EX:No.10 DATE:12/04/25	Develop vector auto regression model for multivariate time series data forecasting.

AIM:

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

ALGORITHM:

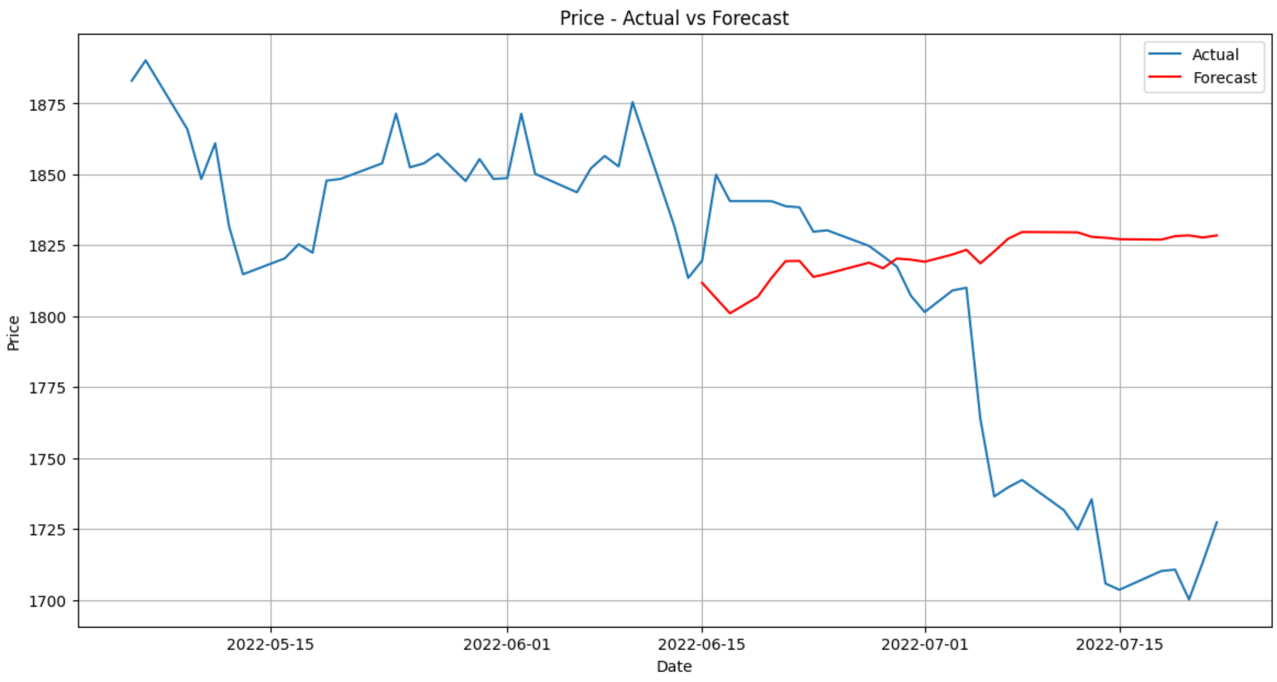
1. Load and preprocess data: Import the dataset, convert 'Year' and 'Month' to datetime, set it as index, and drop unnecessary columns.
2. Handle missing values: Fill any missing values using forward fill to maintain continuity.
3. Split dataset: Divide the data into training (80%) and testing (20%) sets.
4. Fit VAR model: Initialize the VAR model on training data, select optimal lag order (e.g., using AIC), and fit the model.
5. Forecast future values: Use the fitted model to forecast the same number of steps as in the test set.
6. Evaluate and visualize: Plot actual vs forecasted values for each variable to assess model performance.

Code:

```
 import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.api import VAR
from statsmodels.tsa.stattools import adfuller

# Load Dataset
df = pd.read_csv('path_to_your_file.csv', parse_dates=['Date'])
df.set_index('Date', inplace=True)

# Let's select multiple related columns (example: 'Price', 'Open', 'High', 'Low')
# Modify based on your dataset
data = df[['Price', 'Open', 'High', 'Low']]

# 1. ADF Test on each column - Check stationarity
def adf_test(series, name):
    result = adfuller(series.dropna())
    print(f'ADF Test for {name}')
    print(f'ADF Statistic: {result[0]}')
    print(f'p-value: {result[1]}')
    print('---')

for name in data.columns:
    adf_test(data[name], name)

# 2. Differencing if needed (if p-value > 0.05)
```

```
data_diff = data.diff().dropna()

# 3. Train/Test Split
n_obs = 30  # keeping last 30 observations for testing
train = data_diff[:-n_obs]
test = data_diff[-n_obs:]

# 4. Model Training
model = VAR(train)
lag_order = model.select_order()
print(lag_order.summary())

# Best lag order
best_lag = lag_order.selected_orders['aic']  # or bic/hqic
print(f"Selected best lag: {best_lag}")

var_model = model.fit(best_lag)
print(var_model.summary())

# 5. Forecasting
forecast_input = train.values[-best_lag:]
forecast = var_model.forecast(y=forecast_input, steps=n_obs)

# 6. Inverse the differencing to get real forecasted values
forecast_df = pd.DataFrame(forecast, index=test.index, columns=test.columns)

# Add the last known value to restore scale
def invert_transformation(train_data, forecast_data):
    inverted = forecast_data.copy()
    for col in train_data.columns:
        last_val = train_data[col].iloc[-1]
        inverted[col] = forecast_data[col].cumsum() + last_val
    return inverted

forecast_restored = invert_transformation(data.iloc[:-n_obs], forecast_df)

# 7. Visualization
for col in data.columns:
    plt.figure(figsize=(14,7))
    plt.plot(data.index[-2*n_obs:], data[col].iloc[-2*n_obs:], label='Actual')
    plt.plot(forecast_restored.index, forecast_restored[col], label='Forecast', color='red')
    plt.title(f'{col} - Actual vs Forecast')
    plt.xlabel('Date')
    plt.ylabel(col)
    plt.legend()
    plt.grid()
    plt.show()
```

OUTPUT:



RESULT:
Thus, the program to execute the vector auto regression model for multivariate time series data forecasting is completed successfully