**EX:No.10 221501034**

**15/04/25**

**DEVELOP VECTOR AUTO REGRESSION MODEL FOR MULTIVARIATE TIME SERIES DATA FORECASTING.**

**AIM:**

To implement program for Develop neural network-based time series forecasting model.

**ALGORITHM:**

**OBJECTIVE:**

Smooth the electric production data to reduce noise, highlight trends, and prepare for forecasting.

**BACKGROUND:**

1.Time series data has short-term fluctuations.

2.Moving average reduces noise and clarifies trends.

3.Smoothed data improves forecast accuracy and interpretability.

**SCOPE OF THE PROGRAM:**

1.Load and clean dataset

2.Convert date column to datetime

3.Aggregate data monthly and yearly

4.Apply 3-month and 12-month moving averages

5.Plot original vs smoothed data

**ALGORITHM:**

1.Import libraries

2.Load dataset

3.Preprocess and set datetime index

4.Resample data (monthly, yearly)

5.Apply 3-month & 12-month smoothing

6.Visualize results

**PROCESS:**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from statsmodels.tsa.api import VAR**

**from sklearn.metrics import mean\_squared\_error**

**from statsmodels.tsa.stattools import adfuller**

**# Load dataset**

**df = pd.read\_csv('C:\\Users\\Hari\\Desktop\\Exp09\\AAPL.csv', parse\_dates=['Date'])**

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

**# Select relevant columns**

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

**# Rename columns for clarity**

**df.rename(columns={'Close': 'AAPL\_Close', 'Volume': 'AAPL\_Volume'}, inplace=True)**

**# ADF test function**

**def make\_stationary(data):**

**diffed = data.copy()**

**for col in data.columns:**

**result = adfuller(diffed[col])**

**if result[1] > 0.05:**

**print(f"{col} is non-stationary, differencing applied (p={result[1]:.4f})")**

**diffed[col] = diffed[col].diff()**

**return diffed.dropna()**

**# Make the data stationary**

**stationary\_df = make\_stationary(df)**

**# Split into train and test sets**

**n = int(len(stationary\_df) \* 0.8)**

**train = stationary\_df[:n]**

**test = stationary\_df[n:]**

**# Fit VAR model**

**model = VAR(train)**

**results = model.fit(maxlags=3, ic='aic')**

**# Forecast**

**lag\_order = results.k\_ar**

**forecast\_input = train.values[-lag\_order:]**

**forecast = results.forecast(y=forecast\_input, steps=len(test))**

**# Create forecast DataFrame**

**forecast\_df = pd.DataFrame(forecast, index=test.index, columns=['AAPL\_Close\_Pred', 'AAPL\_Volume\_Pred'])**

**# Plot forecast vs actual (Close price only)**

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

**plt.plot(test['AAPL\_Close'], label='Actual')**

**plt.plot(forecast\_df['AAPL\_Close\_Pred'], label='Forecast', color='red')**

**plt.title('AAPL Close Price - VAR Forecast (Differenced Series)')**

**plt.xlabel('Date')**

**plt.ylabel('Differenced Close Price')**

**plt.legend()**

**plt.grid(True)**

**plt.tight\_layout()**

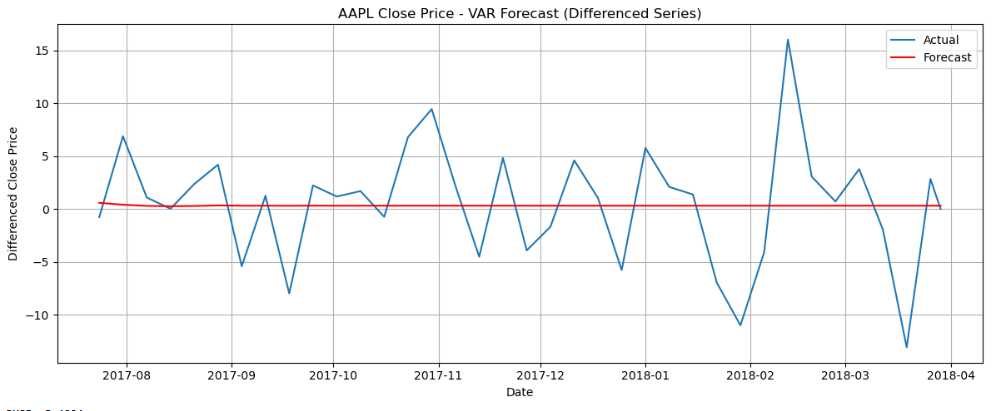
**plt.show()**

**# Evaluate RMSE**

**rmse = np.sqrt(mean\_squared\_error(test['AAPL\_Close'], forecast\_df['AAPL\_Close\_Pred']))**

**print(f'RMSE: {rmse:.4f}')**

**OUTPUT:**

****

**RESULT:**

The program to Develop neural network-based time series forecasting model created and executed successfully.