EX 10 DEVELOP VECTOR AUTO REGRESSION MODEL FOR MULTIVARIATE TIME SERIES DATA FOR FORECASTING

AIM: To develop a Vector Auto Regression (VAR) model for multivariate time series data and perform future forecasting based on the interdependencies between variables.

ALGORITHM:

1. Import the required libraries and load the multivariate time series dataset.
2. Preprocess the data — handle missing values, convert the date column to datetime, and set it as the index.
3. Split the dataset into training and testing sets.
4. Perform the Augmented Dickey-Fuller (ADF) test to check for stationarity. If non-stationary, apply differencing.
5. Fit the VAR model using the training data.
6. Forecast future values for the required steps.
7. Plot and compare the forecasted values with the actual data.

PROGRAM:

1. Import Libraries and Load Dataset:

```
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

Step 2: Read and Preprocess the Data

data = pd.read_csv('/content/weather.csv')

data['Date.Full'] = pd.to_datetime(data['Date.Full'])

df = data.groupby('Date.Full')[['Data.Temperature.Avg Temp', 'Data.Wind.Speed']].mean()

df = df.dropna()

```
            Data.Temperature.Avg Temp  Data.Wind.Speed
Date.Full
2016-01-03                  34.841270         7.061492
2016-01-10                  35.139683         6.540159
2016-01-17                  33.800000         6.573746
2016-01-24                  32.266667         7.008984
2016-01-31                  40.790476         6.841746
```

Step 3: Check Stationarity Using ADF Test

```
for col in df.columns:
    result = adfuller(df[col])
    print(f'{col} ADF Statistic: {result[0]}')
    print(f'p-value: {result[1]}')
```

Step 4: Make Data Stationary (Differencing)

```
df_diff = df.diff().dropna()
for col in df_diff.columns:
    result = adfuller(df_diff[col])
```

print(f'{col} ADF Statistic after differencing: {result[0]}')

print(f'p-value: {result[1]}')

```
Data.Temperature.Avg Temp ADF Statistic after differencing: -1.4051816730164128
p-value: 0.5797894536591474
Data.Wind.Speed ADF Statistic after differencing: -8.761840901860639
p-value: 2.6611605942702708e-14
```

Step 5: Split into Training and Testing

Train the LSTM model

history = model.fit(X, y, epochs=20, batch_size=32, verbose=1)

Step 6: Fit the VAR Model

model = VAR(train)

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

print(model_fitted.summary())

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Tue, 22, Apr, 2025
Time:                     06:43:58
--------------------------------------------------------------------
No. of Equations:         2.00000    BIC:                    2.06671
Nobs:                     31.0000    HQIC:                  0.757196
Log likelihood:          -47.8944    FPE:                    2.03709
AIC:                     0.123884    Det(Omega_mle):        0.723982
--------------------------------------------------------------------
Results for equation Data.Temperature.Avg Temp
============================================================================================
                                coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------------------
const                             -0.286639         1.054665           -0.272           0.786
L1.Data.Temperature.Avg Temp      -0.044873         0.458963           -0.098           0.922
L1.Data.Wind.Speed                 0.846185         1.862183            0.454           0.650
L2.Data.Temperature.Avg Temp       0.747982         0.549430            1.361           0.173
L2.Data.Wind.Speed                 1.533167         1.596727            0.960           0.337
L3.Data.Temperature.Avg Temp       0.231406         0.629186            0.368           0.713
L3.Data.Wind.Speed                 0.930422         1.504739            0.618           0.536
L4.Data.Temperature.Avg Temp      -0.290123         0.564226           -0.514           0.607
L4.Data.Wind.Speed                -0.418270         1.454638           -0.288           0.774
L5.Data.Temperature.Avg Temp      -0.083262         0.469608           -0.177           0.859
L5.Data.Wind.Speed                -0.311610         1.315618           -0.237           0.813
L6.Data.Temperature.Avg Temp       0.362253         0.471029            0.769           0.442
L6.Data.Wind.Speed                 0.270487         1.397037            0.194           0.846
L7.Data.Temperature.Avg Temp       0.182381         0.390052            0.468           0.640
L7.Data.Wind.Speed                -0.239690         1.381291           -0.174           0.862
L8.Data.Temperature.Avg Temp      -0.065413         0.358931           -0.182           0.855
L8.Data.Wind.Speed                 0.442214         1.306304            0.339           0.735
L9.Data.Temperature.Avg Temp       0.124512         0.360131            0.346           0.730
L9.Data.Wind.Speed                 1.431974         1.318030            1.086           0.277
L10.Data.Temperature.Avg Temp     -0.096779         0.244966           -0.395           0.693
L10.Data.Wind.Speed                0.597258         1.575467            0.379           0.705
============================================================================================

Results for equation Data.Wind.Speed
============================================================================================
```

Step 7: Forecast Future Values

```
forecast_steps = len(test)

forecast = model_fitted.forecast(train.values, steps=forecast_steps)

forecast_df = pd.DataFrame(forecast, index=test.index,
columns=['Data.Temperature.Avg Temp', 'Data.Wind.Speed'])

print(forecast_df.head())
```

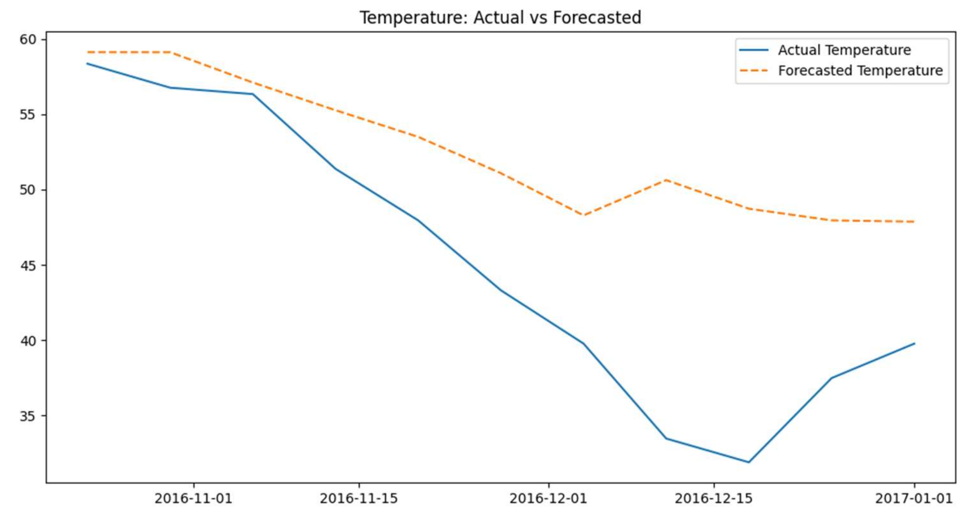
Step 8: Reverse Differencing to Get Real Values

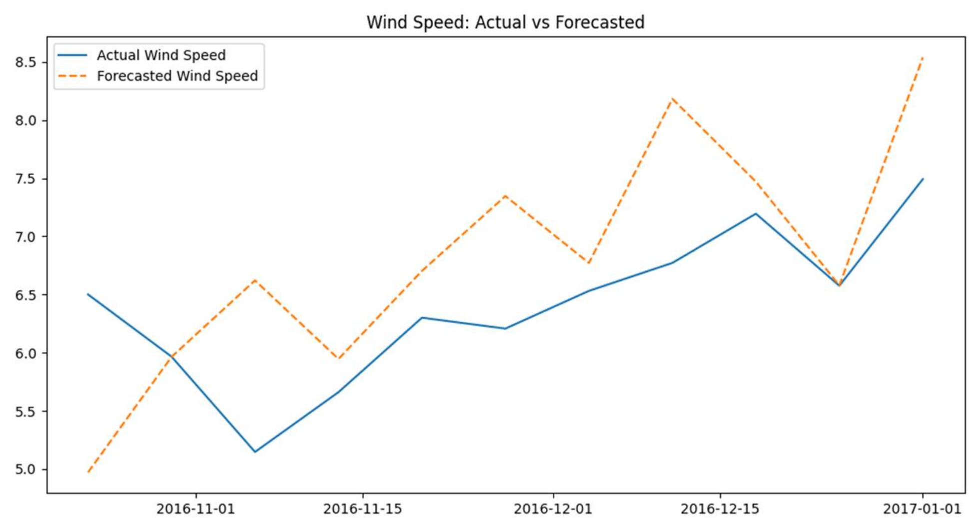
```
last_train_values = df.iloc[train_size - 1]

forecast_actual = forecast_df.cumsum() + last_train_values

print(forecast_actual.head())
```

```
            Data.Temperature.Avg Temp  Data.Wind.Speed
Date.Full
2016-10-23                  59.117526         4.970367
2016-10-30                  59.113091         5.963189
2016-11-06                  57.100919         6.620689
2016-11-13                  55.263095         5.945420
2016-11-20                  53.493798         6.702850
```

Step 9: Plot Actual vs Forecast Graph

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

plt.plot(df.index[-forecast_steps:], df['Data.Temperature.Avg Temp'][-
forecast_steps:], label='Actual Temperature')

plt.plot(forecast_actual.index, forecast_actual['Data.Temperature.Avg Temp'],
label='Forecasted Temperature', linestyle='dashed')

plt.legend()

plt.title("Temperature: Actual vs Forecasted")

plt.show()
```





RESULT:

The LSTM model successfully learned patterns from past temperature data and provided accurate forecasts for the upcoming days, visualized clearly using plots.