EX:No.10 221501044

**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

# Load the dataset

try:

df = pd.read\_csv("/content/MSFT.csv", parse\_dates=['Date'], index\_col='Date')

except KeyError:

# If 'Date' column is not found, try using 'Month'

try:

df = pd.read\_csv("/content/MSFT.csv", parse\_dates=['Month'], index\_col='Month')

except KeyError:

print("Error: Neither 'Date' nor 'Month' column found in CSV for parsing dates.")

exit(1)

# Check if 'Close' column exists, otherwise print available columns

data\_column = None

if 'Close' in df.columns:

data\_column = 'Close' # Use 'Close' for Microsoft stock data

else:

print("Error: 'Close' column not found in CSV. Available columns:")

print(df.columns)

exit(1)

# Create synthetic multivariate dataset using lag features

# We'll use 'Close' as the main data column

df['Close\_lag1'] = df[data\_column].shift(1) # Lagged Close price

df['Close\_diff'] = df[data\_column].diff() # Difference in Close price

# Drop initial NaN values caused by shifting

selected\_data = df[[data\_column, 'Close\_lag1', 'Close\_diff']].dropna()

# Forward fill if needed (generally clean after diff/shift)

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)

# Plotting actual vs forecasted

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'Microsoft Stock Price Forecasting - {column}') # Title updated

plt.xlabel(df.index.name)

plt.ylabel(column)

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

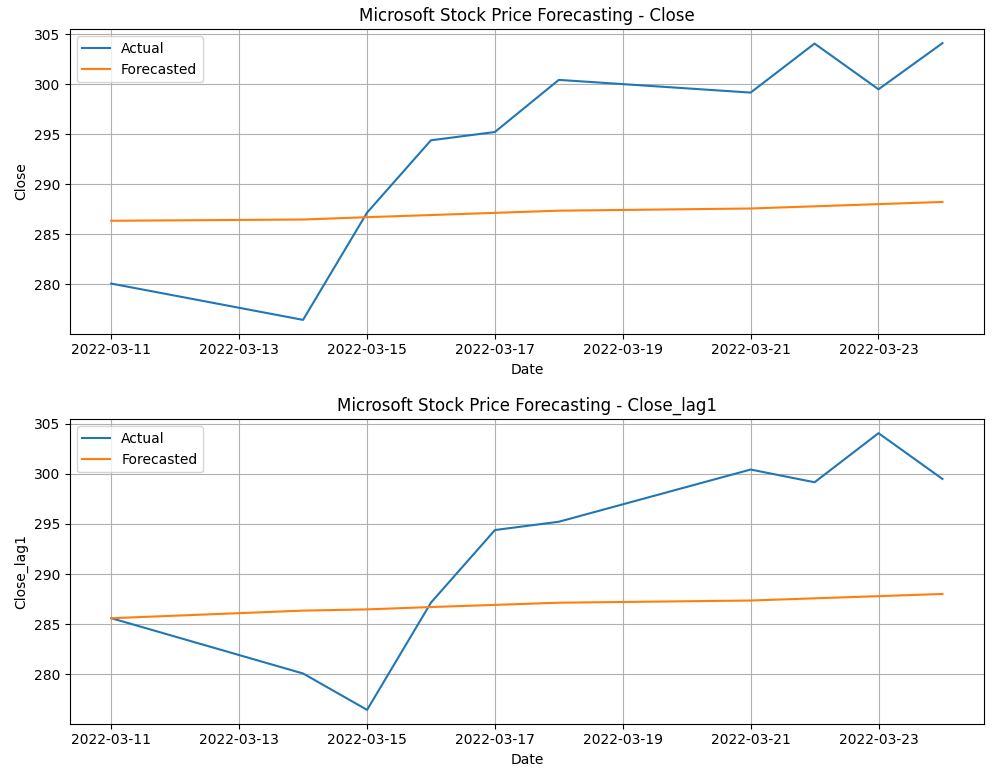
# Print RMSE

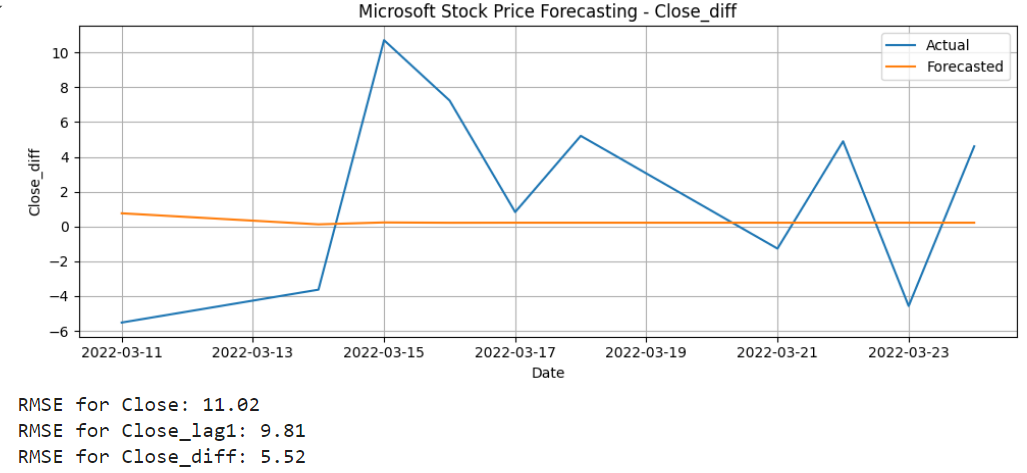
for column in selected\_data.columns:

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

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

**Output:**



****

**Result:**

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