EXP:7 Implement program for decomposing time series data into trend and seasonality.

Aim:

To apply **time series decomposition** on Google Trends data to analyze trends and seasonality.

Procedure and Code:

Step 1: Install Dependencies

!pip install statsmodels --quiet

statsmodels is required for the seasonal decomposition method.

Step 2: Import Required Libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.seasonal import seasonal_decompose

These are needed for data manipulation, plotting, and time series decomposition.

Step 3: Upload the Dataset in Google Colab

from google.colab import files

uploaded = files.upload()

This opens a file upload prompt. Upload your cleaned_weather.csv here.

Step 4: Load and Inspect the Dataset

df = pd.read_csv("cleaned_weather.csv")

print("Available columns in your dataset:", df.columns.tolist())

This loads the dataset and prints all column names. Confirm that 'T' (temperature) exists.

Step 5: Preprocess the Data

df['date'] = pd.to_datetime(df['date']) # Convert date column to datetime

df = df.sort_values(by='date') # Sort by time

time_series_data = df.set_index('date')['T'] # Use 'T' column for temperature

This step ensures time alignment and sets the datetime as the index.

Step 6: Define Helper Functions

def moving_average_smoothing(data, window_size):

return data.rolling(window=window_size, min_periods=1).mean()

def decompose_time_series(data, period):

return seasonal_decompose(data, period=period, model='additive')

These functions help perform moving average smoothing and decomposition.

Step 7: Apply Moving Average Smoothing

window_size = 7 # Weekly smoothing

smoothed_data = moving_average_smoothing(time_series_data, window_size)

Smoothens the short-term fluctuations in the data.

Step 8: Decompose Time Series into Components

period = 30 # Assuming daily data with monthly seasonality

decomposition = decompose_time_series(time_series_data, period)

The period should reflect your dataset's frequency (e.g., 30 for monthly patterns in daily data).

Step 9: Plot Original vs Smoothed Data

plt.figure(figsize=(12, 5))

plt.plot(time_series_data, label='Original Data', linestyle='dashed')

plt.plot(smoothed_data, label=f'Smoothed (Window={window_size})', linewidth=2)

plt.legend()

plt.title('Moving Average Smoothing of Temperature (T)')

plt.grid(True)

plt.show()

Visualizes how smoothing removes noise and highlights the trend.

Step 10: Plot Decomposition Results

fig, axes = plt.subplots(3, 1, figsize=(12, 8), sharex=True)

decomposition.trend.plot(ax=axes[0], title='Trend (Temperature)', legend=False)

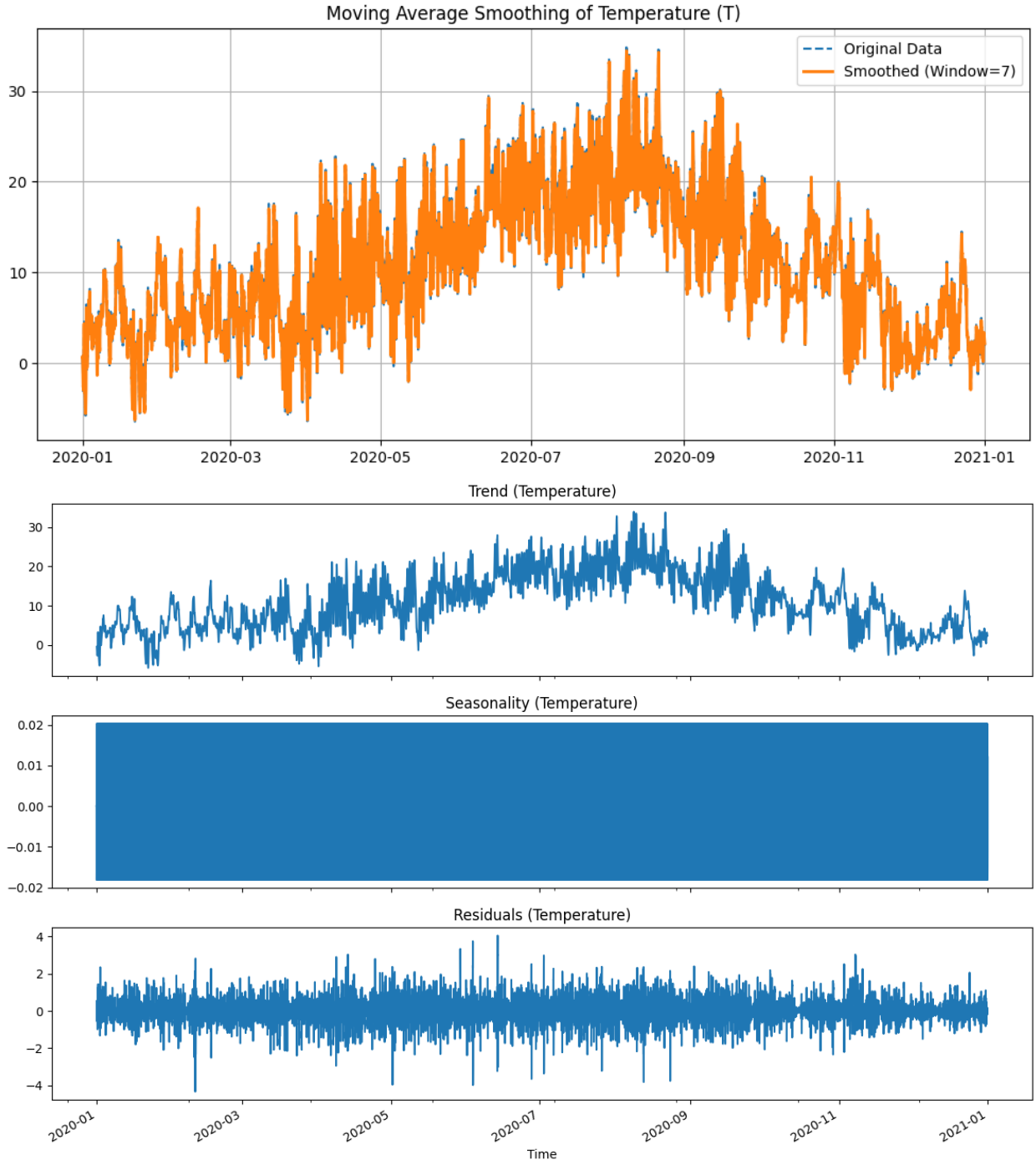
decomposition.seasonal.plot(ax=axes[1], title='Seasonality (Temperature)', legend=False)

decomposition.resid.plot(ax=axes[2], title='Residuals (Temperature)', legend=False)

plt.xlabel("Time")

plt.tight_layout()

plt.show()


Moving Average Smoothing of Temperature (T)
Original Data
Smoothed (Window=7)
30
20
10
0
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01
Trend (Temperature)
30
20
10
0
Seasonality (Temperature)
0.02
0.01
0.00
−0.01
−0.02
Residuals (Temperature)
4
2
0
−2
−4
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01
Time

Result:

The program to decompose the google trends dataset has been successfully implemented and verified