**EX NO :10 DEVELOP VECTOR AUTO REGRESSION MODEL FOR**

**DATE MULTIVARIATE TIME SERIES DATA FORECASTING**

**AIM:**

To develop a Vector AutoRegression (VAR) model for forecasting airline passenger numbers using multivariate time series data derived from lag features.

**ALGORITHM:**

1. Import required libraries and load the dataset.
2. Convert 'Month' to datetime and set as index.
3. Create a lag feature to make the data multivariate.
4. Drop missing values and check stationarity using ADF test.
5. Apply differencing to achieve stationarity.
6. Fit the VAR model and select optimal lag using AIC.
7. Forecast the next 12 months.
8. Convert predictions to original scale using cumulative sum.
9. Plot actual vs. forecasted passenger data.

**PROGRAM:**

# Install required packages

!pip install pandas matplotlib statsmodels --quiet

# Import libraries

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.api import VAR

from statsmodels.tsa.stattools import adfuller

import warnings

warnings.filterwarnings("ignore")

# Load the dataset

df = pd.read\_csv('/content/airline-passengers (1).csv')

# Parse datetime and set index

df['Month'] = pd.to\_datetime(df['Month'])

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

df.columns = ['Passengers']

# Create one lag feature to make it multivariate

df['Lag1'] = df['Passengers'].shift(1)

df.dropna(inplace=True)

# ADF Test

def adf\_test(series, name=''):

result = adfuller(series, autolag='AIC')

print(f"{name} ADF p-value: {result[1]}")

return result[1]

print("ADF Test Results (Before Differencing):")

for col in df.columns:

adf\_test(df[col], col)

# First-order differencing for stationarity

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

# ADF Test After Differencing

print("\nADF Test Results (After Differencing):")

for col in df\_diff.columns:

adf\_test(df\_diff[col], col)

# Train VAR model

model = VAR(df\_diff)

# Use a conservative lag value to avoid overfitting

results = model.fit(maxlags=5, ic='aic')

# Forecast next 12 months

lag\_order = results.k\_ar

forecast\_input = df\_diff.values[-lag\_order:]

forecast = results.forecast(y=forecast\_input, steps=12)

# Convert forecast to DataFrame

forecast\_df = pd.DataFrame(forecast,

index=pd.date\_range(start=df.index[-1] + pd.DateOffset(months=1), periods=12, freq='MS'),

columns=df\_diff.columns)

# Reconstruct original scale

last\_row = df.iloc[-1]

forecast\_actual = forecast\_df.cumsum() + last\_row

# Plot results

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

plt.plot(df['Passengers'], label='Original')

plt.plot(forecast\_actual['Passengers'], label='Forecasted', linestyle='--')

plt.title('VAR Model - Passenger Forecast')

plt.xlabel('Date')

plt.ylabel('Passengers')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

**OUTPUT**

ADF Test Results (Before Differencing):

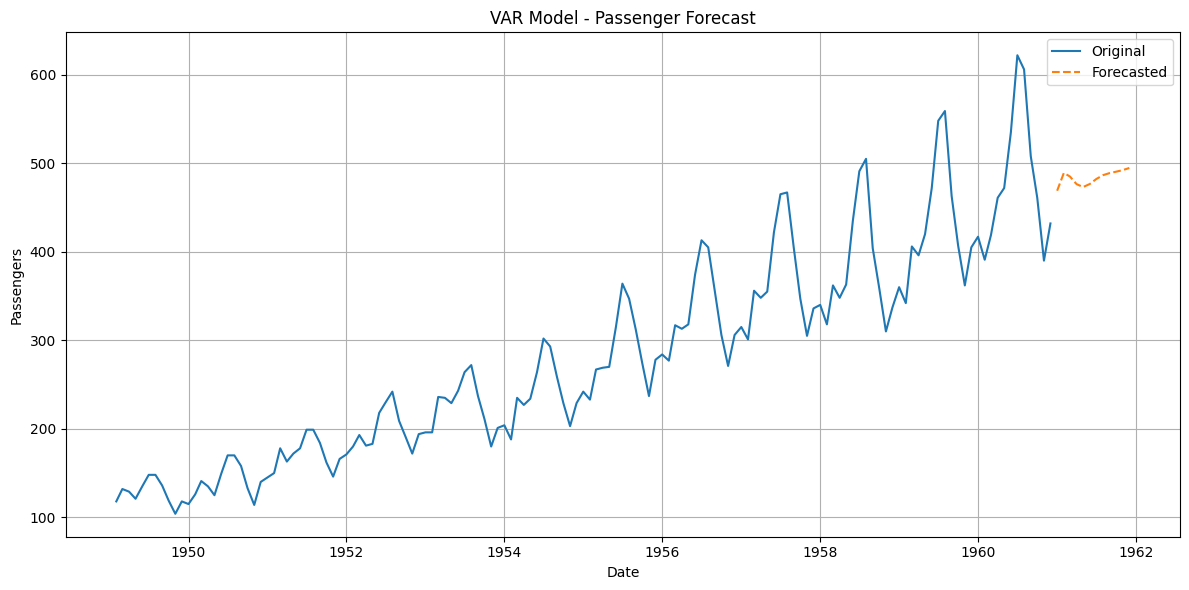
Passengers ADF p-value: 0.991511509628866

Lag1 ADF p-value: 0.9925254163586111

ADF Test Results (After Differencing):

Passengers ADF p-value: 0.053655499158908174

Lag1 ADF p-value: 0.05913488643940043

****

**RESULT:**

The VAR model successfully forecasted the next 12 months of airline passenger numbers using multivariate time series data.