**EX:No.6**

**221501016 MOVING AVERAGE SMOOTHING**

**AND FORECASTING**

**AIM :** To implement moving average and smoothing on a time series dataset.

**PROCEDURE:**

 Read the dataset and convert the date column to datetime format.

 Set the date column as the index for time series analysis.

 Aggregate the data by calculating the monthly mean.

 Apply a 12-month moving average for smoothing.

 Plot the original data, aggregated data, and smoothed data.

 Display the graph to visualize trend estimation and elimination.

**IMPLEMENTATION :**

**Import necessary libraries**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.arima.model import ARIMA

from sklearn.metrics import mean\_squared\_error

from math import sqrt

**Load the dataset**

df = pd.read\_csv('/mnt/data/Electric\_Production.csv')

df.columns = ['Date', 'Production']

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

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

**Visualize original data**

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

plt.plot(df, label='Original Data')

plt.title('Electric Production Over Time')

plt.xlabel('Date')

plt.ylabel('Production')

plt.legend()

plt.show()

**Apply Moving Average Smoothing**

df['Smoothed'] = df['Production'].rolling(window=12).mean()

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

plt.plot(df['Production'], label='Original')

plt.plot(df['Smoothed'], label='12-month Moving Average', color='orange')

plt.title('Smoothed vs Original')

plt.legend()

plt.show()

**Drop NaN values from smoothing**

df\_clean = df.dropna()

**Split into training and test sets**

train\_size = int(len(df\_clean) \* 0.8)

train, test = df\_clean['Smoothed'][:train\_size], df\_clean['Smoothed'][train\_size:]

**Fit an ARIMA model**

model = ARIMA(train, order=(2, 1, 2)) # You can tune this order

model\_fit = model.fit()

**Forecast**

forecast = model\_fit.forecast(steps=len(test))

forecast = pd.Series(forecast, index=test.index)

**Plot actual vs predicted**

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

plt.plot(train, label='Training')

plt.plot(test, label='Test')

plt.plot(forecast, label='Forecast', color='red')

plt.title('Forecast vs Actuals')

plt.legend()

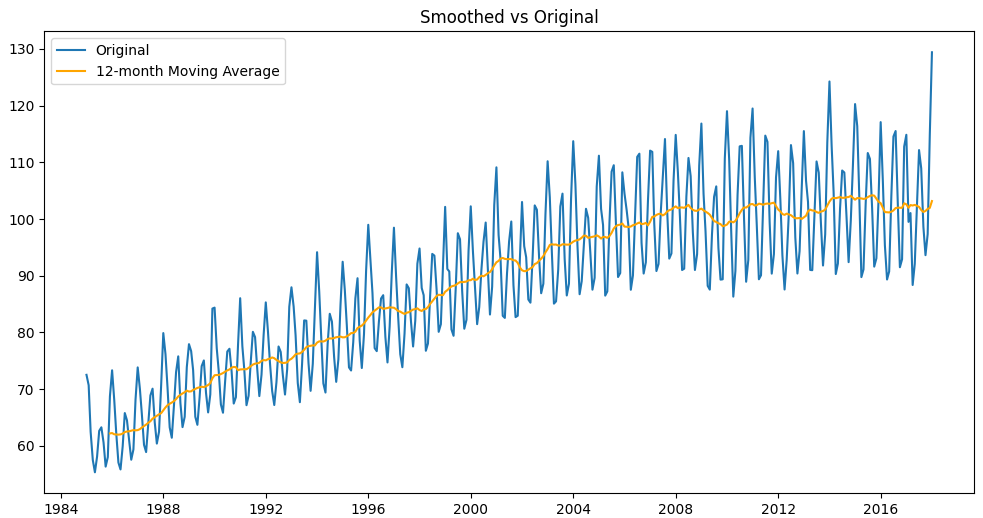
plt.show()

**Evaluate forecast**

rmse = sqrt(mean\_squared\_error(test, forecast))

print(f"RMSE: {rmse:.2f}")

**OUTPUT:**

****

**RESULT :** Thus moving average and smoothing has been implemented.