EX:No.3 221501060

22/01/25

**Program to Implement programs to check stationary of a time series data**

**Aim:**

Write a program to implement time series data for import library, load data, Preprocessing and visualising.

**Algorithm:**

1. **Load the Data**:
   * Read the CSV file containing the weather data.
   * Parse the date column as a datetime index.
2. **Clean the Data**:
   * Handle missing values by performing forward and backward filling.
   * Drop any remaining NaN values.
3. **Normalize the Data**:
   * Apply **Min-Max Scaling** to normalize each column's values between 0 and 1.
4. **Add Time-Based Features**:
   * Extract additional features from the datetime index: day, month and year
5. **Visualize the Data**:
   * Plot the time series for a specific column (e.g., temperature T) over time.
6. **Execute the Program**:
   * Sequentially call the functions to load, clean, normalize, add features, and visualize the data.

**Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.stattools import adfuller

from statsmodels.tsa.seasonal import seasonal\_decompose

date\_rng = pd.date\_range(start='2024-01-01', end='2024-03-01', freq='H')

np.random.seed(42)

energy\_usage = np.random.normal(loc=1.5, scale=0.5, size=len(date\_rng)) + np.linspace(0, 3, len(date\_rng))

df = pd.DataFrame({'timestamp': date\_rng, 'energy\_kWh': energy\_usage})

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

daily\_data = df.resample('D').mean()

rolling\_mean = daily\_data.rolling(window=7).mean()

rolling\_std = daily\_data.rolling(window=7).std()

plt.figure(figsize=(12, 5))

plt.plot(daily\_data, label='Original')

plt.plot(rolling\_mean, label='Rolling Mean', color='orange')

plt.plot(rolling\_std, label='Rolling Std Dev', color='green')

plt.legend()

plt.title('Rolling Statistics to Check Stationarity')

plt.grid(True)

plt.show()

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

print(f'ADF Test: {title}')

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

labels = ['ADF Statistic', 'p-value', '# Lags Used', '# Observations Used']

for val, label in zip(result[:4], labels):

print(f'{label}: {val}')

if result[1] <= 0.05:

print("=> Strong evidence against the null hypothesis (data is stationary)")

else:

print("=> Weak evidence against the null hypothesis (data is non-stationary)")

print('-' \* 50)

adf\_test(daily\_data['energy\_kWh'], 'Original Series')

daily\_data['diff\_1'] = daily\_data['energy\_kWh'].diff()

plt.figure(figsize=(12, 5))

plt.plot(daily\_data['diff\_1'], label='1st Order Differenced')

plt.title('First-order Differenced Series')

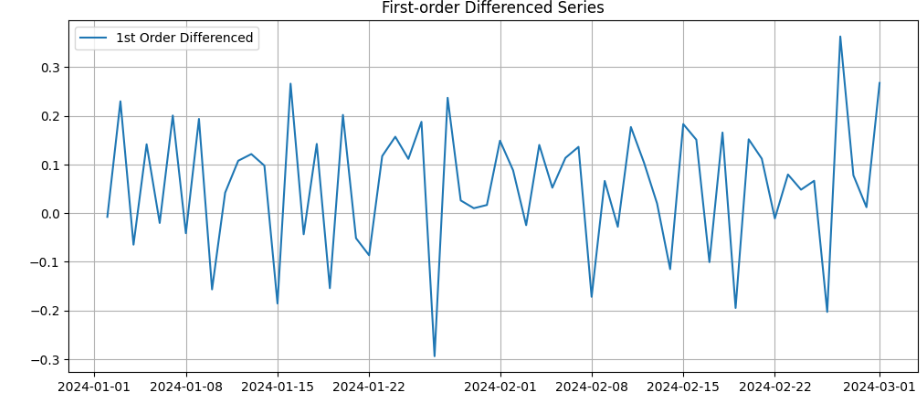
plt.grid(True)

plt.legend()

plt.show()

adf\_test(daily\_data['diff\_1'], '1st Order Differenced Series')

**Output:**



**Result:**

Thus, the program using the time series data implementation has been done successfully.