|  |  |
| --- | --- |
| **EX.N0 : 10** | **Develop vector auto regression model for multivariate time series data forecasting.** |
| **DATE : 08/04/2025** |

**AIM:**

# Develop vector auto regression model for multivariate time series data forecasting.

# ALGORITHM:

# Step 1: Load and Preprocess Data Step 2: Check for Stationarity (ADF Test) Step 3: Fit the VAR Model Step 4: Forecast Future Values Step 5: Format Output

# PROGRAM:

# STEP 1: Install Required Libraries

!pip install pandas statsmodels matplotlib

# STEP 2: Import Libraries

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.api import VAR

from statsmodels.tsa.stattools import adfuller

from google.colab import files

# STEP 3: Upload the Dataset

uploaded = files.upload() # Upload 'climate\_change\_data.csv'

# STEP 4: Load and Preprocess Data

df = pd.read\_csv('climate\_change\_data.csv')

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

df.sort\_values('Date', inplace=True)

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

df = df.drop(columns=['Location', 'Country']) # Drop non-numeric columns

# STEP 5: Check Stationarity Using ADF Test

def adf\_test(series):

result = adfuller(series)

print(f"{series.name}: ADF={result[0]:.4f}, p={result[1]:.4f}")

print("=== Augmented Dickey-Fuller (ADF) Test Results ===")

for column in df.columns:

adf\_test(df[column])

# STEP 6: Fit the VAR Model with Manual Lag

manual\_lag = 2 # Manually specified lag

model = VAR(df)

results = model.fit(manual\_lag)

print(f"\nUsing manual lag order: {manual\_lag}")

# STEP 7: Forecast Future Values

forecast\_steps = 30 # Forecast 30 time steps into the future

forecast\_input = df.values[-manual\_lag:]

forecast = results.forecast(y=forecast\_input, steps=forecast\_steps)

# Create DataFrame for forecast

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

forecast\_df.index = pd.date\_range(start=df.index[-1], periods=forecast\_steps+1, freq='D')[1:]

# STEP 8: Plot Forecast vs Actual

for col in df.columns:

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

plt.plot(df[col][-100:], label=f"Actual {col}")

plt.plot(forecast\_df[col], linestyle='--', label=f"Forecast {col}")

plt.title(f"{col} - Forecast vs Actual")

plt.xlabel("Date")

plt.ylabel(col)

plt.legend()

plt.grid(True)

plt.show()

**OUTPUT:**

=== Augmented Dickey-Fuller (ADF) Test Results ===

Temperature: ADF=-100.8747, p=0.0000

CO2 Emissions: ADF=-99.6286, p=0.0000

Sea Level Rise: ADF=-99.2354, p=0.0000

Precipitation: ADF=-69.3881, p=0.0000

Humidity: ADF=-97.5047, p=0.0000

Wind Speed: ADF=-72.3849, p=0.0000

A graph showing a temperature

AI-generated content may be incorrect.

A graph showing a graph of a graph

AI-generated content may be incorrect.A graph of a graph

AI-generated content may be incorrect.A graph showing a number of blue lines

AI-generated content may be incorrect.

A graph of a weather forecast

AI-generated content may be incorrect.

A graph with blue lines

AI-generated content may be incorrect.

# RESULT:

Thus, the program for Implement programs for a time series data is executed successfully.