**Develop vector auto regression model for multivariate time series data forecasting**

**EX:No. 10**

**DATE: 14/04/25**

# AIM:

To implement a program for developing a vector auto regression model for multivariate time series data forecasting.

## ALGORITHM:

* Import required libraries (pandas, numpy, matplotlib, statsmodels)
* Load the office supply dataset (cleaned\_sales\_data.csv)
* Convert the date column to datetime format and set it as the index
* Select multiple numeric columns for multivariate analysis (e.g., Sales, Quantity)
* Resample the data to monthly frequency to reduce noise and capture long-term trends
* Handle missing values using forward fill
* Split the time series data into training and testing sets (80% train, 20% test)
* Fit a Vector AutoRegression (VAR) model on the training data and determine optimal lag using AIC
* Forecast future values for the test period using the trained VAR model
* Visualize and compare forecasted values with actual data for each variable separately

**CODE:**

import pandas as pd

import numpy as np

from statsmodels.tsa.api import VAR

import matplotlib.pyplot as plt

# Load dataset

df = pd.read\_csv("cleaned\_sales\_data.csv")

df['Order Date'] = pd.to\_datetime(df['Order Date'], errors='coerce')

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

# Monthly total sales

df\_monthly = df['Sales'].resample('MS').sum().to\_frame()

# Add synthetic second column: 3-month rolling mean

df\_monthly['Rolling\_Mean'] = df\_monthly['Sales'].rolling(window=3).mean()

# Drop NaNs caused by rolling

df\_monthly.dropna(inplace=True)

# Optional: normalize to reduce scale issues

df\_monthly = (df\_monthly - df\_monthly.mean()) / df\_monthly.std()

# Split into train/test

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

train = df\_monthly.iloc[:train\_size]

test = df\_monthly.iloc[train\_size:]

# Fit VAR model with fixed lag (safe fallback)

model = VAR(train)

model\_fitted = model.fit(1) # fixed lag of 1 to avoid matrix issues

# Forecast

forecast\_input = train.values[-1:]

forecast\_steps = len(test)

forecast = model\_fitted.forecast(y=forecast\_input, steps=forecast\_steps)

forecast\_df = pd.DataFrame(forecast, index=test.index, columns=df\_monthly.columns)

# Plot forecasts vs actual

for col in df\_monthly.columns:

plt.figure(figsize=(10, 4))

plt.plot(test.index, test[col], label='Actual')

plt.plot(forecast\_df.index, forecast\_df[col], label='Forecast')

plt.title(f'{col} Forecast using VAR')

plt.legend()

plt.tight\_layout()

plt.show()

# OUTPUT:

# **Screenshot 2025-04-30 062210**

**RESULT:**

Thus, the program is implemented and verified successfully.