Title: PM Accelerator TechAssessment Data Scientist

Author: Alex Ordonez

Date: March 21, 2025

Mission: By making industry-leading tools and education available to individuals from all backgrounds, we level the playing field for future PM leaders. This is the PM Accelerator motto, as we grant aspiring and experienced PMs what they need most – Access. We introduce you to industry leaders, surround you with the right PM ecosystem, and discover the new world of AI product management skills.

```
#Importing the necessary Libraries
import pandas as pd
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from pandas.api.types import is_numeric_dtype
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
import shap
from sklearn.linear_model import LinearRegression
```

Analyze the "Global Weather Repository.csv" dataset to forecast future weather trends and showcase data science skills through both basic and advanced techniques.

```
#Reading and inspectioning data
df=pd.read_csv('./GlobalWeatherRepository.csv')
df.head()
```

```
       country     location_name  latitude  longitude        timezone  \
0  Afghanistan             Kabul     34.52      69.18      Asia/Kabul
1      Albania            Tirana     41.33      19.82   Europe/Tirane
2      Algeria           Algiers     36.76       3.05  Africa/Algiers
3      Andorra  Andorra La Vella     42.50       1.52  Europe/Andorra
4       Angola            Luanda     -8.84      13.23   Africa/Luanda
```

```
   last_updated_epoch      last_updated  temperature_celsius  \
0          1715849100  2024-05-16 13:15                 26.6   
1          1715849100  2024-05-16 10:45                 19.0   
2          1715849100  2024-05-16 09:45                 23.0   
3          1715849100  2024-05-16 10:45                  6.3   
4          1715849100  2024-05-16 09:45                 26.0   

   temperature_fahrenheit condition_text  ...  air_quality_PM2.5  \
0                    79.8  Partly Cloudy  ...                8.4   
1                    66.2  Partly cloudy  ...                1.1   
2                    73.4          Sunny  ...               10.4   
3                    43.3  Light drizzle  ...                0.7   
4                    78.8  Partly cloudy  ...              183.4   

   air_quality_PM10  air_quality_us-epa-index air_quality_gb-defra-
index  \
0              26.6                         1                         
1  
1               2.0                         1                         
1  
2              18.4                         1                         
1  
3               0.9                         1                         
1  
4             262.3                         5                        
10  

    sunrise    sunset  moonrise   moonset      moon_phase 
moon_illumination  
0  04:50 AM  06:50 PM  12:12 PM  01:11 AM  Waxing Gibbous                 
55  
1  05:21 AM  07:54 PM  12:58 PM  02:14 AM  Waxing Gibbous                 
55  
2  05:40 AM  07:50 PM  01:15 PM  02:14 AM  Waxing Gibbous                 
55  
3  06:31 AM  09:11 PM  02:12 PM  03:31 AM  Waxing Gibbous                 
55  
4  06:12 AM  05:55 PM  01:17 PM  12:38 AM  Waxing Gibbous                 
55  

[5 rows x 41 columns]
```

The first step that I took was to explore the data, verify types, null and nan values

```
#count of records
len(df)

59633
```

```python
#Verifying columns
df.columns
```

```
Index(['country', 'location_name', 'latitude', 'longitude',
'timezone',
       'last_updated_epoch', 'last_updated', 'temperature_celsius',
       'temperature_fahrenheit', 'condition_text', 'wind_mph',
'wind_kph',
       'wind_degree', 'wind_direction', 'pressure_mb', 'pressure_in',
       'precip_mm', 'precip_in', 'humidity', 'cloud',
'feels_like_celsius',
       'feels_like_fahrenheit', 'visibility_km', 'visibility_miles',
       'uv_index', 'gust_mph', 'gust_kph',
'air_quality_Carbon_Monoxide',
       'air_quality_Ozone', 'air_quality_Nitrogen_dioxide',
       'air_quality_Sulphur_dioxide', 'air_quality_PM2.5',
'air_quality_PM10',
       'air_quality_us-epa-index', 'air_quality_gb-defra-index',
'sunrise',
       'sunset', 'moonrise', 'moonset', 'moon_phase',
'moon_illumination'],
      dtype='object')
```

```python
#Verifying types
df.dtypes
```

```
country                          object
location_name                    object
latitude                        float64
longitude                       float64
timezone                         object
last_updated_epoch                int64
last_updated                     object
temperature_celsius             float64
temperature_fahrenheit          float64
condition_text                   object
wind_mph                        float64
wind_kph                        float64
wind_degree                       int64
wind_direction                   object
pressure_mb                     float64
pressure_in                     float64
precip_mm                       float64
precip_in                       float64
humidity                          int64
cloud                             int64
feels_like_celsius              float64
feels_like_fahrenheit           float64
visibility_km                   float64
visibility_miles                float64
```

```
uv_index                              float64
gust_mph                              float64
gust_kph                              float64
air_quality_Carbon_Monoxide           float64
air_quality_Ozone                     float64
air_quality_Nitrogen_dioxide          float64
air_quality_Sulphur_dioxide           float64
air_quality_PM2.5                     float64
air_quality_PM10                      float64
air_quality_us-epa-index                int64
air_quality_gb-defra-index              int64
sunrise                                object
sunset                                 object
moonrise                               object
moonset                                object
moon_phase                             object
moon_illumination                       int64
dtype: object
```

```
#Localizing Null & Nan values
print("Null Values")
print(df.isnull().sum())
print("")
print("Nan Values")
print(df.isna().sum())
```

```
Null Values
country                         0
location_name                   0
latitude                        0
longitude                       0
timezone                        0
last_updated_epoch              0
last_updated                    0
temperature_celsius             0
temperature_fahrenheit          0
condition_text                  0
wind_mph                        0
wind_kph                        0
wind_degree                     0
wind_direction                  0
pressure_mb                     0
pressure_in                     0
precip_mm                       0
precip_in                       0
humidity                        0
cloud                           0
feels_like_celsius              0
feels_like_fahrenheit           0
visibility_km                   0
```

```
visibility_miles                0
uv_index                        0
gust_mph                        0
gust_kph                        0
air_quality_Carbon_Monoxide     0
air_quality_Ozone               0
air_quality_Nitrogen_dioxide    0
air_quality_Sulphur_dioxide     0
air_quality_PM2.5               0
air_quality_PM10                0
air_quality_us-epa-index        0
air_quality_gb-defra-index      0
sunrise                         0
sunset                          0
moonrise                        0
moonset                         0
moon_phase                      0
moon_illumination               0
dtype: int64

Nan Values
country                         0
location_name                   0
latitude                        0
longitude                       0
timezone                        0
last_updated_epoch              0
last_updated                    0
temperature_celsius             0
temperature_fahrenheit          0
condition_text                  0
wind_mph                        0
wind_kph                        0
wind_degree                     0
wind_direction                  0
pressure_mb                     0
pressure_in                     0
precip_mm                       0
precip_in                       0
humidity                        0
cloud                           0
feels_like_celsius              0
feels_like_fahrenheit           0
visibility_km                   0
visibility_miles                0
uv_index                        0
gust_mph                        0
gust_kph                        0
air_quality_Carbon_Monoxide     0
```

```
air_quality_Ozone                  0
air_quality_Nitrogen_dioxide       0
air_quality_Sulphur_dioxide        0
air_quality_PM2.5                  0
air_quality_PM10                   0
air_quality_us-epa-index           0
air_quality_gb-defra-index         0
sunrise                            0
sunset                             0
moonrise                           0
moonset                            0
moon_phase                         0
moon_illumination                  0
dtype: int64

#Observing Object string values
df[['sunrise','sunset','moonrise','moonset','moon_phase']]

        sunrise    sunset  moonrise   moonset      moon_phase
0      04:50 AM  06:50 PM  12:12 PM  01:11 AM  Waxing Gibbous
1      05:21 AM  07:54 PM  12:58 PM  02:14 AM  Waxing Gibbous
2      05:40 AM  07:50 PM  01:15 PM  02:14 AM  Waxing Gibbous
3      06:31 AM  09:11 PM  02:12 PM  03:31 AM  Waxing Gibbous
4      06:12 AM  05:55 PM  01:17 PM  12:38 AM  Waxing Gibbous
...         ...       ...       ...       ...             ...
59628  06:32 AM  06:38 PM  10:58 PM  09:52 AM  Waning Gibbous
59629  06:02 AM  06:07 PM  10:24 PM  08:41 AM  Waning Gibbous
59630  06:08 AM  06:14 PM  10:27 PM  09:05 AM  Waning Gibbous
59631  06:11 AM  06:18 PM  09:34 PM  10:06 AM  Waning Gibbous
59632  06:00 AM  06:07 PM  09:17 PM  09:59 AM  Waning Gibbous

[59633 rows x 5 columns]

df=df.drop('last_updated_epoch', axis=1)
```

Calculating Measures for input variables is an important step and it would give me an overall idea of how is the data distributed

Also here I'm automating the outliers detection using IQR and Zscore, I used (1.5 * IQR) to put the extreme fences and 3 std Zscore for detecting outliers.

```
#Calculating Important Measures for each numeric column

# Getting IQR and Zscore to provide automatic detection of outliers
stats_df=pd.DataFrame({'Measure':
['Mean','Median','Std','Min','Max','Range','IQR', 'Outliers-IQR',
'Outliers-ZScore']})
for i in df.columns:

    #only numeric columns
    if is_numeric_dtype(df[i]):
```

```
        Q3= df[i].quantile(0.75)
        Q1= df[i].quantile(0.25)
        IQR= Q3 - Q1

        #Calculating ZScore
        df['Zscore']=np.abs(stats.zscore(df[i]))

        #Calculating Upper Fence and Lower Fence for IQR Outliers and
getting outliers
        iqr_out=df[ (df[i] < Q1 - (1.5 * IQR)) |  (df[i] > Q3 + (1.5 *
IQR))]

        #Getting Outliers ZScore based on 3 std away from the mean
        zcore_out=df[df['Zscore']>3]

        # Calculating Masures
        stats_df[i]=[df[i].mean(),
                     df[i].median(),
                     df[i].std(),
                     df[i].min(),
                     df[i].max(),
                     df[i].max() - df[i].min(),
                     IQR,
                     len(iqr_out[i]),
                     len(zcore_out[i])
                    ]
        print("Outliers IQR based:" + i )
        print (iqr_out[i])
        print("")
        print("Outliers ZScore based:" + i )
        print (zcore_out[i])
        print("")
```

```
Outliers IQR based:latitude
Series([], Name: latitude, dtype: float64)

Outliers ZScore based:latitude
Series([], Name: latitude, dtype: float64)

Outliers IQR based:longitude
8         149.2200
58        178.4200
85        139.6900
```

```
89       169.5300
107      171.3800
           ...
59597    159.9500
59614   -175.2000
59619    179.2167
59624   -123.0439
59627    168.3167
Name: longitude, Length: 4588, dtype: float64

Outliers ZScore based:longitude
Series([], Name: longitude, dtype: float64)

Outliers IQR based:temperature_celsius
784      -1.0
3010     45.6
3205     45.7
4373     45.9
4568     46.6
         ...
59147    -1.7
59191    -2.9
59246    -0.1
59275    -5.7
59441    -2.0
Name: temperature_celsius, Length: 1389, dtype: float64

Outliers ZScore based:temperature_celsius
29235    -8.4
29820    -7.9
30600   -10.4
32884   -11.0
33079   -12.1
         ...
57133    -7.6
57328    -7.8
57577    -7.6
58495    -7.5
58661    -8.0
Name: temperature_celsius, Length: 306, dtype: float64

Outliers IQR based:temperature_fahrenheit
784       30.2
3205     114.3
4373     114.6
4568     115.9
5461      28.6
         ...
59063     30.6
59147     28.9
```

```
59191     26.8
59275     21.7
59441     28.4
Name: temperature_fahrenheit, Length: 1369, dtype: float64

Outliers ZScore based:temperature_fahrenheit
29235    16.9
29820    17.8
30600    13.2
32884    12.2
33079    10.3
         ...
57133    18.3
57328    18.0
57577    18.3
58495    18.5
58661    17.5
Name: temperature_fahrenheit, Length: 306, dtype: float64

Outliers IQR based:wind_mph
123      23.0
153      25.5
239      25.5
246      23.0
259      24.2
         ...
59504    22.6
59545    22.8
59550    26.2
59558    24.2
59578    25.1
Name: wind_mph, Length: 890, dtype: float64

Outliers ZScore based:wind_mph
834       106.9
1193      160.8
1827       40.5
3829       40.5
4997       36.7
7248      169.1
7317       36.7
7601     1841.2
8464       43.4
8659       37.8
8675      128.0
9129       50.3
12610      42.9
12897      40.5
13924      36.7
17747      36.7
```

```
19502     36.7
19892     48.5
20087     37.8
22816     47.2
33820     41.2
35380     47.6
39912     36.2
42278     38.5
43180     47.2
43375     39.4
43570     40.5
47658     42.3
48797     43.6
50161     39.1
50551     39.6
51136     37.1
51526     56.6
52696     38.7
52891     38.5
53475     40.0
53670     38.5
53864     36.9
54839     37.4
56593     41.8
Name: wind_mph, dtype: float64

Outliers IQR based:wind_kph
123      37.1
153      41.0
239      41.0
246      37.1
259      38.9
         ...
59504    36.4
59545    36.7
59550    42.1
59558    38.9
59578    40.3
Name: wind_kph, Length: 890, dtype: float64

Outliers ZScore based:wind_kph
834       172.1
1193      258.8
1827       65.2
3829       65.2
4997       59.0
7248      272.2
7317       59.0
7601     2963.2
```

```
8464       69.8
8659       60.8
8675      205.9
9129       81.0
12610      69.1
12897      65.2
13924      59.0
17747      59.0
19502      59.0
19892      78.1
20087      60.8
22816      76.0
33820      66.2
35380      76.7
39912      58.3
42278      61.9
43180      76.0
43375      63.4
43570      65.2
47658      68.0
48797      70.2
50161      63.0
50551      63.7
51136      59.8
51526      91.1
52696      62.3
52891      61.9
53280      58.0
53475      64.4
53670      61.9
53864      59.4
54839      60.1
56593      67.3
Name: wind_kph, dtype: float64

Outliers IQR based:wind_degree
Series([], Name: wind_degree, dtype: int64)

Outliers ZScore based:wind_degree
Series([], Name: wind_degree, dtype: int64)

Outliers IQR based:pressure_mb
119      1033.0
161      1031.0
1894     1031.0
1952     1031.0
2154      997.0
          ...
59580    1033.0
59587    1033.0
```

```
59591    1035.0
59595    1032.0
59596    1033.0
Name: pressure_mb, Length: 2302, dtype: float64

Outliers ZScore based:pressure_mb
30172     971.0
38510    1080.0
45744     972.0
48992     973.0
49582     964.0
49769    3006.0
50941     947.0
52114    3000.0
54449     964.0
54644     969.0
54839     962.0
55034     964.0
Name: pressure_mb, dtype: float64

Outliers IQR based:pressure_in
95       30.39
119      30.50
161      30.45
314      30.39
1764     29.46
         ...
59580    30.50
59587    30.50
59591    30.56
59595    30.47
59596    30.50
Name: pressure_in, Length: 3365, dtype: float64

Outliers ZScore based:pressure_in
30172    28.67
38510    31.89
45744    28.70
48992    28.73
49582    28.47
49769    88.77
50941    27.96
52114    88.59
54449    28.47
54644    28.61
54839    28.41
55034    28.47
Name: pressure_in, dtype: float64

Outliers IQR based:precip_mm
```

```
1        0.10
3        0.30
7        0.13
16       0.25
19       0.16
         ...
59610    0.40
59612    0.08
59619    0.12
59620    0.27
59628    0.84
Name: precip_mm, Length: 11167, dtype: float64

Outliers ZScore based:precip_mm
176      2.09
181      2.00
219      3.01
273      2.05
317      2.42
         ...
59352    1.97
59378    2.35
59417    4.01
59582    3.20
59594    8.52
Name: precip_mm, Length: 762, dtype: float64

Outliers IQR based:precip_in
3        0.01
7        0.01
16       0.01
19       0.01
40       0.01
         ...
59597    0.02
59599    0.02
59610    0.02
59620    0.01
59628    0.03
Name: precip_in, Length: 9371, dtype: float64

Outliers ZScore based:precip_in
176      0.08
181      0.08
219      0.12
273      0.08
317      0.10
         ...
59352    0.08
59378    0.09
```

```
59417    0.16
59582    0.13
59594    0.34
Name: precip_in, Length: 807, dtype: float64

Outliers IQR based:humidity
Series([], Name: humidity, dtype: int64)

Outliers ZScore based:humidity
Series([], Name: humidity, dtype: int64)

Outliers IQR based:cloud
Series([], Name: cloud, dtype: int64)

Outliers ZScore based:cloud
Series([], Name: cloud, dtype: int64)

Outliers IQR based:feels_like_celsius
6816    -4.9
9123    -4.3
28066   -7.1
28612   -5.3
28807   -7.4
         ...
59280   -4.6
59342   -5.2
59426   -4.4
59441   -5.4
59581   -4.1
Name: feels_like_celsius, Length: 1311, dtype: float64

Outliers ZScore based:feels_like_celsius
29040   -13.6
29820   -12.6
30405   -14.6
30600   -14.8
32884   -19.2
         ...
57188   -13.9
57328   -15.5
57577   -14.6
57772   -14.2
58661   -12.7
Name: feels_like_celsius, Length: 314, dtype: float64

Outliers IQR based:feels_like_fahrenheit
6816     23.1
9123     24.2
14117    25.1
28066    19.3
```

```
28612    22.4
          ...
59280    23.7
59342    22.6
59426    24.1
59441    22.2
59581    24.6
Name: feels_like_fahrenheit, Length: 1325, dtype: float64

Outliers ZScore based:feels_like_fahrenheit
29040    7.5
29820    9.4
30405    5.8
30600    5.3
32884   -2.5
         ...
57188    7.1
57328    4.1
57577    5.7
57772    6.5
58661    9.1
Name: feels_like_fahrenheit, Length: 315, dtype: float64

Outliers IQR based:visibility_km
3         2.0
23        0.0
32       24.0
35        7.0
40        7.0
          ...
59597     9.0
59599     9.0
59610     9.0
59624    16.0
59628     9.0
Name: visibility_km, Length: 11632, dtype: float64

Outliers ZScore based:visibility_km
23        0.0
32       24.0
107      24.0
111      24.0
132      23.0
          ...
59375    21.0
59461     0.0
59470    24.0
59549    19.0
59570    21.0
Name: visibility_km, Length: 1620, dtype: float64
```

```
Outliers IQR based:visibility_miles
3         1.0
23        0.0
32       14.0
35        4.0
40        4.0
         ...
59597     5.0
59599     5.0
59610     5.0
59624     9.0
59628     5.0
Name: visibility_miles, Length: 11541, dtype: float64

Outliers ZScore based:visibility_miles
23        0.0
32       14.0
107      14.0
111      14.0
132      14.0
         ...
59375    13.0
59461     0.0
59470    14.0
59549    11.0
59570    13.0
Name: visibility_miles, Length: 1585, dtype: float64

Outliers IQR based:uv_index
Series([], Name: uv_index, dtype: float64)

Outliers ZScore based:uv_index
37958    15.3
39689    15.2
40281    15.0
40567    15.0
40568    15.2
         ...
58746    15.8
58941    15.2
59220    15.2
59415    14.9
59610    15.0
Name: uv_index, Length: 189, dtype: float64

Outliers IQR based:gust_mph
85       32.5
153      30.0
176      35.3
```

```
207      38.5
239      30.0
         ...
59546    30.0
59550    30.1
59561    30.5
59578    36.9
59590    31.3
Name: gust_mph, Length: 1365, dtype: float64

Outliers ZScore based:gust_mph
512       48.0
834      111.4
1193     165.3
1465      43.3
1827      69.8
         ...
56398     47.3
56593     53.3
58569     51.7
58976     48.2
59058     43.7
Name: gust_mph, Length: 128, dtype: float64

Outliers IQR based:gust_kph
85       52.2
153      48.2
176      56.8
207      62.0
239      48.2
         ...
59546    48.3
59550    48.5
59561    49.1
59578    59.4
59590    50.4
Name: gust_kph, Length: 1347, dtype: float64

Outliers ZScore based:gust_kph
512       77.3
834      179.3
1193     266.0
1465      69.6
1827     112.3
         ...
56398     76.1
56593     85.8
58569     83.2
58976     77.6
59058     70.3
```

```
Name: gust_kph, Length: 128, dtype: float64

Outliers IQR based:air_quality_Carbon_Monoxide
4        2964.00
30       1295.10
35       2723.70
36       1335.10
40       1161.60
          ...
59575    1036.00
59600    1052.65
59603    1585.45
59611    1359.75
59628    2099.75
Name: air_quality_Carbon_Monoxide, Length: 5220, dtype: float64

Outliers ZScore based:air_quality_Carbon_Monoxide
78        3471.400
173       3898.600
230       9719.900
245       3845.200
273      19653.301
           ...
59128     5078.250
59279     4253.150
59321     4774.850
59474     3494.650
59516     6184.550
Name: air_quality_Carbon_Monoxide, Length: 761, dtype: float64

Outliers IQR based:air_quality_Ozone
13       188.8
78       303.3
80       161.7
88       160.2
130      173.1
         ...
59213    206.0
59355    151.0
59408    167.0
59559    155.0
59603    182.0
Name: air_quality_Ozone, Length: 1270, dtype: float64

Outliers ZScore based:air_quality_Ozone
13       188.8
78       303.3
130      173.1
191      197.4
275      176.0
```

```
         ...
58001    177.0
58433    180.0
58823    192.0
59213    206.0
59603    182.0
Name: air_quality_Ozone, Length: 596, dtype: float64

Outliers IQR based:air_quality_Nitrogen_dioxide
2         65.100
4         72.700
35        41.800
36       101.500
50        72.700
          ...
59602     73.815
59607     40.515
59618     51.615
59622     70.115
59623     83.250
Name: air_quality_Nitrogen_dioxide, Length: 7474, dtype: float64

Outliers ZScore based:air_quality_Nitrogen_dioxide
36       101.500
79       145.300
103      133.000
230      181.000
231      128.900
          ...
59516    116.735
59517    117.475
59541    114.515
59579     95.645
59600     96.385
Name: air_quality_Nitrogen_dioxide, Length: 1501, dtype: float64

Outliers IQR based:air_quality_Sulphur_dioxide
4         31.500
35        24.800
36       223.200
50        52.000
78        40.100
          ...
59603     76.960
59608     23.125
59611     25.530
59622     38.665
59629     29.415
Name: air_quality_Sulphur_dioxide, Length: 8433, dtype: float64
```

```
Outliers ZScore based:air_quality_Sulphur_dioxide
36       223.200
231      165.900
1007     177.400
1440     200.300
1787     213.600
           ...
58042    165.945
58153    178.155
58169    187.960
58320    181.485
59218    159.470
Name: air_quality_Sulphur_dioxide, Length: 354, dtype: float64

Outliers IQR based:air_quality_PM2.5
4        183.400
35       211.100
36        84.900
68       132.000
78       196.100
           ...
59548     81.030
59589    253.265
59590     66.785
59600     78.625
59601    107.855
Name: air_quality_PM2.5, Length: 4665, dtype: float64

Outliers ZScore based:air_quality_PM2.5
4        183.400
35       211.100
78       196.100
230      714.100
231      228.200
           ...
58571    186.850
59004    234.395
59474    189.995
59516    159.840
59589    253.265
Name: air_quality_PM2.5, Length: 758, dtype: float64

Outliers IQR based:air_quality_PM10
4         262.300
12        114.300
35        268.600
36        107.800
68        178.100
            ...
59589    2729.120
```

```
59590     289.525
59601     627.150
59604     105.080
59630     173.530
Name: air_quality_PM10, Length: 6001, dtype: float64

Outliers ZScore based:air_quality_PM10
230       873.400
273       621.500
468      1002.200
617       682.100
811       544.400
           ...
59211     628.815
59351     646.020
59406     673.585
59589    2729.120
59601     627.150
Name: air_quality_PM10, Length: 609, dtype: float64

Outliers IQR based:air_quality_us-epa-index
4        5
35       5
36       4
68       4
78       5
        ..
59548    4
59589    6
59590    4
59600    4
59601    4
Name: air_quality_us-epa-index, Length: 4726, dtype: int64

Outliers ZScore based:air_quality_us-epa-index
4        5
35       5
78       5
230      6
231      5
        ..
58571    5
59004    5
59474    5
59516    5
59589    6
Name: air_quality_us-epa-index, Length: 851, dtype: int64

Outliers IQR based:air_quality_gb-defra-index
4        10
```

```
35       10
36       10
68       10
78       10
         ..
59590     9
59600    10
59601    10
59611     8
59622     8
Name: air_quality_gb-defra-index, Length: 6563, dtype: int64

Outliers ZScore based:air_quality_gb-defra-index
Series([], Name: air_quality_gb-defra-index, dtype: int64)

Outliers IQR based:moon_illumination
Series([], Name: moon_illumination, dtype: int64)

Outliers ZScore based:moon_illumination
Series([], Name: moon_illumination, dtype: int64)


stats_df.style

<pandas.io.formats.style.Styler at 0x220a0236760>
```

Here I wanted to add certain columns dividing the date getting the month year, weekday and hour that I believe that if the moment in which an observation is taken is really important depending of the month or hour.

Also I am scaling all the numeric values to be between 0-1

```
#Scaling and Adding some columns
scaler=MinMaxScaler()

#Transform 'last_updated' to datetime
df['last_updated']=pd.to_datetime(df['last_updated'])

# Getting date time columns
df['year']=df['last_updated'].dt.year
df['month']=df['last_updated'].dt.month
df['day']=df['last_updated'].dt.day
df['weekday']=df['last_updated'].dt.day_name()
df['hour']=df['last_updated'].dt.hour

# Transforming Month and Hour to catch the cyclic relationship between
hours
df['Hour_sin'] = np.sin(2 * np.pi * df['hour'] / 24)
df['Hour_cos'] = np.cos(2 * np.pi * df['hour'] / 24)
df['Month_sin'] = np.sin(2 * np.pi * df['month'] / 12)
df['Month_cos'] = np.cos(2 * np.pi * df['month'] / 12)
```

```
#Columns to scale
columns = stats_df.columns[1:].tolist()
columns.append('Hour_sin')
columns.append('Hour_cos')
columns.append('Month_sin')
columns.append('Month_cos')

#Scaling values
df_numeric_normalized =
pd.DataFrame(scaler.fit_transform(df[columns]), columns=columns)

df_numeric_normalized
```

```
       latitude  longitude  temperature_celsius
temperature_fahrenheit  \
0      0.719014   0.689521             0.695007
0.694153
1      0.783594   0.550251             0.592443
0.592204
2      0.740256   0.502934             0.646424
0.646177
3      0.794689   0.498617             0.421053
0.420540
4      0.307824   0.531657             0.686910
0.686657
...         ...        ...                  ...                     ..
.
59628  0.491228   0.305523             0.676113
0.676162
59629  0.591117   0.792986             0.636977
0.636432
59630  0.537266   0.619058             0.632928
0.631934
59631  0.245456   0.574130             0.636977
0.637181
59632  0.222686   0.581922             0.639676
0.640180

       wind_mph  wind_kph  wind_degree  pressure_mb  pressure_in
precip_mm  \
0      0.003317  0.003277     0.938719     0.031569     0.031738
0.000000
1      0.002556  0.002568     0.888579     0.031569     0.031574
0.002367
2      0.003915  0.003886     0.777159     0.031083     0.031080
0.000000
3      0.002828  0.002804     0.596100     0.029140     0.029436
```

```
0.007102
4      0.003208  0.003176     0.415042     0.031083     0.031080
0.000000
...         ...       ...          ...          ...          ...
...
59628  0.000000  0.000000     0.136490     0.032054     0.032067
0.019886
59629  0.002828  0.002804     0.977716     0.036911     0.037000
0.000000
59630  0.000272  0.000237     0.529248     0.033511     0.033547
0.000000
59631  0.004296  0.004257     0.225627     0.033997     0.034205
0.000237
59632  0.001468  0.001453     0.178273     0.034483     0.034534
0.000000

       ...  air_quality_Sulphur_dioxide  air_quality_PM2.5
air_quality_PM10  \
0      ...                     0.950464           0.005090
0.237748
1      ...                     0.950455           0.000567
0.234629
2      ...                     0.951719           0.006329
0.236708
3      ...                     0.950464           0.000319
0.234489
4      ...                     0.953440           0.113522
0.267639
...    ...                          ...                ...
...
59628  ...                     0.950727           0.010890
0.237355
59629  ...                     0.953241           0.022123
0.239771
59630  ...                     0.950498           0.017767
0.256381
59631  ...                     0.950516           0.003324
0.235220
59632  ...                     0.950498           0.005846
0.235759

       air_quality_us-epa-index  air_quality_gb-defra-index  \
0                           0.0                    0.000000
1                           0.0                    0.000000
2                           0.0                    0.000000
3                           0.0                    0.000000
4                           0.8                    1.000000
...                         ...                         ...
59628                       0.2                    0.111111
```

```
59629                        0.2                       0.222222
59630                        0.2                       0.222222
59631                        0.0                       0.000000
59632                        0.0                       0.000000

       moon_illumination  Hour_sin  Hour_cos  Month_sin  Month_cos
0                   0.55  0.370590  0.017037       0.75   0.066987
1                   0.55  0.750000  0.066987       0.75   0.066987
2                   0.55  0.853553  0.146447       0.75   0.066987
3                   0.55  0.750000  0.066987       0.75   0.066987
4                   0.55  0.853553  0.146447       0.75   0.066987
...                  ...       ...       ...        ...        ...
59628               0.80  0.982963  0.629410       1.00   0.500000
59629               0.83  0.066987  0.250000       1.00   0.500000
59630               0.82  0.500000  0.000000       1.00   0.500000
59631               0.82  0.629410  0.017037       1.00   0.500000
59632               0.82  0.629410  0.017037       1.00   0.500000

[59633 rows x 33 columns]
```

```
#Combining Categorical and Numerical columns
df_scaled = df.select_dtypes(exclude=['number'])
df_scaled[columns]= df_numeric_normalized
df_scaled
```

```
           country     location_name         timezone
last_updated  \
0      Afghanistan             Kabul       Asia/Kabul 2024-05-16
13:15:00
1          Albania            Tirana    Europe/Tirane 2024-05-16
10:45:00
2          Algeria           Algiers   Africa/Algiers 2024-05-16
09:45:00
3          Andorra  Andorra La Vella   Europe/Andorra 2024-05-16
10:45:00
4           Angola            Luanda    Africa/Luanda 2024-05-16
09:45:00
...            ...               ...              ...
...
59628    Venezuela           Caracas  America/Caracas 2025-03-19
05:30:00
59629      Vietnam             Hanoi     Asia/Bangkok 2025-03-19
16:30:00
59630        Yemen             Sanaa        Asia/Aden 2025-03-19
12:45:00
59631       Zambia            Lusaka    Africa/Lusaka 2025-03-19
11:45:00
59632     Zimbabwe            Harare    Africa/Harare 2025-03-19
11:30:00
```

```
           condition_text wind_direction   sunrise    sunset  moonrise
\
0           Partly Cloudy            NNW  04:50 AM  06:50 PM  12:12 PM

1           Partly cloudy             NW  05:21 AM  07:54 PM  12:58 PM

2                   Sunny              W  05:40 AM  07:50 PM  01:15 PM

3           Light drizzle             SW  06:31 AM  09:11 PM  02:12 PM

4           Partly cloudy            SSE  06:12 AM  05:55 PM  01:17 PM

...                   ...            ...       ...       ...       ...

59628               Clear             NE  06:32 AM  06:38 PM  10:58 PM

59629               Sunny              N  06:02 AM  06:07 PM  10:24 PM

59630               Sunny            SSW  06:08 AM  06:14 PM  10:27 PM

59631  Patchy rain nearby              E  06:11 AM  06:18 PM  09:34 PM

59632               Sunny            ENE  06:00 AM  06:07 PM  09:17 PM


        moonset  ... air_quality_Sulphur_dioxide air_quality_PM2.5  \
0      01:11 AM  ...                    0.950464          0.005090
1      02:14 AM  ...                    0.950455          0.000567
2      02:14 AM  ...                    0.951719          0.006329
3      03:31 AM  ...                    0.950464          0.000319
4      12:38 AM  ...                    0.953440          0.113522
...         ...  ...                         ...               ...
59628  09:52 AM  ...                    0.950727          0.010890
59629  08:41 AM  ...                    0.953241          0.022123
59630  09:05 AM  ...                    0.950498          0.017767
59631  10:06 AM  ...                    0.950516          0.003324
59632  09:59 AM  ...                    0.950498          0.005846

       air_quality_PM10  air_quality_us-epa-index  air_quality_gb-
defra-index  \
0              0.237748                       0.0
0.000000
1              0.234629                       0.0
0.000000
2              0.236708                       0.0
0.000000
3              0.234489                       0.0
0.000000
4              0.267639                       0.8
1.000000
```

```
...                  ...                       ...
...
59628          0.237355                       0.2
0.111111
59629          0.239771                       0.2
0.222222
59630          0.256381                       0.2
0.222222
59631          0.235220                       0.0
0.000000
59632          0.235759                       0.0
0.000000

       moon_illumination  Hour_sin  Hour_cos  Month_sin  Month_cos
0                   0.55  0.370590  0.017037       0.75   0.066987
1                   0.55  0.750000  0.066987       0.75   0.066987
2                   0.55  0.853553  0.146447       0.75   0.066987
3                   0.55  0.750000  0.066987       0.75   0.066987
4                   0.55  0.853553  0.146447       0.75   0.066987
...                  ...       ...       ...        ...        ...
59628               0.80  0.982963  0.629410       1.00   0.500000
59629               0.83  0.066987  0.250000       1.00   0.500000
59630               0.82  0.500000  0.000000       1.00   0.500000
59631               0.82  0.629410  0.017037       1.00   0.500000
59632               0.82  0.629410  0.017037       1.00   0.500000

[59633 rows x 45 columns]
```

When I was exploring the data I notice that the columns of sunset, sunrise, moonrise, moonset were in a string format not 0 - 23 so here I am paring those columns to be 0-23

```
#Parsing Hour of the day for sunrise

df_scaled['split1']=df_scaled['sunrise'].str[:2]
df_scaled['split2']=df_scaled['sunrise'].str[-2:]
df_scaled['sunrise']=df_scaled.apply(lambda x: np.nan if not
x['split1'].isnumeric() else

(int(x['split1']) if x['split2']=='AM' and int(x['split1'])!=12 else

(0 if x['split2']=='AM' and int(x['split1'])==12 else

(int(x['split1'])+12 if int(x['split1'])!=12

else 12))),axis=1)
df_scaled=df_scaled.drop('split1', axis=1)
df_scaled=df_scaled.drop('split2', axis=1)
```

```python
#Parsing Hour of the day for sunset

df_scaled['split1']=df_scaled['sunset'].str[:2]
df_scaled['split2']=df_scaled['sunset'].str[-2:]
df_scaled['sunset']=df_scaled.apply(lambda x: np.nan if not
x['split1'].isnumeric() else

(int(x['split1']) if x['split2']=='AM' and int(x['split1'])!=12 else

(0 if x['split2']=='AM' and int(x['split1'])==12 else

(int(x['split1'])+12 if int(x['split1'])!=12

else 12))),axis=1)
df_scaled=df_scaled.drop('split1', axis=1)
df_scaled=df_scaled.drop('split2', axis=1)

#Parsing Hour of the day for moonrise

df_scaled['split1']=df_scaled['moonrise'].str[:2]
df_scaled['split2']=df_scaled['moonrise'].str[-2:]
df_scaled['moonrise']=df_scaled.apply(lambda x: np.nan if not
x['split1'].isnumeric() else

(int(x['split1']) if x['split2']=='AM' and int(x['split1'])!=12 else

(0 if x['split2']=='AM' and int(x['split1'])==12 else

(int(x['split1'])+12 if int(x['split1'])!=12

else 12))),axis=1)
df_scaled=df_scaled.drop('split1', axis=1)
df_scaled=df_scaled.drop('split2', axis=1)

#Parsing Hour of the day for moonset

df_scaled['split1']=df_scaled['moonset'].str[:2]
df_scaled['split2']=df_scaled['moonset'].str[-2:]
df_scaled['moonset']=df_scaled.apply(lambda x: np.nan if not
x['split1'].isnumeric() else

(int(x['split1']) if x['split2']=='AM' and int(x['split1'])!=12 else

(0 if x['split2']=='AM' and int(x['split1'])==12 else

(int(x['split1'])+12 if int(x['split1'])!=12

else 12))),axis=1)
df_scaled=df_scaled.drop('split1', axis=1)
df_scaled=df_scaled.drop('split2', axis=1)
```

```
df_scaled

            country     location_name         timezone
last_updated  \
0     Afghanistan             Kabul       Asia/Kabul 2024-05-16
13:15:00
1         Albania            Tirana    Europe/Tirane 2024-05-16
10:45:00
2         Algeria           Algiers   Africa/Algiers 2024-05-16
09:45:00
3         Andorra  Andorra La Vella   Europe/Andorra 2024-05-16
10:45:00
4          Angola            Luanda    Africa/Luanda 2024-05-16
09:45:00
...           ...               ...              ...
...
59628   Venezuela           Caracas  America/Caracas 2025-03-19
05:30:00
59629     Vietnam             Hanoi     Asia/Bangkok 2025-03-19
16:30:00
59630       Yemen             Sanaa        Asia/Aden 2025-03-19
12:45:00
59631      Zambia            Lusaka    Africa/Lusaka 2025-03-19
11:45:00
59632    Zimbabwe            Harare    Africa/Harare 2025-03-19
11:30:00

            condition_text wind_direction  sunrise  sunset  moonrise
moonset  \
0            Partly Cloudy            NNW        4      18      12.0
1.0
1            Partly cloudy             NW        5      19      12.0
2.0
2                    Sunny              W        5      19      13.0
2.0
3            Light drizzle             SW        6      21      14.0
3.0
4            Partly cloudy            SSE        6      17      13.0
0.0
...                    ...            ...      ...     ...       ...
...
59628                Clear             NE        6      18      22.0
9.0
59629                Sunny              N        6      18      22.0
8.0
59630                Sunny            SSW        6      18      22.0
9.0
59631  Patchy rain nearby              E        6      18      21.0
10.0
59632                Sunny            ENE        6      18      21.0
```

```
9.0

       ... air_quality_Sulphur_dioxide air_quality_PM2.5
air_quality_PM10  \
0      ...                    0.950464          0.005090
0.237748
1      ...                    0.950455          0.000567
0.234629
2      ...                    0.951719          0.006329
0.236708
3      ...                    0.950464          0.000319
0.234489
4      ...                    0.953440          0.113522
0.267639
...    ...                         ...               ...
...
59628  ...                    0.950727          0.010890
0.237355
59629  ...                    0.953241          0.022123
0.239771
59630  ...                    0.950498          0.017767
0.256381
59631  ...                    0.950516          0.003324
0.235220
59632  ...                    0.950498          0.005846
0.235759

       air_quality_us-epa-index  air_quality_gb-defra-index  \
0                           0.0                    0.000000
1                           0.0                    0.000000
2                           0.0                    0.000000
3                           0.0                    0.000000
4                           0.8                    1.000000
...                         ...                         ...
59628                       0.2                    0.111111
59629                       0.2                    0.222222
59630                       0.2                    0.222222
59631                       0.0                    0.000000
59632                       0.0                    0.000000

       moon_illumination  Hour_sin  Hour_cos  Month_sin  Month_cos
0                   0.55  0.370590  0.017037       0.75   0.066987
1                   0.55  0.750000  0.066987       0.75   0.066987
2                   0.55  0.853553  0.146447       0.75   0.066987
3                   0.55  0.750000  0.066987       0.75   0.066987
4                   0.55  0.853553  0.146447       0.75   0.066987
...                  ...       ...       ...        ...        ...
59628               0.80  0.982963  0.629410       1.00   0.500000
59629               0.83  0.066987  0.250000       1.00   0.500000
59630               0.82  0.500000  0.000000       1.00   0.500000
```

```
59631                    0.82  0.629410  0.017037       1.00   0.500000
59632                    0.82  0.629410  0.017037       1.00   0.500000

[59633 rows x 45 columns]
```

Here I start plotting, correlation matrix to revie correlations, of course there are some variables that I expecto to be highly correlated like temperature_celsius and temperature_fahrenheit

```
#Building a correlation Matrix
correlation_matrix = df_scaled[columns].corr()
correlation_matrix
```

```
                             latitude  longitude  temperature_celsius  \
latitude                     1.000000  -0.020472            -0.360750

longitude                   -0.020472   1.000000             0.092861

temperature_celsius         -0.360750   0.092861             1.000000

temperature_fahrenheit      -0.360738   0.092847             0.999997

wind_mph                     0.020455   0.024700             0.065894

wind_kph                     0.020485   0.024626             0.065820

wind_degree                  0.166279   0.064298            -0.047451

pressure_mb                  0.055717  -0.076264            -0.274208

pressure_in                  0.055882  -0.075219            -0.274797

precip_mm                   -0.053977   0.054391             0.020900

precip_in                   -0.052638   0.054409             0.020956

humidity                    -0.075062  -0.177452            -0.345254

cloud                       -0.048952   0.014072            -0.174460

feels_like_celsius          -0.384608   0.093044             0.981122

feels_like_fahrenheit       -0.384603   0.093047             0.981117

visibility_km               -0.030959   0.129263             0.089081

visibility_miles            -0.032864   0.128075             0.093096

uv_index                    -0.176908  -0.023290             0.531248
```

```
gust_mph                        0.005723   0.020047              0.080642
gust_kph                        0.005746   0.020032              0.080661
air_quality_Carbon_Monoxide    -0.036401   0.106860             -0.083877
air_quality_Ozone               0.105086   0.033032              0.288131
air_quality_Nitrogen_dioxide    0.243139   0.138328             -0.287191
air_quality_Sulphur_dioxide     0.056135   0.077484             -0.081419
air_quality_PM2.5              -0.008816   0.071967             -0.061818
air_quality_PM10               -0.001675   0.030109              0.039651
air_quality_us-epa-index        0.060135   0.126316             -0.057699
air_quality_gb-defra-index      0.061964   0.122535             -0.047209
moon_illumination              -0.000289  -0.001131             -0.014185
Hour_sin                       -0.117487  -0.703979             -0.233213
Hour_cos                       -0.291952   0.140608              0.076940
Month_sin                       0.000908  -0.001179             -0.229576
Month_cos                      -0.000320   0.003569             -0.354801

                              temperature_fahrenheit  wind_mph
wind_kph  \
latitude                                   -0.360738  0.020455
0.020485
longitude                                   0.092847  0.024700
0.024626
temperature_celsius                         0.999997  0.065894
0.065820
temperature_fahrenheit                      1.000000  0.065910
0.065836
wind_mph                                    0.065910  1.000000
0.999992
wind_kph                                    0.065836  0.999992
1.000000
wind_degree                                -0.047436  0.003051
0.003043
pressure_mb                                -0.274201 -0.055280 -
0.055250
pressure_in                                -0.274791 -0.055476 -
0.055445
```

```
precip_mm                                          0.020910  0.001028
0.001002
precip_in                                          0.020966  0.000430
0.000403
humidity                                          -0.345254 -0.065447 -
0.065415
cloud                                             -0.174451  0.009650
0.009661
feels_like_celsius                                 0.981122  0.048397
0.048339
feels_like_fahrenheit                              0.981117  0.048416
0.048358
visibility_km                                      0.089086  0.055764
0.055695
visibility_miles                                   0.093099  0.057769
0.057707
uv_index                                           0.531252  0.049038
0.048987
gust_mph                                           0.080655  0.952600
0.952626
gust_kph                                           0.080674  0.952597
0.952623
air_quality_Carbon_Monoxide                       -0.083877 -0.087321 -
0.087301
air_quality_Ozone                                  0.288146  0.070747
0.070747
air_quality_Nitrogen_dioxide                      -0.287183 -0.108204 -
0.108187
air_quality_Sulphur_dioxide                       -0.081417 -0.043374 -
0.043360
air_quality_PM2.5                                 -0.061810 -0.046413 -
0.046441
air_quality_PM10                                   0.039658  0.033776
0.033713
air_quality_us-epa-index                          -0.057699 -0.065643 -
0.065681
air_quality_gb-defra-index                        -0.047210 -0.058086 -
0.058119
moon_illumination                                 -0.014187  0.008497
0.008491
Hour_sin                                          -0.233204 -0.063190 -
0.063147
Hour_cos                                           0.076925 -0.029292 -
0.029262
Month_sin                                         -0.229575  0.007970
0.007973
Month_cos                                         -0.354802 -0.043862 -
0.043886
```

```
                              wind_degree  pressure_mb  pressure_in  \
latitude                         0.166279     0.055717     0.055882
longitude                        0.064298    -0.076264    -0.075219
temperature_celsius             -0.047451    -0.274208    -0.274797
temperature_fahrenheit          -0.047436    -0.274201    -0.274791
wind_mph                         0.003051    -0.055280    -0.055476
wind_kph                         0.003043    -0.055250    -0.055445
wind_degree                      1.000000    -0.035822    -0.035533
pressure_mb                     -0.035822     1.000000     0.999873
pressure_in                     -0.035533     0.999873     1.000000
precip_mm                        0.012641    -0.062187    -0.061979
precip_in                        0.013013    -0.061343    -0.061138
humidity                        -0.037420     0.006319     0.006198
cloud                            0.010097    -0.024254    -0.024236
feels_like_celsius              -0.053672    -0.273277    -0.273845
feels_like_fahrenheit           -0.053649    -0.273279    -0.273847
visibility_km                   -0.077087    -0.014825    -0.015371
visibility_miles                -0.077014    -0.013735    -0.014210
uv_index                         0.027623    -0.073106    -0.073561
gust_mph                        -0.001167    -0.082262    -0.082546
gust_kph                        -0.001144    -0.082263    -0.082546
air_quality_Carbon_Monoxide      0.018423     0.026819     0.027279
air_quality_Ozone                0.072918    -0.105671    -0.105882
air_quality_Nitrogen_dioxide     0.067661     0.101824     0.102609
air_quality_Sulphur_dioxide      0.040488     0.039040     0.039241
air_quality_PM2.5               -0.009652     0.037686     0.037811
air_quality_PM10                -0.020724     0.008870     0.008719
air_quality_us-epa-index        -0.000689     0.054726     0.054910
air_quality_gb-defra-index      -0.000876     0.048552     0.048720
moon_illumination               -0.015165     0.004325     0.004248
Hour_sin                        -0.171327     0.110166     0.109265
Hour_cos                        -0.113386    -0.116270    -0.115996
Month_sin                       -0.038939     0.068977     0.068855
Month_cos                       -0.066503     0.124114     0.124354

                              precip_mm  ...  
air_quality_Sulphur_dioxide  \
latitude                      -0.053977  ...  
0.056135
longitude                      0.054391  ...  
0.077484
temperature_celsius            0.020900  ...                     -
0.081419
temperature_fahrenheit         0.020910  ...                     -
0.081417
wind_mph                       0.001028  ...                     -
0.043374
wind_kph                       0.001002  ...                     -
0.043360
```

```
wind_degree                       0.012641  ...
0.040488
pressure_mb                      -0.062187  ...
0.039040
pressure_in                      -0.061979  ...
0.039241
precip_mm                         1.000000  ...                      -
0.020338
precip_in                         0.998237  ...                      -
0.018903
humidity                          0.183946  ...                      -
0.063358
cloud                             0.214786  ...                      -
0.070107
feels_like_celsius                0.049077  ...                      -
0.079585
feels_like_fahrenheit             0.049081  ...                      -
0.079607
visibility_km                    -0.046091  ...                      -
0.036296
visibility_miles                 -0.056418  ...                      -
0.039468
uv_index                         -0.067069  ...                      -
0.071125
gust_mph                          0.041330  ...                      -
0.059642
gust_kph                          0.041318  ...                      -
0.059666
air_quality_Carbon_Monoxide       0.009413  ...
0.275804
air_quality_Ozone                -0.086634  ...                      -
0.030622
air_quality_Nitrogen_dioxide     -0.027682  ...
0.366879
air_quality_Sulphur_dioxide      -0.020338  ...
1.000000
air_quality_PM2.5                -0.047968  ...
0.252439
air_quality_PM10                 -0.042026  ...
0.114911
air_quality_us-epa-index         -0.074251  ...
0.283694
air_quality_gb-defra-index       -0.070285  ...
0.280532
moon_illumination                 0.005482  ...                      -
0.001131
Hour_sin                         -0.044366  ...                      -
0.065229
Hour_cos                          0.076860  ...                      -
```

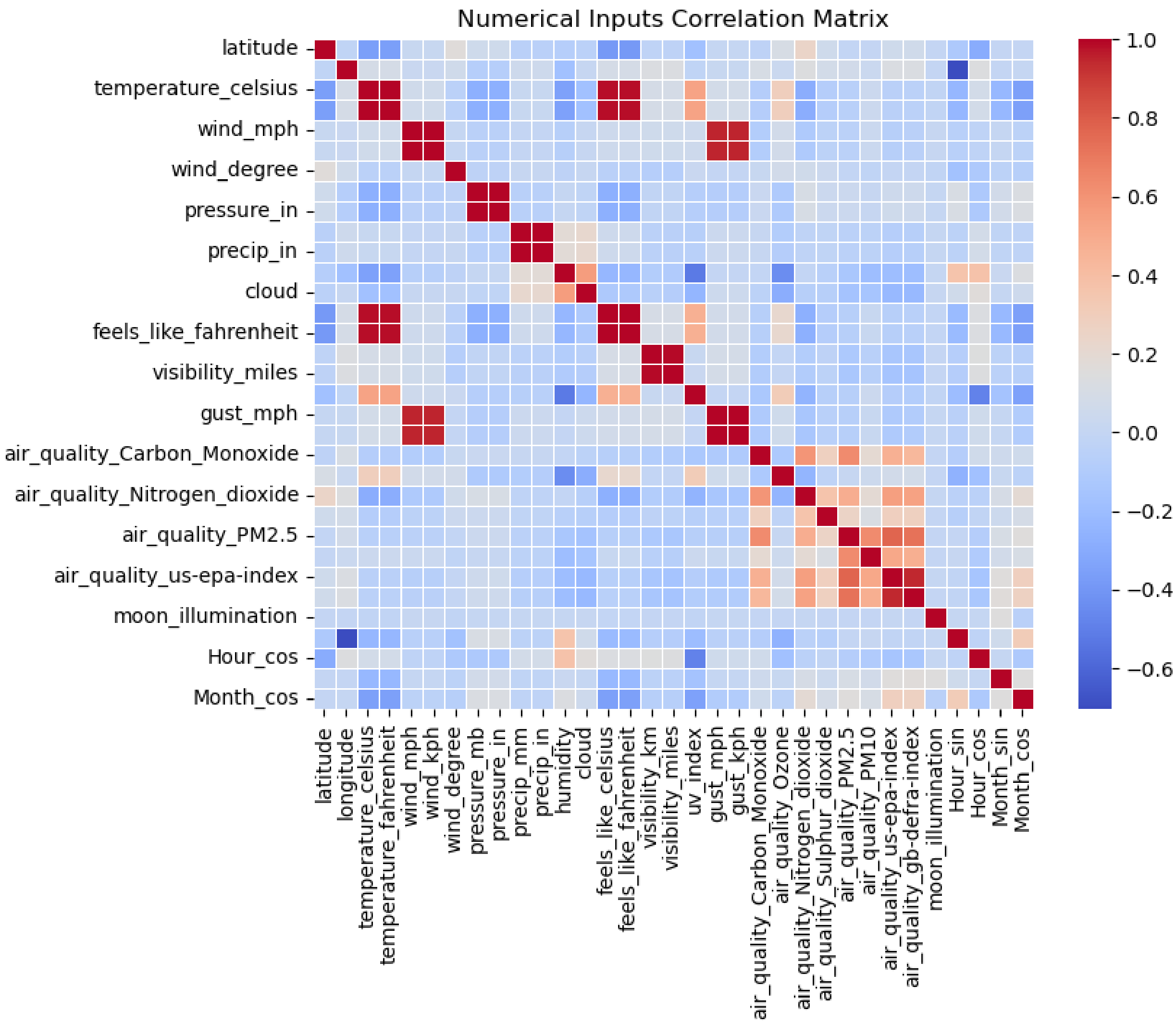
```
0.014875
Month_sin                             -0.018893  ...
0.042896
Month_cos                             -0.028547  ...
0.090919

                              air_quality_PM2.5  air_quality_PM10  \
latitude                              -0.008816         -0.001675
longitude                              0.071967          0.030109
temperature_celsius                   -0.061818          0.039651
temperature_fahrenheit                -0.061810          0.039658
wind_mph                              -0.046413          0.033776
wind_kph                              -0.046441          0.033713
wind_degree                           -0.009652         -0.020724
pressure_mb                            0.037686          0.008870
pressure_in                            0.037811          0.008719
precip_mm                             -0.047968         -0.042026
precip_in                             -0.046252         -0.040685
humidity                              -0.135112         -0.194318
cloud                                 -0.160064         -0.150691
feels_like_celsius                    -0.068467          0.020729
feels_like_fahrenheit                 -0.068490          0.020719
visibility_km                         -0.123629         -0.059455
visibility_miles                      -0.134258         -0.065278
uv_index                              -0.066759          0.036126
gust_mph                              -0.071034          0.020886
gust_kph                              -0.071053          0.020879
air_quality_Carbon_Monoxide            0.632877          0.200310
air_quality_Ozone                      0.009886          0.046597
air_quality_Nitrogen_dioxide           0.486304          0.188252
air_quality_Sulphur_dioxide            0.252439          0.114911
air_quality_PM2.5                      1.000000          0.627611
air_quality_PM10                       0.627611          1.000000
air_quality_us-epa-index               0.769686          0.513657
air_quality_gb-defra-index             0.720709          0.480097
moon_illumination                     -0.002465         -0.001317
Hour_sin                              -0.005076          0.009843
Hour_cos                              -0.080553         -0.103605
Month_sin                              0.104827          0.072800
Month_cos                              0.160459          0.106816

                              air_quality_us-epa-index  \
latitude                                      0.060135
longitude                                     0.126316
temperature_celsius                          -0.057699
temperature_fahrenheit                       -0.057699
wind_mph                                     -0.065643
wind_kph                                     -0.065681
wind_degree                                  -0.000689
```

```
pressure_mb                              0.054726
pressure_in                              0.054910
precip_mm                               -0.074251
precip_in                               -0.071787
humidity                                -0.196531
cloud                                   -0.222682
feels_like_celsius                      -0.068775
feels_like_fahrenheit                   -0.068819
visibility_km                           -0.140602
visibility_miles                        -0.153333
uv_index                                -0.082052
gust_mph                                -0.105203
gust_kph                                -0.105229
air_quality_Carbon_Monoxide              0.471069
air_quality_Ozone                        0.072649
air_quality_Nitrogen_dioxide             0.548457
air_quality_Sulphur_dioxide              0.283694
air_quality_PM2.5                        0.769686
air_quality_PM10                         0.513657
air_quality_us-epa-index                 1.000000
air_quality_gb-defra-index               0.940809
moon_illumination                        0.003262
Hour_sin                                -0.013703
Hour_cos                                -0.149723
Month_sin                                0.159906
Month_cos                                0.284928

                              air_quality_gb-defra-index
moon_illumination  \
latitude                                        0.061964           -
0.000289
longitude                                       0.122535           -
0.001131
temperature_celsius                            -0.047209           -
0.014185
temperature_fahrenheit                         -0.047210           -
0.014187
wind_mph                                       -0.058086
0.008497
wind_kph                                       -0.058119
0.008491
wind_degree                                    -0.000876           -
0.015165
pressure_mb                                     0.048552
0.004325
pressure_in                                     0.048720
0.004248
precip_mm                                      -0.070285
0.005482
```

```
precip_in                                            -0.067839
0.005246
humidity                                             -0.196240
0.007630
cloud                                                -0.221123
0.012135
feels_like_celsius                                   -0.059620          -
0.011345
feels_like_fahrenheit                                -0.059669          -
0.011333
visibility_km                                        -0.140553          -
0.002241
visibility_miles                                     -0.154479          -
0.003847
uv_index                                             -0.088459          -
0.018348
gust_mph                                             -0.093380
0.006626
gust_kph                                             -0.093412
0.006650
air_quality_Carbon_Monoxide                           0.433213          -
0.001637
air_quality_Ozone                                     0.075244          -
0.003162
air_quality_Nitrogen_dioxide                          0.540302
0.002133
air_quality_Sulphur_dioxide                           0.280532          -
0.001131
air_quality_PM2.5                                     0.720709          -
0.002465
air_quality_PM10                                      0.480097          -
0.001317
air_quality_us-epa-index                              0.940809
0.003262
air_quality_gb-defra-index                            1.000000
0.000157
moon_illumination                                     0.000157
1.000000
Hour_sin                                             -0.015624
0.006133
Hour_cos                                             -0.135998
0.006608
Month_sin                                             0.163650
0.153978
Month_cos                                             0.272945          -
0.055846

                              Hour_sin  Hour_cos  Month_sin  Month_cos
```

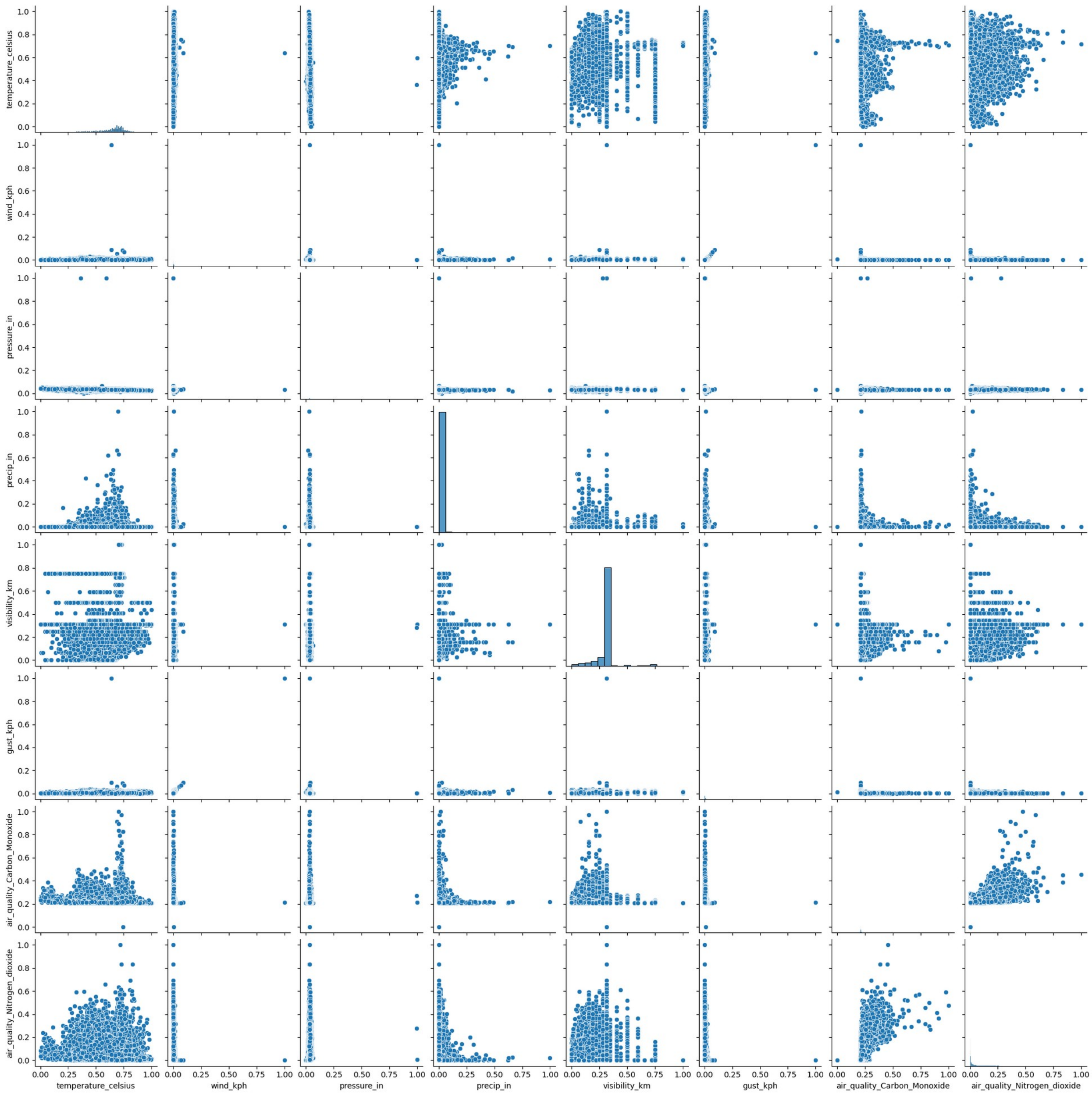
latitude	-0.117487	-0.291952	0.000908	-0.000320
longitude	-0.703979	0.140608	-0.001179	0.003569
temperature_celsius	-0.233213	0.076940	-0.229576	-0.354801
temperature_fahrenheit	-0.233204	0.076925	-0.229575	-0.354802
wind_mph	-0.063190	-0.029292	0.007970	-0.043862
wind_kph	-0.063147	-0.029262	0.007973	-0.043886
wind_degree	-0.171327	-0.113386	-0.038939	-0.066503
pressure_mb	0.110166	-0.116270	0.068977	0.124114
pressure_in	0.109265	-0.115996	0.068855	0.124354
precip_mm	-0.044366	0.076860	-0.018893	-0.028547
precip_in	-0.044819	0.074977	-0.018558	-0.028431
humidity	0.362730	0.371332	0.024546	0.132608
cloud	0.062810	0.165537	0.002372	0.050845
feels_like_celsius	-0.212557	0.128124	-0.224689	-0.353872
feels_like_fahrenheit	-0.212563	0.128146	-0.224698	-0.353886
visibility_km	-0.077997	0.142437	-0.042055	-0.066737
visibility_miles	-0.082728	0.136490	-0.048847	-0.071232
uv_index	-0.203808	-0.488760	-0.164533	-0.349198
gust_mph	-0.057325	0.060583	0.006230	-0.094471
gust_kph	-0.057360	0.060566	0.006216	-0.094514
air_quality_Carbon_Monoxide	-0.079739	0.061676	0.052961	0.056585
air_quality_Ozone	-0.258583	-0.174392	0.019732	-0.044031
air_quality_Nitrogen_dioxide	-0.050640	-0.048762	0.096101	0.190322
air_quality_Sulphur_dioxide	-0.065229	-0.014875	0.042896	0.090919
air_quality_PM2.5	-0.005076	-0.080553	0.104827	0.160459
air_quality_PM10	0.009843	-0.103605	0.072800	0.106816

```
air_quality_us-epa-index      -0.013703 -0.149723   0.159906   0.284928
air_quality_gb-defra-index    -0.015624 -0.135998   0.163650   0.272945
moon_illumination              0.006133  0.006608   0.153978  -0.055846
Hour_sin                       1.000000 -0.030083   0.055253   0.308274
Hour_cos                      -0.030083  1.000000   0.003468  -0.116881
Month_sin                      0.055253  0.003468   1.000000   0.148197
Month_cos                      0.308274 -0.116881   0.148197   1.000000

[33 rows x 33 columns]
#Plotting Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm',
fmt=".2f", linewidths=0.5)
plt.title("Numerical Inputs Correlation Matrix")
plt.show()
```



Aside from the variables in different measures, we can see that the air quality between some components has high correlations; still, I don't believe it is good to remove those.

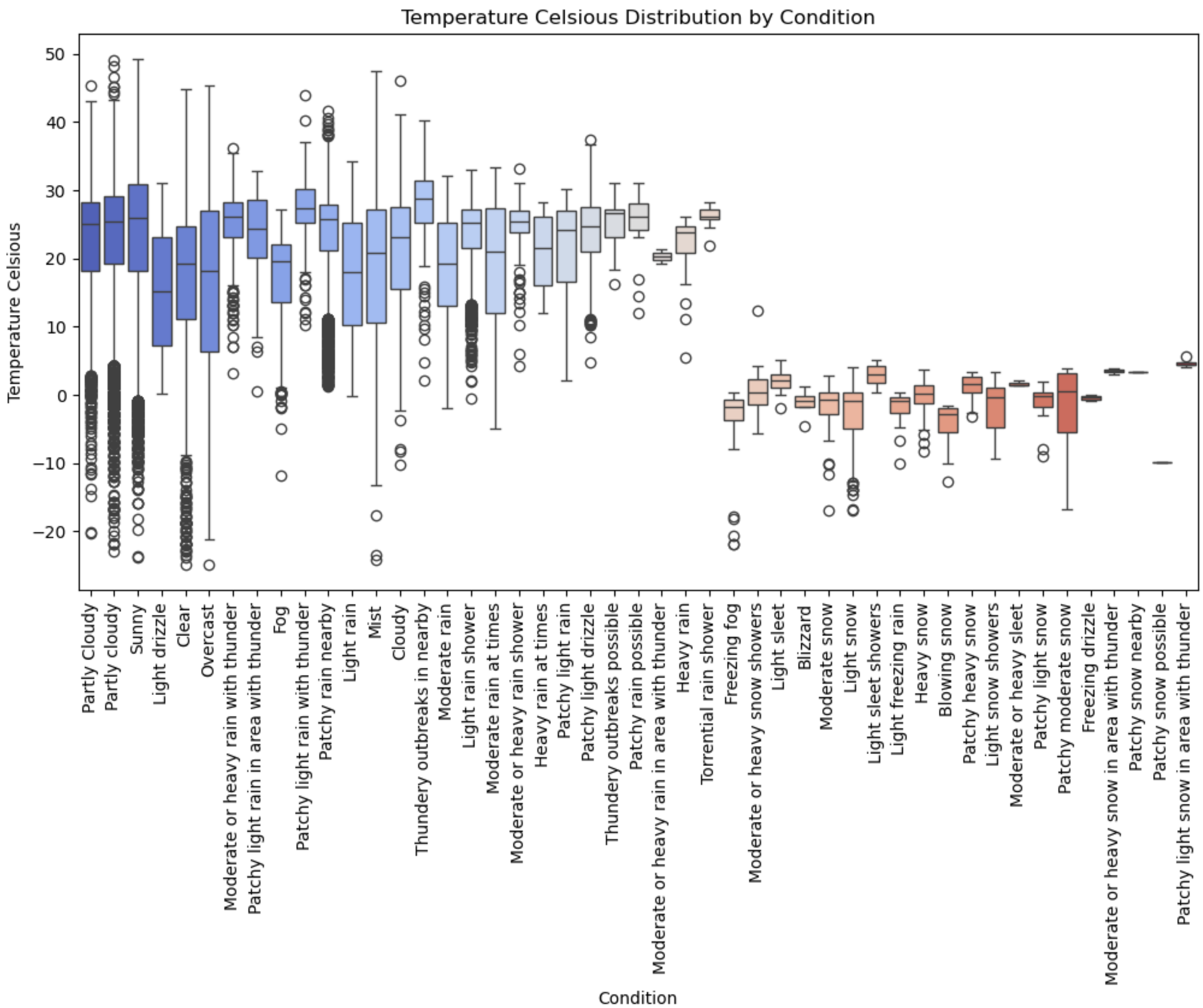
```
#Plotting some Interest Variables to get relations

sns.pairplot(df_scaled[['temperature_celsius','wind_kph','pressure_in'
,'precip_in','visibility_km','gust_kph','air_quality_Carbon_Monoxide',
'air_quality_Nitrogen_dioxide']])

plt.show()
```



some interest variables relations some of the with linear relation

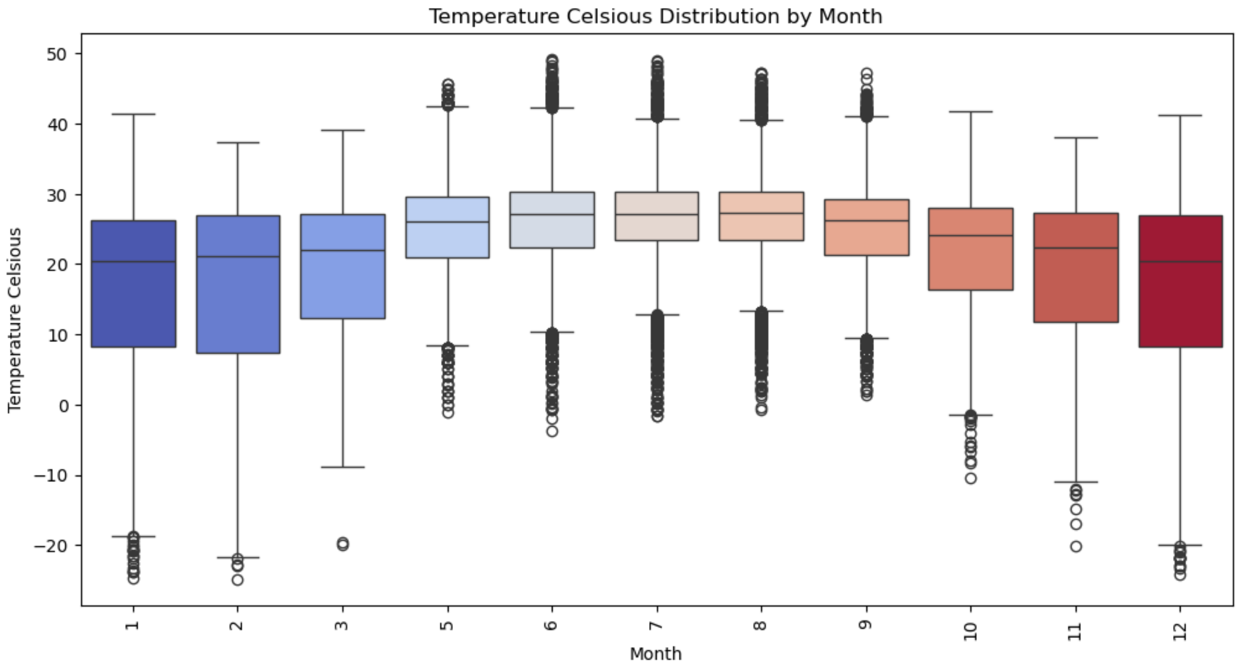
```
#Plotting boxplots for temperature and condition

plt.figure(figsize=(12, 6))
sns.boxplot(
    x=df['condition_text'],
    y=df['temperature_celsius'],
    hue=df['condition_text'],
    palette="coolwarm",
    legend=False
)
```

```python
plt.xticks(rotation=90)
plt.xlabel("Condition")
plt.ylabel("Temperature Celsious")
plt.title("Temperature Celsious Distribution by Condition")
plt.show()
```

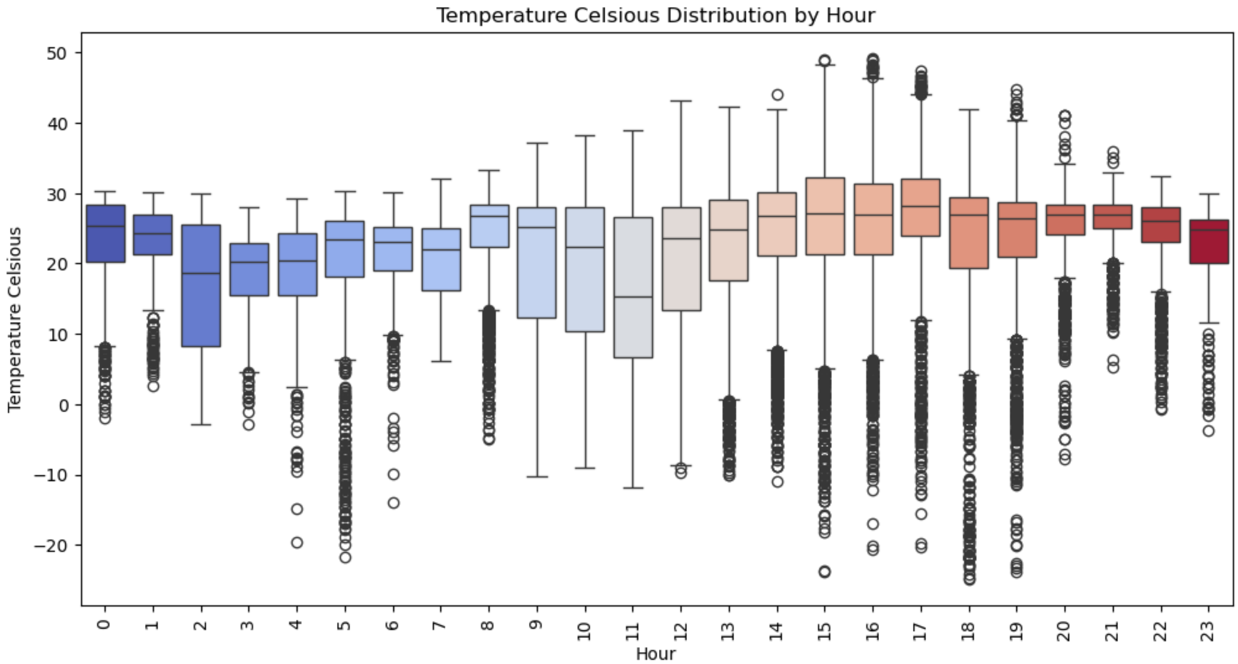



```python
plt.figure(figsize=(12, 6))
sns.boxplot(
    x=df['month'],
    y=df['temperature_celsius'],
    hue=df['month'],
    palette="coolwarm",
    legend=False
)

plt.xticks(rotation=90)
plt.xlabel("Month")
```

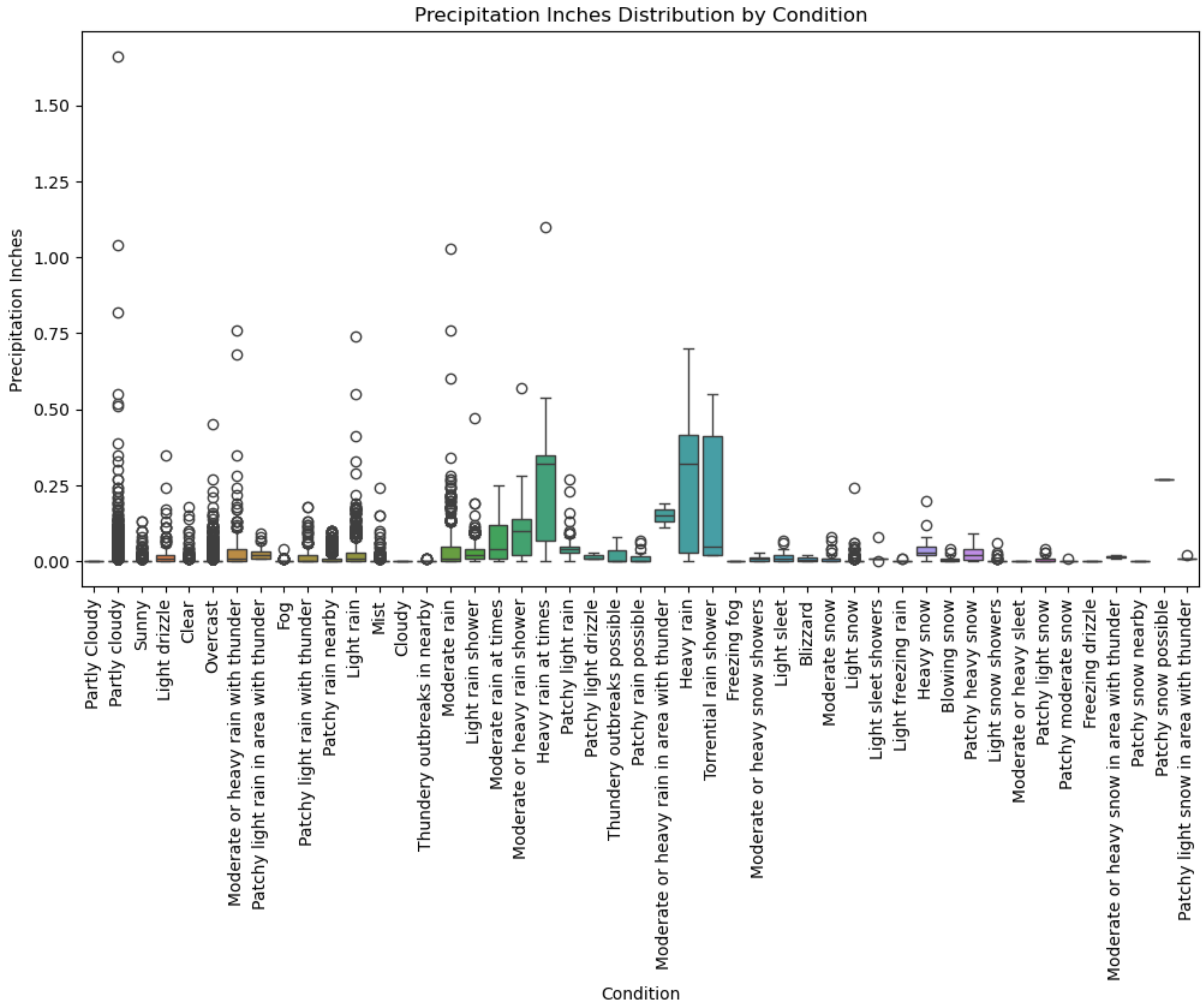
```
plt.ylabel("Temperature Celsious")
plt.title("Temperature Celsious Distribution by Month")
plt.show()
```



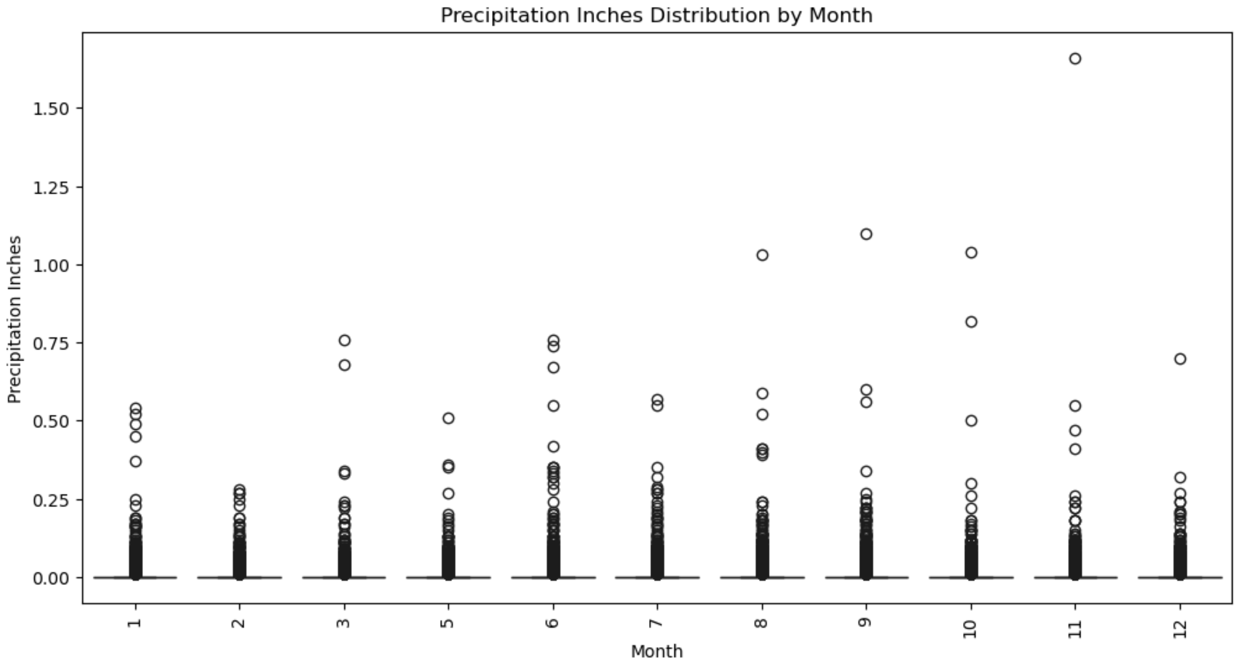

```
plt.figure(figsize=(12, 6))
sns.boxplot(
    x=df['hour'],
    y=df['temperature_celsius'],
    hue=df['hour'],
    palette="coolwarm",
    legend=False
)

plt.xticks(rotation=90)
plt.xlabel("Hour")
plt.ylabel("Temperature Celsious")
plt.title("Temperature Celsious Distribution by Hour")
plt.show()
```



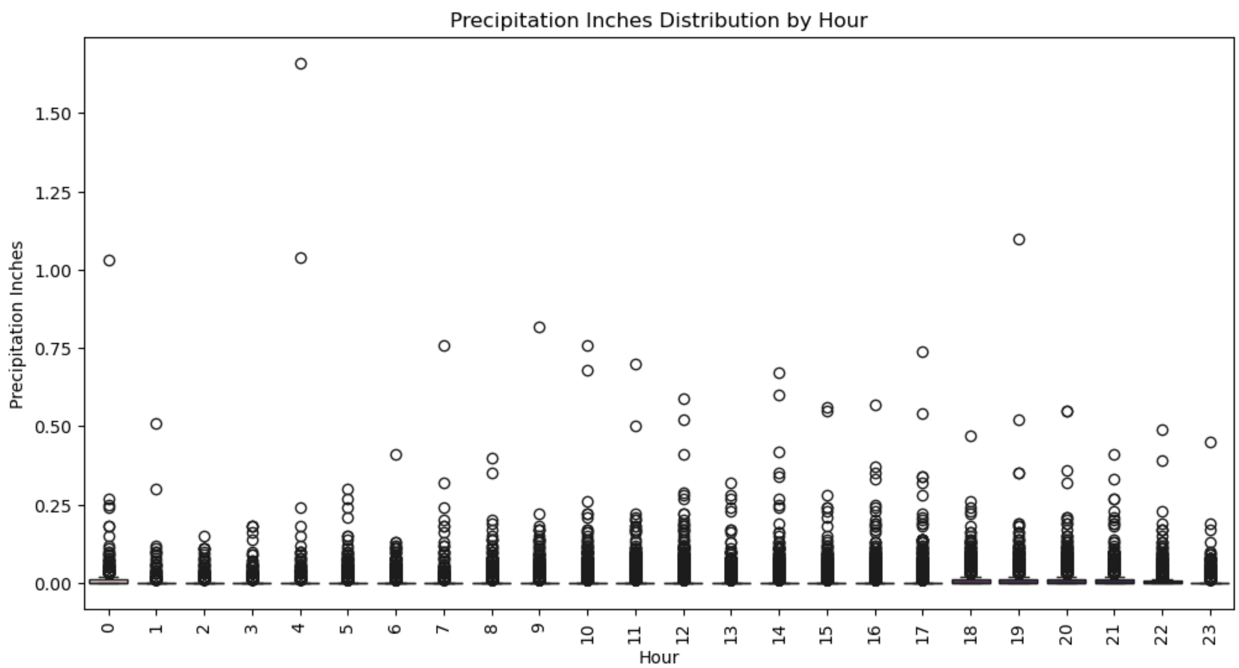

```
#Plotting boxplots for temperature and condition

plt.figure(figsize=(12, 6))

sns.boxplot(
    x=df['condition_text'],
    y=df['precip_in'],
    hue=df['condition_text'],
    legend=False
)

plt.xticks(rotation=90)
plt.xlabel("Condition")
plt.ylabel("Precipitation Inches")
plt.title("Precipitation Inches Distribution by Condition")
plt.show()
```




```
#Plotting boxplots for temperature and condition

plt.figure(figsize=(12, 6))

sns.boxplot(
    x=df['month'],
    y=df['precip_in'],
    hue=df['month'],
    legend=False
)

plt.xticks(rotation=90)
plt.xlabel("Month")
plt.ylabel("Precipitation Inches")
plt.title("Precipitation Inches Distribution by Month")
plt.show()
```




```
#Plotting boxplots for temperature and condition

plt.figure(figsize=(12, 6))

sns.boxplot(
    x=df['hour'],
    y=df['precip_in'],
    hue=df['hour'],
    legend=False
)

plt.xticks(rotation=90)
plt.xlabel("Hour")
plt.ylabel("Precipitation Inches")
plt.title("Precipitation Inches Distribution by Hour")
plt.show()
```



For the modeling part, I only did it thinking of one particular location because the way the dataset is formatted in one day can have multiple measures for distinct locations, and we know that the location, of course, affects the weather. In this case I chose Tegucigalpa, Honduras.

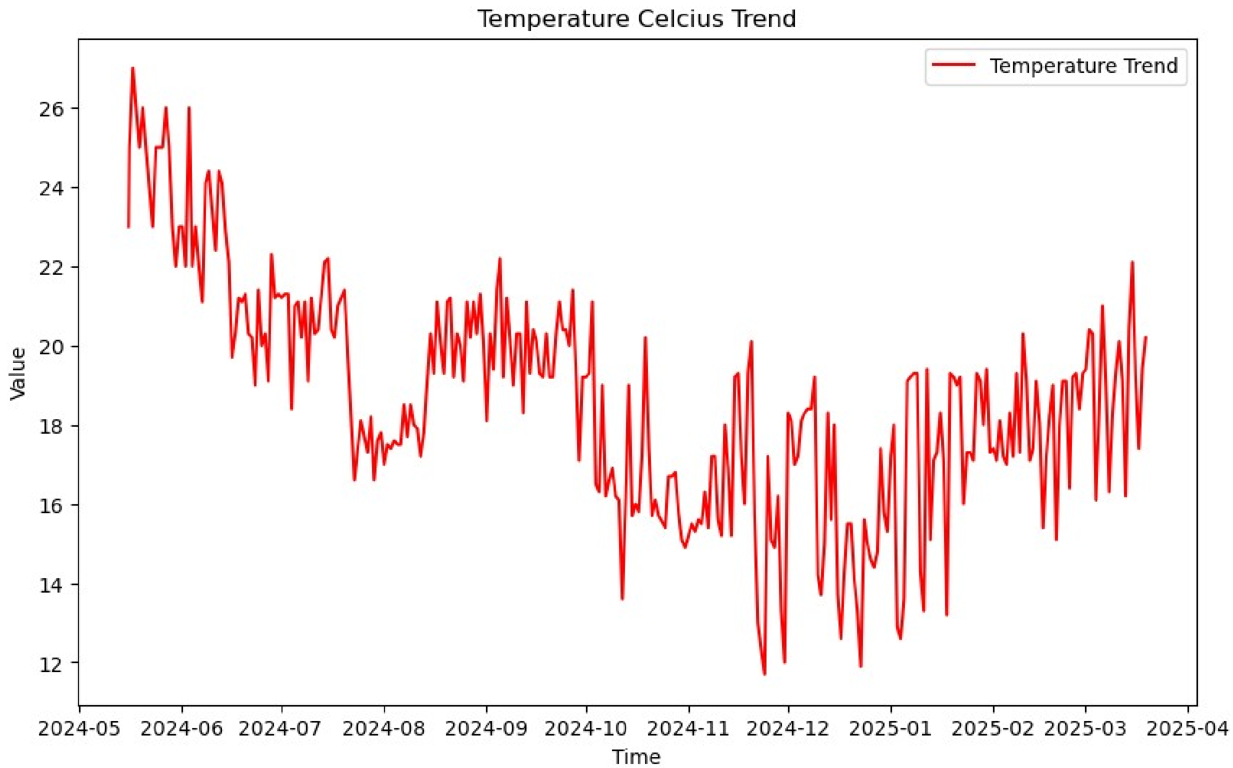
```
#Setting a location for prediction analysis

location='Tegucigalpa'
df_scaled_location= df_scaled[df_scaled['location_name']==location]
df_scaled_location= df_scaled_location.sort_values(by='last_updated')
df_scaled_location
```

```
        country location_name            timezone        last_updated
\
74     Honduras   Tegucigalpa  America/Tegucigalpa 2024-05-16 02:45:00

269    Honduras   Tegucigalpa  America/Tegucigalpa 2024-05-16 08:00:00

464    Honduras   Tegucigalpa  America/Tegucigalpa 2024-05-17 10:00:00

656    Honduras   Tegucigalpa  America/Tegucigalpa 2024-05-18 08:30:00

850    Honduras   Tegucigalpa  America/Tegucigalpa 2024-05-19 08:15:00

...         ...           ...                  ...                 ...

58732  Honduras   Tegucigalpa  America/Tegucigalpa 2025-03-15 03:45:00

58927  Honduras   Tegucigalpa  America/Tegucigalpa 2025-03-16 03:45:00
```

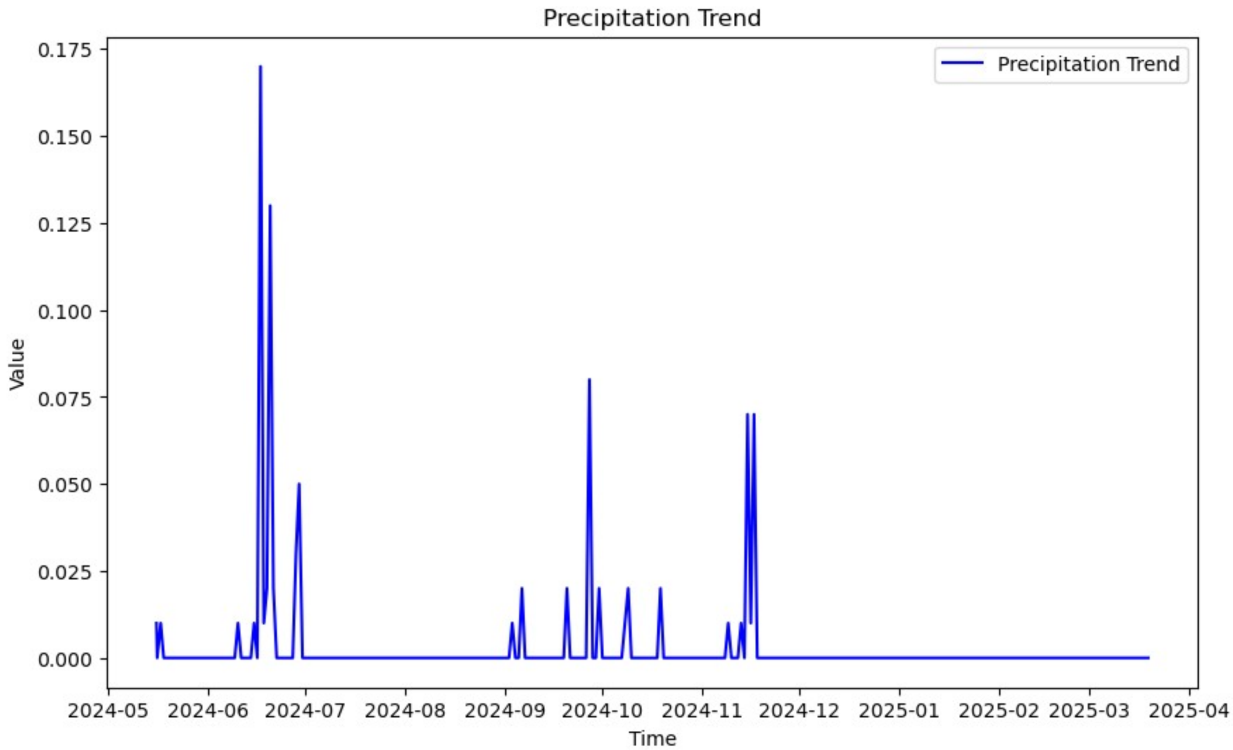
```
59122  Honduras   Tegucigalpa  America/Tegucigalpa 2025-03-17 03:45:00

59317  Honduras   Tegucigalpa  America/Tegucigalpa 2025-03-18 03:45:00

59512  Honduras   Tegucigalpa  America/Tegucigalpa 2025-03-19 03:30:00


      condition_text wind_direction  sunrise  sunset  moonrise
moonset  ...  \
74     Partly cloudy            WSW        5      18      12.0
0.0  ...
269    Partly cloudy            NNE        5      18      12.0
0.0  ...
464    Partly cloudy            NNW        5      18      13.0
1.0  ...
656    Partly cloudy              N        5      18      14.0
2.0  ...
850         Overcast              N        5      18      15.0
2.0  ...
...              ...            ...      ...     ...       ...      ..
.  ...
58732  Partly cloudy            SSW        5      17      19.0
6.0  ...
58927          Clear            SSW        5      17      20.0
7.0  ...
59122  Partly cloudy              N        5      18      20.0
7.0  ...
59317  Partly cloudy              N        5      18      21.0
8.0  ...
59512  Partly cloudy            NNE        5      18      22.0
9.0  ...

      air_quality_Sulphur_dioxide air_quality_PM2.5  air_quality_PM10
\
74                        0.950579          0.011658          0.237583

269                       0.950541          0.011658          0.238103

464                       0.950617          0.001372          0.234844

656                       0.950788          0.004223          0.235884

850                       0.950569          0.010481          0.237469

...                            ...               ...               ...

58732                     0.951993          0.013641          0.237284

58927                     0.952239          0.015131          0.237636
```

```
59122                        0.951360           0.018914           0.238387

59317                        0.950674           0.006648           0.235923

59512                        0.950709           0.006075           0.235806


       air_quality_us-epa-index  air_quality_gb-defra-index  \
74                          0.2                    0.111111
269                         0.2                    0.111111
464                         0.0                    0.000000
656                         0.0                    0.000000
850                         0.2                    0.111111
...                         ...                         ...
58732                       0.2                    0.111111
58927                       0.2                    0.222222
59122                       0.2                    0.222222
59317                       0.0                    0.000000
59512                       0.0                    0.000000

       moon_illumination  Hour_sin  Hour_cos  Month_sin  Month_cos
74                  0.55  0.750000  0.933013       0.75   0.066987
269                 0.55  0.933013  0.250000       0.75   0.066987
464                 0.64  0.750000  0.066987       0.75   0.066987
656                 0.73  0.933013  0.250000       0.75   0.066987
850                 0.81  0.933013  0.250000       0.75   0.066987
...                  ...       ...       ...        ...        ...
58732               0.99  0.853553  0.853553       1.00   0.500000
58927               0.96  0.853553  0.853553       1.00   0.500000
59122               0.92  0.853553  0.853553       1.00   0.500000
59317               0.86  0.853553  0.853553       1.00   0.500000
59512               0.79  0.853553  0.853553       1.00   0.500000

[303 rows x 45 columns]
```

```
#Plotting Temperature trend for Location

plt.figure(figsize=(10,6))
plt.plot(df_scaled_location['last_updated'],scaler.inverse_transform(d
f_scaled_location[columns])[:,columns.index("temperature_celsius")],
label="Temperature Trend", color="red")
plt.xlabel("Time")
plt.ylabel("Value")
plt.title("Temperature Celcius Trend")
plt.legend()
plt.show()
```




```
#Plotting Precipitation trend for Location

plt.figure(figsize=(10,6))
plt.plot(df_scaled_location['last_updated'],scaler.inverse_transform(d
f_scaled_location[columns])[:,columns.index("precip_in")],
label="Precipitation Trend", color="blue")
plt.xlabel("Time")
plt.ylabel("Value")
plt.title("Precipitation Trend")
plt.legend()
plt.show()
```



The first model that I will use is an LSTM neural network because I am used to working with it and looking at the plots for precipitation and temperature, a complex model is needed

```
# Method to create sequences for LSTM Model

def create_sequences(data, time_steps=5):
    X, y = [], []
    for i in range(len(data) - time_steps):
        X.append(data[i:i+time_steps])
        y.append(data[i+time_steps])
    return np.array(X), np.array(y)

#Creating sequences for LSTM
X, y = create_sequences(df_scaled_location[columns].to_numpy())

#Spliting data in training and validation
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

#creating model structure
model = Sequential()
model.add(LSTM(units=248, return_sequences=False,
input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dense(units=128))
model.add(Dense(units=64))
model.add(Dense(units=len(columns)))
```

```python
model.compile(optimizer='adam', loss='mean_squared_error')

#Repeating the dataset to improve training
train_ds = tf.data.Dataset.from_tensor_slices(
    (X_train, y_train)).shuffle(len(X_train)).repeat().batch(32)
test_ds = tf.data.Dataset.from_tensor_slices((X_test,
y_test)).batch(32)

# Train the model
history = model.fit(train_ds, epochs=20, batch_size=32,
validation_data=(X_test, y_test), steps_per_epoch=500)
```

```
Epoch 1/20
500/500 [==============================] - 10s 15ms/step - loss:
0.0099 - val_loss: 0.0083
Epoch 2/20
500/500 [==============================] - 6s 12ms/step - loss: 0.0066
- val_loss: 0.0096
Epoch 3/20
500/500 [==============================] - 7s 14ms/step - loss: 0.0054
- val_loss: 0.0126
Epoch 4/20
500/500 [==============================] - 6s 13ms/step - loss: 0.0036
- val_loss: 0.0161
Epoch 5/20
500/500 [==============================] - 4s 7ms/step - loss: 0.0020
- val_loss: 0.0253
Epoch 6/20
500/500 [==============================] - 7s 14ms/step - loss: 0.0011
- val_loss: 0.0323
Epoch 7/20
500/500 [==============================] - 6s 13ms/step - loss:
5.8780e-04 - val_loss: 0.0368
Epoch 8/20
500/500 [==============================] - 5s 10ms/step - loss:
3.8987e-04 - val_loss: 0.0308
Epoch 9/20
500/500 [==============================] - 3s 7ms/step - loss:
2.9084e-04 - val_loss: 0.0327
Epoch 10/20
500/500 [==============================] - 3s 7ms/step - loss:
2.5168e-04 - val_loss: 0.0301
Epoch 11/20
500/500 [==============================] - 3s 7ms/step - loss:
1.7950e-04 - val_loss: 0.0312
Epoch 12/20
500/500 [==============================] - 3s 7ms/step - loss:
1.7881e-04 - val_loss: 0.0279
Epoch 13/20
500/500 [==============================] - 3s 7ms/step - loss:
```

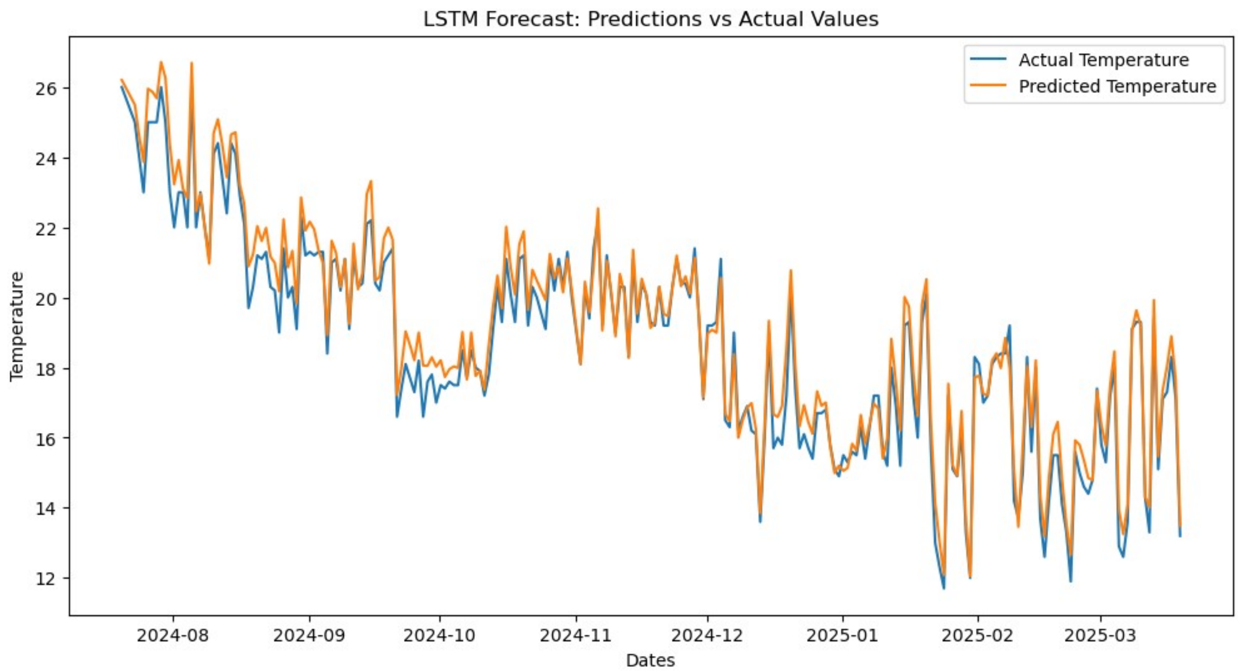
```
1.5592e-04 - val_loss: 0.0268
Epoch 14/20
500/500 [==============================] - 3s 7ms/step - loss:
1.7166e-04 - val_loss: 0.0269
Epoch 15/20
500/500 [==============================] - 4s 7ms/step - loss:
1.0265e-04 - val_loss: 0.0246
Epoch 16/20
500/500 [==============================] - 3s 7ms/step - loss:
1.2973e-04 - val_loss: 0.0252
Epoch 17/20
500/500 [==============================] - 3s 7ms/step - loss:
8.7254e-05 - val_loss: 0.0236
Epoch 18/20
500/500 [==============================] - 3s 7ms/step - loss:
1.0811e-04 - val_loss: 0.0238
Epoch 19/20
500/500 [==============================] - 3s 7ms/step - loss:
9.0295e-05 - val_loss: 0.0222
Epoch 20/20
500/500 [==============================] - 4s 7ms/step - loss:
8.2237e-05 - val_loss: 0.0201
```

Training goes smoothly with good loss measures

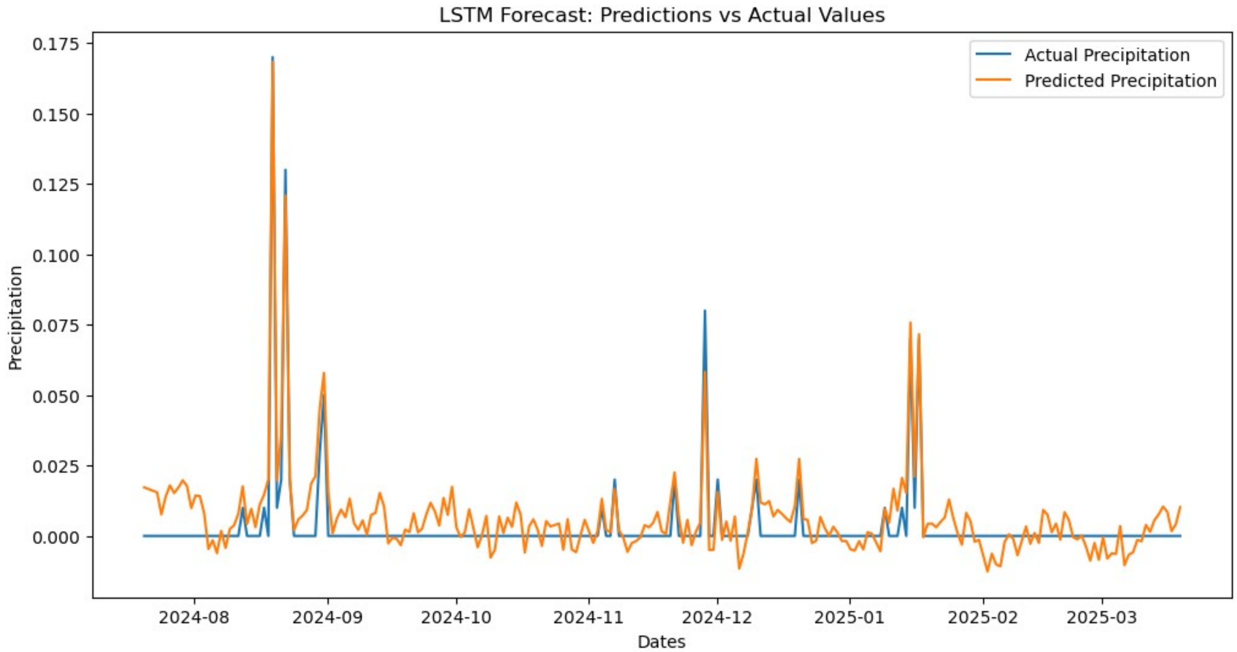
```
#Predicting Fit Values
train_predictions = model.predict(X_train)

# Reverse scaling to get back original values
train_predictions_rescaled =
scaler.inverse_transform(train_predictions)
y_train_rescaled = scaler.inverse_transform(y_train)

# Calculate metrics
mse_train = mean_squared_error(y_train_rescaled,
train_predictions_rescaled)
rmse_train = np.sqrt(mse_train)
mae_train = mean_absolute_error(y_train_rescaled,
train_predictions_rescaled)

# Print results
print(f"Train Mean Squared Error (MSE): {mse_train:.4f}")
print(f"Train Root Mean Squared Error (RMSE): {rmse_train:.4f}")
print(f"Train Mean Absolute Error (MAE): {mae_train:.4f}")

8/8 [==============================] - 0s 3ms/step
Train Mean Squared Error (MSE): 1362.2510
Train Root Mean Squared Error (RMSE): 36.9087
Train Mean Absolute Error (MAE): 9.2302
```

In this case we used 3 error measures mse rmse and mae

```
#Plot Fit vs Actual for temperature
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-len(y_train_rescaled):],
y_train_rescaled[:,columns.index("temperature_celsius")],
label="Actual Temperature")
plt.plot(df_scaled_location['last_updated'][-
len(train_predictions_rescaled):], train_predictions_rescaled[:,
columns.index("temperature_celsius")], label="Predicted Temperature")
plt.xlabel("Dates")
plt.ylabel("Temperature")
plt.title("LSTM Forecast: Predictions vs Actual Values")
plt.legend()
plt.show()
```




```
#Plot Fit vs Actual for temperature
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-len(y_train_rescaled):],
y_train_rescaled[:,columns.index("precip_in")], label="Actual
Precipitation")
plt.plot(df_scaled_location['last_updated'][-
len(train_predictions_rescaled):], train_predictions_rescaled[:,
columns.index("precip_in")], label="Predicted Precipitation")
plt.xlabel("Dates")
plt.ylabel("Precipitation")
plt.title("LSTM Forecast: Predictions vs Actual Values")
plt.legend()
plt.show()
```

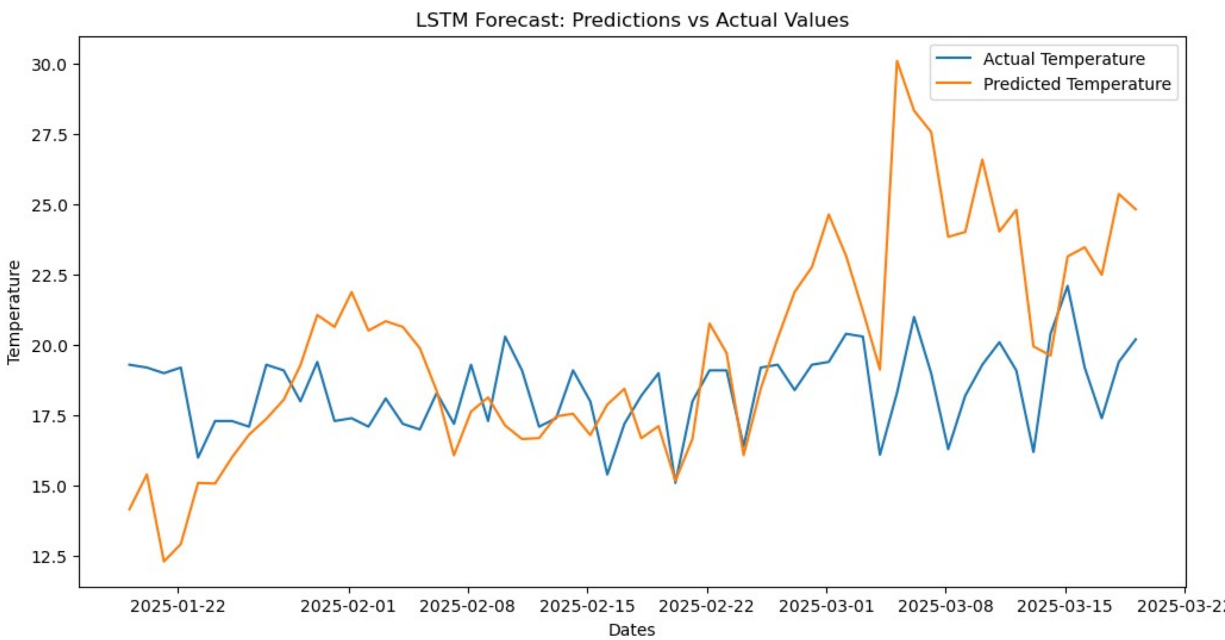


For what we can see in the plot the model does a good job fitting, would requiere more training for it to fit perfectly but that could not be good due to overfitting.

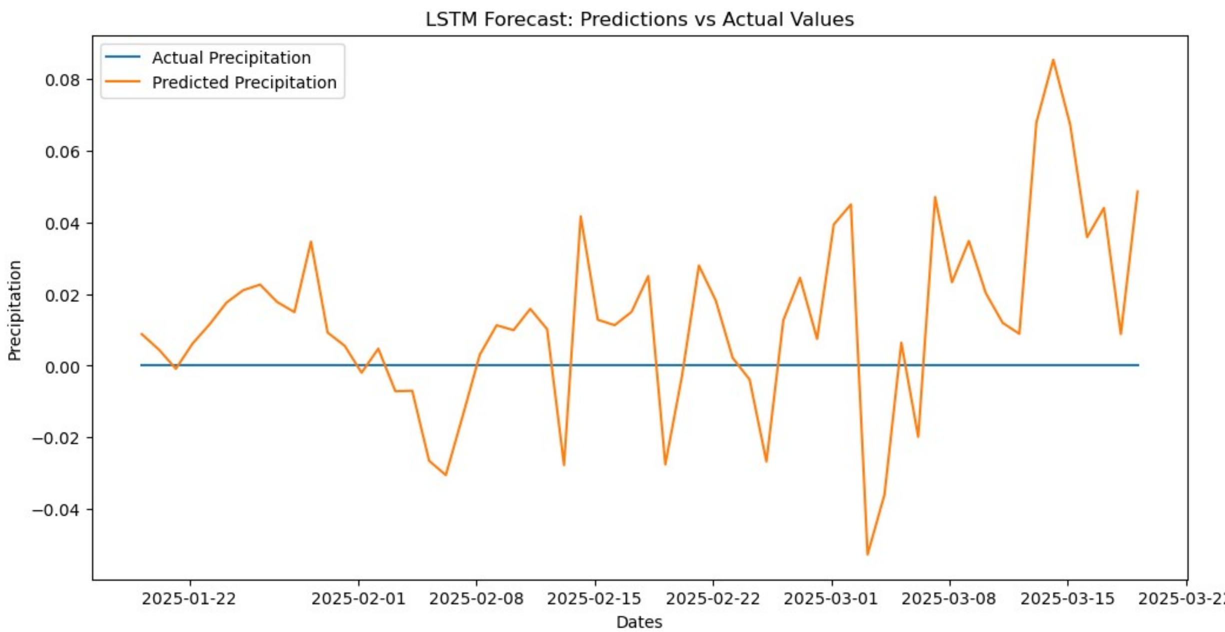
```
#Predicting Test Values

test_predictions = model.predict(X_test)

# Reverse scaling to get back original values
test_predictions_rescaled = scaler.inverse_transform(test_predictions)
y_test_rescaled = scaler.inverse_transform(y_test)

# Calculate metrics
mse_test = mean_squared_error(y_test_rescaled,
test_predictions_rescaled)
rmse_test = np.sqrt(mse_train)
mae_test = mean_absolute_error(y_test_rescaled,
test_predictions_rescaled)
r2_test = r2_score(y_test_rescaled, test_predictions_rescaled)

# Print results
print(f"Test Mean Squared Error (MSE): {mse_test:.4f}")
print(f"Test Root Mean Squared Error (RMSE): {rmse_test:.4f}")
print(f"Test Mean Absolute Error (MAE): {mae_test:.4f}")

2/2 [==============================] - 0s 11ms/step
Test Mean Squared Error (MSE): 8681.2693
Test Root Mean Squared Error (RMSE): 36.9087
Test Mean Absolute Error (MAE): 24.5857
```

Considerable Higher measures for the validation set, not as much to call it overfitting.

```
#Plot Predicted vs Actual for temperature

plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-len(y_test_rescaled):],
y_test_rescaled[:, columns.index("temperature_celsius")],
label="Actual Temperature")
plt.plot(df_scaled_location['last_updated'][-
len(test_predictions_rescaled):], test_predictions_rescaled[:,
columns.index("temperature_celsius")], label="Predicted Temperature")
plt.xlabel("Dates")
plt.ylabel("Temperature")
plt.title("LSTM Forecast: Predictions vs Actual Values")
plt.legend()
plt.show()
```




```
#Plot Predicted vs Actual for temperature

plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-len(y_test_rescaled):],
y_test_rescaled[:, columns.index("precip_in")], label="Actual
Precipitation")
plt.plot(df_scaled_location['last_updated'][-
len(test_predictions_rescaled):], test_predictions_rescaled[:,
columns.index("precip_in")], label="Predicted Precipitation")
plt.xlabel("Dates")
plt.ylabel("Precipitation")
plt.title("LSTM Forecast: Predictions vs Actual Values")
plt.legend()
plt.show()
```



We can see that the model tries to simulate the complex behaviour of temperature but struggles to do so. other reason could be that Im trying to predict multiple measures at the same time, maybe just focusing on temperature or precipitation alone would deliver better results.

The next model to implement would be Random Forest

Same process of split, train transform to the original scale, calculating error measures and plotting

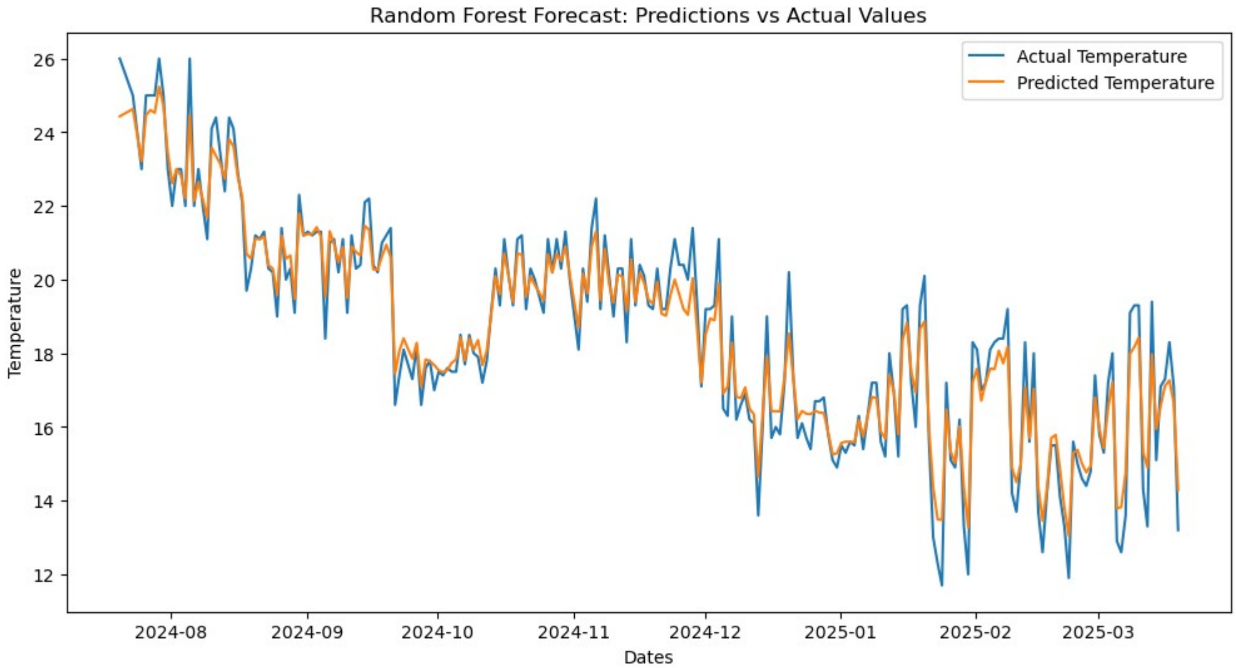
```
#Preparing data for Random Forest Model
X_flat = X.reshape((X.shape[0], X.shape[1] * X.shape[2]))

#Training and test Split
X_train, X_test, y_train, y_test = train_test_split(X_flat, y,
test_size=0.2, shuffle=False)

#Model Fit
rf_model = RandomForestRegressor(n_estimators=100)
rf_model.fit(X_train, y_train)

#Prediction Fit
rf_train_preds = rf_model.predict(X_train)

# Reverse scaling to get back original values
rf_train_preds_rescaled = scaler.inverse_transform(rf_train_preds)
rfy_train_rescaled = scaler.inverse_transform(y_train)

#Prediction Test
rf_test_preds = rf_model.predict(X_test)

# Reverse scaling to get back original values
```

```
rf_test_preds_rescaled = scaler.inverse_transform(rf_test_preds)
rfy_test_rescaled = scaler.inverse_transform(y_test)

# Error Measures
mse_train_rf = mean_squared_error(rfy_train_rescaled,
rf_train_preds_rescaled)
rmse_train_rf = np.sqrt(mse_train_rf)
train_mae_rf = mean_absolute_error(rfy_train_rescaled,
rf_train_preds_rescaled)
mse_test_rf = mean_squared_error(rfy_test_rescaled,
rf_test_preds_rescaled)
rmse_test_rf = np.sqrt(mse_test_rf)
test_mae_rf = mean_absolute_error(rfy_test_rescaled,
rf_test_preds_rescaled)

print(f"Random Forest Train MSE: {mse_train_rf:.4f}")
print(f"Random Forest Train RMSE: {rmse_train_rf:.4f}")
print(f"Random Forest Train MAE: {train_mae_rf:.4f}")
print(f"Random Forest Test MSE: {mse_test_rf:.4f}")
print(f"Random Forest Test RMSE: {rmse_test_rf:.4f}")
print(f"Random Forest Test MAE: {test_mae_rf:.4f}")
```

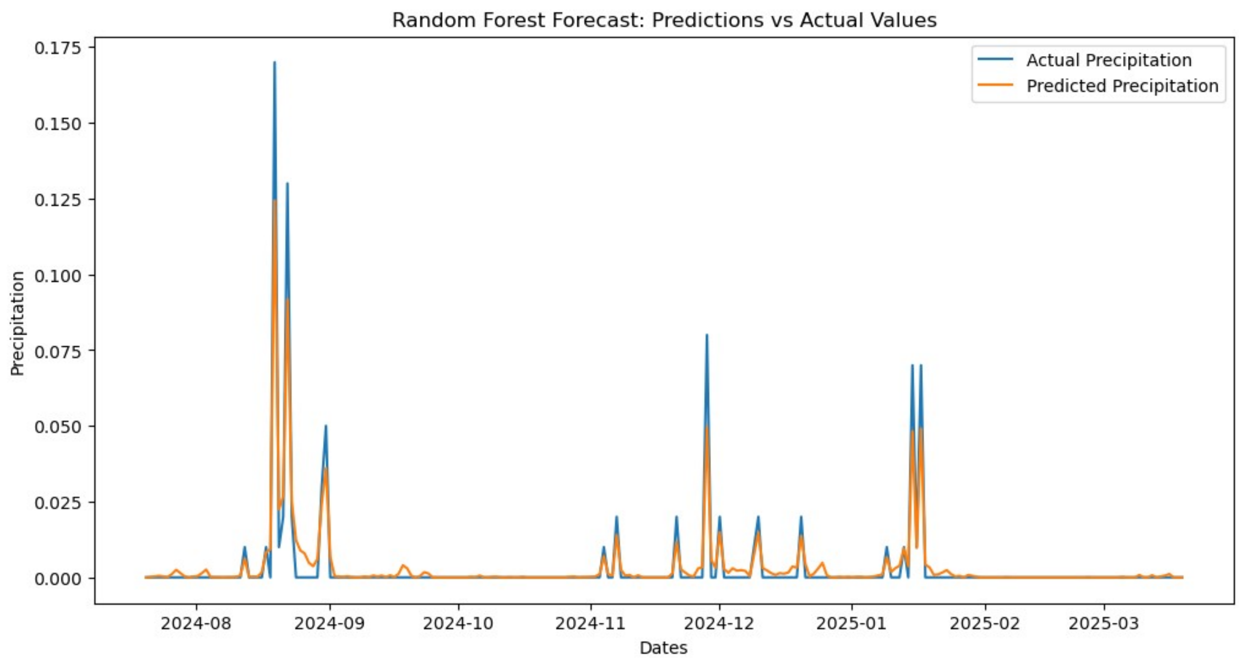
```
Random Forest Train MSE: 178.4549
Random Forest Train RMSE: 13.3587
Random Forest Train MAE: 3.1959
Random Forest Test MSE: 2548.6511
Random Forest Test RMSE: 50.4842
Random Forest Test MAE: 7.8328
```

To my surprise th random forest model has better error measures

```
# Plotting Prediction of fit values vs actual values
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-
len(rfy_train_rescaled):], rfy_train_rescaled[:,
columns.index("temperature_celsius")], label="Actual Temperature")
plt.plot(df_scaled_location['last_updated'][-
len(rf_train_preds_rescaled):], rf_train_preds_rescaled[:,
columns.index("temperature_celsius")], label="Predicted Temperature")
plt.xlabel("Dates")
plt.ylabel("Temperature")
plt.title("Random Forest Forecast: Predictions vs Actual Values")
plt.legend()
plt.show()
```

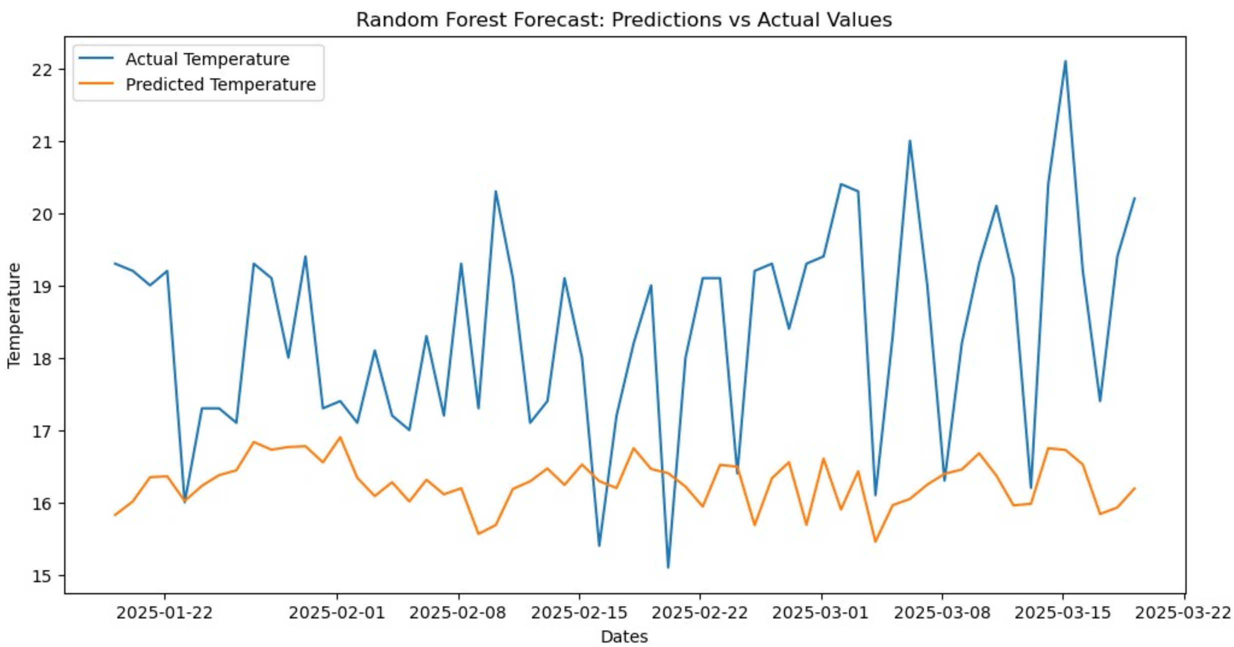



```
# Plotting Prediction of fit values vs actual values
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-
len(rfy_train_rescaled):], rfy_train_rescaled[:,
columns.index("precip_in")], label="Actual Precipitation")
plt.plot(df_scaled_location['last_updated'][-
len(rf_train_preds_rescaled):], rf_train_preds_rescaled[:,
columns.index("precip_in")], label="Predicted Precipitation")
plt.xlabel("Dates")
plt.ylabel("Precipitation")
plt.title("Random Forest Forecast: Predictions vs Actual Values")
plt.legend()
plt.show()
```

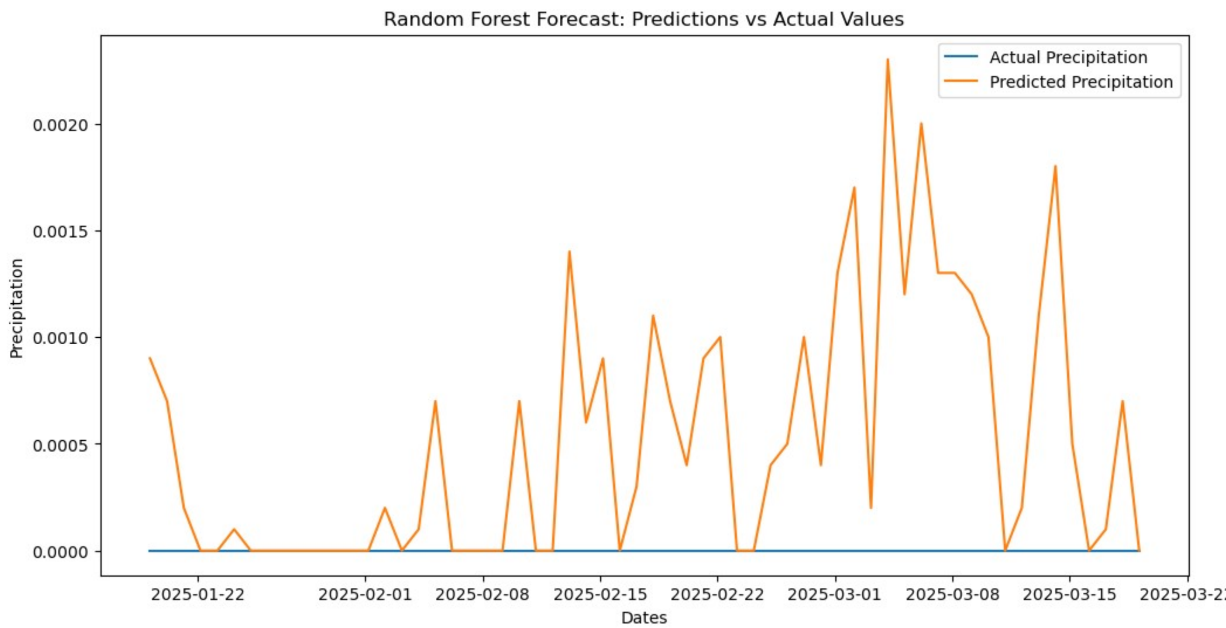


It does a really good job fitting the model

```
# Plotting Prediction of test values vs actual values
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-len(rfy_test_rescaled):],
rfy_test_rescaled[:, columns.index("temperature_celsius")],
label="Actual Temperature")
plt.plot(df_scaled_location['last_updated'][-
len(rf_test_preds_rescaled):], rf_test_preds_rescaled[:,
columns.index("temperature_celsius")], label="Predicted Temperature")
plt.xlabel("Dates")
plt.ylabel("Temperature")
plt.title("Random Forest Forecast: Predictions vs Actual Values")
plt.legend()
plt.show()
```




```
# Plotting Prediction of test values vs actual values
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-len(rfy_test_rescaled):],
rfy_test_rescaled[:, columns.index("precip_in")], label="Actual
Precipitation")
plt.plot(df_scaled_location['last_updated'][-
len(rf_test_preds_rescaled):], rf_test_preds_rescaled[:,
columns.index("precip_in")], label="Predicted Precipitation")
plt.xlabel("Dates")
plt.ylabel("Precipitation")
plt.title("Random Forest Forecast: Predictions vs Actual Values")
plt.legend()
plt.show()
```



For this model it is closer the prediction to the actual value, but it doesn't try to simulate the complex trend of the measures.

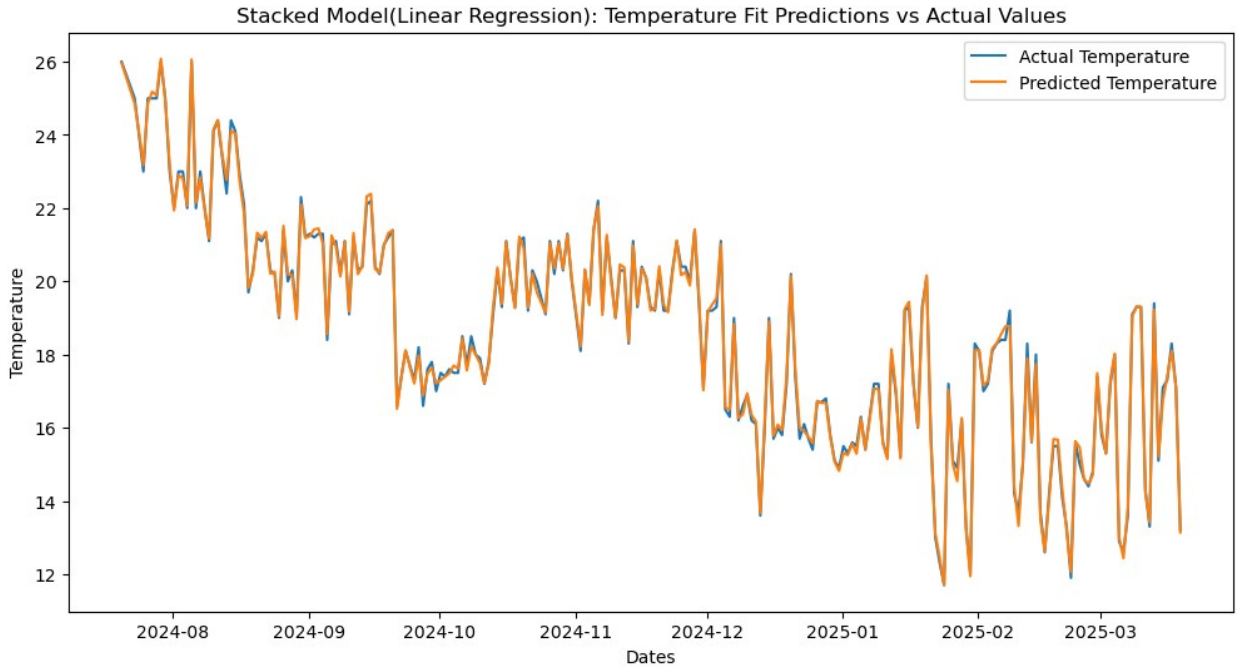
No I will combine the 2 models using a stack model in this case a linear regression to see if I can improve the performance.

```
# Training an Ensemble of models stacking them

X_train_stack = np.column_stack((train_predictions, rf_train_preds))
X_test_stack = np.column_stack((test_predictions, rf_test_preds))

#Using Linear Regresion as Meta Model to combine predictions
meta_model = LinearRegression()
meta_model.fit(X_train_stack, y_train)

# Predict using the stacked model
stacked_train_preds = meta_model.predict(X_train_stack)
stacked_test_preds = meta_model.predict(X_test_stack)

# Reverse scaling to get back original values
stacked_train_preds_rescaled =
scaler.inverse_transform(stacked_train_preds)
stacked_test_preds_rescaled =
scaler.inverse_transform(stacked_test_preds)

# Evaluate the performance of the stacked model
train_mse = mean_squared_error(rfy_train_rescaled,
stacked_train_preds_rescaled)
train_rmse = np.sqrt(mse_train_rf)
train_mae = mean_absolute_error(rfy_train_rescaled,
stacked_train_preds_rescaled)
```

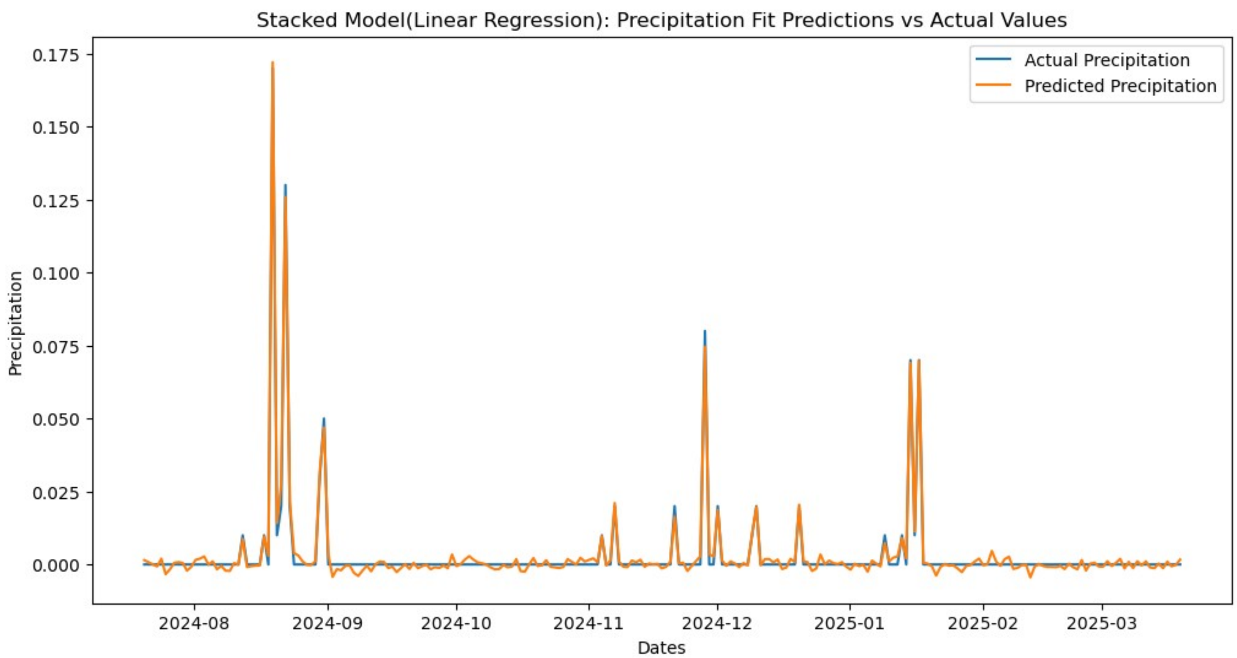
```
test_mse = mean_squared_error(rfy_test_rescaled,
stacked_test_preds_rescaled)
test_rmse = np.sqrt(test_mse)
test_mae = mean_absolute_error(rfy_test_rescaled,
stacked_test_preds_rescaled)

print(f"Stacked Model Train MSE: {train_mse:.4f}")
print(f"Stacked Model Train RMSE: {train_rmse:.4f}")
print(f"Stacked Model Train MAE: {train_mae:.4f}")
print(f"Stacked Model Test MSE: {test_mse:.4f}")
print(f"Stacked Model Test RMSE: {test_rmse:.4f}")
print(f"Stacked Model Test MAE: {test_mae:.4f}")

Stacked Model Train MSE: 11.9690
Stacked Model Train RMSE: 13.3587
Stacked Model Train MAE: 0.8627
Stacked Model Test MSE: 5177.3622
Stacked Model Test RMSE: 71.9539
Stacked Model Test MAE: 16.2126
```

Error measures in a good area mainly on the training set on the validation still the random forest seem to perform better.

```
# Plotting Prediction of fit values vs actual values
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-
len(rfy_train_rescaled):], rfy_train_rescaled[:,
columns.index("temperature_celsius")], label="Actual Temperature")
plt.plot(df_scaled_location['last_updated'][-
len(stacked_train_preds_rescaled):], stacked_train_preds_rescaled[:,
columns.index("temperature_celsius")], label="Predicted Temperature")
plt.xlabel("Dates")
plt.ylabel("Temperature")
plt.title("Stacked Model(Linear Regression): Temperature Fit
Predictions vs Actual Values")
plt.legend()
plt.show()
```

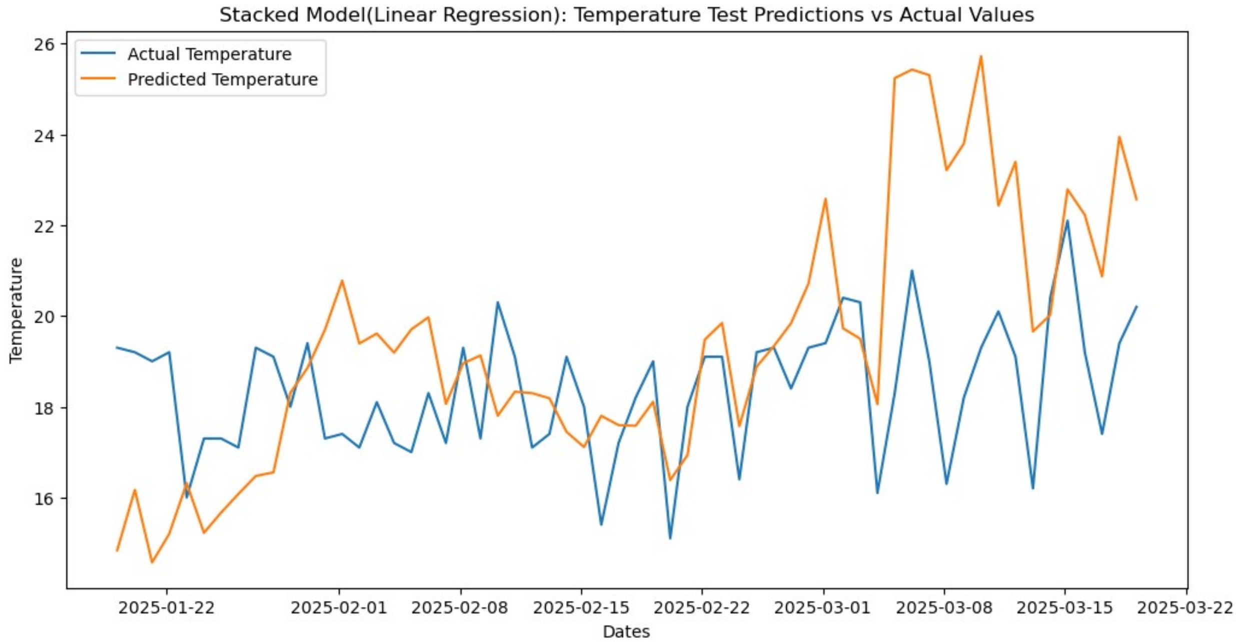



```
# Plotting Prediction of fit values vs actual values
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-
len(rfy_train_rescaled):], rfy_train_rescaled[:,
columns.index("precip_in")], label="Actual Precipitation")
plt.plot(df_scaled_location['last_updated'][-
len(stacked_train_preds_rescaled):], stacked_train_preds_rescaled[:,
columns.index("precip_in")], label="Predicted Precipitation")
plt.xlabel("Dates")
plt.ylabel("Precipitation")
plt.title("Stacked Model(Linear Regression): Precipitation Fit
Predictions vs Actual Values")
plt.legend()
plt.show()
```

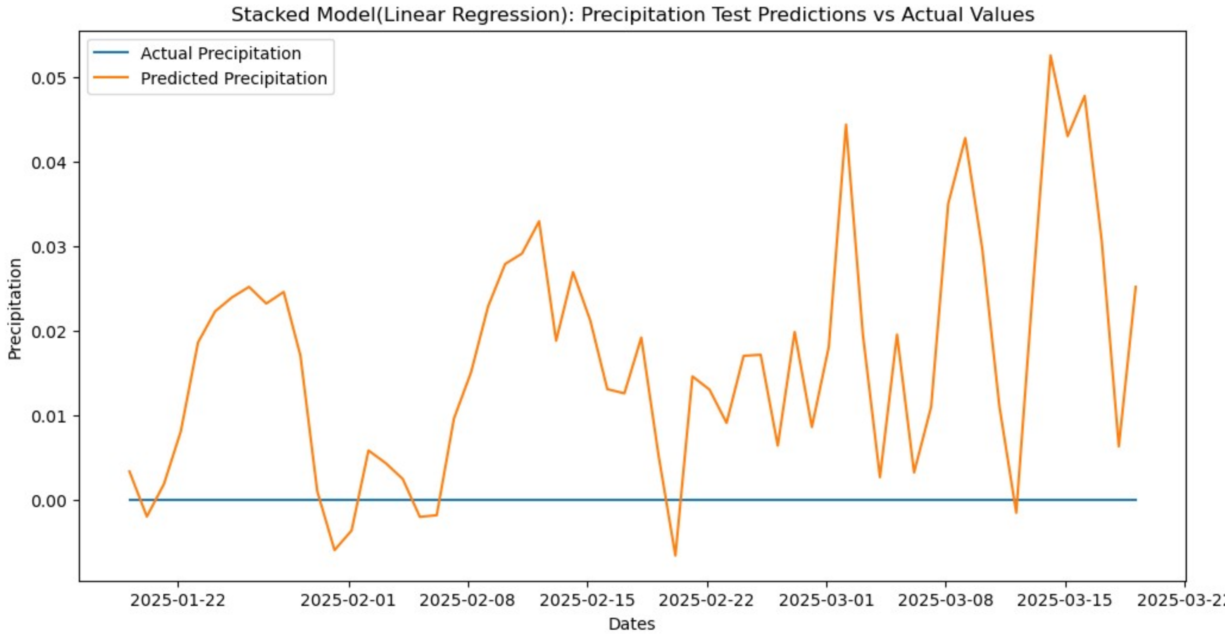


We can see that the model almost perfectly fits the training data

```
# Plotting Prediction of test values vs actual values
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-len(rfy_test_rescaled):],
rfy_test_rescaled[:, columns.index("temperature_celsius")],
label="Actual Temperature")
plt.plot(df_scaled_location['last_updated'][-
len(stacked_test_preds_rescaled):], stacked_test_preds_rescaled[:, 2],
label="Predicted Temperature")
plt.xlabel("Dates")
plt.ylabel("Temperature")
plt.title("Stacked Model(Linear Regression): Temperature Test
Predictions vs Actual Values")
plt.legend()
plt.show()
```




```
# Plotting Prediction of test values vs actual values
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-len(rfy_test_rescaled):],
rfy_test_rescaled[:, columns.index("precip_in")], label="Actual
Precipitation")
plt.plot(df_scaled_location['last_updated'][-
len(stacked_test_preds_rescaled):], stacked_test_preds_rescaled[:,
columns.index("precip_in")], label="Predicted Precipitation")
plt.xlabel("Dates")
plt.ylabel("Precipitation")
plt.title("Stacked Model(Linear Regression): Precipitation Test
Predictions vs Actual Values")
plt.legend()
plt.show()
```



The model in comparison with the LSTM alone it got closer to the prediction, the random forest still perfoms better but this one tries to simulate thos high and low points also.

Now instead of using a linear regression as the stacked model, I used the Random Forest again.

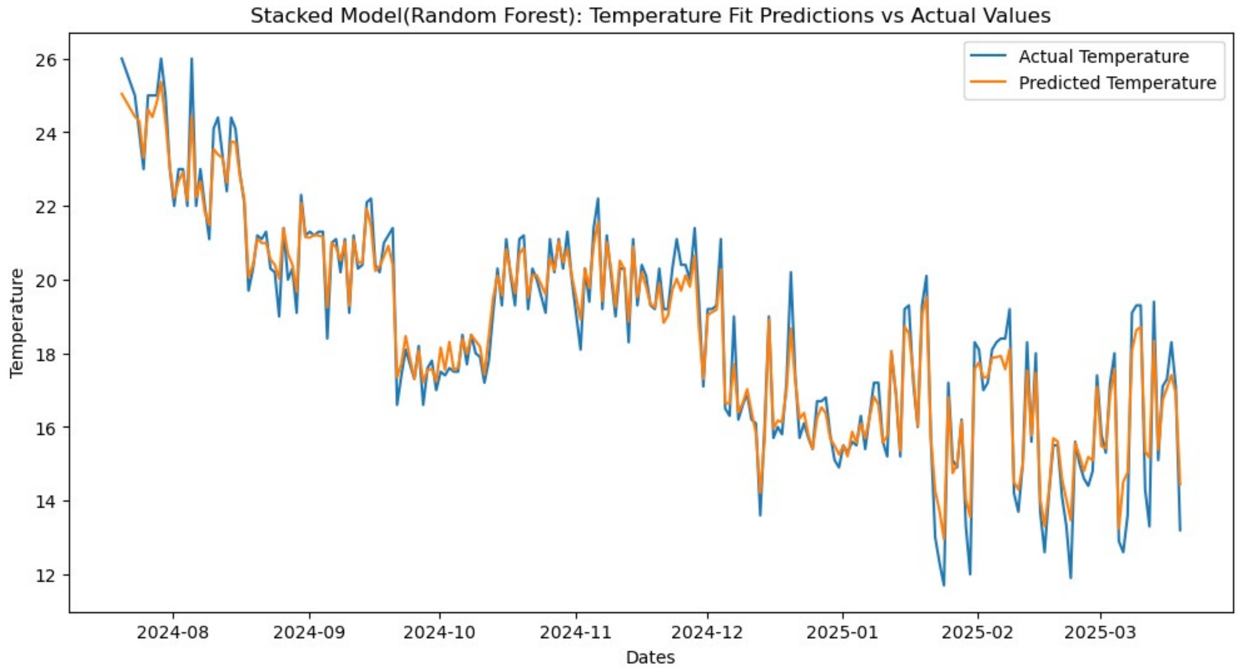
```
# Training an Ensemble of models stacking them

X_train_stack = np.column_stack((train_predictions, rf_train_preds))
X_test_stack = np.column_stack((test_predictions, rf_test_preds))

#Using RandomForest as Meta Model to combine predictions
meta_model = RandomForestRegressor(n_estimators=50)
meta_model.fit(X_train_stack, y_train)

# Predict using the stacked model
stacked_train_preds = meta_model.predict(X_train_stack)
stacked_test_preds = meta_model.predict(X_test_stack)

# Reverse scaling to get back original values
stacked_train_preds_rescaled =
scaler.inverse_transform(stacked_train_preds)
stacked_test_preds_rescaled =
scaler.inverse_transform(stacked_test_preds)

# Evaluate the performance of the stacked model
train_mse = mean_squared_error(rfy_train_rescaled,
stacked_train_preds_rescaled)
train_rmse = np.sqrt(mse_train_rf)
train_mae = mean_absolute_error(rfy_train_rescaled,
```

```
stacked_train_preds_rescaled)
test_mse = mean_squared_error(rfy_test_rescaled,
stacked_test_preds_rescaled)
test_rmse = np.sqrt(test_mse)
test_mae = mean_absolute_error(rfy_test_rescaled,
stacked_test_preds_rescaled)

print(f"Stacked Model Train MSE: {train_mse:.4f}")
print(f"Stacked Model Train RMSE: {train_rmse:.4f}")
print(f"Stacked Model Train MAE: {train_mae:.4f}")
print(f"Stacked Model Test MSE: {test_mse:.4f}")
print(f"Stacked Model Test RMSE: {test_rmse:.4f}")
print(f"Stacked Model Test MAE: {test_mae:.4f}")
```

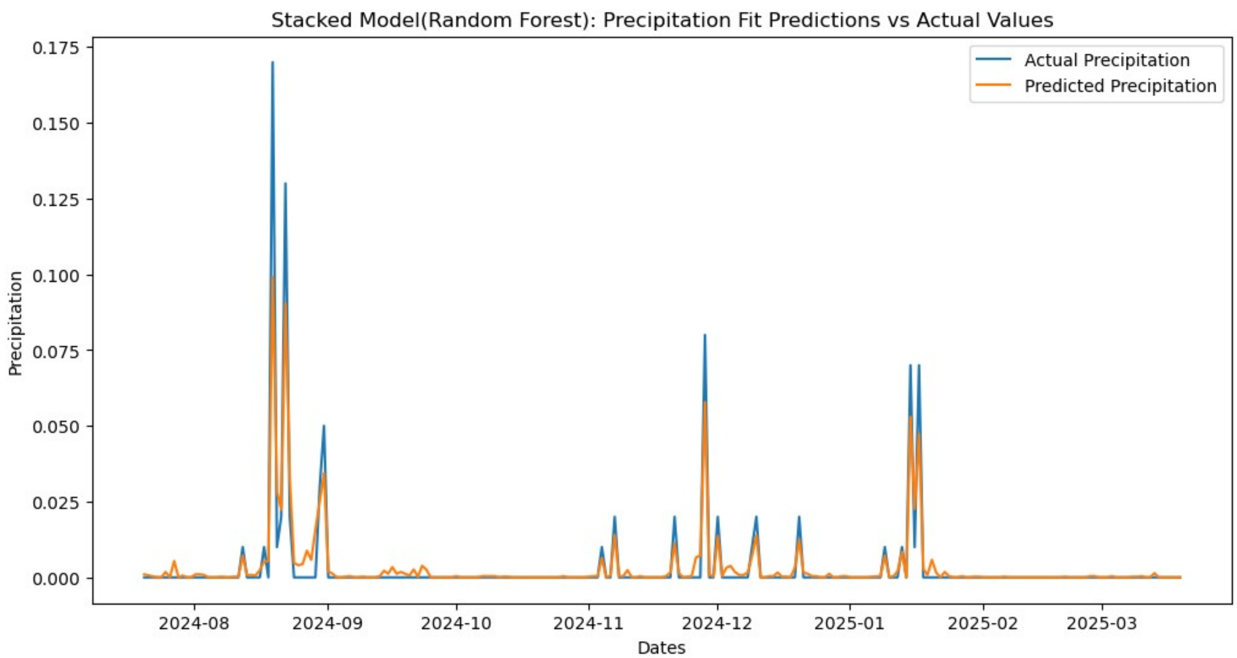
```
Stacked Model Train MSE: 94.3645
Stacked Model Train RMSE: 13.3587
Stacked Model Train MAE: 1.9406
Stacked Model Test MSE: 2696.4991
Stacked Model Test RMSE: 51.9278
Stacked Model Test MAE: 7.7604
```

Good Error Measures slightly higher in the validation set than the first random forest

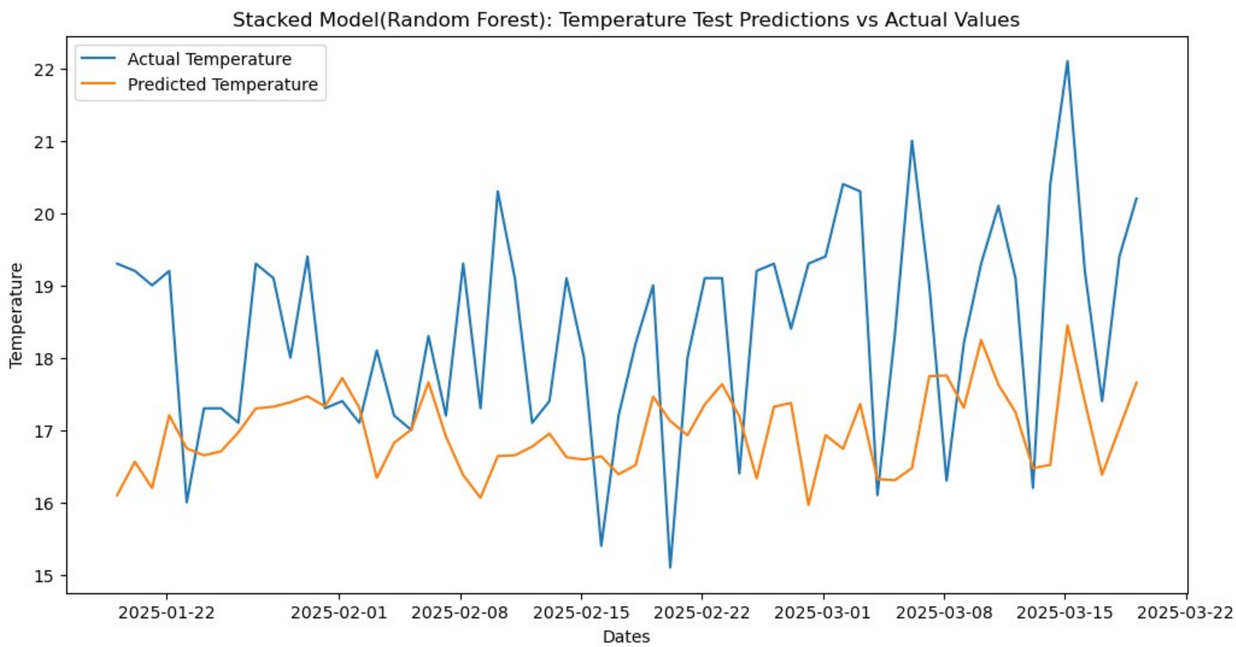
```
# Plotting Prediction of fit values vs actual values
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-
len(rfy_train_rescaled):], rfy_train_rescaled[:,
columns.index("temperature_celsius")], label="Actual Temperature")
plt.plot(df_scaled_location['last_updated'][-
len(stacked_train_preds_rescaled):], stacked_train_preds_rescaled[:,
columns.index("temperature_celsius")], label="Predicted Temperature")
plt.xlabel("Dates")
plt.ylabel("Temperature")
plt.title("Stacked Model(Random Forest): Temperature Fit Predictions
vs Actual Values")
plt.legend()
plt.show()
```



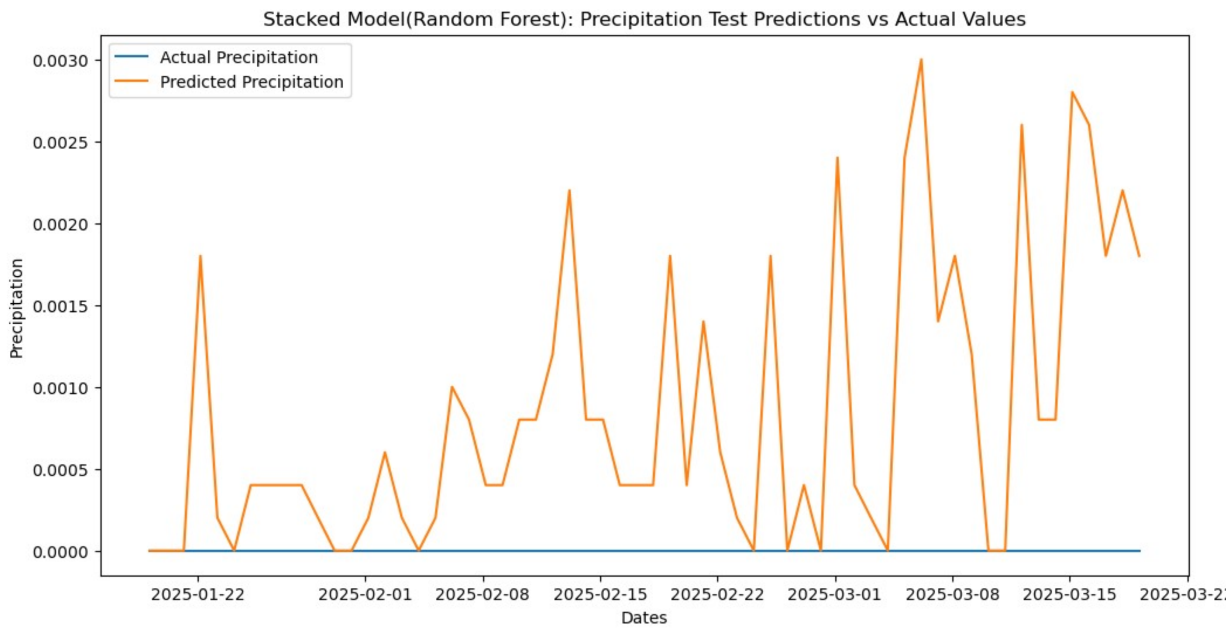

```python
# Plotting Prediction of fit values vs actual values
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-
len(rfy_train_rescaled):], rfy_train_rescaled[:,
columns.index("precip_in")], label="Actual Precipitation")
plt.plot(df_scaled_location['last_updated'][-
len(stacked_train_preds_rescaled):], stacked_train_preds_rescaled[:,
columns.index("precip_in")], label="Predicted Precipitation")
plt.xlabel("Dates")
plt.ylabel("Precipitation")
plt.title("Stacked Model(Random Forest): Precipitation Fit Predictions
vs Actual Values")
plt.legend()
plt.show()
```




```
# Plotting Prediction of test values vs actual values
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-len(rfy_test_rescaled):],
rfy_test_rescaled[:, columns.index("temperature_celsius")],
label="Actual Temperature")
plt.plot(df_scaled_location['last_updated'][-
len(stacked_test_preds_rescaled):], stacked_test_preds_rescaled[:,
columns.index("temperature_celsius")], label="Predicted Temperature")
plt.xlabel("Dates")
plt.ylabel("Temperature")
plt.title("Stacked Model(Random Forest): Temperature Test Predictions
vs Actual Values")
plt.legend()
plt.show()
```




```
# Plotting Prediction of test values vs actual values
plt.figure(figsize=(12,6))
plt.plot(df_scaled_location['last_updated'][-len(rfy_test_rescaled):],
rfy_test_rescaled[:, columns.index("precip_in")], label="Actual
Precipitation")
plt.plot(df_scaled_location['last_updated'][-
len(stacked_test_preds_rescaled):], stacked_test_preds_rescaled[:,
columns.index("precip_in")], label="Predicted Precipitation")
plt.xlabel("Dates")
plt.ylabel("Precipitation")
plt.title("Stacked Model(Random Forest): Precipitation Test
Predictions vs Actual Values")
plt.legend()
plt.show()
```



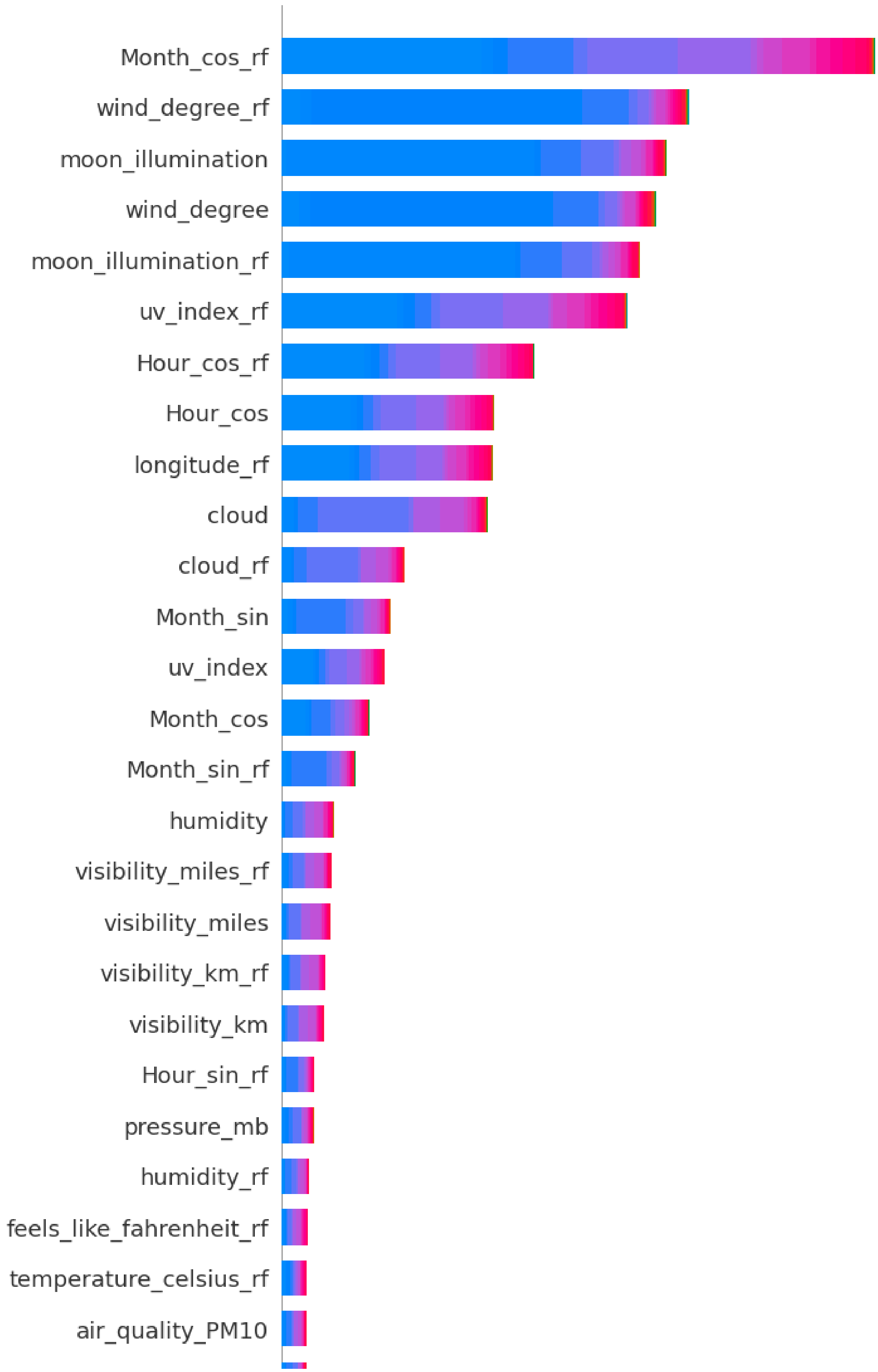
The model in my opinion is trying to catch some of the variation of the measures, i believe is the influence of the lstm in the model, next steps to improve the model would be to add more stacked models to the equation and try to tune hyperparameters for better performance.

Now I will show based on the final random forest model some variable importance for the model

```
# In this case cause the features were duplicated cause we have the predicted values for the lstm model and the random regression model
# Creating new_columns with both models features
new_columns = columns + [column + "_rf" for column in columns]

#Variable importance

feature_importance = meta_model.feature_importances_
feature_names = new_columns

# Create a DataFrame
importance_df = pd.DataFrame({"Feature": feature_names, "Importance": feature_importance})
importance_df = importance_df.sort_values(by="Importance", ascending=False)

# Display importance
importance_df.style

<pandas.io.formats.style.Styler at 0x220edc266a0>
```

We can see that moon ilumination, wind measures also the month are the most important variables, which in logical terms make sense.

```
#Variable importance using SHAP
explainer = shap.Explainer(meta_model, X_train_stack)
shap_values = explainer(X_test_stack,check_additivity=False)
shap.summary_plot(shap_values, X_test_stack, new_columns,
class_names=columns, max_display=X_test_stack.shape[1])
```


Month_cos_rf
wind_degree_rf
moon_illumination
wind_degree
moon_illumination_rf
uv_index_rf
Hour_cos_rf
Hour_cos
longitude_rf
cloud
cloud_rf
Month_sin
uv_index
Month_cos
Month_sin_rf
humidity
visibility_miles_rf
visibility_miles
visibility_km_rf
visibility_km
Hour_sin_rf
pressure_mb
humidity_rf
feels_like_fahrenheit_rf
temperature_celsius_rf
air_quality_PM10

Same as before moon ilumination, wind measures also the month are the most important variables, which in logical terms make sense.