**EX:No.10 221501017**

**22/04/25**

**DEVELOP VECTOR AUTO REGRESSION MODEL FOR MULTIVARIATE TIME SERIES DATA FORECASTING.**

**AIM:**

To implement program for Develop neural network-based time series forecasting model.

**ALGORITHM:**

**OBJECTIVE:**

Smooth the electric production data to reduce noise, highlight trends, and prepare for forecasting.

**BACKGROUND:**

1.Time series data has short-term fluctuations.

2.Moving average reduces noise and clarifies trends.

3.Smoothed data improves forecast accuracy and interpretability.

**SCOPE OF THE PROGRAM:**

1.Load and clean dataset

2.Convert date column to datetime

3.Aggregate data monthly and yearly

4.Apply 3-month and 12-month moving averages

5.Plot original vs smoothed data

**ALGORITHM:**

1.Import libraries

2.Load dataset

3.Preprocess and set datetime index

4.Resample data (monthly, yearly)

5.Apply 3-month & 12-month smoothing

6.Visualize results

**PROCESS:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.api import VAR

from sklearn.metrics import mean\_squared\_error

from statsmodels.tsa.stattools import adfuller

**# Load dataset**

df = pd.read\_csv('/content/gold\_price\_dataset.csv', parse\_dates=['DATE'])

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

**# Rename for easier access**

df.rename(columns={'IPG2211A2N': 'Electric\_Production'}, inplace=True)

**# Add dummy second variable (simulated temperature trend)**

df['Dummy\_Temp'] = df['Electric\_Production'].rolling(window=3, min\_periods=1).mean()

**# Drop NA caused by rolling**

df.dropna(inplace=True)

**# ADF test function**

def make\_stationary(data):

diffed = data.copy()

for col in data.columns:

result = adfuller(diffed[col])

if result[1] > 0.05:

print(f"{col} is non-stationary, differencing applied (p={result[1]:.4f})")

diffed[col] = diffed[col].diff()

return diffed.dropna()

**# Make both series stationary**

stationary\_df = make\_stationary(df)

**# Split into train and test**

n = int(len(stationary\_df) \* 0.8)

train = stationary\_df[:n]

test = stationary\_df[n:]

**# Fit VAR model**

model = VAR(train)

results = model.fit(maxlags=3, ic='aic')

**# Forecast**

lag\_order = results.k\_ar

forecast\_input = train.values[-lag\_order:]

forecast = results.forecast(y=forecast\_input, steps=len(test))

**# Forecast DataFrame**

forecast\_df = pd.DataFrame(forecast, index=test.index, columns=['Electric\_Production\_Pred', 'Dummy\_Temp\_Pred'])

**# Plot results**

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

plt.plot(test['Electric\_Production'], label='Actual')

plt.plot(forecast\_df['Electric\_Production\_Pred'], label='Forecast', color='red')

plt.title('Electric Production - VAR Forecast (Differenced Series)')

plt.xlabel('Date')

plt.ylabel('Differenced Value')

plt.legend()

plt.grid(True)

plt.tight\_layout()

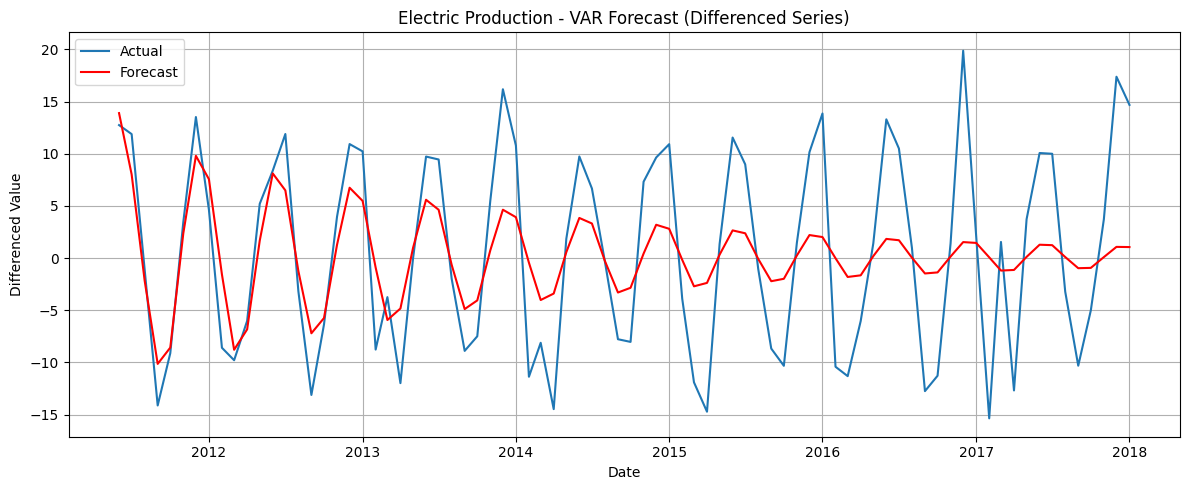
plt.show()

**# RMSE Evaluation**

rmse = np.sqrt(mean\_squared\_error(test['Electric\_Production'], forecast\_df['Electric\_Production\_Pred']))

print(f'RMSE: {rmse:.4f}')

**OUTPUT:**



**\**

**RESULT:**

The program to Develop neural network-based time series forecasting model created and executed successfully.