EX:No.10 221501026

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

**Aim:**

Write a program to develop vector auto regression model for multivariate time series data forecasting.

**Algorithm:**

1.**Import necessary libraries**:  
Import numpy, pandas, matplotlib.pyplot, VAR from statsmodels.tsa.api, and mean\_squared\_error from sklearn.metrics.

2.**Load the dataset**:  
Read the weather data CSV file, parse the 'Date' column as datetime, and set it as the index.

3.**Select multiple relevant columns**:  
Choose important features for multivariate analysis like 'Temperature', 'Humidity', and 'WindSpeed'.

4.**Handle missing values**:  
Use forward fill method (ffill) to fill in any missing values in the dataset.

5.**Split the data into training and testing sets**:  
Reserve the last n observations for testing (e.g., n\_obs = 10), and use the rest for training the model.

6.**Fit the VAR model**:  
Initialize the VAR model with the training dataset and fit it to learn interdependencies among the variables.

7.**Forecast future values**:  
Forecast the next n\_obs steps using the trained VAR model.  
Convert the forecasted values into a DataFrame with the same structure as the original data.

8.**Visualize actual vs. forecasted values**:  
For each selected variable (e.g., Temperature, Humidity, WindSpeed), plot the actual and forecasted values on the same graph.

9.**Evaluate model performance**:  
Calculate the Root Mean Squared Error (RMSE) between actual and forecasted values for each variable.

10.**Display the RMSE values**.

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.api import VAR

from sklearn.metrics import mean\_squared\_error

from google.colab import files

# Upload file (if not done already)

uploaded = files.upload()

filename = next(iter(uploaded))

# Load dataset

df = pd.read\_csv(filename)

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

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

# Use 'Close' price (adjust if your target column is named differently)

df['Price'] = df['Close']

# Create lag and difference features

df['Price\_lag1'] = df['Price'].shift(1)

df['Price\_diff'] = df['Price'].diff()

# Prepare selected data

selected\_data = df[['Price', 'Price\_lag1', 'Price\_diff']].dropna()

selected\_data = selected\_data.fillna(method='ffill')

# Forecast horizon

n\_obs = 10

df\_train = selected\_data[:-n\_obs]

df\_test = selected\_data[-n\_obs:]

# Fit VAR model

model = VAR(df\_train)

model\_fit = model.fit()

# Forecast

forecast = model\_fit.forecast(df\_train.values[-model\_fit.k\_ar:], steps=n\_obs)

forecast\_df = pd.DataFrame(forecast, index=df\_test.index, columns=selected\_data.columns)

# Plot actual vs forecast

for column in selected\_data.columns:

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

plt.plot(df\_test[column], label='Actual')

plt.plot(forecast\_df[column], label='Forecasted')

plt.title(f'{column} Forecast vs Actual')

plt.xlabel('Date')

plt.ylabel(column)

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

# RMSE

for column in selected\_data.columns:

rmse = np.sqrt(mean\_squared\_error(df\_test[column], forecast\_df[column]))

print(f'RMSE for {column}: {rmse:.2f}')

**Output:**

A graph with a line and a line

AI-generated content may be incorrect.

**Result:**

Thus, the program to develop vector auto regression model for multivariate time series data forecasting was done.