Date : 27/03/25

EX .NO : 6 Implement program to apply moving average smoothing for data preparation and time series forecasting

AIM:

To Implement program to apply moving average smoothing for data preparation and time series forecasting

PROCEDURE:

1. Data Loading

- First, load the Google stock price dataset using Pandas.
- Ensure the Date column is recognized as a date using parse_dates.
- Set Date as the index to make it suitable for time series analysis.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv('google_stock_prices.csv', parse_dates=['Date'],
index_col='Date')
print(data.head())
```

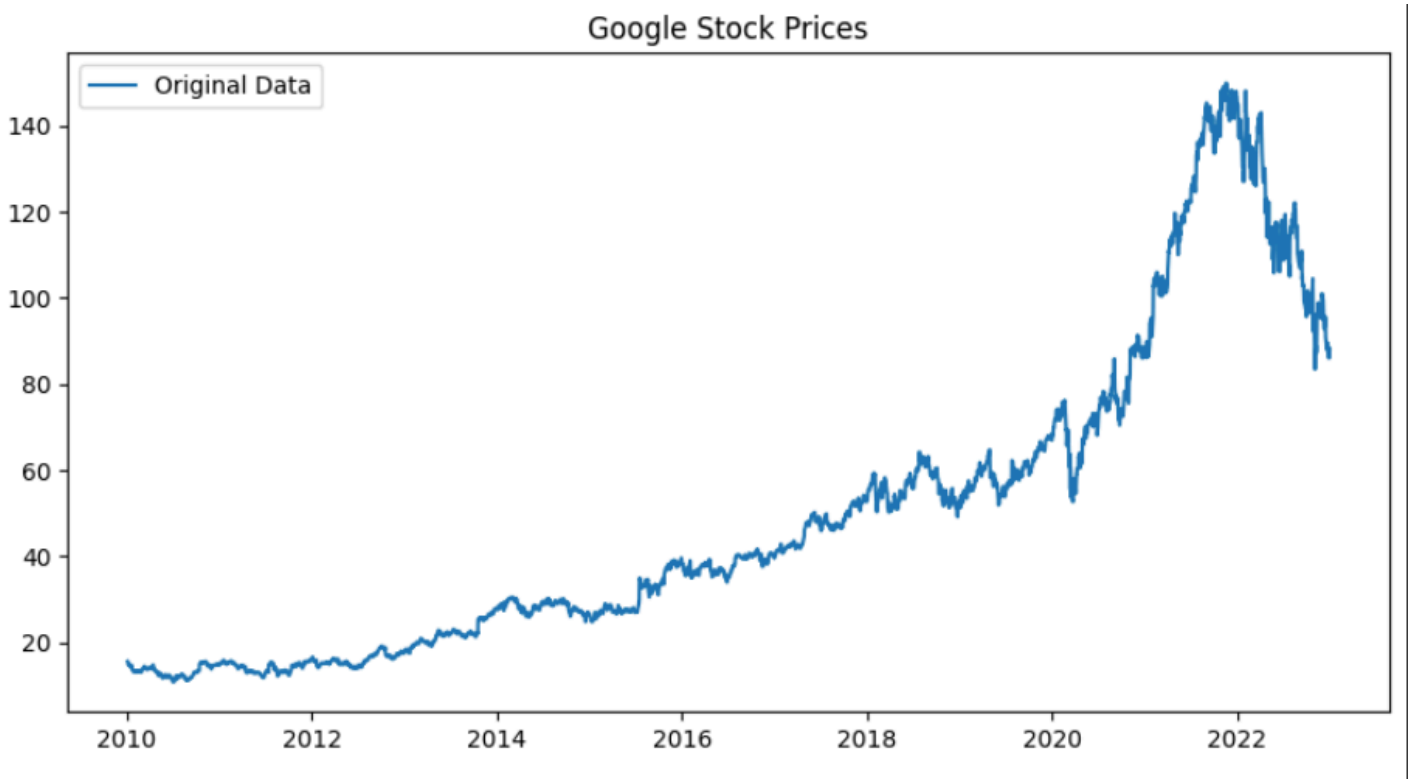
Explanation: This helps us check the structure of the data and ensure it is correctly formatted for further analysis.

2. Visualization of Original Data

- Plot the original data using **Matplotlib** to observe the stock price trend.

```
plt.figure(figsize=(10, 5))
plt.plot(data['Close'], label='Original Data')
plt.title('Google Stock Prices')
plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.legend()
plt.grid(True)
plt.show()
```

Explanation: Visualizing the data helps us identify trends, seasonality, and any anomalies.



3. Moving Average Smoothing

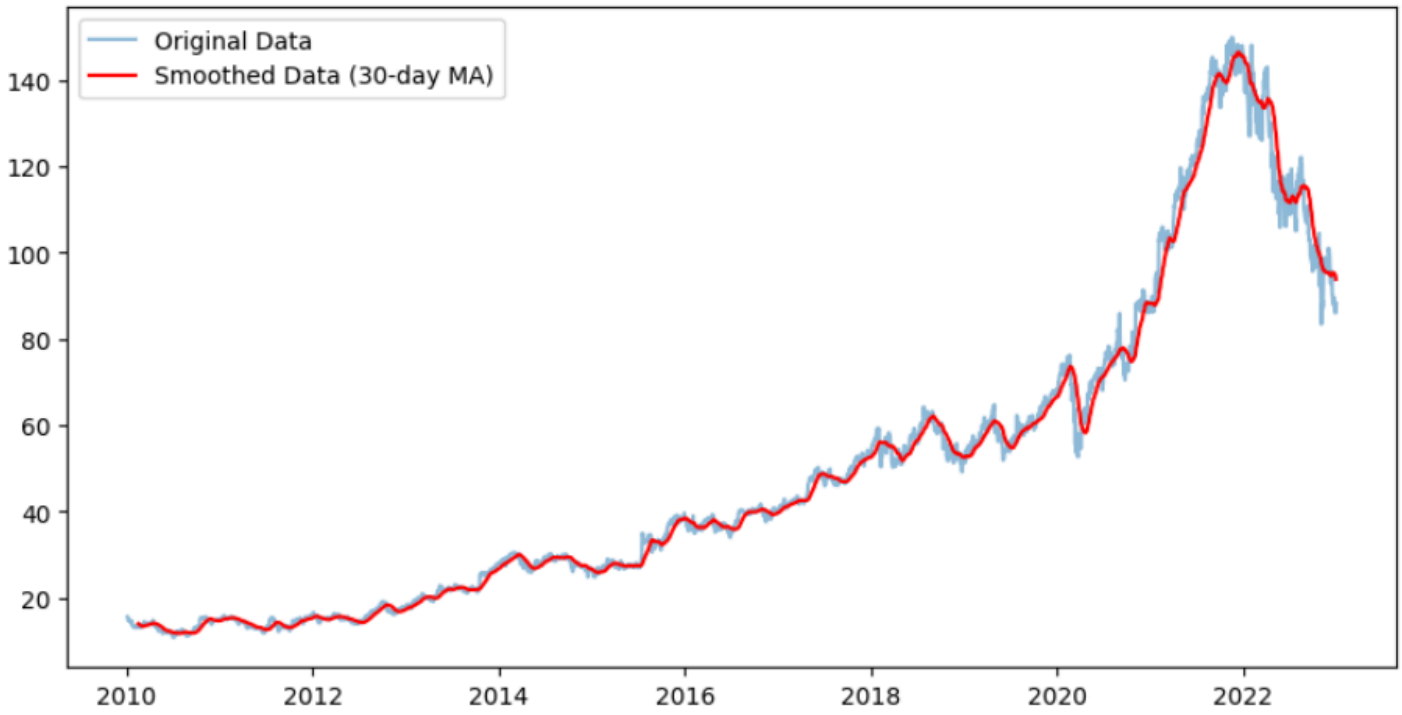
- Apply a **30-day simple moving average (SMA)** using the `.rolling()` method.
- This reduces noise and reveals the underlying trend.

```
window_size = 30

smoothed_data = data['Close'].rolling(window=window_size).mean()

plt.figure(figsize=(10, 5))

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

plt.plot(smoothed_data, label='Smoothed Data (30-day MA)', color='red')

plt.title('Moving Average Smoothing')

plt.xlabel('Date')

plt.ylabel('Closing Price')

plt.legend()

plt.grid(True)

plt.show()
```

Explanation:

- `rolling(window=30)` creates a moving window of 30 days.
- `.mean()` calculates the average within the window.
- The red line represents the smoothed data, making the trend clearer.

.



4. Forecasting Using Naive Trend Extension

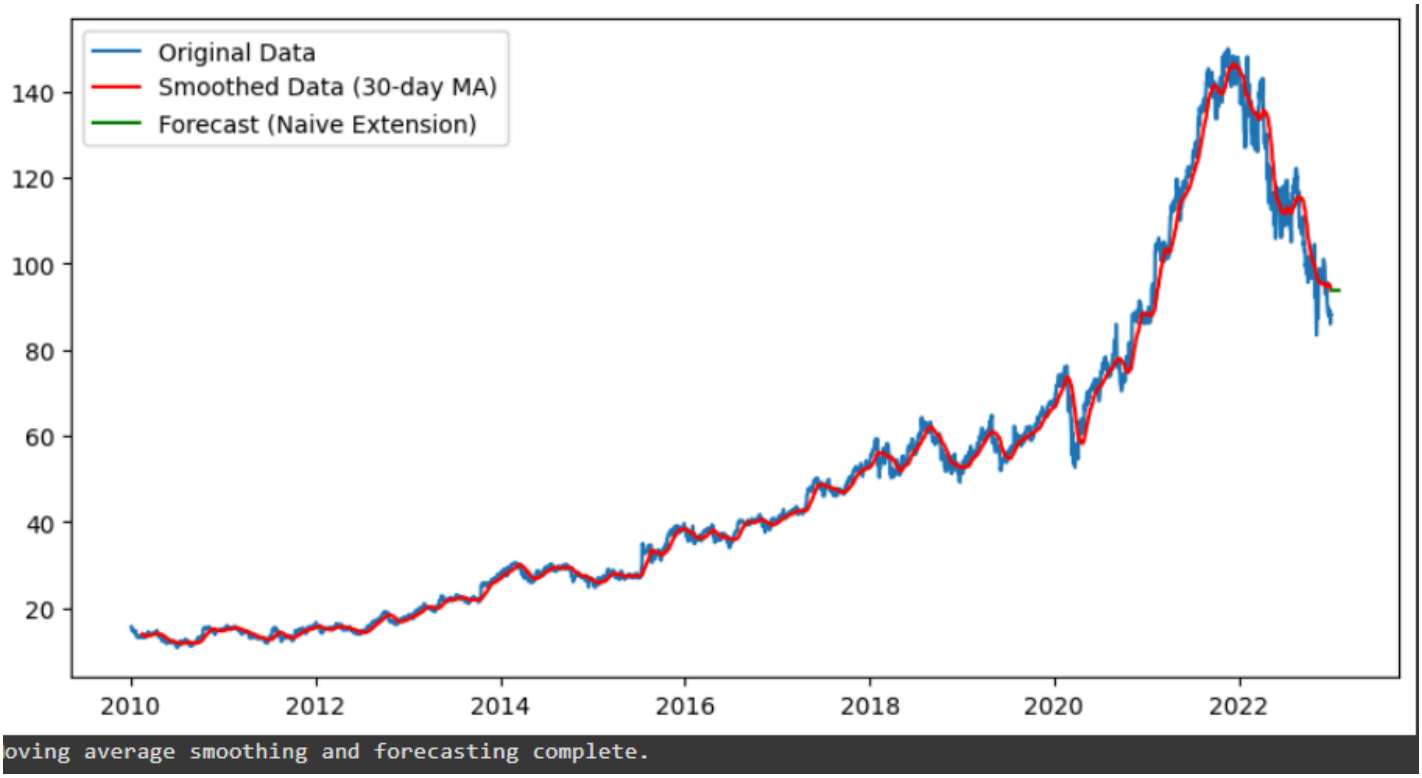
- Perform a simple time series forecast using **naive trend extension**.
- Assume that the last smoothed value will persist as a prediction for future dates.

```
forecast_period = 30

last_trend_value = smoothed_data.dropna().iloc[-1]

forecast_dates = pd.date_range(data.index[-1] +
pd.Timedelta(days=1), periods=forecast_period)

forecast = pd.Series(last_trend_value,
index=forecast_dates)
```

```
plt.figure(figsize=(10, 5))

plt.plot(data['Close'], label='Original Data')

plt.plot(smoothed_data, label='Smoothed Data (30-day
MA)', color='red')

plt.plot(forecast, label='Forecast (Naive Extension)',
color='green')

plt.title('Forecasting using Moving Average Smoothing')

plt.xlabel('Date')

plt.ylabel('Closing Price')

plt.legend()

plt.grid(True)

plt.show()
```

Explanation:

- `last_trend_value` captures the most recent smoothed value.
- `forecast_dates` generates future dates for the next 30 days.
- `forecast` assumes no major changes, following the last known trend.



RESULT :

Thus the Implementation programs for estimating & eliminating trends in time series data- aggregation, smoothing is done successfully.