Exp No: 5 **Implement programs to check stationary of a time series data**

Date:21/3/25

Objective:

The objective of this implementation is to check whether a given time series dataset is stationary or not. Stationarity is an essential property for many time series models, such as ARIMA, as they assume that the statistical properties (mean, variance, and autocorrelation) remain constant over time.

Background & Scope:

Stationarity in time series analysis ensures that past observations can be used effectively to predict future values. A non-stationary series can be transformed into a stationary one using techniques like differencing or logarithmic transformations.

Two key tests used for stationarity assessment:

1. Augmented Dickey-Fuller (ADF) Test: Tests for the presence of a unit root, indicating whether a series is stationary.
2. Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test: Checks whether a time series is trend-stationary or needs transformation.

This implementation applies stationarity tests on an economic dataset and performs transformations to achieve stationarity if needed.

Implementation Steps:

1. **Import Required Libraries**

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import warnings

from statsmodels.tsa.stattools import adfuller, kpss
```

2. Load and Preprocess Data

```
# Ignore warnings to prevent InterpolationWarning

warnings.filterwarnings("ignore")

# Load the dataset

file_path = "AirPassengers.csv"  # Update path if needed

df = pd.read_csv(file_path)
```

```
# Convert DATE column to datetime and set as index
df['Month'] = pd.to_datetime(df['Month'])
df.set_index('DATE', inplace=True)
# Ensure there are no missing values
df.dropna(inplace=True)
```

2. **Function to Perform Stationarity Tests**

```
def stationarity_tests(series):
    print("\nPerforming Stationarity Tests...\n")
    # ADF Test (Augmented Dickey-Fuller)
    adf_test = adfuller(series, autolag='AIC')
    print(f"ADF Test:\nTest Statistic: {adf_test[0]}\np-value: {adf_test[1]}\n")
    # KPSS Test (Kwiatkowski-Phillips-Schmidt-Shin)
    try:
        kpss_test = kpss(series, regression='c', nlags='auto')  # Used 'auto' to avoid warnings
        print(f"KPSS Test:\nTest Statistic: {kpss_test[0]}\np-value: {kpss_test[1]}\n")
    except ValueError as e:
        print(f"KPSS Test could not be performed: {e}\n")
    except Exception as e:
        print(f"An unexpected error occurred during the KPSS test: {e}\n")
```

4. Apply Tests on Original Data

```
print(" Stationarity Test for Original Data:")
stationarity_tests(df['#Passengers'])
```

5. Differencing to Remove Trend

```
df['Value_Diff'] = df['#Passengers'].diff()
df.dropna(inplace=True)  # Drop NA after differencing
print("\n Stationarity Test after Differencing:")
stationarity_tests(df['Value_Diff'])
```

6. Log Differencing for Variance Stabilization

```
df['Log_Value'] = np.log(df['#Passengers'])
df['Log_Value_Diff'] = df['Log_Value'].diff()
```

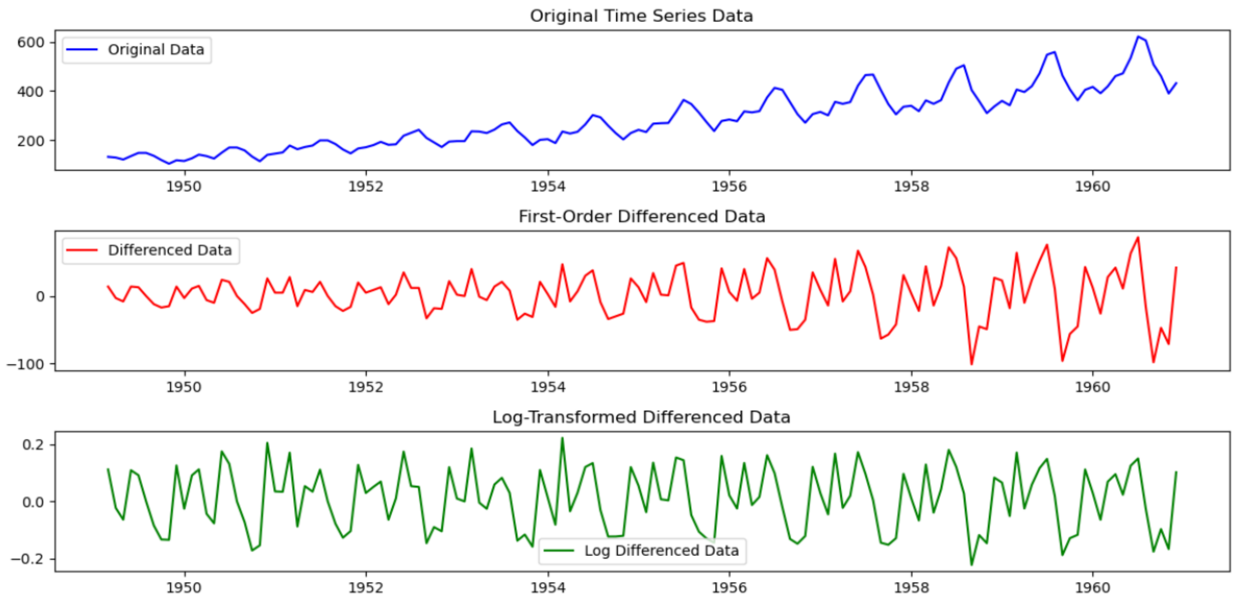
```
df.dropna(inplace=True)  # Drop NA after log differencing

print("\n Stationarity Test after Log Differencing:")
stationarity_tests(df['Log_Value_Diff'])
```

7. Visualizing Data Transformations

```
plt.figure(figsize=(12, 6))
plt.subplot(3, 1, 1)
plt.plot(df['#Passengers'], label="Original Data", color='blue')
plt.title("Original Time Series Data")
plt.legend()
plt.subplot(3, 1, 2)
plt.plot(df['Value_Diff'], label="Differenced Data", color='red')
plt.title("First-Order Differenced Data")
plt.legend()
plt.subplot(3, 1, 3)
plt.plot(df['Log_Value_Diff'], label="Log Differenced Data", color='green')
plt.title("Log-Transformed Differenced Data")
plt.legend()
plt.tight_layout()
plt.show()
```

Output :



Conclusion:

This implementation successfully checks the stationarity of a time series dataset using ADF and KPSS tests. If the data is found to be non-stationary, differencing and log transformations are applied to make it stationary. The final plots visually demonstrate the effectiveness of these transformations. This ensures that the dataset is ready for further time series modeling, such as ARIMA forecasting.