan potensi hujan. Sebaliknya, defisit tekanan uap yang tinggi biasanya terkait dengan kondisi kering dan rendahnya kemungkinan hujan.

Kami memutuskan untuk menghapus outlier dari data Train.

```
# Menghapus outlier pada rain sum

if 'rain_sum (mm)' in train.columns:
    # Filter data dengan rain_sum < 150
    train = train[train['rain_sum (mm)'] < 150]
```

Test

```
daily_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5892 entries, 0 to 5891
Data columns (total 18 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   ID                                 5892 non-null   object 
 1   location_id                        5892 non-null   int64  
 2   time                               5892 non-null   object 
 3   temperature_2m_max (°C)            5892 non-null   float64
 4   temperature_2m_min (°C)            5892 non-null   float64
 5   temperature_2m_mean (°C)           5892 non-null   float64
 6   apparent_temperature_max (°C)      5892 non-null   float64
 7   apparent_temperature_min (°C)      5892 non-null   float64
 8   apparent_temperature_mean (°C)     5892 non-null   float64
 9   sunrise (iso8601)                  5892 non-null   object 
 10  sunset (iso8601)                   5892 non-null   object 
 11  daylight_duration (s)              5892 non-null   float64
 12  sunshine_duration (s)              5890 non-null   float64
 13  wind_speed_10m_max (km/h)          5892 non-null   float64
 14  wind_gusts_10m_max (km/h)          5892 non-null   float64
 15  wind_direction_10m_dominant (°)    5892 non-null   float64
 16  shortwave_radiation_sum (MJ/m²)    5890 non-null   float64
 17  et0_fao_evapotranspiration (mm)    5890 non-null   float64
dtypes: float64(13), int64(1), object(4)
memory usage: 828.7+ KB
```

```
daily_test.head()
```

	ID	location_id	time	temperature_2m_max (°C)	temperature_2m_min (°C)	t
0	loc1_20230701	1	2023-07-01	31.0	24.7	
1	loc1_20230702	1	2023-07-02	30.4	24.8	
2	loc1_20230703	1	2023-07-03	30.6	24.1	
3	loc1_20230704	1	2023-07-04	31.0	23.7	
4	loc1_20230705	1	2023-07-05	32.3	23.8	

```
hourly_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 141408 entries, 0 to 141407
Data columns (total 32 columns):
 #   Column                                  Non-Null Count   Dtype  
---  ------                                  --------------   -----  
 0   location_id                             141408 non-null  int64  
 1   time                                    141408 non-null  object 
 2   temperature_2m (°C)                     141408 non-null  float64
 3   relative_humidity_2m (%)                141408 non-null  float64
 4   dew_point_2m (°C)                       141408 non-null  float64
 5   apparent_temperature (°C)               141408 non-null  float64
 6   pressure_msl (hPa)                      141408 non-null  float64
 7   surface_pressure (hPa)                  141408 non-null  float64
 8   cloud_cover (%)                         141408 non-null  float64
 9   cloud_cover_low (%)                     141408 non-null  float64
 10  cloud_cover_mid (%)                     141408 non-null  float64
 11  cloud_cover_high (%)                    141408 non-null  float64
 12  et0_fao_evapotranspiration (mm)         141388 non-null  float64
 13  vapour_pressure_deficit (kPa)           141408 non-null  float64
 14  wind_speed_10m (km/h)                   141408 non-null  float64
 15  wind_speed_100m (km/h)                  141408 non-null  float64
 16  wind_direction_10m (°)                  141408 non-null  float64
 17  wind_direction_100m (°)                 141408 non-null  float64
 18  wind_gusts_10m (km/h)                   141408 non-null  float64
 19  soil_temperature_0_to_7cm (°C)          141408 non-null  float64
 20  soil_temperature_7_to_28cm (°C)         141408 non-null  float64
 21  soil_temperature_28_to_100cm (°C)       141408 non-null  float64
 22  soil_temperature_100_to_255cm (°C)      141408 non-null  float64
 23  soil_moisture_0_to_7cm (m³/m³)          141408 non-null  float64
 24  soil_moisture_7_to_28cm (m³/m³)         141408 non-null  float64
 25  soil_moisture_28_to_100cm (m³/m³)       141408 non-null  float64
 26  soil_moisture_100_to_255cm (m³/m³)      141408 non-null  float64
 27  shortwave_radiation_instant (W/m²)      141388 non-null  float64
 28  direct_radiation_instant (W/m²)         141387 non-null  float64
 29  diffuse_radiation_instant (W/m²)        141387 non-null  float64
 30  direct_normal_irradiance_instant (W/m²) 141387 non-null  float64
 31  terrestrial_radiation_instant (W/m²)    141408 non-null  float64
dtypes: float64(30), int64(1), object(1)
memory usage: 34.5+ MB
```

```
hourly_test.head()
```

	location_id	time	temperature_2m (°C)	relative_humidity_2m (%)	dew_point_2m (°C)
0	1	2023-07-01 00:00:00	25.7	88.0	23.5
1	1	2023-07-01 01:00:00	25.4	89.0	23.5
2	1	2023-07-01 02:00:00	25.2	90.0	23.3
3	1	2023-07-01 03:00:00	25.0	89.0	23.1
4	1	2023-07-01 04:00:00	24.9	89.0	23.1

Statistic Summary

```
daily_test.describe().T
```

	count	mean	std	min	25%
location_id	5892.0	6.500000	3.452346	1.00	3.7500
temperature_2m_max (°C)	5892.0	31.672794	2.526868	23.70	29.9000
temperature_2m_min (°C)	5892.0	22.863764	2.203253	13.70	21.8000
temperature_2m_mean (°C)	5892.0	26.731908	2.023901	19.70	25.5000
apparent_temperature_max (°C)	5892.0	36.321572	4.153023	25.90	34.5000
apparent_temperature_min (°C)	5892.0	26.360149	3.309636	12.30	24.5000
apparent_temperature_mean (°C)	5892.0	30.564596	2.677379	20.50	28.9000
daylight_duration (s)	5892.0	43489.217838	1010.700076	41924.06	42539.5025
sunshine_duration (s)	5890.0	36356.149888	6662.023244	0.00	35792.1275
wind_speed_10m_max (km/h)	5892.0	15.022115	5.036290	4.30	11.2000
wind_gusts_10m_max (km/h)	5892.0	33.129209	8.034490	15.50	27.0000
wind_direction_10m_dominant (°)	5892.0	172.792261	81.713324	0.00	109.0000

```
hourly_test.describe().T
```

	count	mean	std	min	25
location_id	141408.0	6.500000	3.452065	1.000	3.75
temperature_2m (°C)	141408.0	26.736298	3.618797	13.700	24.30
relative_humidity_2m (%)	141408.0	75.268853	16.555214	18.000	65.00
dew_point_2m (°C)	141408.0	21.492761	2.607763	3.900	19.90
apparent_temperature (°C)	141408.0	30.564277	4.191615	12.300	28.10
pressure_msl (hPa)	141408.0	1011.017280	2.046538	1003.600	1009.60
surface_pressure (hPa)	141408.0	995.668935	22.822709	928.200	998.70
cloud_cover (%)	141408.0	68.845094	36.021262	0.000	36.00
cloud_cover_low (%)	141408.0	15.799693	24.047353	0.000	1.00
cloud_cover_mid (%)	141408.0	14.136279	21.742731	0.000	0.00
cloud_cover_high (%)	141408.0	60.419941	41.453258	0.000	11.00
et0_fao_evapotranspiration (mm)	141388.0	0.193546	0.253840	0.000	0.00

vapour_pressure_deficit (kPa)	141408.0	0.989517	0.863188	0.000	0.33
wind_speed_10m (km/h)	141408.0	8.020065	5.161231	0.000	4.10
wind_speed_100m (km/h)	141408.0	11.930232	7.593405	0.000	6.10
wind_direction_10m (°)	141408.0	182.145522	86.735284	1.000	112.00
wind_direction_100m (°)	141408.0	178.471557	84.586776	0.000	109.00
wind_gusts_10m (km/h)	141408.0	18.758739	10.371804	1.100	10.10
soil_temperature_0_to_7cm (°C)	141408.0	28.700876	4.224373	17.100	25.80
soil_temperature_7_to_28cm (°C)	141408.0	28.654879	2.671294	20.500	26.80
soil_temperature_28_to_100cm (°C)	141408.0	28.444104	2.362760	22.300	26.80
soil_temperature_100_to_255cm (°C)	141408.0	28.196140	2.161322	22.700	26.70
soil_moisture_0_to_7cm (m³/m³)	141408.0	0.233067	0.124423	0.010	0.12
soil_moisture_7_to_28cm (m³/m³)	141408.0	0.225090	0.120796	0.000	0.13
soil_moisture_28_to_100cm (m³/m³)	141408.0	0.224342	0.106782	0.011	0.13
soil_moisture_100_to_255cm (m³/m³)	141408.0	0.278363	0.082457	0.135	0.19
shortwave_radiation_instant	141388.0	239.189554	347.704308	0.000	0.00

Missing Value and Duplicate

```
# Check for duplicate rows and missing values
print("\nNumber of duplicate rows in daily_test:", daily_test.duplicated().sum(
print("Number of duplicate rows in hourly_test:", hourly_test.duplicated().sum(
print("\nMissing values in daily_test:\n", daily_test.isnull().sum())
print("\nMissing values in hourly_test:\n", hourly_test.isnull().sum())
```

```
Number of duplicate rows in daily_test: 0
Number of duplicate rows in hourly_test: 0

Missing values in daily_test:
 ID                                0
location_id                       0
time                              0
temperature_2m_max (°C)           0
temperature_2m_min (°C)           0
```

```
temperature_2m_mean (°C)           0
apparent_temperature_max (°C)      0
apparent_temperature_min (°C)      0
apparent_temperature_mean (°C)     0
sunrise (iso8601)                  0
sunset (iso8601)                   0
daylight_duration (s)              0
sunshine_duration (s)              2
wind_speed_10m_max (km/h)          0
wind_gusts_10m_max (km/h)          0
wind_direction_10m_dominant (°)    0
shortwave_radiation_sum (MJ/m²)    2
et0_fao_evapotranspiration (mm)    2
dtype: int64

Missing values in hourly_test:
 location_id                          0
time                                  0
temperature_2m (°C)                   0
relative_humidity_2m (%)              0
dew_point_2m (°C)                     0
apparent_temperature (°C)             0
pressure_msl (hPa)                    0
surface_pressure (hPa)                0
cloud_cover (%)                       0
cloud_cover_low (%)                   0
cloud_cover_mid (%)                   0
cloud_cover_high (%)                  0
et0_fao_evapotranspiration (mm)      20
vapour_pressure_deficit (kPa)         0
wind_speed_10m (km/h)                 0
wind_speed_100m (km/h)                0
wind_direction_10m (°)                0
wind_direction_100m (°)               0
wind_gusts_10m (km/h)                 0
soil_temperature_0_to_7cm (°C)        0
soil_temperature_7_to_28cm (°C)       0
soil_temperature_28_to_100cm (°C)     0
soil_temperature_100_to_255cm (°C)    0
soil_moisture_0_to_7cm (m³/m³)        0
soil_moisture_7_to_28cm (m³/m³)       0
soil_moisture_28_to_100cm (m³/m³)     0
soil_moisture_100_to_255cm (m³/m³)    0
shortwave_radiation_instant (W/m²)   20
direct_radiation_instant (W/m²)      21
diffuse_radiation_instant (W/m²)     21
direct_normal_irradiance_instant (W/m²)    21
terrestrial_radiation_instant (W/m²)        0
dtype: int64
```

Terdapat 2 row missing value pada data daily_test dan 20/21 row missing value pada data hourly_test.

```
# Baris dengan missing values pada data Daily
missing_rows = daily_test[daily_test.isnull().any(axis=1)]
missing_rows
```

	ID	location_id	time	temperature_2m_max (°C)	temperature_2m_mi (°C
1469	loc3_20241030	3	2024-10-30	31.3	22
4906	loc10_20241030	10	2024-10-30	33.3	23

Missing Value pada daily_test akan diimputasi secara manual dengan melihat data yang serupa pada waktu sebelumnya.

```
# Isi nilai manual untuk missing data di lokasi 3 dan 10
daily_test.loc[(daily_test['location_id'] == 3) & (daily_test['et0_fao_evapotra
daily_test.loc[(daily_test['location_id'] == 3) & (daily_test['sunshine_duratio
daily_test.loc[(daily_test['location_id'] == 3) & (daily_test['shortwave_radiat
daily_test.loc[(daily_test['location_id'] == 10) & (daily_test['et0_fao_evapotr
daily_test.loc[(daily_test['location_id'] == 10) & (daily_test['sunshine_durati
daily_test.loc[(daily_test['location_id'] == 10) & (daily_test['shortwave_radia

# Cek hasil perubahan
missing_rows = daily_test[daily_test.isnull().any(axis=1)]
missing_rows
```

ID	location_id	time	temperature_2m_max (°C)	temperature_2m_min (°C)	temperature_

Untuk data hourly_test, dilakukan imputasi dengan mengisi data berdasarkan rata-rata tiap lokasi.

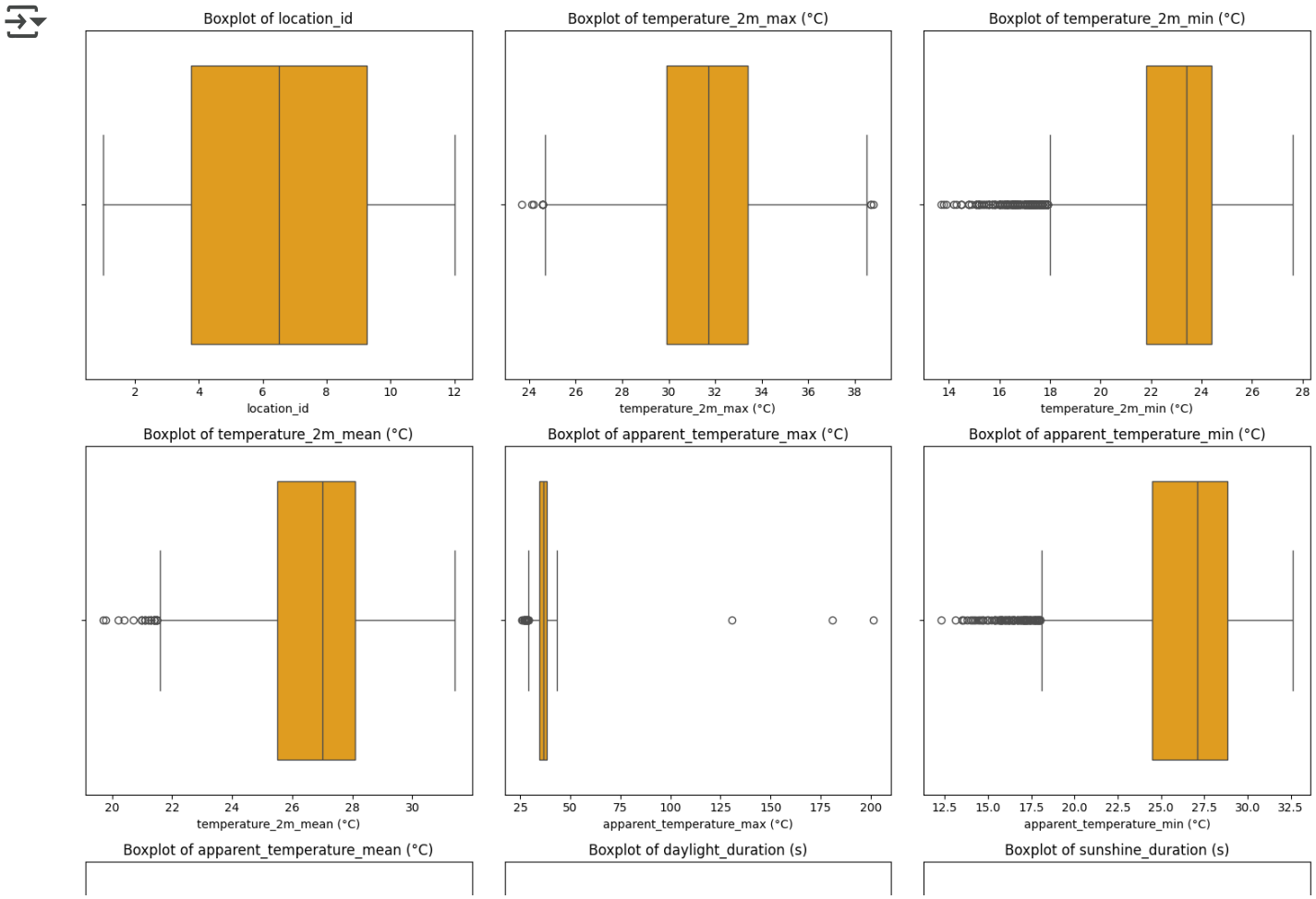
```
# Imputasi NaN berdasarkan rata-rata per lokasi
columns_to_impute = [
    'et0_fao_evapotranspiration (mm)', 'shortwave_radiation_instant (W/m²)',
    'direct_radiation_instant (W/m²)', 'diffuse_radiation_instant (W/m²)',
    'direct_normal_irradiance_instant (W/m²)'
]

# Fungsi untuk imputasi
def impute_with_group_mean(df, column):
    return df.groupby('location_id')[column].transform(lambda x: x.fillna(x.mea

# Mengisi NaN untuk setiap kolom yang perlu diimputasi
for col in columns_to_impute:
    hourly_test[col] = impute_with_group_mean(hourly_test, col)

# Cek hasil imputasi
print(hourly_test.isnull().sum())
```

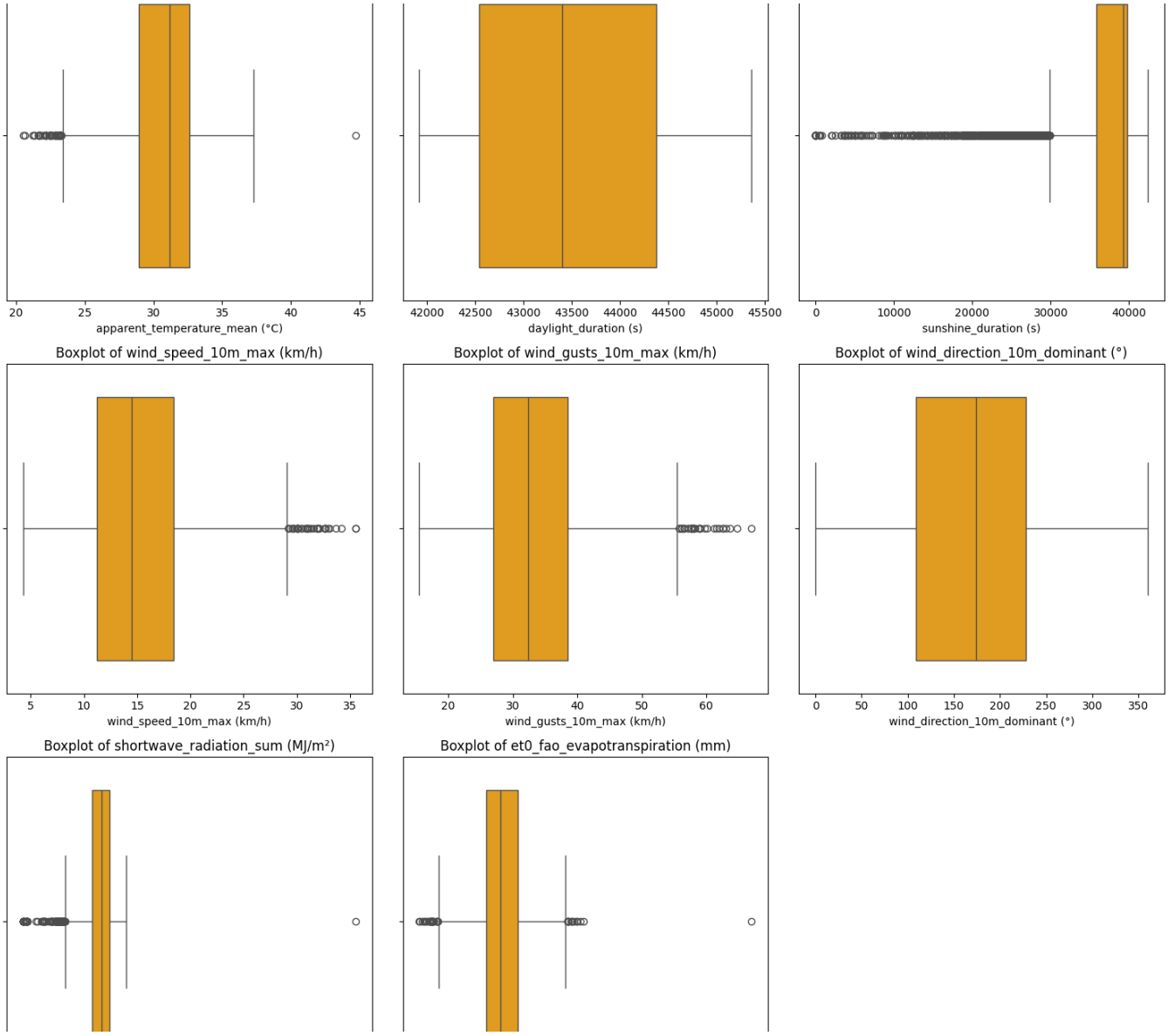
```
location_id                             0
time                                    0
temperature_2m (°C)                     0
relative_humidity_2m (%)                0
dew_point_2m (°C)                       0
apparent_temperature (°C)               0
pressure_msl (hPa)                      0
surface_pressure (hPa)                  0
cloud_cover (%)                         0
cloud_cover_low (%)                     0
cloud_cover_mid (%)                     0
cloud_cover_high (%)                    0
et0_fao_evapotranspiration (mm)         0
vapour_pressure_deficit (kPa)           0
wind_speed_10m (km/h)                   0
wind_speed_100m (km/h)                  0
wind_direction_10m (°)                  0
wind_direction_100m (°)                 0
wind_gusts_10m (km/h)                   0
soil_temperature_0_to_7cm (°C)          0
soil_temperature_7_to_28cm (°C)         0
soil_temperature_28_to_100cm (°C)       0
soil_temperature_100_to_255cm (°C)      0
soil_moisture_0_to_7cm (m³/m³)          0
soil_moisture_7_to_28cm (m³/m³)         0
soil_moisture_28_to_100cm (m³/m³)       0
soil_moisture_100_to_255cm (m³/m³)      0
shortwave_radiation_instant (W/m²)      0
direct_radiation_instant (W/m²)         0
diffuse_radiation_instant (W/m²)        0
direct_normal_irradiance_instant (W/m²) 0
terrestrial_radiation_instant (W/m²)    0
dtype: int64
```

Missing value berhasil diimputasi.

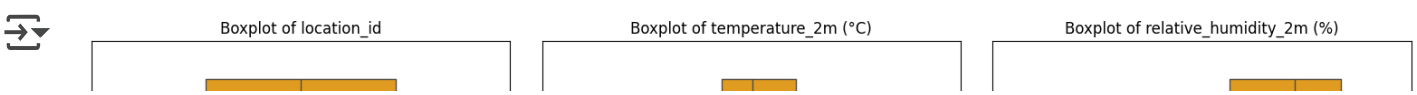
Outlier

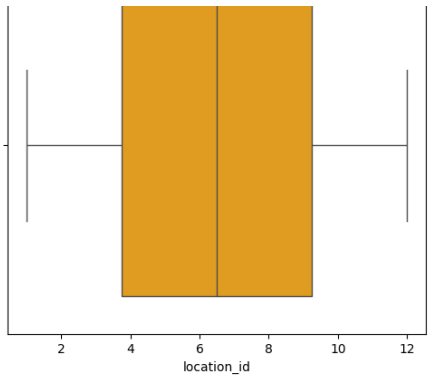
```
# Boxplot Daily
# Pilih hanya kolom numerik
numeric_columns = daily_test.select_dtypes(include=['float64', 'int64']).columr

# Tentukan ukuran grid untuk subplot
num_cols = 3  # Jumlah kolom dalam grid
num_rows = (len(numeric_columns) + num_cols - 1) // num_cols  # Hitung jumlah b

# Plot boxplot untuk setiap kolom numerik
plt.figure(figsize=(15, 5 * num_rows))  # Ukuran gambar menyesuaikan jumlah bar
for i, column in enumerate(numeric_columns, 1):
    plt.subplot(num_rows, num_cols, i)
    sns.boxplot(data=daily_test, x=column, color='orange')
    plt.title(f'Boxplot of {column}')
    plt.xlabel(column)

# Mengatur tata letak untuk menghindari overlap
plt.tight_layout()
plt.show()
```

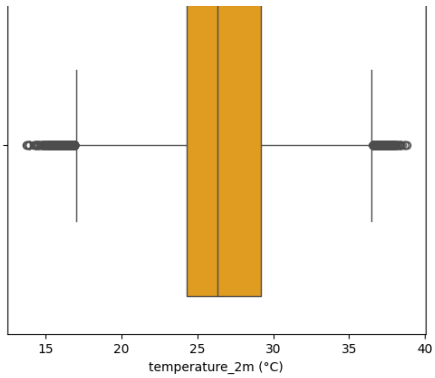


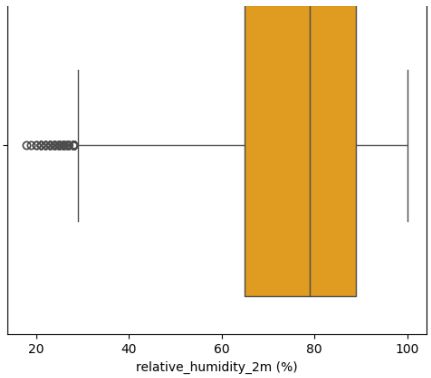



```
# Boxplot Hourly
# Pilih hanya kolom numerik
numeric_columns = hourly_test.select_dtypes(include=['float64', 'int64']).colum

# Tentukan ukuran grid untuk subplot
num_cols = 3  # Jumlah kolom dalam grid
num_rows = (len(numeric_columns) + num_cols - 1) // num_cols  # Hitung jumlah b

# Plot boxplot untuk setiap kolom numerik
plt.figure(figsize=(15, 5 * num_rows))  # Ukuran gambar menyesuaikan jumlah bar
for i, column in enumerate(numeric_columns, 1):
    plt.subplot(num_rows, num_cols, i)
    sns.boxplot(data=hourly_test, x=column, color='orange')
    plt.title(f'Boxplot of {column}')
    plt.xlabel(column)

# Mengatur tata letak untuk menghindari overlap
plt.tight_layout()
plt.show()
```

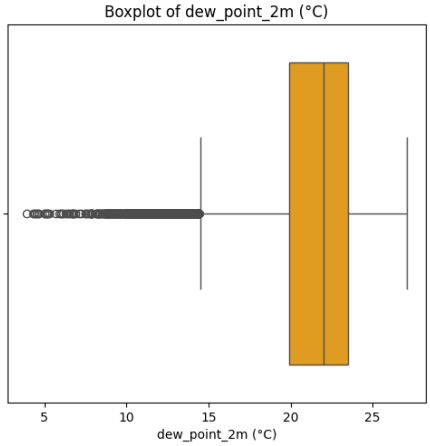




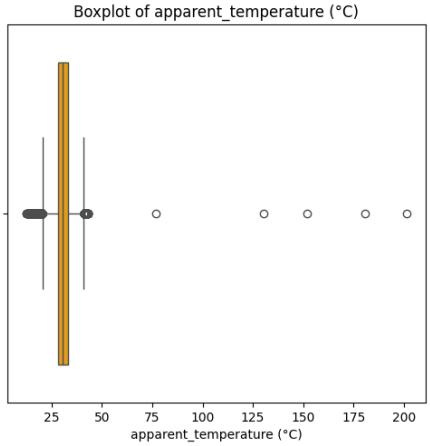




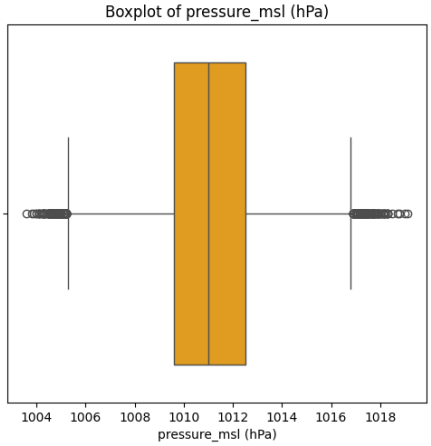
Boxplot of dew_point_2m (°C)



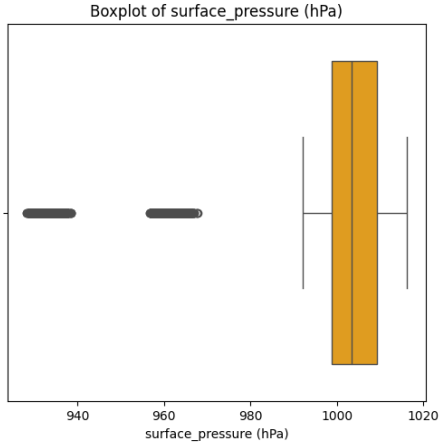
Boxplot of apparent_temperature (°C)



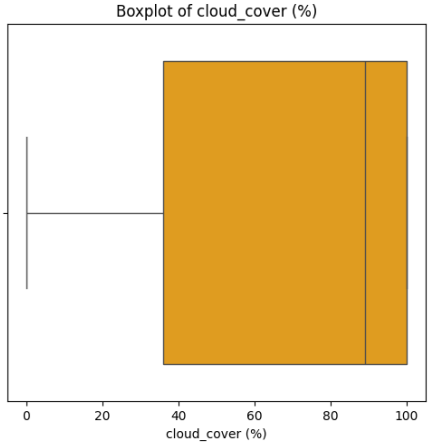
Boxplot of pressure_msl (hPa)



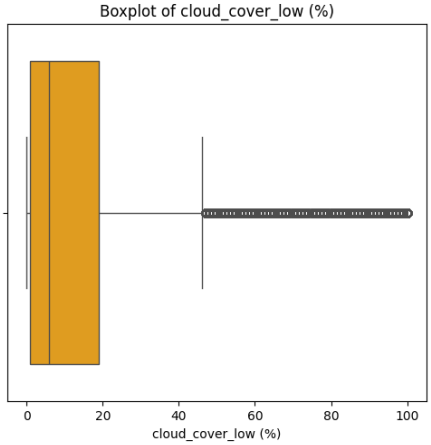
Boxplot of surface_pressure (hPa)



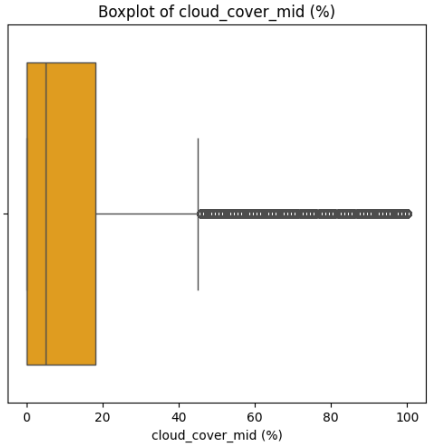
Boxplot of cloud_cover (%)



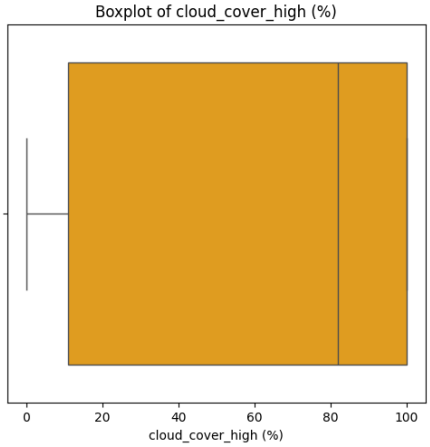
Boxplot of cloud_cover_low (%)



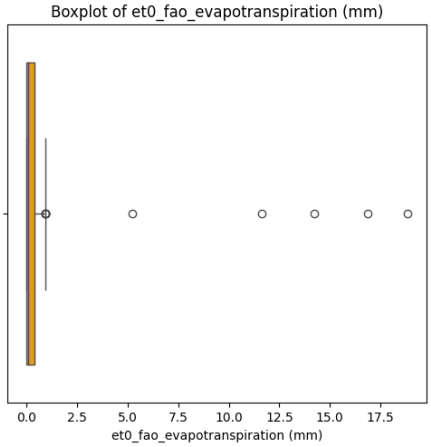
Boxplot of cloud_cover_mid (%)



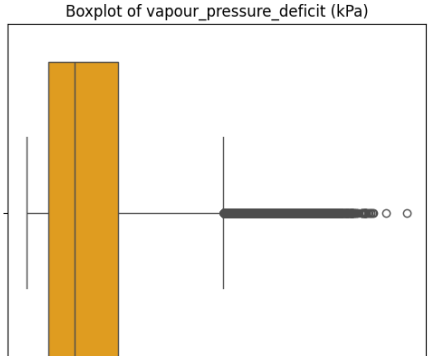
Boxplot of cloud_cover_high (%)



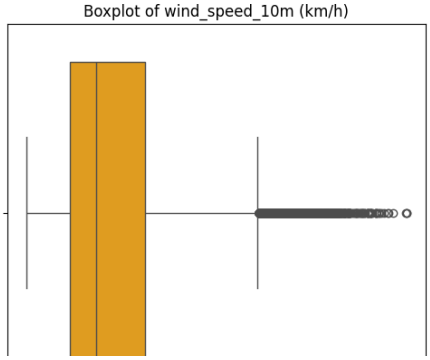
Boxplot of et0_fao_evapotranspiration (mm)



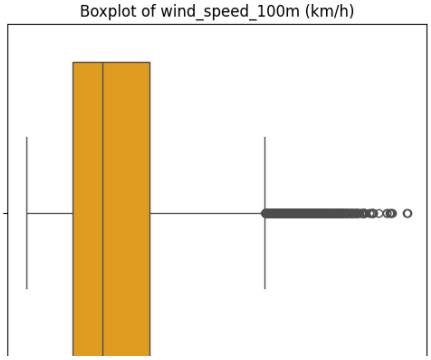
Boxplot of vapour_pressure_deficit (kPa)

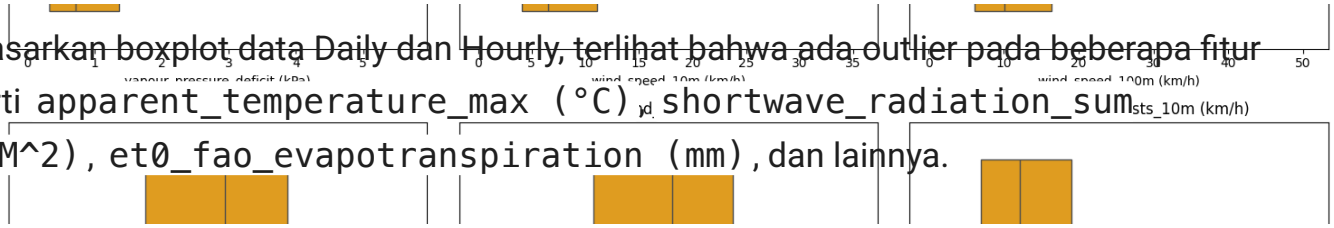


Boxplot of wind_speed_10m (km/h)



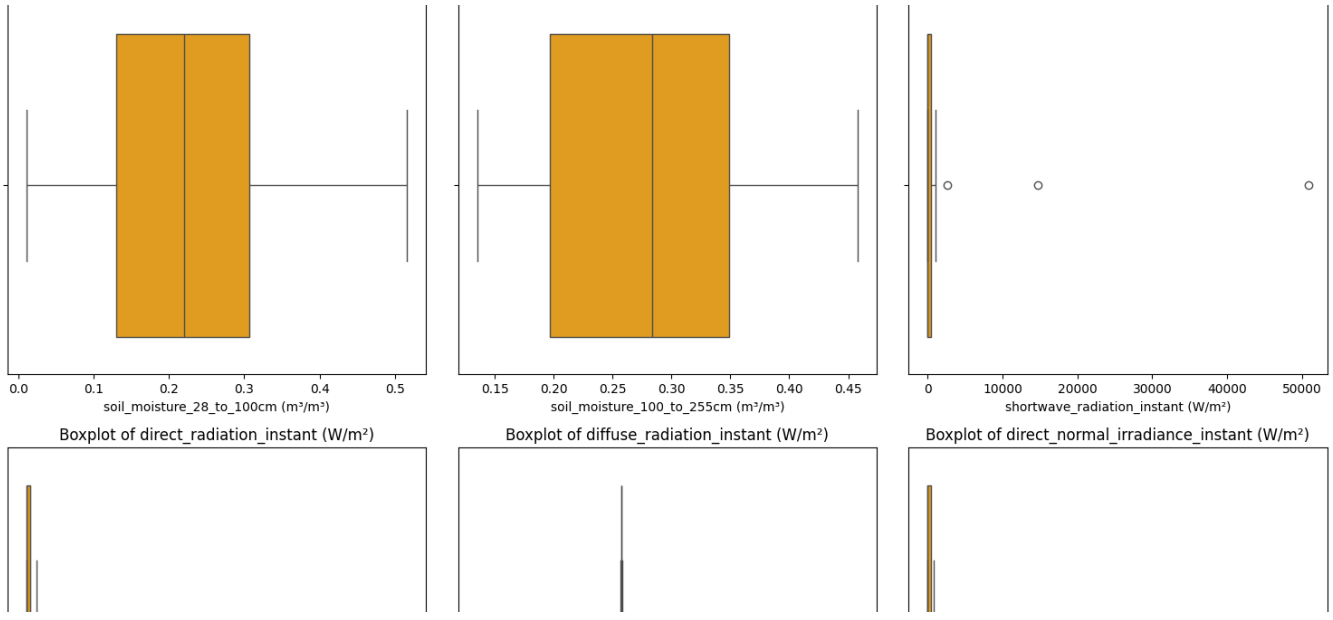
Boxplot of wind_speed_100m (km/h)



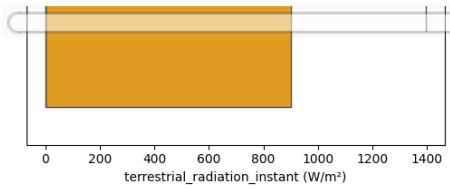


Berdasarkan boxplot data Daily dan Hourly, terlihat bahwa ada outlier pada beberapa fitur seperti `apparent_temperature_max (°C)`, `shortwave_radiation_sum (MJ/M^2)`, `et0_fao_evapotranspiration (mm)`, dan lainnya.


```
# Melihat row dengan outlier
# Kriteria outlier
criteria = (
    (daily_test['apparent_temperature_max (°C)'] > 100) |
    (daily_test['apparent_temperature_mean (°C)'] > 40) |
    (daily_test['shortwave_radiation_sum (MJ/m²)'] > 80) |
    (daily_test['et0_fao_evapotranspiration (mm)'] > 16)
)

# Baris yang memenuhi kriteria
filtered_rows = daily_test[criteria]

# Fungsi untuk mendapatkan baris sebelum dan sesudah
def get_surrounding_rows(df, index):
    start = max(index - 1, 0)
    end = min(index + 2, len(df))  # +2 karena slicing [start:end] eksklusif di
    return df.iloc[start:end]

# Mendapatkan semua baris dengan tambahan sebelum dan sesudah
result = pd.concat([get_surrounding_rows(daily_test, idx) for idx in filtered_r

# Tampilkan kolom tertentu (optional)
result = result[['location_id', 'time',
                 'apparent_temperature_max (°C)',
                 'apparent_temperature_mean (°C)',
                 'shortwave_radiation_sum (MJ/m²)',
                 'et0_fao_evapotranspiration (mm)']]

# Output hasil
result
```



	location_id	time	apparent_temperature_max (°C)	apparent_temperature_mean (°C)
1468	3	2024-10-29	38.0	31.5
1469	3	2024-10-30	201.4	37.3
1470	3	2024-10-31	40.3	31.6
4904	10	2024-10-28	41.0	32.1



Untuk mengatasi outlier pada data test, kami memutuskan untuk mengganti nilainya secara manual dengan melihat data pada hari-hari yang berdekatan.

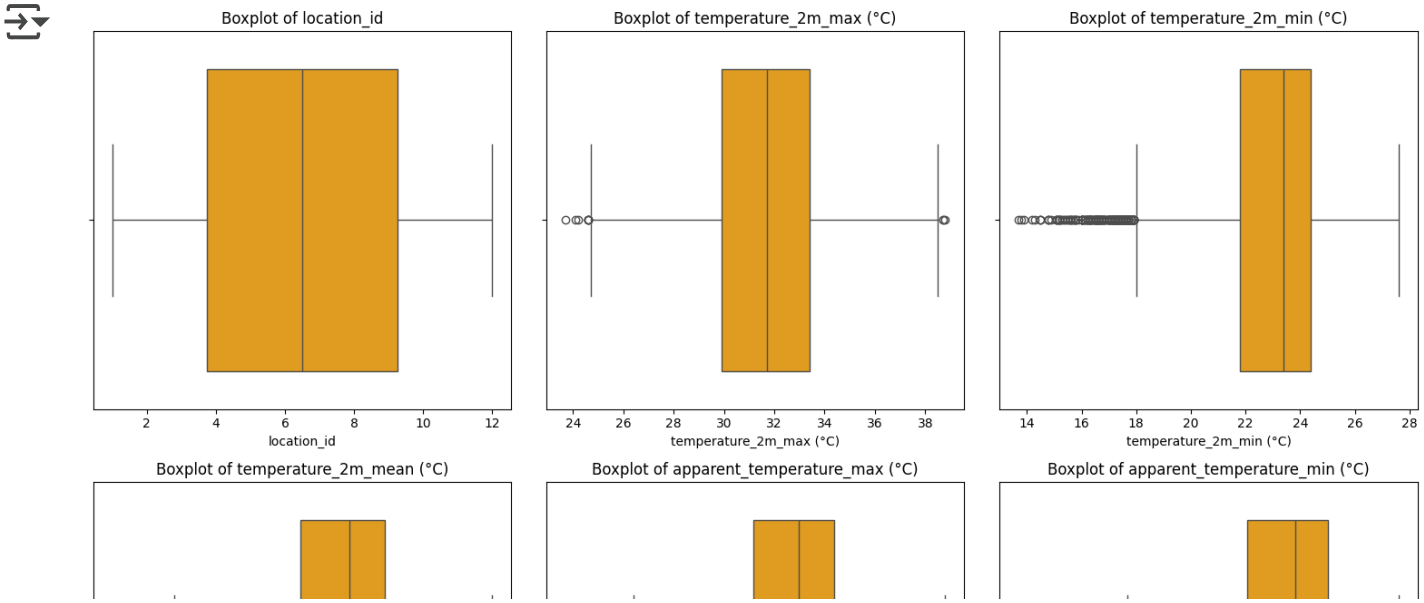
```
# 1. Untuk lokasi 10, apparent_temperature_max (°C) > 100 diganti dengan 39.5
daily_test.loc[
    (daily_test['location_id'] == 10) &
    (daily_test['apparent_temperature_max (°C)'] > 100),
    'apparent_temperature_max (°C)'
] = 39.5

# 2. Untuk lokasi 3, apparent_temperature_max (°C) > 100 diganti dengan 39.1
daily_test.loc[
    (daily_test['location_id'] == 3) &
    (daily_test['apparent_temperature_max (°C)'] > 100),
    'apparent_temperature_max (°C)'
] = 39.1

# 3. Untuk lokasi 10, apparent_temperature_mean (°C) > 40 diganti dengan 33.5
daily_test.loc[
    (daily_test['location_id'] == 10) &
    (daily_test['apparent_temperature_mean (°C)'] > 40),
    'apparent_temperature_mean (°C)'
] = 33.5

# 4. Untuk lokasi 10, shortwave_radiation_sum (MJ/m²) > 40 diganti dengan 20.33
daily_test.loc[
    (daily_test['location_id'] == 10) &
```

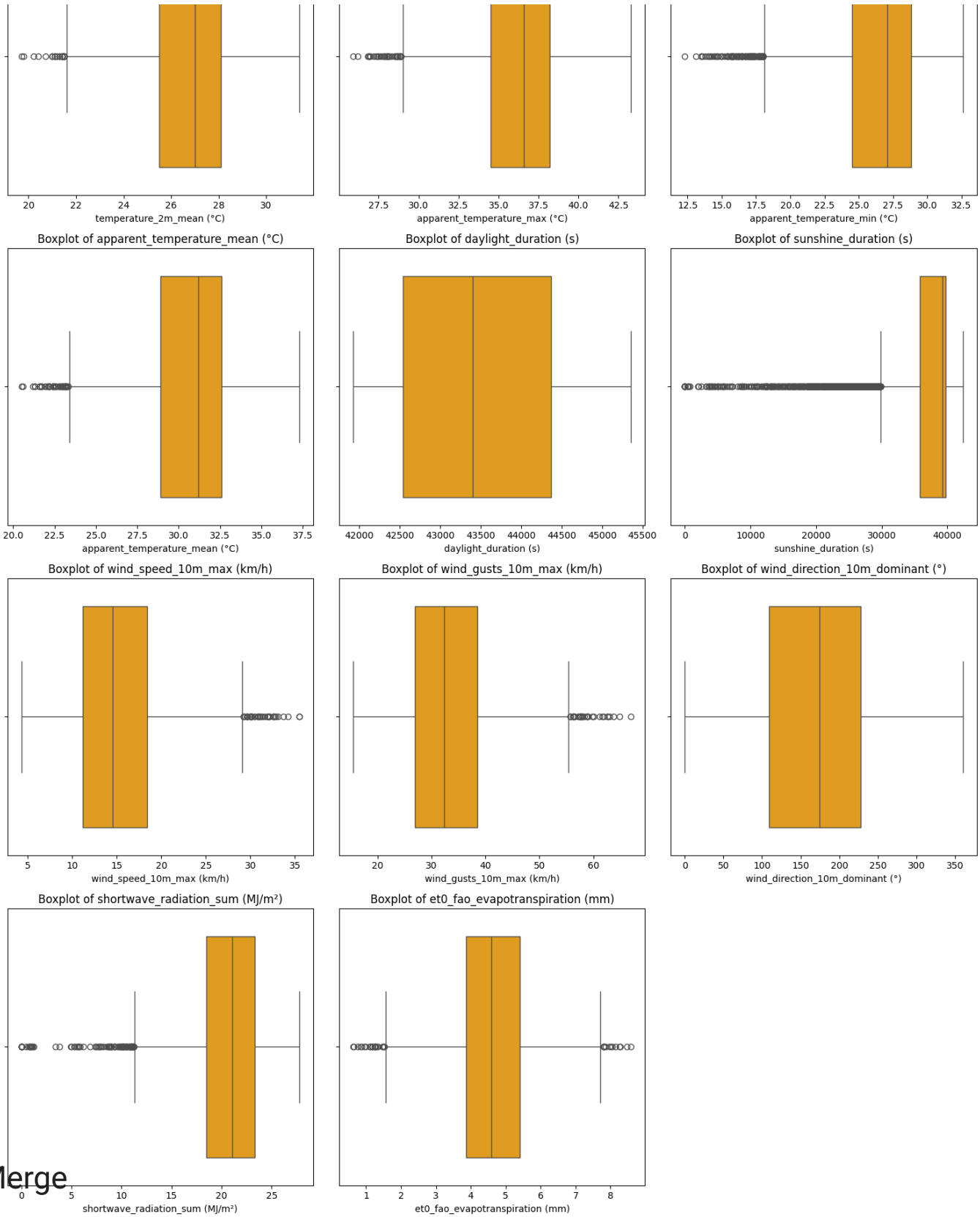
```
    (daily_test['shortwave_radiation_sum (MJ/m²)'] > 40),
    'shortwave_radiation_sum (MJ/m²)'
] = 20.33

# 5. Untuk lokasi 10, et0_fao_evapotranspiration (mm) > 14 diganti dengan 4.43
daily_test.loc[
    (daily_test['location_id'] == 10) &
    (daily_test['et0_fao_evapotranspiration (mm)'] > 14),
    'et0_fao_evapotranspiration (mm)'
] = 4.43

# Tampilkan hasil perubahan
print(daily_test.loc[[1468, 1469, 1470, 4904, 4905, 4906, 4907]])
```

```
                   ID  location_id        time  temperature_2m_max (°C)  \
1468   loc3_20241029            3  2024-10-29                     33.3
1469   loc3_20241030            3  2024-10-30                     31.3
1470   loc3_20241031            3  2024-10-31                     32.4
4904  loc10_20241028           10  2024-10-28                     34.3
4905  loc10_20241029           10  2024-10-29                     33.0
4906  loc10_20241030           10  2024-10-30                     33.3
4907  loc10_20241031           10  2024-10-31                     31.3

      temperature_2m_min (°C)  temperature_2m_mean (°C)  \
1468                     22.7                      27.4
1469                     22.8                      25.9
1470                     22.7                      26.4
4904                     23.3                      27.7
4905                     23.6                      27.5
4906                     23.6                      27.1
4907                     22.1                      26.4

      apparent_temperature_max (°C)  apparent_temperature_min (°C)  \
1468                           38.0                           26.6
1469                           39.1                           26.0
1470                           40.3                           26.7
4904                           41.0                           27.5
4905                           39.5                           27.6
4906                           39.5                           27.2
4907                           38.5                           25.3

      apparent_temperature_mean (°C) sunrise (iso8601)  sunset (iso8601)
1468                            31.5  2024-10-29T05:11  2024-10-29T17:33
1469                            37.3  2024-10-30T05:10  2024-10-30T17:33
1470                            31.6  2024-10-31T05:10  2024-10-31T17:33
4904                            32.1  2024-10-28T04:57  2024-10-28T17:20
4905                            36.0  2024-10-29T04:57  2024-10-29T17:20
4906                            33.5  2024-10-30T04:57  2024-10-30T17:20
4907                            31.3  2024-10-31T04:56  2024-10-31T17:20

      daylight_duration (s)  sunshine_duration (s)  wind_speed_10m_max (k
1468               44524.20               40573.28
1469               44547.82               40573.28
1470               44571.25               30510.11
4904               44549.59               42056.65
```

```
4905                44574.67                42078.56
4906                44599.56                35623.18
4907                44624.24                35623.18

      wind_gusts_10m_max (km/h)  wind_direction_10m_dominant (°)  \
1468                       36.4                            225.0
1469                       38.2                            267.0
1470                       28.1                            244.0
4904                       32.8                            258.0
4905                       28.8                            269.0
4906                       29.2                            256.0
4907                       23.0                            256.0

      shortwave_radiation_sum (MJ/m²)  et0_fao_evapotranspiration (mm)
1468                            24.34                             5.36
1469                            24.34                             5.36
1470                            20.74                             4.32
```

```
# Boxplot setelah menangani outlier
# Pilih hanya kolom numerik
numeric_columns = daily_test.select_dtypes(include=['float64', 'int64']).columr

# Tentukan ukuran grid untuk subplot
num_cols = 3  # Jumlah kolom dalam grid
num_rows = (len(numeric_columns) + num_cols - 1) // num_cols  # Hitung jumlah b

# Plot boxplot untuk setiap kolom numerik
plt.figure(figsize=(15, 5 * num_rows))  # Ukuran gambar menyesuaikan jumlah bar
for i, column in enumerate(numeric_columns, 1):
    plt.subplot(num_rows, num_cols, i)
    sns.boxplot(data=daily_test, x=column, color='orange')
    plt.title(f'Boxplot of {column}')
    plt.xlabel(column)

# Mengatur tata letak untuk menghindari overlap
plt.tight_layout()
plt.show()
```





Merge

Fitur Radiasi pada data `hourly` memiliki satuan W/m2, untuk menyatukannya dengan data `daily`, akan diubah terlebih dulu satuannya menjadi MJ/m2 sebelum dihitung sumasinya.

```
# Konversi ke satuan MJ/m²
sum_col = [
    'direct_radiation_instant (W/m²)', 'diffuse_radiation_instant (W/m²)',
    'direct_normal_irradiance_instant (W/m²)', 'terrestrial_radiation_instant (

for col in sum_col:
    new_col_name = col.replace(' (W/m²)', ' (MJ/m²)')  # Mengubah nama kolom
    hourly_test[new_col_name] = (hourly_test[col] * 3600) / 1_000_000

# Hitung sum per hari
sum_col2 = [
    'direct_radiation_instant (MJ/m²)', 'diffuse_radiation_instant (MJ/m²)',
    'direct_normal_irradiance_instant (MJ/m²)', 'terrestrial_radiation_instant

hourly_test['time'] = pd.to_datetime(hourly_test['time'])
hourly_test['date'] = hourly_test['time'].dt.date

hourly_sum = hourly_test.groupby(['location_id', 'date'])[sum_col2].sum()
hourly_sum.reset_index(inplace=True)


# Pengecualian kolom yang dijumlahkan
exclude_columns = [
    'direct_radiation_instant (MJ/m²)','diffuse_radiation_instant (MJ/m²)',
    'direct_normal_irradiance_instant (MJ/m²)','terrestrial_radiation_instant (
    'direct_radiation_instant (W/m²)', 'diffuse_radiation_instant (W/m²)',
    'direct_normal_irradiance_instant (W/m²)', 'terrestrial_radiation_instant (
    'shortwave_radiation_instant (W/m²)'
]
numeric_columns = hourly_test.select_dtypes(include=['float64']).columns
numeric_columns_to_include = [col for col in numeric_columns if col not in excl

# Definisikan fungsi agregasi
aggregation_functions = {col: ['mean', 'max', 'min', 'std'] for col in numeric_

# Agregasi hourly_test menjadi harian
hourly_aggregated = hourly_test.groupby(['location_id', 'date']).agg(aggregatic

# Flatten MultiIndex columns
hourly_aggregated.columns = ['_'.join(col).strip() for col in hourly_aggregated
hourly_aggregated.reset_index(inplace=True)


# Merge hourly_sum and hourly_aggregated
hourly_agg = pd.merge(hourly_aggregated, hourly_sum, on=['location_id', 'date']

daily_test['time'] = pd.to_datetime(daily_test['time'])
hourly_agg['date'] = pd.to_datetime(hourly_agg['date'])

daily_combined = pd.merge(daily_test, hourly_agg,
                          how='left',
```

```
                          left_on=['location_id', 'time'],
                          right_on=['location_id', 'date'])

test = pd.merge(daily_combined, loc, how='left', on='location_id')

test = test.drop(columns=['date', # drop kolom redundant
                          'temperature_2m (°C)_mean', 'temperature_2m (°C)_ma
                          'apparent_temperature (°C)_mean',   'apparent_tempe
                          'wind_speed_10m (km/h)_max', 'wind_gusts_10m (km/h)
test
```

	ID	location_id	time	temperature_2m_max (°C)	temperature_2m_mi (°C
0	loc1_20230701	1	2023-07-01	31.0	24
1	loc1_20230702	1	2023-07-02	30.4	24
2	loc1_20230703	1	2023-07-03	30.6	24
3	loc1_20230704	1	2023-07-04	31.0	23
4	loc1_20230705	1	2023-07-05	32.3	23
...	...	...	...	...	
5887	loc12_20241029	12	2024-10-29	34.8	24
5888	loc12_20241030	12	2024-10-30	34.3	25
5889	loc12_20241031	12	2024-10-31	35.2	24
5890	loc12_20241101	12	2024-11-01	31.4	23
5891	loc12_20241102	12	2024-11-02	33.2	24

5892 rows × 117 columns

```
# Mengecek jumlah nilai kosong di setiap kolom
missing_values = test.isnull().sum()

# Menampilkan jumlah nilai kosong untuk semua kolom
print("Jumlah Nilai Kosong di Setiap Kolom:")
print(missing_values[missing_values > 0])  # Hanya menampilkan kolom yang memil

# Menampilkan jumlah total nilai kosong di seluruh dataset
total_missing = test.isnull().sum().sum()
print(f"\nTotal Nilai Kosong di Dataset: {total_missing}")
```

```
Jumlah Nilai Kosong di Setiap Kolom:
Series([], dtype: int64)

Total Nilai Kosong di Dataset: 0
```

Feature Engineering

```
def get_season(month):
    if month in [12, 1, 2]:
        return 'rainy'
    elif month in [3, 4, 5]:
        return 'transitional_1'
    elif month in [6, 7, 8]:
        return 'dry'
    elif month in [9, 10, 11]:
        return 'transitional_2'

def categorize_wind_speed(speed):
    if speed < 10:
        return 'low'
    elif 10 <= speed < 20:
        return 'moderate'
    elif 20 <= speed < 30:
        return 'high'
    else:
        return 'storm'

# Fungsi untuk mengkategorikan temperatur mean
def categorize_temperature_mean(temp):
    if temp < 15:
        return 'cold'
    elif 15 <= temp < 30:
        return 'moderate'
    else:
        return 'hot'
```

```
def feature_engineering(df):
    # Mengonversi kolom 'time' menjadi datetime
    df['time'] = pd.to_datetime(df['time'])

    # Ekstrak fitur dari waktu
    df['day'] = df['time'].dt.day
    df['month'] = df['time'].dt.month
    df['year'] = df['time'].dt.year
    df['day_of_year'] = df['time'].dt.dayofyear
    df['week_of_year'] = df['time'].dt.isocalendar().week
    df['day_of_week'] = df['time'].dt.dayofweek
    df['is_weekend'] = df['day_of_week'].apply(lambda x: 1 if x >= 5 else 0)

    # Temperatur
    df['temperature_range'] = df['temperature_2m_max (°C)'] - df['temperature_2
    df['daily_temperature_range'] = df['temperature_2m_max (°C)'] - df['tempera
    df['temperature_gradient'] = df['temperature_2m_mean (°C)'].diff().fillna(0
    df['temp_pressure_humidity_interaction'] = (
        df['temperature_2m_mean (°C)'] *
        df['relative_humidity_2m (%)_mean'] /
        df['pressure_msl (hPa)_mean']
    )
    df['temp_humidity_interaction'] = df['temperature_2m_mean (°C)'] * df['rela
    df['temp_elevation_interaction'] = df['temperature_2m_mean (°C)'] / df['ele

    # Sunshine dan Radiasi
    df['sunshine_ratio'] = df['sunshine_duration (s)'] / df['daylight_duration
    df['radiation_daylight_interaction'] = df['shortwave_radiation_sum (MJ/m²)'
    df['radiation_gradient'] = df['shortwave_radiation_sum (MJ/m²)'].diff().fil

    # Elevasi dan Interaksi
    df['elevation_group'] = pd.cut(df['elevation'], bins=[0, 100, 500, 1000, 30

    # Angin
    df['wind_speed_range'] = df['wind_speed_10m_max (km/h)'] - df['wind_speed_1
    df['wind_speed_gradient'] = df['wind_speed_10m (km/h)_mean'].diff().fillna(
    df['wind_pressure_ratio'] = df['wind_speed_10m (km/h)_mean'] / df['pressure

    # Geografi
    df['abs_latitude'] = df['x'].abs()
    df['latitude_longitude_interaction'] = df['x'] * df['y']
    df['distance_from_equator'] = df['x'].apply(lambda lat: np.sqrt(lat**2))

    # Elevasi Berdasarkan Kelompok
    df['elevation_mean_by_group'] = df.groupby('elevation_group')['elevation'].

    # Cloud dan Kelembapan
    df['heavy_cloud_indicator'] = df['cloud_cover (%)_mean'] * df['vapour_press
    df['season'] = df['month'].apply(get_season)

    # Wind Speed and Direction (Features with Transformation)
```

```
df['wind_speed_max_min_difference'] = df['wind_speed_10m_max (km/h)'] - df[
df['wind_direction_diff'] = df['wind_direction_10m_dominant (°)'] - df['wir

# Latitude-Longitude Interaction Features
df['latitude_longitude_interaction'] = df['x'] * df['y']

df.rename(columns={'apparent_temperature (°C)_max': 'apparent_temperature_m

# Fitur Temporal
df['sin_day_of_year'] = np.sin(2 * np.pi * df['day_of_year'] / 365.25)
df['cos_day_of_year'] = np.cos(2 * np.pi * df['day_of_year'] / 365.25)

# Fitur Atmosferik
df['dewpoint_deficit'] = df['temperature_2m_mean (°C)'] - df['dew_point_2m
df['humidity_pressure_ratio'] = df['relative_humidity_2m (%)_mean'] / df['p

# Fitur Awan
df['cloud_variability'] = df['cloud_cover (%)_max'] - df['cloud_cover (%)_m
df['high_cloud_to_total_ratio'] = df['cloud_cover_high (%)_mean'] / (df['cl

# Fitur Tanah
df['soil_moisture_ratio'] = df['soil_moisture_0_to_7cm (m³/m³)_mean'] / (df
df['soil_moisture_variability'] = df['soil_moisture_0_to_7cm (m³/m³)_max']

# Fitur Angin
df['storm_indicator'] = ((df['wind_gusts_10m_max (km/h)'] > 30) & (df['pres
df['wind_energy_potential'] = 0.5 * (df['wind_speed_10m (km/h)_mean'] ** 3)

# Fitur Radiasi
df['sunshine_to_radiation_ratio'] = df['sunshine_duration (s)'] / (df['shor

# Fitur Interaksi
df['humidity_radiation_interaction'] = df['relative_humidity_2m (%)_mean']
df['wind_pressure_interaction'] = df['wind_speed_10m (km/h)_mean'] * df['pr

df['wind_speed_category'] = df['wind_speed_10m (km/h)_mean'].apply(categori

# Kategori temperatur mean
df['temperature_category_mean'] = df['temperature_2m_mean (°C)'].apply(cate

# Mengubah tipe data
categorical_features = ['elevation_group', 'season', 'wind_speed_category',
for feature in categorical_features:
    df[feature] = df[feature].astype('category')
df['week_of_year'] = df['week_of_year'].astype('int32')

df['temp_humidity_interaction_squared'] = df['temp_humidity_interaction'] *
df['wind_pressure_interaction_squared'] = df['wind_pressure_interaction'] *
df['radiation_sunshine_interaction'] = df['shortwave_radiation_sum (MJ/m²)'
df['log_dewpoint_deficit'] = np.log1p(df['dewpoint_deficit'])
df['soil_temperature_difference'] = df['soil_temperature_0_to_7cm (°C)_mean
```

```
    df['wind_direction_radians'] = np.radians(df['wind_direction_10m_dominant (
    df['extreme_temp'] = (df['temperature_2m_max (°C)'] > 35).astype(int)
    df['high_humidity'] = (df['relative_humidity_2m (%)_max'] > 90).astype(int)
    df['low_pressure'] = (df['pressure_msl (hPa)_min'] < 1000).astype(int)

    columns_to_drop = [
      'ID',
      'location_id',
      'time',
      'sunrise (iso8601)',
      'sunset (iso8601)',
      'x',
      'y',
      'daily_temperature_range'
    ]

    # Drop kolom dari train
    df = df.drop(columns=columns_to_drop)

    return df


train = feature_engineering(train)
test = feature_engineering(test)
```

```
train
```

	temperature_2m_max (°C)	temperature_2m_min (°C)	temperature_2m_mean (°C)	apparent
0	26.8	24.2	25.4	
1	30.5	24.4	26.6	
2	30.1	24.2	26.4	
3	31.8	23.5	27.5	
4	30.7	24.4	27.5	
...	...	...	...	
19699	33.3	22.5	27.4	
19700	32.9	24.3	27.5	
19701	33.1	24.6	27.8	
19702	33.5	25.1	28.2	
19703	33.5	23.9	27.9	

19698 rows × 162 columns

test

	temperature_2m_max (°C)	temperature_2m_min (°C)	temperature_2m_mean (°C)	apparent_
0	31.0	24.7	27.3	
1	30.4	24.8	27.0	
2	30.6	24.1	26.9	
3	31.0	23.7	27.1	
4	32.3	23.8	27.5	
...	...	...	...	
5887	34.8	24.8	29.4	
5888	34.3	25.0	29.0	
5889	35.2	24.8	29.1	
5890	31.4	23.3	26.8	
5891	33.2	24.9	28.0	

5892 rows × 161 columns

Modelling

Feature Selection

Mutual Information

```
# Step 1: Transformasi log pada target
train['log_rain_sum'] = np.log1p(train['rain_sum (mm)'])  # log(1 + y)
y_train = train['log_rain_sum']

# Step 2: Pisahkan fitur numerik dan kategorik
numerical_features = train.select_dtypes(include=['number']).drop(columns=['rai
categorical_features = train.select_dtypes(exclude=['number']).columns.tolist()

# Menampilkan semua fitur sebelum seleksi
all_features = numerical_features.columns.tolist() + categorical_features
print("Fitur Sebelum Seleksi:")
print(all_features)
```

```
print(f"Jumlah Fitur Sebelum Seleksi: {len(all_features)}\n")

# Step 3: Hitung Mutual Information untuk fitur numerik
mi_scores = mutual_info_regression(numerical_features, y_train, random_state=SE
mi_scores = pd.Series(mi_scores, index=numerical_features.columns).sort_values(

# Step 4: Tampilkan skor Mutual Information
print("Mutual Information Scores:")
print(mi_scores)

# Step 5: Pilih fitur numerik dengan skor MI di atas threshold
threshold = 0.01  # Atur threshold sesuai kebutuhan
selected_numerical_features = mi_scores[mi_scores > threshold].index.tolist()

# Gabungkan fitur numerik terpilih dengan fitur kategorik
final_selected_features = selected_numerical_features + categorical_features

# Step 6: Dataset Train dan Test dengan fitur terpilih
X_train_selected = train[final_selected_features]
X_test_selected = test[final_selected_features]

# Menampilkan semua fitur setelah seleksi
print("\nFitur Setelah Seleksi:")
print(final_selected_features)
print(f"Jumlah Fitur Setelah Seleksi: {len(final_selected_features)}\n")
```

```
Fitur Sebelum Seleksi:
['temperature_2m_max (°C)', 'temperature_2m_min (°C)', 'temperature_2m_mean
Jumlah Fitur Sebelum Seleksi: 161

Mutual Information Scores:
relative_humidity_2m (%)_mean             0.531087
humidity_pressure_ratio                   0.524927
log_dewpoint_deficit                      0.516994
dewpoint_deficit                          0.516049
soil_moisture_0_to_7cm (m³/m³)_max        0.471909
                                              ...
elevation_mean_by_group                   0.004576
soil_temperature_100_to_255cm (°C)_std    0.004138
wind_speed_gradient                       0.001696
storm_indicator                           0.000000
low_pressure                              0.000000
Length: 156, dtype: float64

Fitur Setelah Seleksi:
['relative_humidity_2m (%)_mean', 'humidity_pressure_ratio', 'log_dewpoint_
Jumlah Fitur Setelah Seleksi: 151
```

Handle Multicollinearity

```
# Step 1: Pilih hanya fitur numerik
numerical_features = X_train_selected.select_dtypes(include=['number'])

# Step 2: Hitung matriks korelasi
correlation_matrix = numerical_features.corr().abs()

# Step 3: Identifikasi pasangan fitur dengan korelasi di atas 0.95
upper_triangle = correlation_matrix.where(np.triu(np.ones(correlation_matrix.sh
high_correlation_pairs = [
    (column, index, upper_triangle.loc[index, column])
    for column in upper_triangle.columns
    for index in upper_triangle.index
    if upper_triangle.loc[index, column] > 0.95
]

# Konversi pasangan korelasi tinggi ke DataFrame
high_correlation_df = pd.DataFrame(high_correlation_pairs, columns=['Feature 1'

# Tampilkan DataFrame pasangan korelasi tinggi
print("High Correlation Pairs (> 0.95):")
print(high_correlation_df)

# Step 4: Drop salah satu fitur dari setiap pasangan berdasarkan MI
features_to_drop = set()
for col1, col2, _ in high_correlation_pairs:
    if col1 in mi_scores and col2 in mi_scores:
        # Drop fitur dengan skor MI lebih rendah
        if mi_scores[col1] < mi_scores[col2]:
            features_to_drop.add(col1)
        else:
            features_to_drop.add(col2)

# Hapus fitur yang dipilih untuk di-drop
X_train_selected_final = X_train_selected.drop(columns=features_to_drop)
X_test_selected_final = X_test_selected.drop(columns=features_to_drop)

# Tampilkan fitur yang di-drop dan fitur tersisa
print("\nFeatures to Drop (Multicollinearity > 0.95):")
print(pd.DataFrame(list(features_to_drop), columns=['Dropped Features']))

print("\nRemaining Features After Multicollinearity Analysis:")
print(pd.DataFrame(X_train_selected_final.columns.tolist(), columns=['Remaining
```

```
High Correlation Pairs (> 0.95):
                        Feature 1                        Feature 2  Correlati
0       humidity_pressure_ratio  relative_humidity_2m (%)_mean     0.9999
1          log_dewpoint_deficit  relative_humidity_2m (%)_mean     0.9885
2          log_dewpoint_deficit        humidity_pressure_ratio     0.9885
3              dewpoint_deficit  relative_humidity_2m (%)_mean     0.9952
4              dewpoint_deficit        humidity_pressure_ratio     0.9950
..                          ...                            ...          .
101  surface_pressure (hPa)_max    surface_pressure (hPa)_mean     0.9999
```

```
102  surface_pressure (hPa)_min                    elevation     0.9982
103  surface_pressure (hPa)_min   surface_pressure (hPa)_mean     0.9997
104  surface_pressure (hPa)_min    surface_pressure (hPa)_max     0.9995
105     pressure_msl (hPa)_mean        pressure_msl (hPa)_max     0.9800

[106 rows x 3 columns]

Features to Drop (Multicollinearity > 0.95):
                          Dropped Features
0                                    month
1     soil_temperature_28_to_100cm (°C)_min
2           wind_direction_10m_dominant (°)
3    soil_temperature_28_to_100cm (°C)_mean
4           shortwave_radiation_sum (MJ/m²)
5                      log_dewpoint_deficit
6                       cloud_cover (%)_std
7                         cloud_variability
8                       wind_pressure_ratio
9    soil_moisture_100_to_255cm (m³/m³)_min
10               wind_pressure_interaction
11                 pressure_msl (hPa)_mean
12       temp_humidity_interaction_squared
13              wind_gusts_10m (km/h)_mean
14      temp_pressure_humidity_interaction
15     soil_moisture_7_to_28cm (m³/m³)_mean
16                           week_of_year
17                  wind_energy_potential
18      vapour_pressure_deficit (kPa)_mean
19      soil_moisture_7_to_28cm (m³/m³)_min
20       soil_moisture_0_to_7cm (m³/m³)_std
21      et0_fao_evapotranspiration (mm)_std
22              wind_speed_10m (km/h)_mean
23    soil_moisture_28_to_100cm (m³/m³)_min
24                  sunshine_duration (s)
25              surface_pressure (hPa)_max
26    soil_temperature_28_to_100cm (°C)_max
27                  distance_from_equator
28  soil_temperature_100_to_255cm (°C)_mean
29          et0_fao_evapotranspiration (mm)
30      soil_temperature_7_to_28cm (°C)_min
31   soil_temperature_100_to_255cm (°C)_max
32     soil_temperature_7_to_28cm (°C)_mean
33                  daylight_duration (s)
34                      temperature_range
35           radiation_sunshine_interaction
36      soil_moisture_0_to_7cm (m³/m³)_mean
37       soil_moisture_0_to_7cm (m³/m³)_min
38         direct_radiation_instant (MJ/m²)
39                  humidity_pressure_ratio
```

Model Selection

```
# Data Preparation
X_train = X_train_selected_final.copy()  # Selected features
X_test = X_test_selected_final.copy()  # Test set features

# Encoding Categoric Variable
# Identify categorical columns
categorical_cols = X_train.select_dtypes(include=['category', 'object']).columns

# Perform one-hot encoding
X_train = pd.get_dummies(X_train, columns=categorical_cols, drop_first=True)
X_test = pd.get_dummies(X_test, columns=categorical_cols, drop_first=True)

# Split train data into train and validation sets
X_train_split, X_valid_split, y_train_split, y_valid_split = train_test_split(
    X_train, y_train, test_size=0.2, random_state=SEED
)


# Pipelines untuk model
pipelines = {
    'RandomForest': Pipeline([
        ('model', RandomForestRegressor(random_state=SEED))
    ]),
    'GradientBoosting': Pipeline([
        ('model', GradientBoostingRegressor(random_state=SEED))
    ]),
    'XGBoost': Pipeline([
        ('model', xgb.XGBRegressor(random_state=SEED))
    ]),
    'LightGBM': Pipeline([
        ('model', lgb.LGBMRegressor(random_state=SEED))
    ]),
    'KNN': Pipeline([
        ('model', KNeighborsRegressor())
    ]),
    'DecisionTree': Pipeline([
        ('model', DecisionTreeRegressor(random_state=SEED))
    ]),
    'CatBoost': Pipeline([
        ('model', cb.CatBoostRegressor(random_state=SEED, verbose=100))
    ]),
    'ExtraTrees': Pipeline([
        ('model', ExtraTreesRegressor(random_state=SEED))
    ])
}

# Fungsi untuk evaluasi model
def evaluate_model(model, X_train, y_train, X_val, y_val):
    model.fit(X_train, y_train)

    # Prediksi pada validation set
```

```
    y_pred = model.predict(X_val)
    y_pred = np.expm1(y_pred)  # Transformasi balik dari log scale

    mse = mean_squared_error(np.expm1(y_val), y_pred)
    rmse = np.sqrt(mse)
    mae = mean_squared_error(np.expm1(y_val), y_pred)

    return mse, rmse, mae, y_pred

# Evaluasi setiap model dan tampilkan metriknya
for name, model in pipelines.items():
    print(f"Evaluating {name}...")
    mse, rmse, mae, y_pred = evaluate_model(model, X_train_split, y_train_split
    print(f"  MSE: {mse:.4f}")
    print(f"  RMSE: {rmse:.4f}")
    print(f"  MAE: {mae:.4f}")
    print()
```

```
Evaluating RandomForest...
  MSE: 26.4462
  RMSE: 5.1426
  MAE: 26.4462

Evaluating GradientBoosting...
  MSE: 29.9983
  RMSE: 5.4771
  MAE: 29.9983

Evaluating XGBoost...
  MSE: 25.1408
  RMSE: 5.0141
  MAE: 25.1408

Evaluating LightGBM...
[LightGBM] [Warning] Found whitespace in feature_names, replace with unde
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 17831
[LightGBM] [Info] Number of data points in the train set: 15758, number o
[LightGBM] [Info] Start training from score 1.433722
  MSE: 25.5286
  RMSE: 5.0526
  MAE: 25.5286

Evaluating KNN...
  MSE: 109.1175
  RMSE: 10.4459
  MAE: 109.1175

Evaluating DecisionTree...
  MSE: 63.8181
  RMSE: 7.9886
  MAE: 63.8181
```

```
Evaluating CatBoost...
Learning rate set to 0.063297
0:      learn: 1.0738301        total: 87.2ms   remaining: 1m 27s
100:    learn: 0.3689015        total: 3.99s    remaining: 35.5s
200:    learn: 0.3361255        total: 6.32s    remaining: 25.1s
300:    learn: 0.3128620        total: 10.5s    remaining: 24.5s
400:    learn: 0.2950227        total: 13.2s    remaining: 19.7s
500:    learn: 0.2801070        total: 15.6s    remaining: 15.5s
600:    learn: 0.2674599        total: 17.9s    remaining: 11.9s
700:    learn: 0.2559946        total: 20.3s    remaining: 8.64s
800:    learn: 0.2454285        total: 25.2s    remaining: 6.27s
900:    learn: 0.2360960        total: 29s      remaining: 3.18s
999:    learn: 0.2272012        total: 31.3s    remaining: 0us
  MSE: 23.9109
  RMSE: 4.8899
  MAE: 23.9109

Evaluating ExtraTrees...
  MSE: 29.9662
  RMSE: 5.4741
  MAE: 29.9662
```

Kami memutuskan untuk memilih model **Catboost** karena memiliki MSE paling kecil di antara semua model

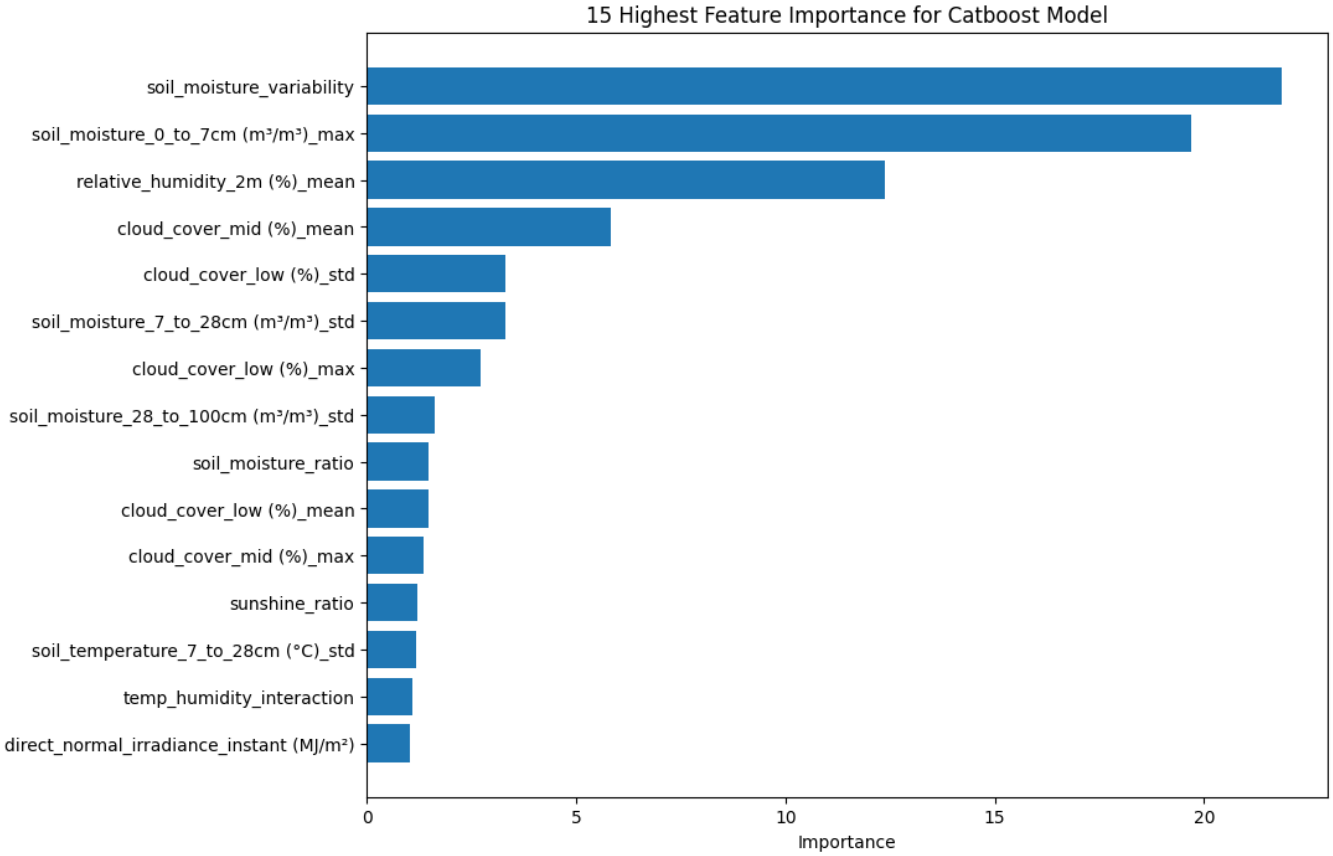
Final Model

```
final_model = cb.CatBoostRegressor(iterations = 1500, loss_function = 'RMSE', d
                                   learning_rate = 0.05, l2_leaf_reg = 10, bagg
                                   random_strength = 0.7, od_type = 'Iter', ear
                                   random_state=SEED, verbose=100)
final_model.fit(X_train_split, y_train_split)

# Predict
y_valid_pred_log = final_model.predict(X_valid_split)
y_valid_pred = np.expm1(y_valid_pred_log)  # Transform back from log scale
y_valid_pred = np.maximum(0, y_valid_pred)  # Ensure no negative values

# Calculate MSE, RMSE, and R² for validation set
mse = mean_squared_error(np.expm1(y_valid_split), y_valid_pred)  # Mean Squared
rmse = np.sqrt(mse)  # Root Mean Squared Error
r2 = r2_score(np.expm1(y_valid_split), y_valid_pred)  # R² Score

# Print results
print(f"Validation MSE: {mse:.4f}")
print(f"Validation RMSE: {rmse:.4f}")
print(f"Validation R²: {r2:.4f}")
```

```
0:      learn: 1.0866643        total: 73.2ms   remaining: 1m 49s
100:    learn: 0.3798032        total: 7.25s    remaining: 1m 40s
200:    learn: 0.3500507        total: 11.8s    remaining: 1m 16s
300:    learn: 0.3304125        total: 16.7s    remaining: 1m 6s
400:    learn: 0.3166791        total: 22.9s    remaining: 1m 2s
500:    learn: 0.3050348        total: 27.7s    remaining: 55.3s
600:    learn: 0.2941329        total: 32.3s    remaining: 48.3s
700:    learn: 0.2845845        total: 38.7s    remaining: 44.1s
800:    learn: 0.2753838        total: 44.2s    remaining: 38.6s
900:    learn: 0.2671926        total: 49s      remaining: 32.6s
1000:   learn: 0.2600318        total: 54.7s    remaining: 27.3s
1100:   learn: 0.2534204        total: 58.9s    remaining: 21.3s
1200:   learn: 0.2470418        total: 1m 1s    remaining: 15.2s
1300:   learn: 0.2410040        total: 1m 4s    remaining: 9.88s
1400:   learn: 0.2351298        total: 1m 8s    remaining: 4.82s
1499:   learn: 0.2296996        total: 1m 10s   remaining: 0us
Validation MSE: 24.0627
Validation RMSE: 4.9054
Validation R²: 0.7698
```

Feature Importance

```
feat_importance = final_model.get_feature_importance()
feature_importance_df = pd.DataFrame({'Feature': X_train_split.columns, 'Import
feature_importance_df = feature_importance_df.sort_values(by='Importance', asce
feature_importance_df = feature_importance_df[:15]

plt.figure(figsize=(10, 8))
plt.barh(feature_importance_df['Feature'], feature_importance_df['Importance'])
plt.xlabel('Importance')
plt.title('15 Highest Feature Importance for Catboost Model')
plt.gca().invert_yaxis()
plt.show()
```



Final Training, Save Model, Prediction, Postprocessing, and Submission

```
final_model = cb.CatBoostRegressor(iterations = 1500, loss_function = 'RMSE', d
                                   learning_rate = 0.05, l2_leaf_reg = 10, bagg
                                   random_strength = 0.7, od_type = 'Iter', ear
                                   random_state=SEED, verbose=100)
final_model.fit(X_train, y_train)
with open("catboost_model.pkl", "wb") as file:
    pickle.dump(final_model, file)
```

```
0:      learn: 1.0876736        total: 33.9ms   remaining: 50.9s
100:    learn: 0.3816508        total: 2.63s    remaining: 36.4s
200:    learn: 0.3527529        total: 5.39s    remaining: 34.9s
300:    learn: 0.3327612        total: 10.4s    remaining: 41.5s
400:    learn: 0.3193953        total: 14.3s    remaining: 39.1s
500:    learn: 0.3085748        total: 17.1s    remaining: 34s
600:    learn: 0.2989896        total: 19.7s    remaining: 29.5s
700:    learn: 0.2906059        total: 22.3s    remaining: 25.5s
800:    learn: 0.2831081        total: 26.7s    remaining: 23.3s
900:    learn: 0.2758967        total: 29.5s    remaining: 19.6s
1000:   learn: 0.2690483        total: 32s      remaining: 16s
1100:   learn: 0.2630589        total: 34.5s    remaining: 12.5s
1200:   learn: 0.2573161        total: 37s      remaining: 9.21s
1300:   learn: 0.2518355        total: 41.7s    remaining: 6.38s
1400:   learn: 0.2462732        total: 44.2s    remaining: 3.12s
1499:   learn: 0.2412607        total: 46.6s    remaining: 0us
```

```
with open("catboost_model.pkl", "rb") as file:
    final_cb_model = pickle.load(file)


train = pd.concat([X_train_selected_final, y_train], axis=1)
test = X_test_selected_final


# Simpan hasil ke file CSV
train.to_csv("train_titik.csv", index=False)
test.to_csv("test_titik.csv", index=False)

# Unduh file
files.download("train_titik.csv")
files.download("test_titik.csv")

print("Gabungan data selesai! File disimpan sebagai train_titik.csv dan test_ti
```

```
Gabungan data selesai! File disimpan sebagai train_titik.csv dan test_titik
```

```
!gdown 1Kd9XFHuE7jeK51en8zbpCHjiWjwo90BZ

sample = pd.read_csv('/content/sample_submission.csv')
```

```
Downloading...
From: https://drive.google.com/uc?id=1Kd9XFHuE7jeK51en8zbpCHjiWjwo90BZ
To: /content/sample_submission.csv
100% 102k/102k [00:00<00:00, 9.01MB/s]
```

```
# Predict on Test Data
y_test_pred_log = final_model.predict(X_test)
y_test_pred = np.expm1(y_test_pred_log)  # Transform back from log scale
y_test_pred = np.maximum(0, y_test_pred)  # Ensure no negative values

# Prepare Submission for Kaggle
# Replace 'ID' with the actual ID column in your test data
submission = pd.DataFrame({'ID': sample['ID'], 'rain_sum (mm)': y_test_pred})

# Save to CSV
submission.to_csv('submission.csv', index=False)
print("Submission file saved as 'submission.csv'")
```

```
Submission file saved as 'submission.csv'
```