[2]:
```
import pandas as pd
import numpy as np
import datetime
import matplotlib.pyplot as plt
from termcolor import colored
from statsmodels.tsa.seasonal import seasonal_decompose
from pmdarima import auto_arima
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima_model import ARIMA, ARIMAResults
from statsmodels.tsa.statespace.sarimax import SARIMAX, SARIMAXResults
from datetime import datetime
from sklearn.metrics import mean_squared_error
```

```
/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 (
```

[3]:
```
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)
```

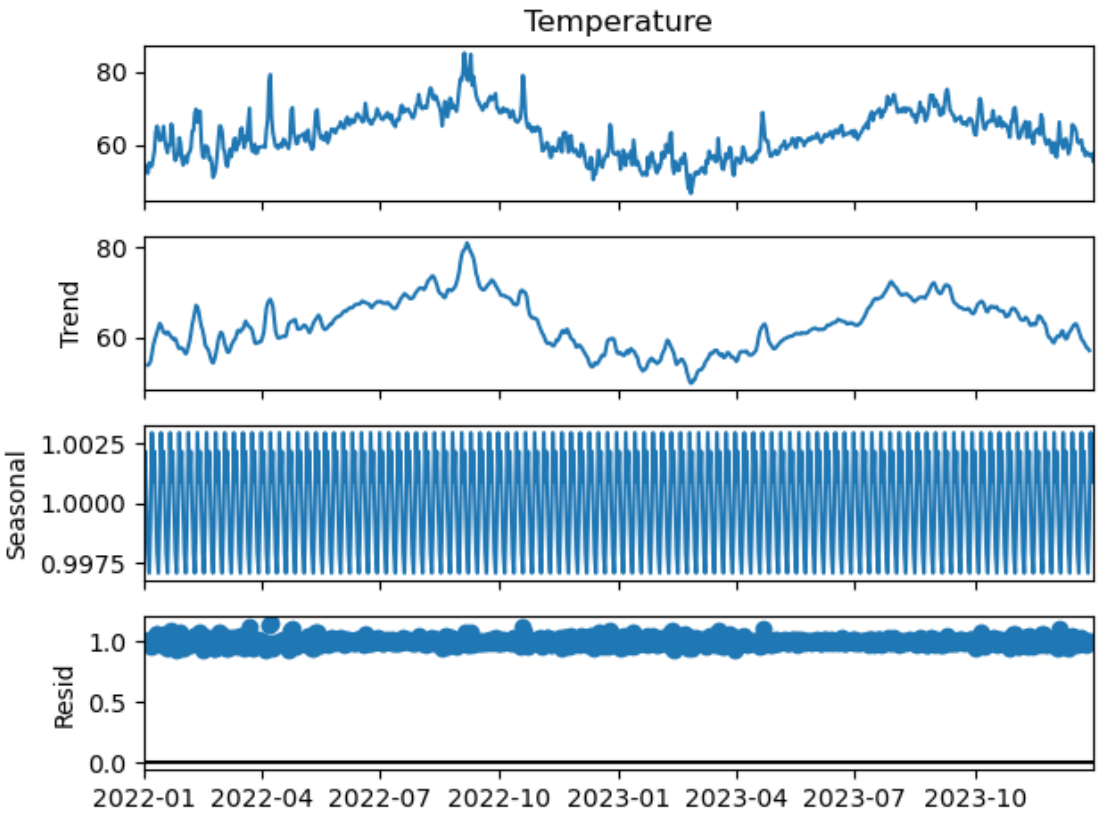
[4]:
```
weather_22["Date"]= pd.to_datetime(weather_22[['Year', 'Month', 'Day']])
weather_23["Date"]= pd.to_datetime(weather_23[['Year', 'Month', 'Day']])
weather_24["Date"]= pd.to_datetime(weather_24[['Year', 'Month', 'Day']])
```

[5]:
```
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"].replace("R", ""))
weather22_merged = weather22_merged.assign(Temperature = lambda x:␣
 ↪x["Temperature"].replace("L", ""))
weather22_merged = weather22_merged.assign(Temperature = lambda x:␣
 ↪x["Temperature"].replace("F", ""))
weather22_merged = weather22_merged.assign(Temperature = lambda x:␣
 ↪x["Temperature"]*(9/5)+32)
```

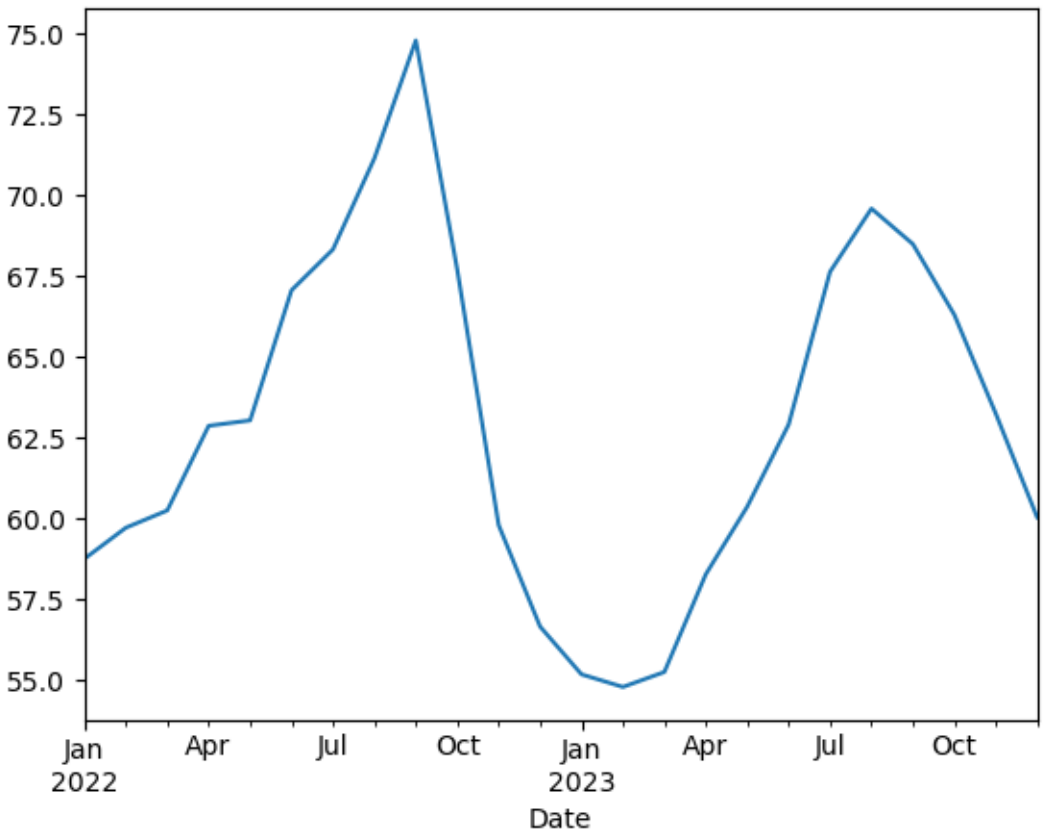
```
weather22_merged = weather22_merged.set_index("Date")

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"].replace("R", ""))
weather23_merged = weather23_merged.assign(Temperature = lambda x:␣
 ↪x["Temperature"].replace("L", ""))
weather23_merged = weather23_merged.assign(Temperature = lambda x:␣
 ↪x["Temperature"].replace("F", ""))
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"].replace("R", ""))
weather24_merged = weather24_merged.assign(Temperature = lambda x:␣
 ↪x["Temperature"].replace("L", ""))
weather24_merged = weather24_merged.assign(Temperature = lambda x:␣
 ↪x["Temperature"].replace("F", ""))
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()
```

[6]:
```
train_data = pd.concat([weather22_merged, weather23_merged])
train_data = train_data[["Temperature"]]
train_data.index.freq = "D"

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

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



[8]:
```
temp_30_tr = train_data['Temperature'].resample('MS').mean()
temp_30_te = test_data['Temperature'].resample('MS').mean()
temp_30_tr.plot()
plt.savefig('Temp_EWMA.png')
```



[9]:
```
adfuller(train_data["Temperature"])
```

[9]:
```
(-2.887766167368436,
 0.04678923333845198,
 7,
 722,
 {'1%': -3.439439614524914,
  '5%': -2.865551414233055,
  '10%': -2.5689061365397747},
 3079.044474206762)
```

[28]:
```
# Using the best parameters, we are going to fit the SARIMAX model
model = SARIMAX(train_data["Temperature"],
                order=(1,1,0),
                seasonal_order=(1,1,0,90),
                enforce_invertibility = False)
results = model.fit()
results.summary()
```

```
This problem is unconstrained.
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            3     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  2.16222D+00    |proj g|=  5.84103D-02

At iterate    5    f=  2.15008D+00    |proj g|=  3.51458D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    3      6      9      1     0     0   6.628D-07   2.150D+00
  F =   2.1500821674098387

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
```

[28]:

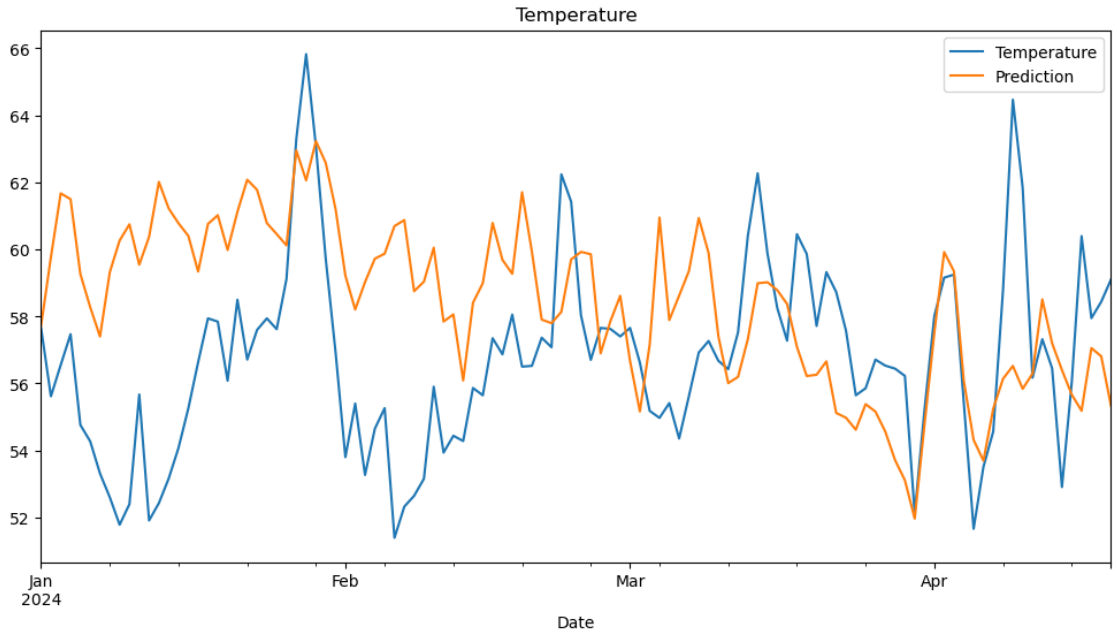
Dep. Variable:	Temperature	**No. Observations:**	730
Model:	SARIMAX(1, 1, 0)x(1, 1, 0, 90)	**Log Likelihood**	-1569.560
Date:	Fri, 10 May 2024	**AIC**	3145.120
Time:	23:18:44	**BIC**	3158.500
Sample:	01-01-2022 - 12-31-2023	**HQIC**	3150.314
Covariance Type:	opg		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
ar.L1	0.0563	0.034	1.641	0.101	-0.011	0.124
ar.S.L90	-0.4739	0.027	-17.310	0.000	-0.528	-0.420
sigma2	7.6817	0.337	22.799	0.000	7.021	8.342

Ljung-Box (L1) (Q):	0.13	**Jarque-Bera (JB):**	53.87
Prob(Q):	0.72	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.49	**Skew:**	0.01
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.42

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

[29]:
```
start = len(train_data)
end = len(train_data)+ len(test_data)-1
predictions = results.predict(start=start, end=end, dynamic=False).
 ↪rename("Prediction")

title = "Temperature"
ax= test_data["Temperature"].plot(legend=True, figsize=(12,6), title=title)
predictions.plot(legend=True)
ax.autoscale(axis='x', tight=True)
```



[30]:
```
print(colored("MSE",color="red", attrs=["bold"]),
      "(SARIMAX(0, 1, 0)x(1, 1, 0, 90)):",
      colored(round(mean_squared_error(test_data["Temperature"],␣
 ↪predictions),4), "red", attrs=['bold']))
```

```
MSE (SARIMAX(0, 1, 0)x(1, 1, 0, 90)): 15.0787
```

[10]:
```
idx = pd.date_range("2024-01-01", periods=180, 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, 180), 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(2024,5,1), datetime(2024,7,1), color='red', alpha=0.1)
plt.legend()
#plt.show()
plt.savefig('Temp_Forecast.png')
```

```
0.0015637860082304436 2.2271755838806233
```