EX:No.10 **2215010148**

22/04/25

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

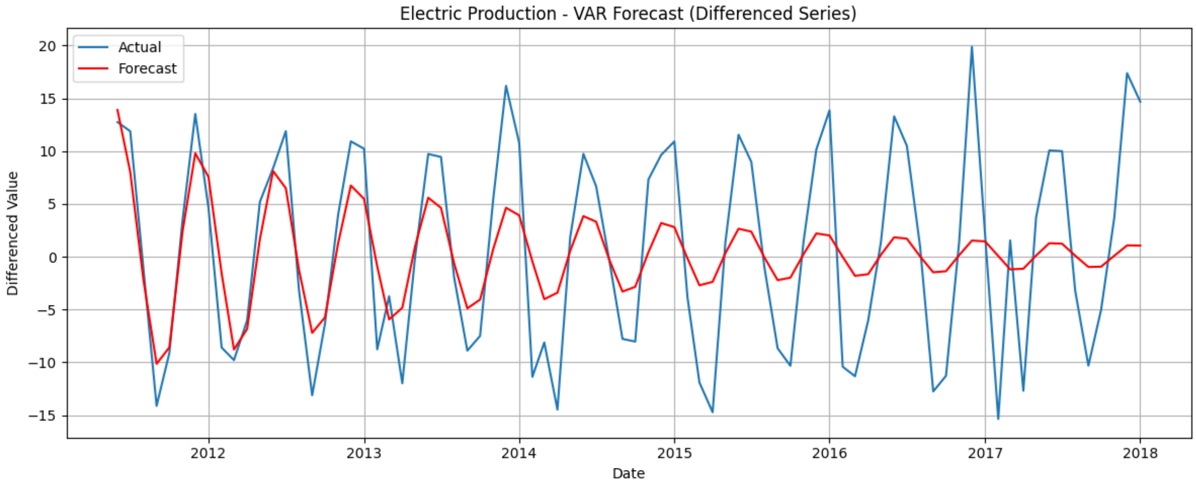
PROCESS:

```
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('/content/gold_price_dataset.csv', parse_dates=['DATE'])
df.set_index('DATE', inplace=True)

# Rename for easier access
df.rename(columns={'IPG2211A2N': 'Electric_Production'}, inplace=True)

# Add dummy second variable (simulated temperature trend)
df['Dummy_Temp'] = df['Electric_Production'].rolling(window=3, min_periods=1).mean()

# Drop NA caused by rolling
df.dropna(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 both series stationary
stationary_df = make_stationary(df)

# Split into train and test
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))

# Forecast DataFrame
forecast_df = pd.DataFrame(forecast, index=test.index, columns=['Electric_Production_Pred', 'Dummy_Temp_Pred'])

# Plot results
plt.figure(figsize=(12,5))
plt.plot(test['Electric_Production'], label='Actual')
plt.plot(forecast_df['Electric_Production_Pred'], label='Forecast', color='red')
plt.title('Electric Production - VAR Forecast (Differenced Series)')
plt.xlabel('Date')
plt.ylabel('Differenced Value')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# RMSE Evaluation
rmse = np.sqrt(mean_squared_error(test['Electric_Production'], forecast_df['Electric_Production_Pred']))
print(f'RMSE: {rmse:.4f}')
```

OUTPUT:



\

RESULT:

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