# **Time Series Stationarity Check**

### **Aim**

To analyze and determine the stationarity of a given time series using the Augmented Dickey-Fuller (ADF) test and visualize its properties through rolling statistics and autocorrelation plots.

### **Algorithm**

1. Import necessary libraries.
2. Read the time series data from a CSV file.
3. Apply the Augmented Dickey-Fuller (ADF) test.
4. Calculate and plot rolling mean and rolling standard deviation.
5. Plot the Autocorrelation Function (ACF).
6. Determine whether the time series is stationary based on the p-value.
7. Display results and visualization.

## **Prerequisites**

Ensure the following Python libraries are installed:

pip install pandas numpy matplotlib statsmodels

## **Code Explanation**

### **Importing Required Libraries**

****import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.stattools import adfuller

* pandas: For handling time series data.
* numpy: For numerical operations.
* matplotlib.pyplot: For visualization.
* adfuller from statsmodels.tsa.stattools: Performs the Augmented Dickey-Fuller test to check stationarity.

### **Function: check\_stationarity**

****def check\_stationarity(timeseries):

result = adfuller(timeseries)

p\_value = result[1]

critical\_values = result[4]

if p\_value < 0.05:

stationary = True

status = "Time Series is Stationary"

else:

stationary = False

status = "Time Series is Non-Stationary"

* This function applies the ADF test to determine if the time series is stationary.
* If the p-value is below 0.05, the null hypothesis (non-stationarity) is rejected, and the series is considered stationary.

### **Visualization**

**** fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(12, 8))

timeseries.plot(ax=ax1, title="Original Time Series", color='blue')

ax1.set\_ylabel("Value")

roll\_mean = timeseries.rolling(window=12).mean()

roll\_std = timeseries.rolling(window=12).std()

timeseries.plot(ax=ax2, label='Original', color='blue')

roll\_mean.plot(ax=ax2, label='Rolling Mean', color='red')

roll\_std.plot(ax=ax2, label='Rolling Std', color='black')

ax2.set\_ylabel("Value")

ax2.legend(loc='best')

ax2.set\_title("Rolling Mean & Standard Deviation")

* The function generates three subplots:
  + The first plot shows the original time series.
  + The second plot displays the rolling mean and rolling standard deviation to visually assess stationarity.

### **Autocorrelation Function (ACF)**

**** try:

from statsmodels.graphics.tsaplots import plot\_acf

plot\_acf(timeseries, ax=ax3, lags=20)

ax3.set\_title("Autocorrelation Function (ACF)")

plt.tight\_layout()

plt.show()

except ImportError:

print("Warning: statsmodels not installed. ACF plot not generated.")

* The third plot represents the autocorrelation function, which helps analyze dependencies in the time series.

### **Reading and Processing Data**

****data = pd.read\_csv('/content/Microsoft\_Stock.csv', parse\_dates=['Date'], index\_col='Date')

timeseries = data['Close']

is\_stationary, p\_value, status, fig = check\_stationarity(timeseries)

* The CSV file containing stock prices is read using pandas.read\_csv.
* The Close price column is extracted as a time series.
* The check\_stationarity function is called to analyze the time series.

### **Output Results**

****print(f"Stationarity: {is\_stationary}")

print(f"P-value: {p\_value}")

print(f"Status: {status}")

plt.show()

* The function returns:
  + is\_stationary: Boolean indicating stationarity.
  + p\_value: The test's significance level.
  + status: A textual summary of the stationarity check.
* The results are printed to the console.

## **Result**

This script provides a comprehensive method to check the stationarity of time series data using statistical and visual techniques. If the time series is non-stationary, techniques such as differencing or transformation may be required before further analysis.