EX:No.2 221501043

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 matplotlib.pyplot as plt

from statsmodels.tsa.stattools import adfuller

# Load the dataset

file\_path = "E:/221501043/daily-minimum-temperatures-in-me.csv"

df = pd.read\_csv(file\_path)

# Convert 'Date' column to datetime format and set it as index

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

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

# Rename the temperature column for easier access

df.rename(columns={'Daily minimum temperatures': 'Temperature'}, inplace=True)

# Convert Temperature column to numeric, forcing errors to NaN and then dropping them

df['Temperature'] = pd.to\_numeric(df['Temperature'], errors='coerce')

df.dropna(inplace=True)

# Perform Augmented Dickey-Fuller test

adf\_test = adfuller(df['Temperature'])

adf\_result = {

"ADF Statistic": adf\_test[0],

"p-value": adf\_test[1],

"Critical Values": adf\_test[4]

}

# Print ADF test results

print("ADF Statistic:", adf\_result["ADF Statistic"])

print("p-value:", adf\_result["p-value"])

for key, value in adf\_result["Critical Values"].items():

print(f"Critical Value ({key}): {value}")

# Plot scatter graph of time series data

plt.figure(figsize=(10, 5))

plt.scatter(df.index, df['Temperature'], s=10, alpha=0.5)

plt.xlabel('Date')

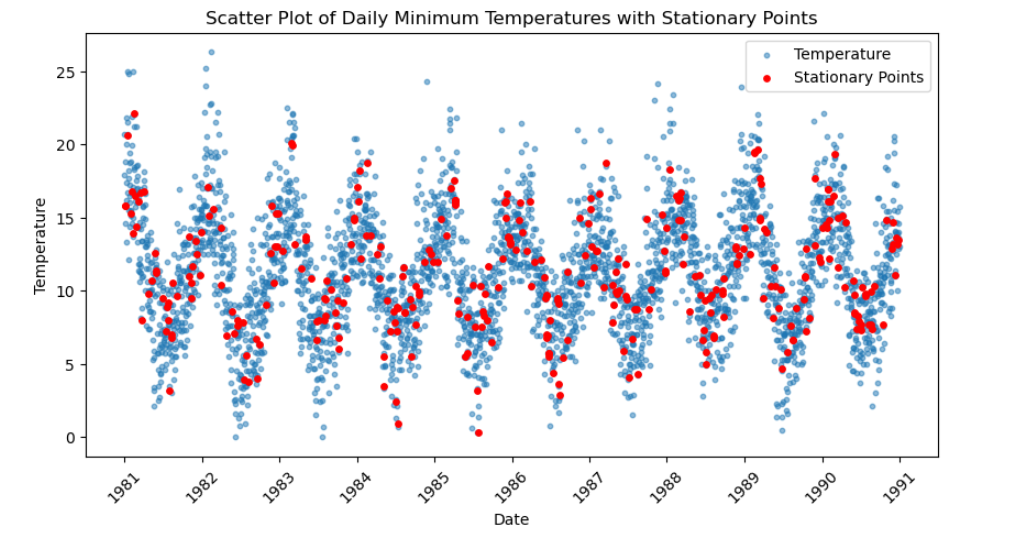
plt.ylabel('Temperature')

plt.title('Scatter Plot of Daily Minimum Temperatures')

plt.xticks(rotation=45)

plt.show()

**Output:**



**Result:**

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