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

**EX:No.10**

**DATE:12/04/25**

**AIM:**

To implement a Vector AutoRegression (VAR) model for forecasting multivariate time series data using AAPL stock data.

**ALGORITHM:**

1. Import necessary libraries and load the multivariate time series dataset (e.g., AAPL.csv).
2. Convert the 'Date' column to datetime format, set it as index, and select relevant features (e.g., Close, Volume).
3. Handle missing values and check for stationarity; apply differencing if necessary.
4. Split the dataset into training and testing sets.
5. Fit the VAR model on the training data and determine the optimal lag order.
6. Generate forecasts using the trained VAR model on the test data.
7. Reverse differencing (if applied), visualize the results, and evaluate the model using metrics like MAE and RMSE.

**CODE:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.api import VAR

from statsmodels.tools.eval\_measures import rmse, meanabs

# 1. Load dataset

data = pd.read\_csv('/content/AAPL.csv')

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

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

# 2. Select relevant features (multivariate)

df = data[['Close', 'Volume']].copy()

# 3. Handle missing values

df = df.fillna(method='ffill')

# 4. Split into train and test

n\_obs = 30  # number of observations for testing

train, test = df[:-n\_obs], df[-n\_obs:]

# 5. Check for stationarity - Difference the series

train\_diff = train.diff().dropna()

# 6. Fit VAR model

model = VAR(train\_diff)

lag\_order = model.select\_order(maxlags=15)

print("Selected Lag Order:\n", lag\_order.summary())

selected\_lag = lag\_order.aic  # choose based on AIC

var\_model = model.fit(selected\_lag)

# 7. Forecast

forecast\_input = train\_diff.values[-selected\_lag:]

forecast = var\_model.forecast(y=forecast\_input, steps=n\_obs)

# 8. Convert forecast to DataFrame and reverse differencing

forecast\_df = pd.DataFrame(forecast, index=test.index, columns=['Close', 'Volume'])

forecast\_cumsum = forecast\_df.cumsum()

last\_known = train.iloc[-1]

forecast\_values = forecast\_cumsum + last\_known

# 9. Evaluation

print("\n Evaluation Metrics (Close Price):")

print("MAE:", meanabs(test['Close'], forecast\_values['Close']))

print("RMSE:", rmse(test['Close'], forecast\_values['Close']))

# 10. Plotting

plt.figure(figsize=(12, 6))

plt.plot(train['Close'], label='Train Close')

plt.plot(test['Close'], label='Actual Close', color='green')

plt.plot(forecast\_values['Close'], label='Forecast Close', color='red')

plt.title('AAPL Close Price Forecast using VAR')

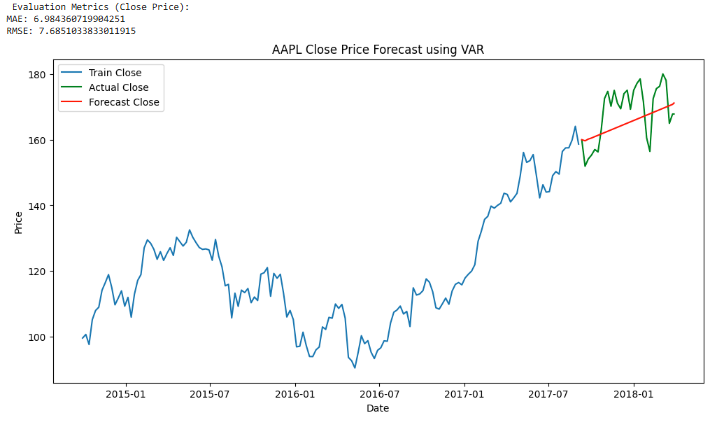
plt.xlabel('Date')

plt.ylabel('Price')

plt.legend()

plt.show()

**OUTPUT:**

****

**RESULT:**

Thus the program has been completed and verified successfully.