Decoding and Explanation of a Python Function for Time Series Stationarity Analysis Using the Augmented Dickey-Fuller Test

---

1. Purpose of the Function:

The `adf_test` function is designed to:

- Test the stationarity of a time series dataset using the Augmented Dickey-Fuller (ADF) test.
- Visualize the time series and save the plot with stationarity information.
- Output a message indicating whether the series is stationary or non-stationary based on the test results.

Stationarity is crucial in time series analysis, as many forecasting models (e.g., ARIMA) require the data to be stationary.

---

2. Conditions for Decision-Making:

The function evaluates the stationarity based on:

1. **Test Statistic:** It compares the test statistic to critical values at 1%, 5%, and 10% significance levels.
    - If the test statistic is smaller than any of these critical values, there is evidence to reject the null hypothesis.
2. **p-value:** If the p-value is less than 0.05, it indicates stationarity.

Decision Logic:

- **Stationary:**
    - Test statistic < any critical value **and** p-value < 0.05.
    - Reject the null hypothesis (data is stationary).
- **Non-stationary:**
    - If the above condition fails, the null hypothesis is accepted (data is non-stationary).

---

3. Code Explanation:

a. Imports and Parameters:

```
python
Copy code
from statsmodels.tsa.stattools import adfuller
```

- `adfuller`: Function from `statsmodels` to perform the ADF test.

```
python
Copy code
def adf_test(timeseries, df, pollutant):
```

- `timeseries`: Time series data (e.g., a pandas Series).
- `df`: A DataFrame containing the full dataset.
- `pollutant`: A specific column in `df` being analyzed.

b. ADF Test Execution:

```
python
Copy code
    dftest = adfuller(timeseries, autolag='AIC')
```

- Runs the ADF test on the `timeseries`.
- **Returns:** A tuple:
    - Test Statistic: A measure to compare against critical values.
    - p-value: Probability of observing results assuming the null hypothesis.
    - #Lags Used: Number of lags in the test.
    - Number of Observations Used: Size of the sample used.
    - Critical Values: Threshold values at different confidence levels.

c. Format Results:

```
python
Copy code
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic', 'p-value',
'#Lags Used', 'Number of Observations Used'])
    for key, value in dftest[4].items():
        dfoutput['Critical Value (%s)' % key] = value
```

- `dfoutput`: Stores ADF results in a user-friendly format (pandas Series).
- Includes test statistic, p-value, lags, observations, and critical values (1%, 5%, 10%).

d. Display Results:

```
python
Copy code
    print(dfoutput)
```

- Outputs the formatted results.

e. Stationarity Check:

```
python
Copy code
    if (ans['Test Statistic'] < ans["Critical Value (1%)"] or
        ans['Test Statistic'] < ans["Critical Value (5%)"] or
        ans['Test Statistic'] < ans["Critical Value (10%)"] and ans["p-
value"] < 0.05):
        message = "Stationarity based on ADH"
    else:
        message = "Non-stationarity based on ADH"
```

- Checks the test statistic against critical values **and** evaluates the p-value.
- Assigns a message indicating the result of the test.

f. Plot and Save Results:

```
python
Copy code
    plt.plot(df.index, df[pollutant], label=pollutant)
    plt.legend(loc='best')
    plt.title("{}_{}_2013 to 2021".format(message, pollutant))
    plt.savefig("{}_ADH.png".format(pollutant))
    plt.show()
```

- Plots the time series data.
- Adds a title based on the stationarity message.
- Saves the plot with a filename derived from the pollutant.

g. Return the Message:

```
python
Copy code
    return message
```

- Returns a string summarizing the result of the stationarity test.

---

4. Variables and Their Roles:

Variable	Stored Value	Role
`dftest`	Tuple with ADF test results.	Raw output from `adfuller` for stationarity analysis.
`dfoutput`	Formatted results of the ADF test as a pandas Series.	User-friendly summary of the test results.
`ans`	Alias for `dfoutput`.	Holds the same test summary for condition checks.
`message`	String: "Stationarity based on ADH" or "Non-stationarity based on ADH".	Final decision message based on test conditions.

This function effectively integrates statistical testing with data visualization to analyze and interpret the stationarity of a time series. Let me know if you'd like to dive deeper into any aspect!