TimeSeries_Weather_Forecast

April 28, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from datetime import datetime
```

```
/Users/rickytrujillo/anaconda3/lib/python3.11/site-
packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

[28]:
```
weather_22 = pd.read_csv("weather_22.csv", index_col=0)
weather_23 = pd.read_csv("weather_23.csv", index_col=0)
weather_24 = pd.read_csv("weather_24.csv", index_col=0)
```

[29]:
```
weather_22["Date"]= pd.to_datetime(weather_22[['Year', 'Month', 'Day']])
weather_23["Date"]= pd.to_datetime(weather_23[['Year', 'Month', 'Day']])
weather_24["Date"]= pd.to_datetime(weather_24[['Year', 'Month', 'Day']])
```

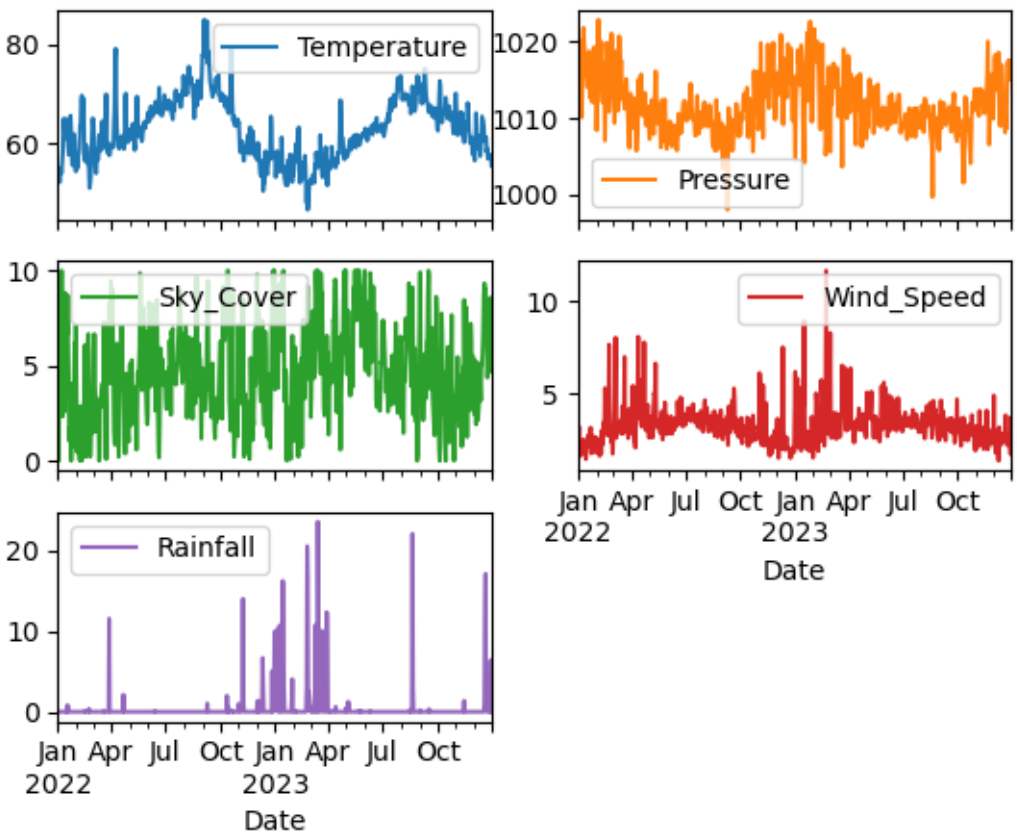
[30]:
```
weather22_merged = weather_22.groupby(["Month", "Day"]).mean().reset_index()
weather22_merged = weather22_merged.drop(["Year", "Month", "Day", "Hour"],␣
 ↪axis=1)
weather22_merged = weather22_merged.assign(Temperature = lambda x:␣
 ↪x["Temperature"]*(9/5)+32)
weather22_merged = weather22_merged.set_index("Date")

weather23_merged = weather_23.groupby(["Month", "Day"]).mean().reset_index()
weather23_merged = weather23_merged.drop(["Year", "Month", "Day", "Hour"],␣
 ↪axis=1)
weather23_merged = weather23_merged.assign(Temperature = lambda x:␣
 ↪x["Temperature"]*(9/5)+32)
weather23_merged = weather23_merged.set_index("Date")

weather24_merged = weather_24.groupby(["Month", "Day"]).mean().reset_index()
weather24_merged = weather24_merged.drop(["Year", "Month", "Day", "Hour"],␣
 ↪axis=1)
```

```
weather24_merged = weather24_merged.assign(Temperature = lambda x:␣
 ↪x["Temperature"]*(9/5)+32)
weather24_merged = weather24_merged.set_index("Date")
weather24_merged = weather24_merged.dropna()
```

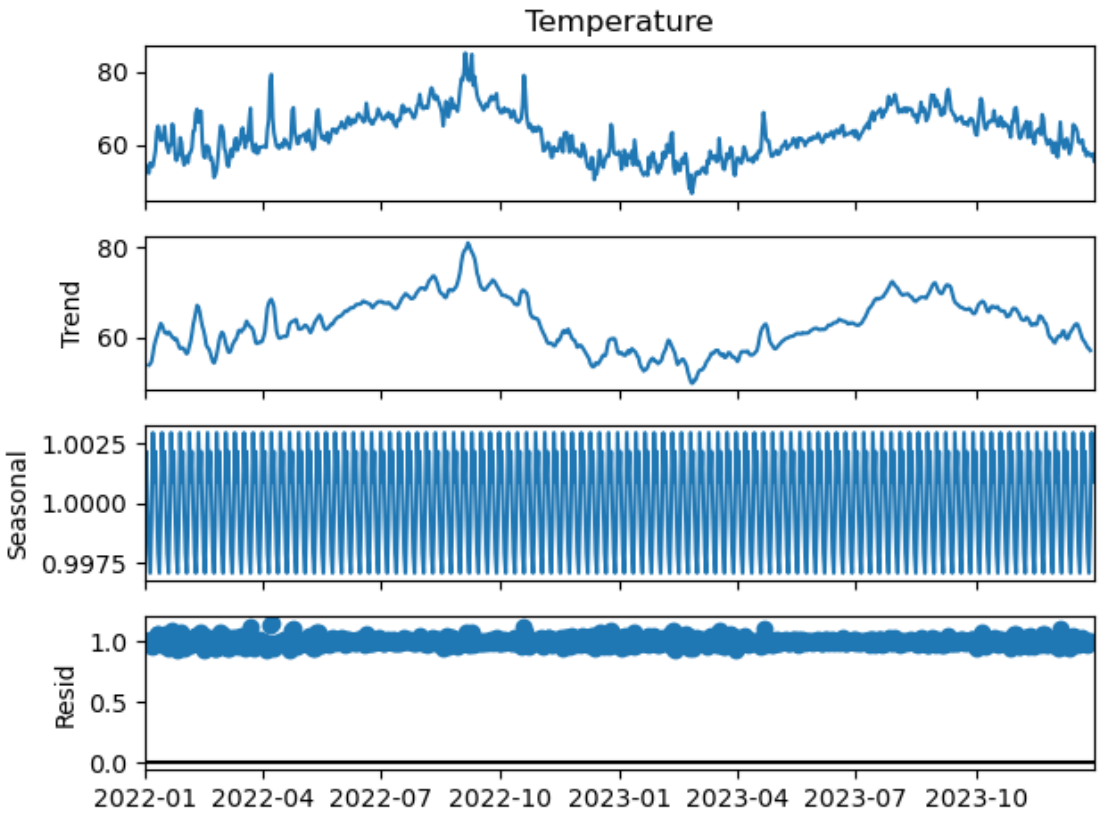
[31]:
```
train_data = pd.concat([weather22_merged, weather23_merged])
train_data.index.freq = "D"

test_data = weather24_merged
test_data.index.freq = "D"
```

[33]:
```
train_data.to_csv("weather_22_23.csv")
```

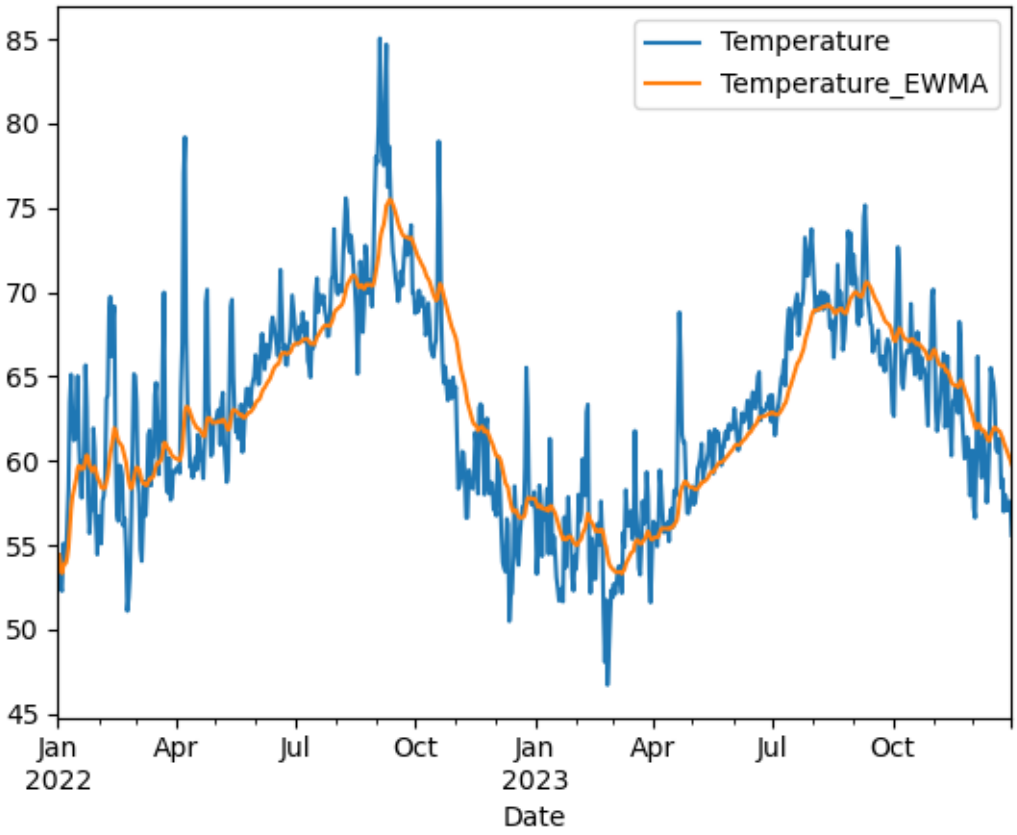
[6]:
```
# Stationary_Variables
# Sky Cover
# Rainfall

# Non-Stationary Variables
# Temperature
# Pressure
# Wind SPeed
train_data.plot(subplots=True, layout=(3,2))
plt.savefig('Stationary.png')
```

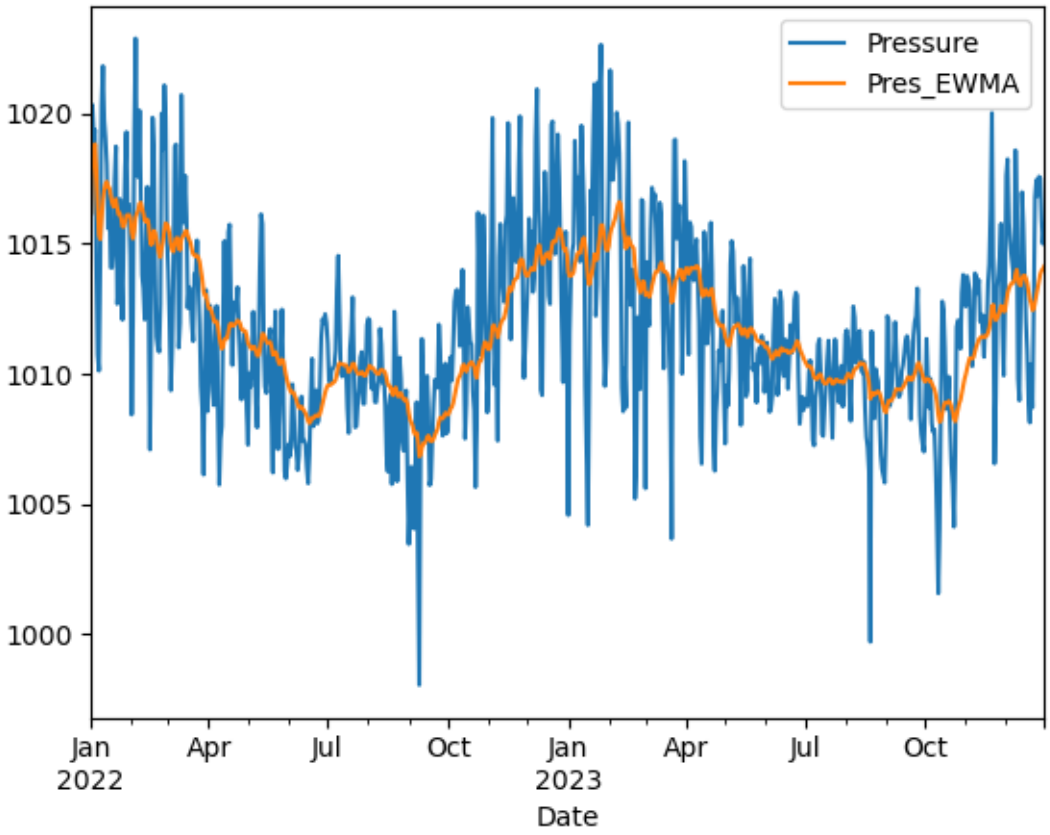


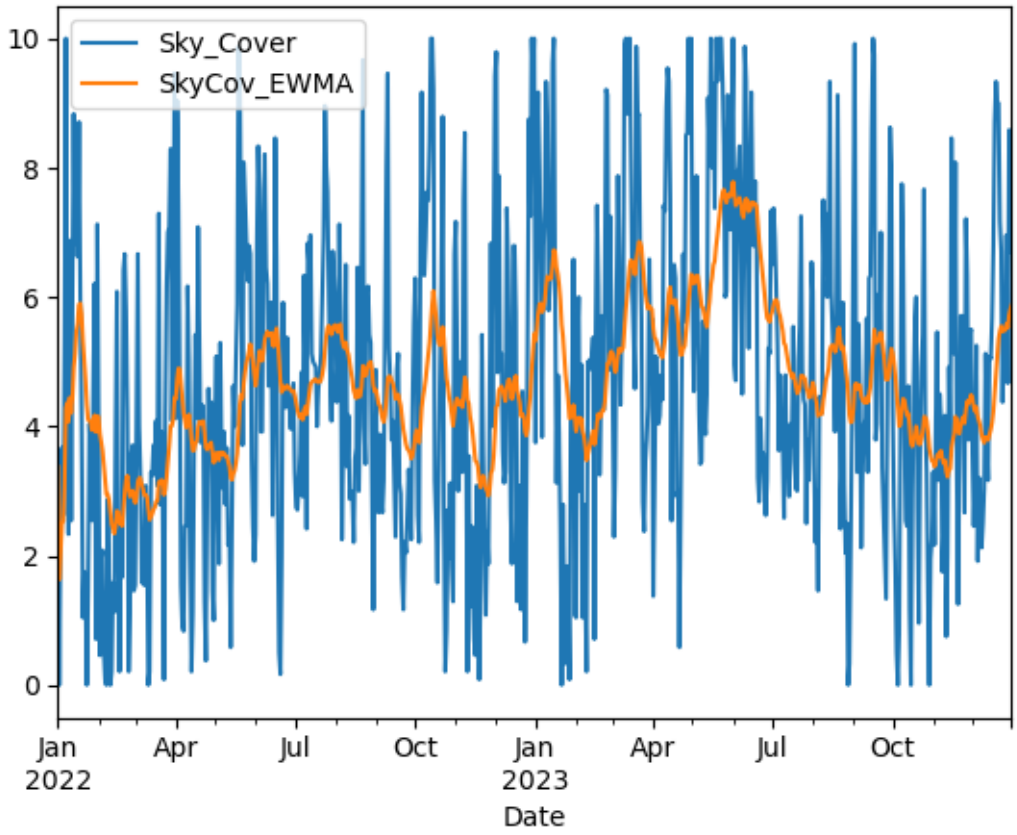
[7]:
```
# ETS Decomposition
result = seasonal_decompose(train_data["Temperature"], model="multiplicative")
result.plot();
plt.savefig('ETS_Temp.png')
```

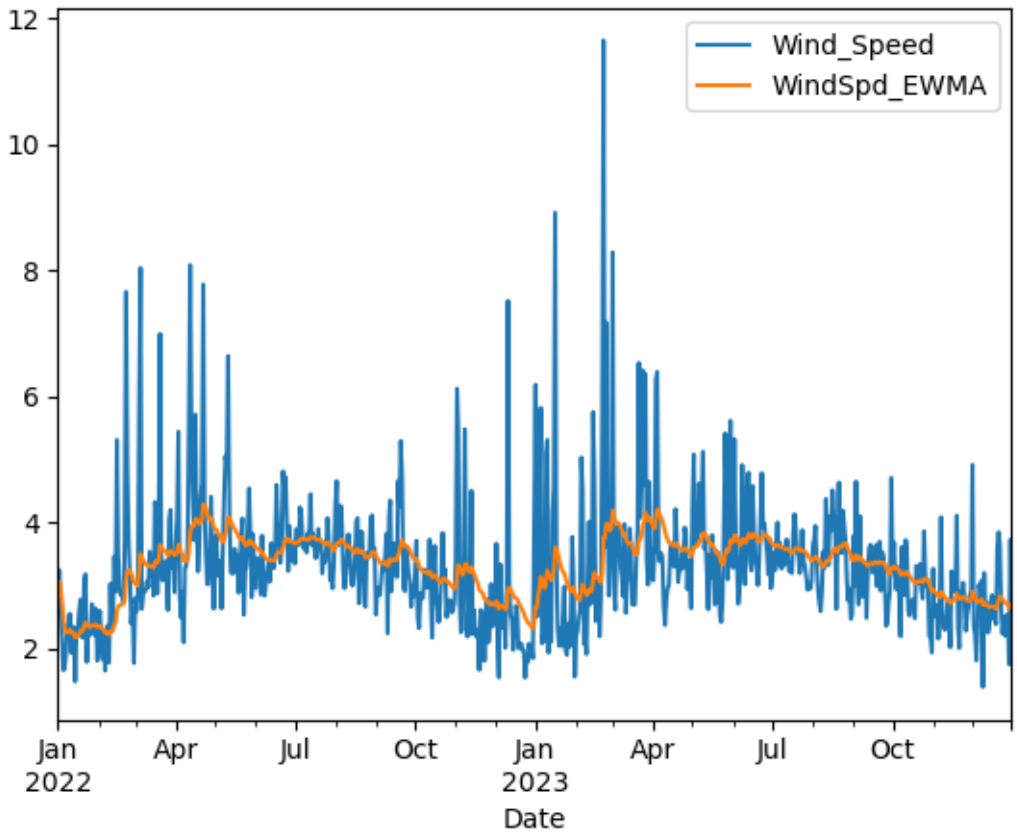


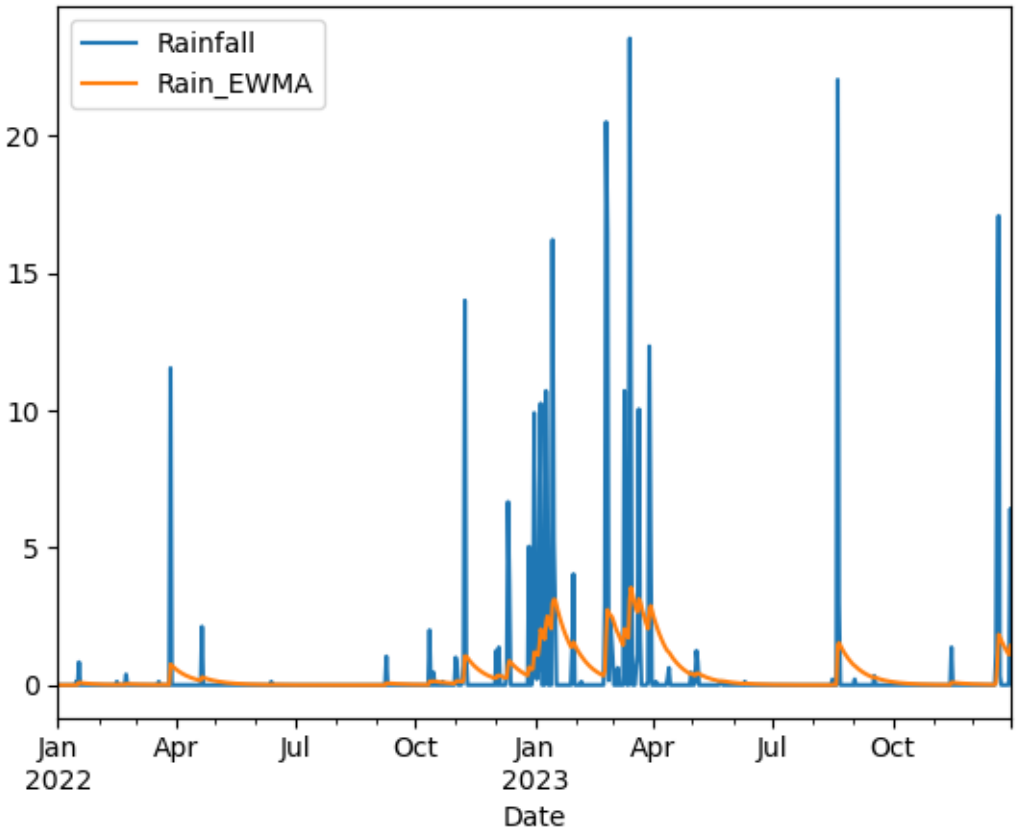
[8]:
```
# EWMA
# Window size of 30 days ~ 1 month
train_data["Temperature_EWMA"] = train_data["Temperature"].ewm(span=30).mean()
train_data[["Temperature", "Temperature_EWMA"]].plot();
plt.savefig('Temp_EWMA.png')

train_data["Pres_EWMA"] = train_data["Pressure"].ewm(span=30).mean()
train_data[["Pressure", "Pres_EWMA"]].plot();
plt.savefig('Pres_EWMA.png')

train_data["SkyCov_EWMA"] = train_data["Sky_Cover"].ewm(span=30).mean()
train_data[["Sky_Cover", "SkyCov_EWMA"]].plot();
plt.savefig('SkyCover_EWMA.png')

train_data["WindSpd_EWMA"] = train_data["Wind_Speed"].ewm(span=30).mean()
train_data[["Wind_Speed", "WindSpd_EWMA"]].plot();
plt.savefig('Wind_EWMA.png')

train_data["Rain_EWMA"] = train_data["Rainfall"].ewm(span=30).mean()
train_data[["Rainfall", "Rain_EWMA"]].plot();
plt.savefig('Rain_EWMA.png')
```


Temperature
Temperature_EWMA
85
80
75
70
65
60
55
50
45
Jan
2022
Apr
Jul
Oct
Jan
2023
Apr
Jul
Oct
Date


Pressure
Pres_EWMA
1020
1015
1010
1005
1000
Jan
2022
Apr
Jul
Oct
Jan
2023
Apr
Jul
Oct
Date

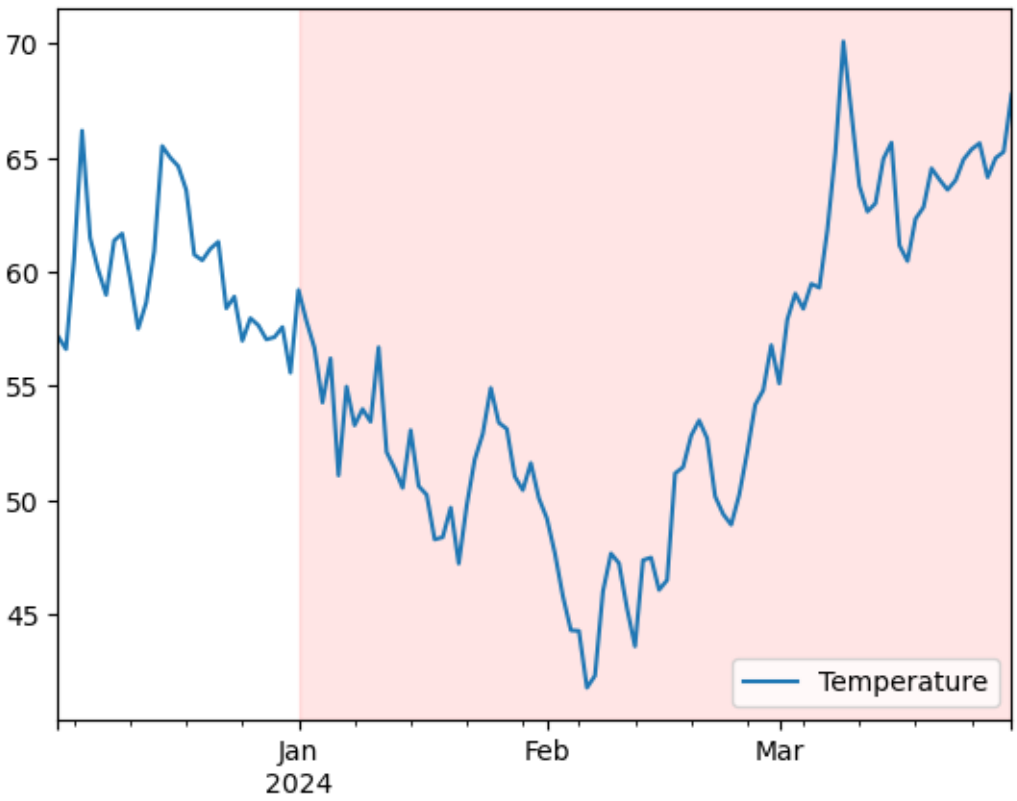

Sky_Cover
SkyCov_EWMA
10
8
6
4
2
0
Jan
2022
Apr
Jul
Oct
Jan
2023
Apr
Jul
Oct
Date


Wind_Speed
WindSpd_EWMA
12
10
8
6
4
2
Jan
2022
Apr
Jul
Oct
Jan
2023
Apr
Jul
Oct
Date



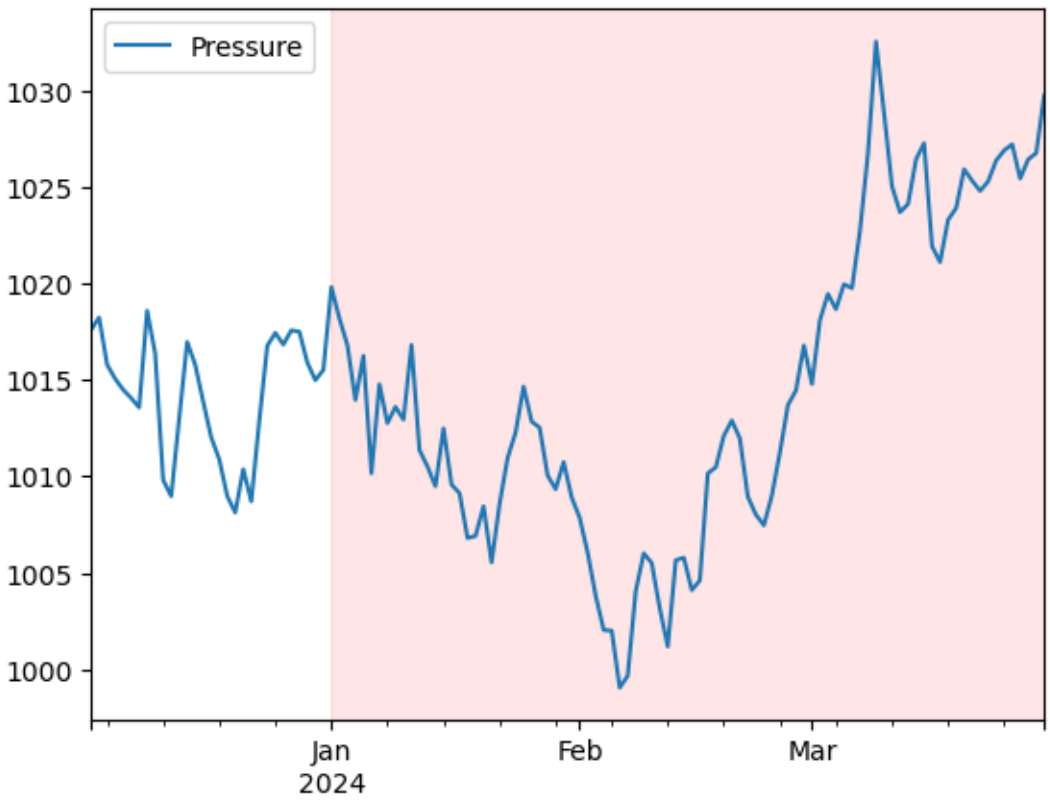
[22]:
```
idx = pd.date_range("2024-01-01", periods=90, freq="D")

np.random.seed(1)
train_data["d1_Temp"] = train_data["Temperature"].diff()
mu = train_data["d1_Temp"].mean()
sigma = train_data["d1_Temp"].std()
print(mu, sigma)
temp = pd.DataFrame(np.random.normal(mu, sigma, 90), index=idx,␣
 ↪columns=["Fcast"])
temp["forecast"] = train_data["Temperature"].iloc[-1]+ temp["Fcast"].cumsum()
pd.concat([train_data.iloc[-30:]["Temperature"], temp["forecast"]]).
 ↪plot(label="Temperature")
plt.axvspan(datetime(2024,1,1), datetime(2024,4,1), color='red', alpha=0.1)
plt.legend()
#plt.show()
plt.savefig('Temp_Forecast.png')
```

```
0.0015637860082304436 2.2271755838806233
```



[23]:
```
np.random.seed(1)
train_data["d1_Pres"] = train_data["Pressure"].diff()
mu = train_data["d1_Pres"].mean()
sigma = train_data["d1_Pres"].std()
print(mu,sigma)
temp = pd.DataFrame(np.random.normal(mu, sigma, 90), index=idx,␣
 ↪columns=["Fcast"])
temp["forecast"] = train_data["Pressure"].iloc[-1]+ temp["Fcast"].cumsum()
pd.concat([train_data.iloc[-30:]["Pressure"], temp["forecast"]]).
 ↪plot(label="Pressure")
plt.axvspan(datetime(2024,1,1), datetime(2024,4,1), color='red', alpha=0.1)
plt.legend()
#plt.show()
plt.savefig('Pres_Forecast.png')
```

```
-0.0008973479652492414 2.6447068863504186
```



[24]:
```
np.random.seed(1)
train_data["d1_Wind"] = train_data["Wind_Speed"].diff()
mu = train_data["d1_Wind"].mean()
sigma = train_data["d1_Wind"].std()
max_wind = 6

wind = [0]*90

for i in range(90):
    rand = np.random.normal(mu, sigma)
    if (i==0):
        wind[i] =train_data["Wind_Speed"].iloc[-1] + rand
    else:
        if(wind[i-1]+rand < 0):
            wind[i] = wind[i-1]
        elif(wind[i-1]+rand>max_wind):
            wind[i] = wind[i-1] - rand
        else:
            wind[i] = wind[i-1] + rand

temp = pd.DataFrame(wind, index=idx, columns=["forecast"])
```

```
pd.concat([train_data.iloc[-30:]["Wind_Speed"], temp["forecast"]]).
  ↪plot(label="Wind_Speed")
plt.axvspan(datetime(2024,1,1), datetime(2024,4,1), color='red', alpha=0.1)
plt.legend()
#plt.show()
plt.savefig('Wind_Forecast.png')
```