EXP No:5 Implement programs for estimating & eliminating trend in time series data- aggregation, smoothing.

Aim:

To estimate and eliminate trends in time series data using aggregation, smoothing, and - polynomial trend estimation for better analysis and forecasting.

Objectives:

1. Load and preprocess time series data.
2. Aggregate data using monthly resampling.
3. Apply smoothing techniques like moving averages.
4. Extract trends using polynomial fitting.
5. Visualize and compare original vs. processed data.

Background:

Time series data, such as electricity production, often exhibits trends and seasonality that need to be removed for accurate analysis and forecasting. **Trend estimation and elimination** are crucial preprocessing steps in time series modeling.

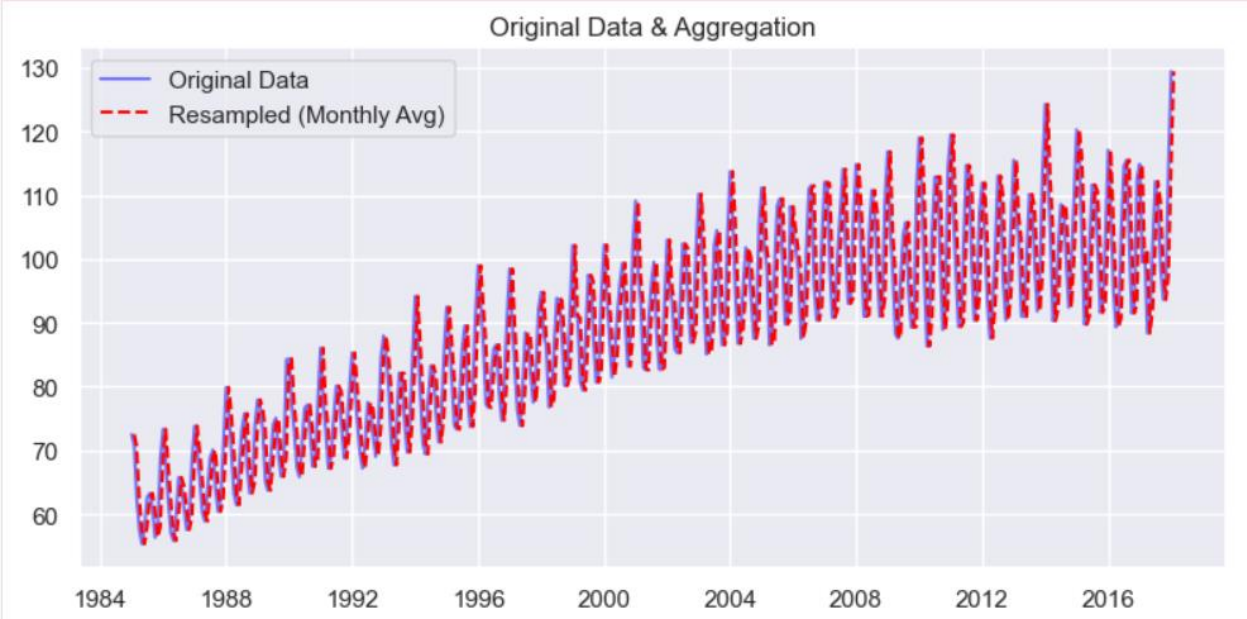
- **Aggregation (Resampling):** Converting daily data into monthly averages helps reduce noise and highlights long-term trends.
- **Moving Average Smoothing:** Averages values over a specific window (e.g., 12 months) to smooth out fluctuations.
- **Polynomial Trend Estimation:** Fits a polynomial function to the data to extract the underlying trend and remove it for further analysis.

Code:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from numpy.polynomial.polynomial import Polynomial


# Load dataset

file_path = r"C:\Users\Lenovo\Downloads\Electric_Production.csv"

df = pd.read_csv(file_path)
```

```
# Convert DATE to datetime and set as index
df['DATE'] = pd.to_datetime(df['DATE'])
df.set_index('DATE', inplace=True)

# 1. Aggregation (Resampling) - Monthly Average
df_resampled = df.resample('M').mean()

# 2. Smoothing using Moving Average
df_resampled['MA_12'] = df_resampled['IPG2211A2N'].rolling(window=12).mean()

# 3. Trend Estimation using Polynomial Fitting
def polynomial_detrend(series, degree=2):
    x = np.arange(len(series))
    coeffs = Polynomial.fit(x, series.dropna(), degree).convert().coef
    trend_estimate = sum(c * x**i for i, c in enumerate(coeffs))
    return trend_estimate

estimated_trend = polynomial_detrend(df_resampled['IPG2211A2N'])

# Visualization (Now Only 3 Graphs)
fig, axes = plt.subplots(3, 1, figsize=(8, 12))



# 1. Original Data with Aggregation
axes[0].plot(df.index, df['IPG2211A2N'], alpha=0.5, label='Original Data', color='blue')
axes[0].plot(df_resampled.index, df_resampled['IPG2211A2N'], color='red', linestyle='dashed',
label='Resampled (Monthly Avg)')
axes[0].set_title('Original Data & Aggregation')
```

axes[0].legend()



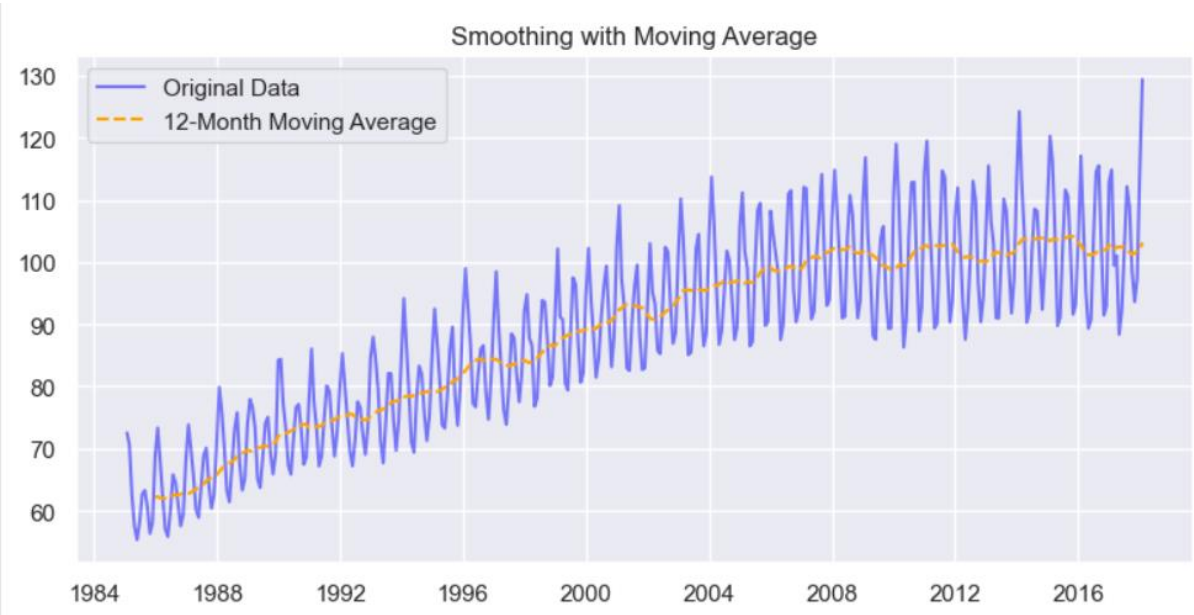
2. Smoothing using Moving Average

axes[1].plot(df_resampled.index, df_resampled['IPG2211A2N'], alpha=0.5, label='Original Data', color='blue')

axes[1].plot(df_resampled.index, df_resampled['MA_12'], color='orange', linestyle='dashed', label='12-Month Moving Average')

axes[1].set_title('Smoothing with Moving Average')

axes[1].legend()



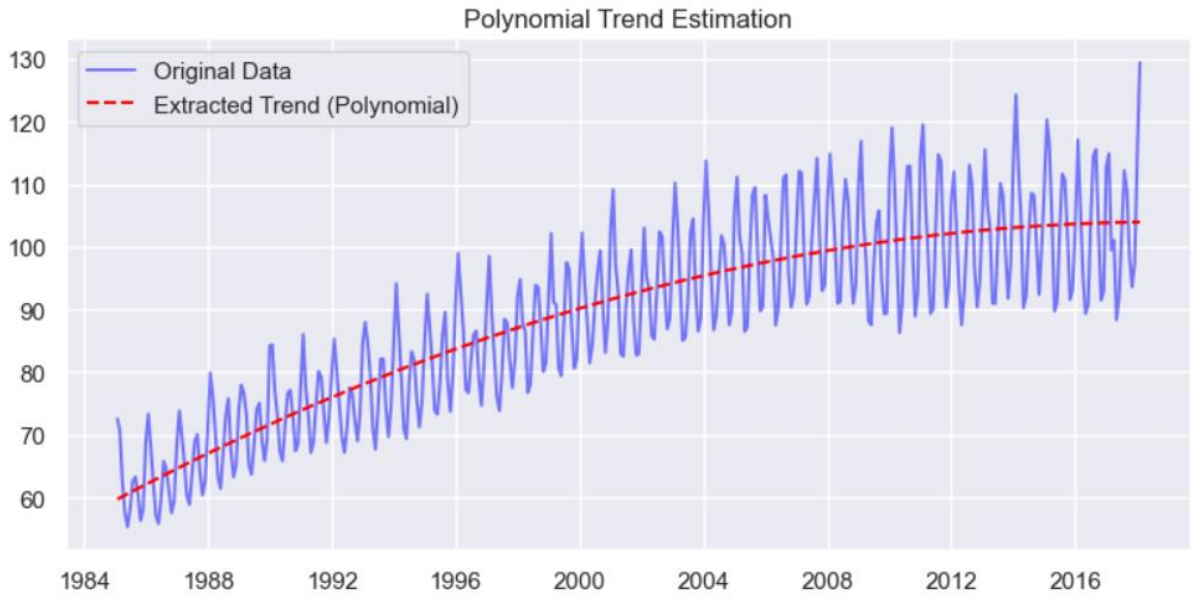
3. Polynomial Trend Estimation

axes[2].plot(df_resampled.index, df_resampled['IPG2211A2N'], alpha=0.5, label='Original Data', color='blue')

axes[2].plot(df_resampled.index, estimated_trend, color='red', linestyle='dashed', label='Extracted Trend (Polynomial)')

axes[2].set_title('Polynomial Trend Estimation')

axes[2].legend()



plt.tight_layout()

plt.show()

Result:

Thus programs for estimating & eliminating trend in time series data- aggregation, smoothing techniques have been implemented successfully.