| **EXP:9**  **17/04/2025** | **Develop vector auto regression model for multivariate time series data forecasting.** |
| --- | --- |

**AIM:**

To develop a Vector Auto Regression (VAR) model for forecasting multiple interdependent time series variables—such as car sales, price, engine size, and horsepower—using historical data.

**PROCEDURE:**

**1) Import Necessary Libraries**

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

**2)** **Load and Preprocess the Data**

**df = pd.read\_csv("Car\_sales.csv")**

**df['Latest\_Launch'] = pd.to\_datetime(df['Latest\_Launch'], errors='coerce')**

**df = df.dropna(subset=['Latest\_Launch'])**

**df = df[['Latest\_Launch', 'Sales\_in\_thousands', 'Price\_in\_thousands', 'Engine\_size', 'Horsepower']]**

**df = df.dropna()**

**df = df.groupby('Latest\_Launch').mean().sort\_index()**

**3)Check Stationarity and Difference the Data**

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

**4)Split Dataset into Train and Test Sets**

n\_obs = 10

train = df\_diff[:-n\_obs]

test = df\_diff[-n\_obs:]

**5)Fit the VAR Model**

model = VAR(train)

results = model.fit(maxlags=15, ic='aic')

**6) Inverse Differencing for Actual Values**

last\_values = df.iloc[-n\_obs - 1:-1]

forecast\_values = forecast\_df.cumsum() + last\_values.values

actual = df[-n\_obs:]

predicted = pd.DataFrame(forecast\_values, index=test.index, columns=test.columns)

**7)Forecasting**

forecast\_input = train.values[-results.k\_ar:]

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

forecast\_df = pd.DataFrame(forecast, index=test.index, columns=test.columns)

### **8)Visualize Results**

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

for col in df.columns:

plt.plot(actual.index, actual[col], label=f'Actual {col}')

plt.plot(predicted.index, predicted[col], linestyle='--', label=f'Forecast {col}')

plt.legend()

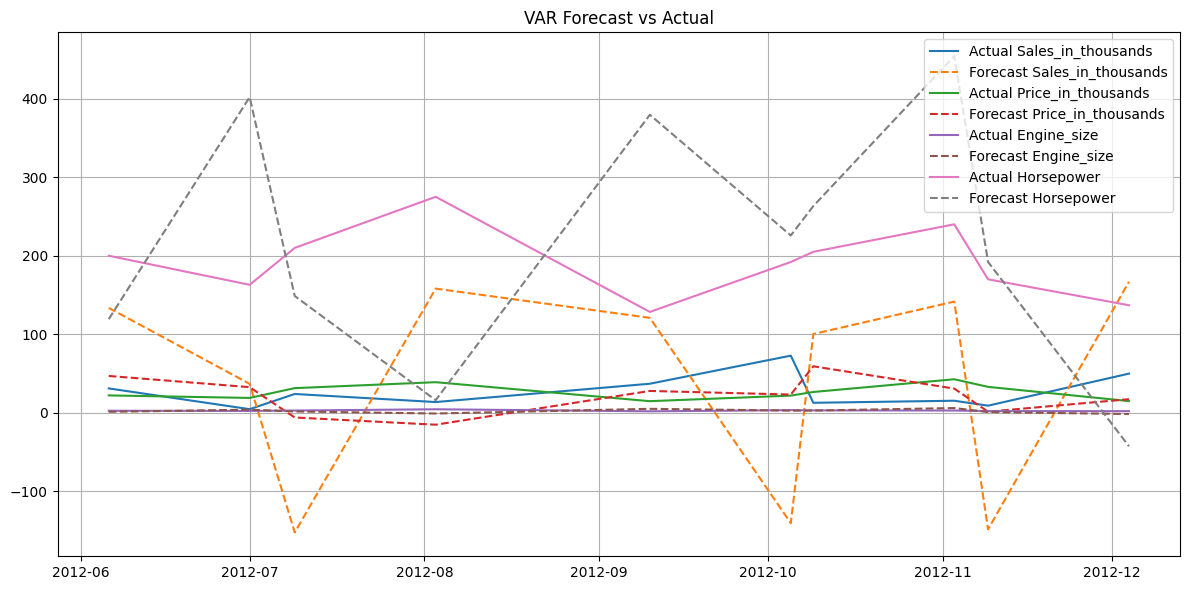
plt.title('VAR Forecast vs Actual')

plt.grid(True)

plt.tight\_layout()

plt.show()

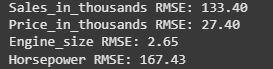
**OUTPUT:**

****

### **9)Calculate RMSE for Each Variable\**

for col in df.columns:

print(f'{col} RMSE: {rmse(actual[col], predicted[col]):.2f}')

**OUTPUT**  


**RESULT:**

Thus the program has been executed successfully