**EX:No.5 221501025**

**25/03/25**

**IMPLEMENT PROGRAMS FOR ESTIMATING & ELIMINATING TREND IN TIME SERIES DATA – AGGREGATION, SMOOTHING.**

**AIM:**

To Implement programs for estimating & eliminating trend in time series data- aggregation, smoothing.

**ALGORITHM:**

1. **Import Libraries**: Load required Python libraries (pandas, numpy, matplotlib, seaborn).
2. **Load Dataset**: Read the apple\_stock\_prices.csv file using pandas.read\_csv().
3. **Preprocess Data**:
   * Strip leading/trailing spaces from column names.
   * Check if the required column exists.
   * Convert the DATE column to datetime format and set it as the index.
   * Handle missing values by replacing them with the mean.
4. **Aggregation**:
   * Resample data to compute **monthly mean** (M).
   * Resample data to compute **yearly mean** (Y).
5. **Smoothing**:
   * Compute **3-month rolling mean** to reduce short-term fluctuations.
   * Compute **12-month rolling mean** for long-term trend analysis.
6. **Visualization**:
   * Plot **original vs. aggregated data** (monthly, yearly).
   * Plot **original vs. smoothed data** (3-month, 12-month rolling means).
7. **Output Result**: Display a confirmation message.

**PROCESS:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Load the Apple stock data

file\_path = "C:\\Users\\Lenovo\\Desktop\\tsa\_exp\_4\\apple\_stock\_prices.csv"

data = pd.read\_csv(file\_path)

# Ensure column names have no leading/trailing spaces

data.columns = data.columns.str.strip()

# Convert 'Date' column to datetime and set it as the index

data['Date'] = pd.to\_datetime(data['Date'])

data.set\_index('Date', inplace=True)

# Use 'Adj Close' as the relevant column for analysis

column\_name = 'Adj Close'

# Handle missing values by filling with the mean

data[column\_name].fillna(data[column\_name].mean(), inplace=True)

# Aggregation: Resample to monthly and yearly averages

monthly\_data = data[column\_name].resample('M').mean()

yearly\_data = data[column\_name].resample('Y').mean()

# Smoothing: Calculate rolling means (3-day and 12-day)

rolling\_mean\_3 = data[column\_name].rolling(window=3).mean()

rolling\_mean\_12 = data[column\_name].rolling(window=12).mean()

# Plot original data vs monthly/yearly aggregated data

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

plt.plot(data[column\_name], label='Original Data', alpha=0.5)

plt.plot(monthly\_data, label='Monthly Aggregated', linestyle='dashed')

plt.plot(yearly\_data, label='Yearly Aggregated', linestyle='dotted')

plt.title('Apple Stock Prices: Aggregated Data')

plt.xlabel('Time')

plt.ylabel('Adjusted Close Price')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

# Plot original data vs smoothed data

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

plt.plot(data[column\_name], label='Original Data', alpha=0.5)

plt.plot(rolling\_mean\_3, label='3-day Rolling Mean', linestyle='dashed')

plt.plot(rolling\_mean\_12, label='12-day Rolling Mean', linestyle='dotted')

plt.title('Apple Stock Prices: Smoothing Techniques')

plt.xlabel('Time')

plt.ylabel('Adjusted Close Price')

plt.legend()

plt.grid(True)

plt.tight\_layout()

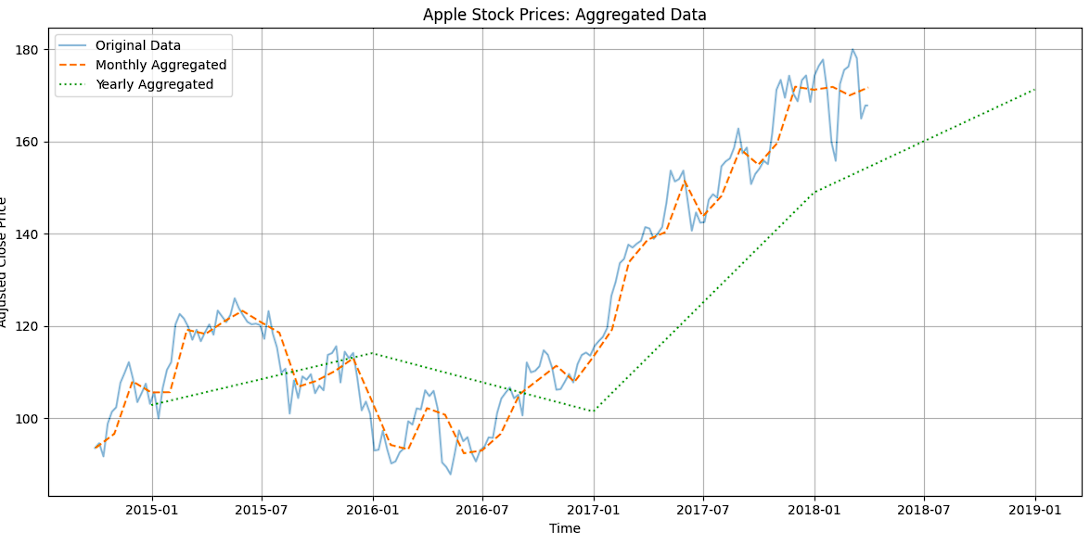
plt.show()

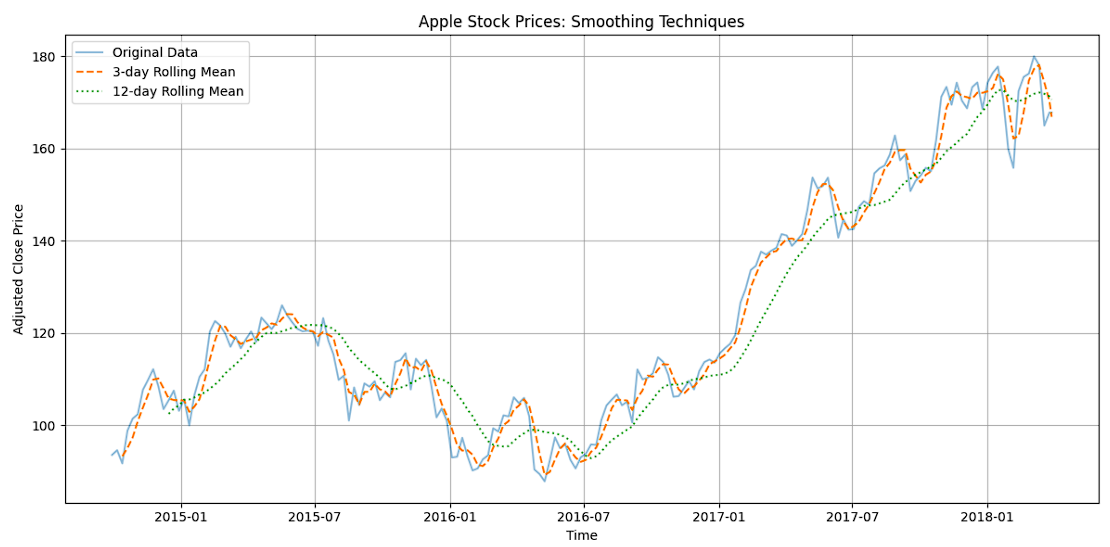
print("Aggregation and smoothing techniques applied successfully.")

**# Result:**

print("Aggregation and smoothing techniques applied successfully on the time series dataset.")

**OUTPUT:**

****

****

**RESULT:**

The program to Implement estimating & eliminating trend in time series data- aggregation, smoothing is created and executed successfully.