EX:No.10 **DATE:12/04/2**	**Develop vector auto regression model for multivariate time series data forecasting**

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]
```

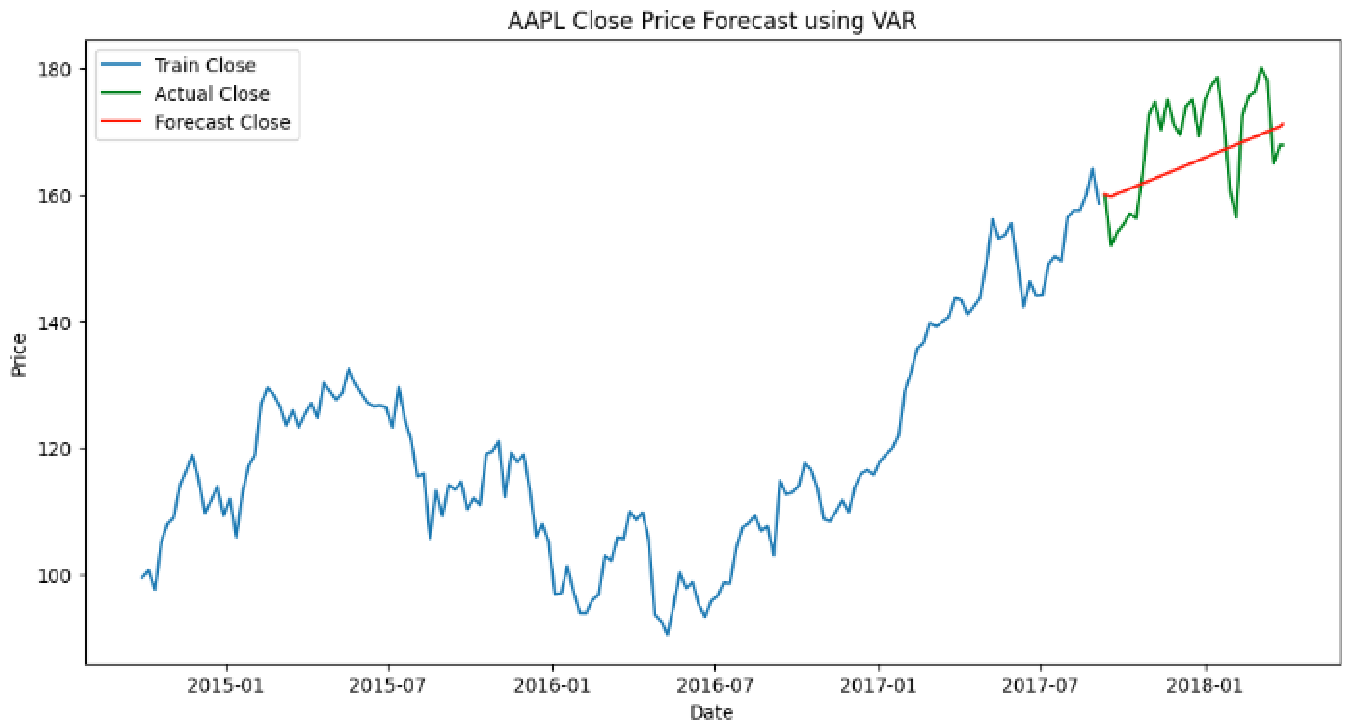
```
forecast_values = forecast_cumsum + last_known

# 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:

```
 Evaluation Metrics (Close Price):
MAE: 6.984360719904251
RMSE: 7.685103383301915
```



RESULT:

Thus the program has been completed and verified successfully.