THIRD ORDER TIME-VARYING AUTOREGRESSION MODEL ON NON-STATIONARY SIGNAL

Task: Signal Processing

For Remote Internship at UiT, The Arctic University of Norway

Soham Das

IITG, B.Tech in Engineering Physics, 1st Year

Contents

1. Problem Statement

The task for your selection involves working with a non-stationary signal, such as the El Centro ground motion data, or alternatively, creating a simulated non-stationary signal with similar characteristics. Your objective is to apply a 3rd-order Time-Varying Autoregressive (TVAR) model to the signal and analyze its behavior. This will involve estimating the AR coefficients over time using a sliding window approach. Once the coefficients are obtained, you are required to create a coefficient cluster map, also known as a heatmap, to visualize how these coefficients evolve over time. The heatmap should have time or time windows on the X-axis, AR coefficient indices (1st, 2nd, and 3rd) on the Y-axis, and the color gradient should represent the magnitude of the coefficients.

In addition to generating the heatmap, you must provide an analysis explaining how the changes in the AR coefficients reflect the signal's non-stationary behavior. The analysis should include a discussion of any trends, patterns, or anomalies observed in the coefficient cluster map. The deliverables for this task include a clear and well-labeled heatmap and a short report explaining your methodology, results, and interpretation.

2. Abstract

For this task, I have first generated a Non-stationary Time Series. Since Time-varying Autoregression (TVAR) Models can operate on a non-stationary time series, I have not converted the time series to stationary. Since the statsmodels library does not directly support time-varying parameters, I have created a custom implementation. I have then generated a heatmap of the autoregression (AR) coefficients, and drawn conclusions regarding the non-stationary nature of the data from it.

3. Solution

3.1 Libraries used

pandas is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool.

NumPy has functions for working in domain of linear algebra, Fourier transform, and matrices. It finds applications in various fields like Data Science, Machine Learning, Data visualization and others.

seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

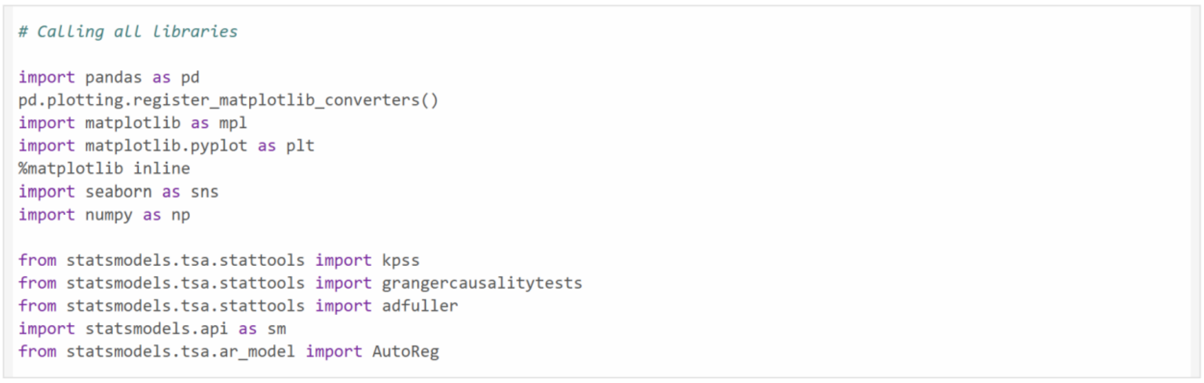
matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

matplotlib.pyplot is a state-based interface to matplotlib. It provides an implicit, MATLAB-like way of plotting. It also opens figures on your screen, and acts as the figure GUI manager.

statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration.

statsmodels.tsa contains model classes and functions that are useful for time series analysis. Such models include univariate autoregressive models (AR), vector autoregressive models (VAR), univariate autoregressive moving average models (ARMA), Markov switching dynamic regression and autoregression. It includes descriptive statistics for time series as well as the corresponding theoretical properties of ARMA or related processes. It also includes methods to work with autoregressive and moving average lag-polynomials. Additionally, related statistical tests and some useful helper functions are available.

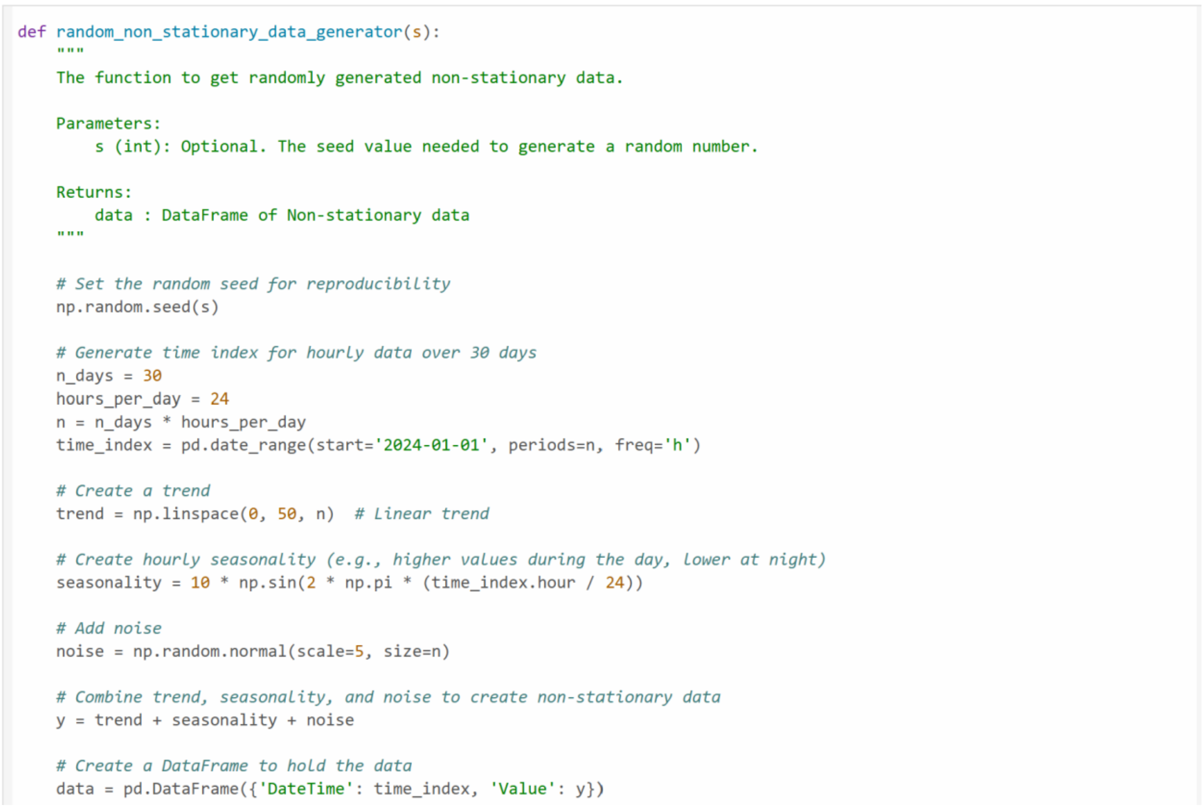
statsmodels.api contains cross-sectional models and methods.



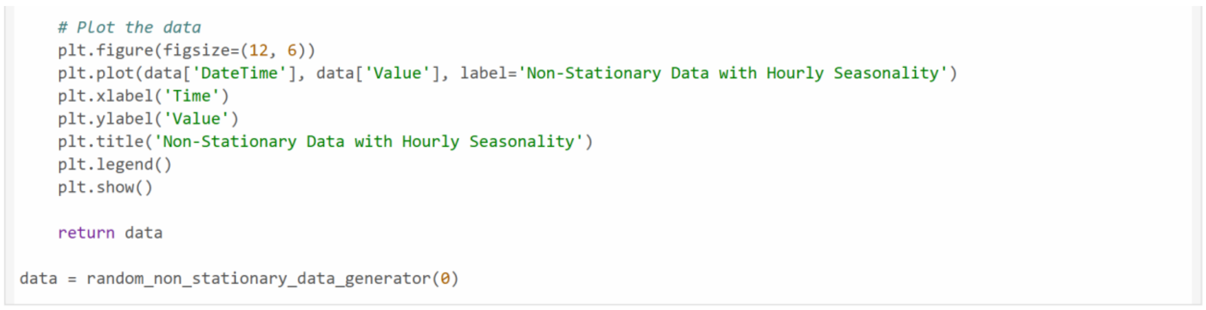
```python
# Calling all libraries

import pandas as pd
pd.plotting.register_matplotlib_converters()
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import numpy as np

from statsmodels.tsa.stattools import kpss
from statsmodels.tsa.stattools import grangercausalitytests
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
from statsmodels.tsa.ar_model import AutoReg
```

Figure 1: Libraries used

3.2 Generating Synthetic Non-stationary Time Series



```python
def random_non_stationary_data_generator(s):
    """
    The function to get randomly generated non-stationary data.

    Parameters:
        s (int): Optional. The seed value needed to generate a random number.

    Returns:
        data : DataFrame of Non-stationary data
    """

    # Set the random seed for reproducibility
    np.random.seed(s)

    # Generate time index for hourly data over 30 days
    n_days = 30
    hours_per_day = 24
    n = n_days * hours_per_day
    time_index = pd.date_range(start='2024-01-01', periods=n, freq='h')

    # Create a trend
    trend = np.linspace(0, 50, n)  # Linear trend

    # Create hourly seasonality (e.g., higher values during the day, lower at night)
    seasonality = 10 * np.sin(2 * np.pi * (time_index.hour / 24))

    # Add noise
    noise = np.random.normal(scale=5, size=n)

    # Combine trend, seasonality, and noise to create non-stationary data
    y = trend + seasonality + noise

    # Create a DataFrame to hold the data
    data = pd.DataFrame({'DateTime': time_index, 'Value': y})
```



```
    # Plot the data
    plt.figure(figsize=(12, 6))
    plt.plot(data['DateTime'], data['Value'], label='Non-Stationary Data with Hourly Seasonality')
    plt.xlabel('Time')
    plt.ylabel('Value')
    plt.title('Non-Stationary Data with Hourly Seasonality')
    plt.legend()
    plt.show()

    return data

data = random_non_stationary_data_generator(0)
```

Figure 2: Generating Synthetic Non-stationary Time Series

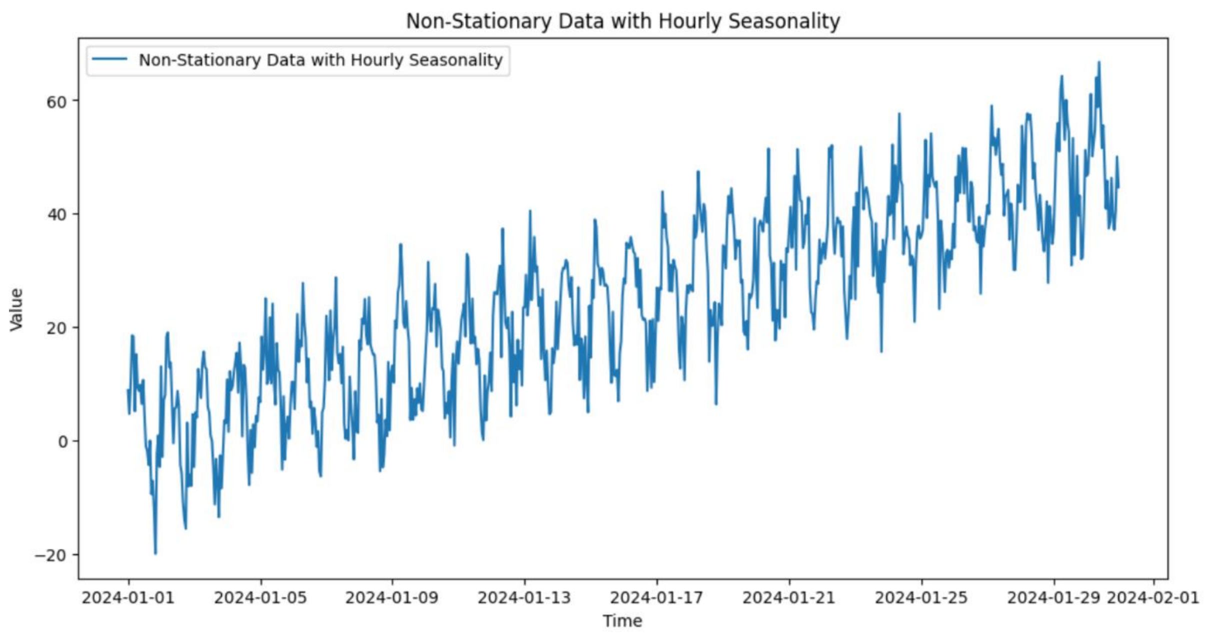


Figure 3: Plotting Non-stationary Time Series on a Line chart

3.3 Testing Stationarity

A time series is stationary when mean, correlation and variance remain constant at all sections of the data. Thus, in the absence of stationary time series, the applied model will have differing accuracy at different time points.

To check for the stationarity of time series, various tests can be used like Augmented Dickey-Fuller (ADF) test, Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test, Phillips–Perron test, Zivot-Andrews test, Ljung Test P, Durbin-Watson (DW) test and others.

ADF and **KPSS** tests have been used here.

Augmented Dickey-Fuller (ADF) Test

Null Hypothesis (H_0): The time series has a unit root, indicating it is non-stationary.

Alternate Hypothesis (H_1): The time series does not have a unit root, indicating it is stationary.

Test Statistic: The ADF test statistic is compared to critical values from the ADF distribution to determine whether the null hypothesis can be rejected.

Decision Rule: If the test statistic is less than the critical value, the null hypothesis is rejected, and the series is considered stationary. Otherwise, if the test statistic is greater than the critical value, the null hypothesis is not rejected, and the series is considered non-stationary.

Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test

Null Hypothesis (H_0): The series is stationary around a deterministic trend.

Alternate Hypothesis (H_1): The series has a unit root, indicating it is non-stationary.

Test Statistic: The KPSS test statistic is compared to critical values to determine whether the null hypothesis can be rejected.

Decision Rule: If the test statistic is greater than the critical value, the null hypothesis is rejected, and the series is considered non-stationary. If the test statistic is less than the critical value, the null hypothesis is not rejected, and the series is considered stationary.

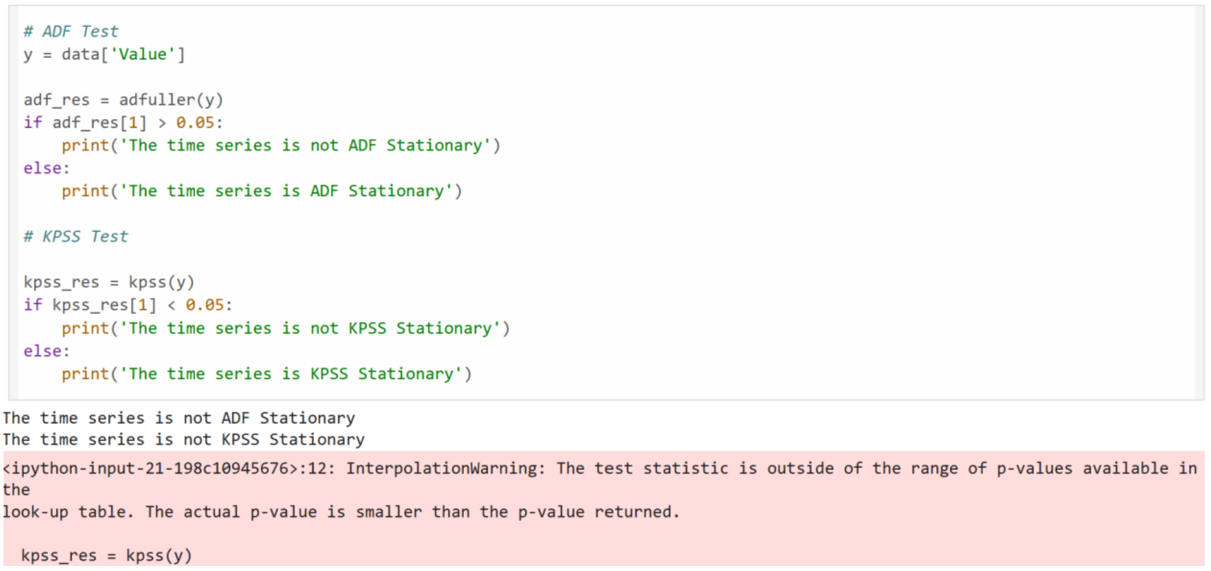
ADF & KPSS Test Cases

Case 1: Both tests conclude that the series is not stationary - The series is not stationary

Case 2: Both tests conclude that the series is stationary - The series is stationary

Case 3: KPSS indicates stationarity and ADF indicates non-stationarity - The series is trend stationary. Trend needs to be removed to make series strict stationary. The detrended series is checked for stationarity.

Case 4: KPSS indicates non-stationarity and ADF indicates stationarity - The series is difference stationary. Differencing is to be used to make series stationary. The differenced series is checked for stationarity.



```
# ADF Test
y = data['Value']

adf_res = adfuller(y)
if adf_res[1] > 0.05:
    print('The time series is not ADF Stationary')
else:
    print('The time series is ADF Stationary')

# KPSS Test

kpss_res = kpss(y)
if kpss_res[1] < 0.05:
    print('The time series is not KPSS Stationary')
else:
    print('The time series is KPSS Stationary')
```

```
The time series is not ADF Stationary
The time series is not KPSS Stationary
<ipython-input-21-198c10945676>:12: InterpolationWarning: The test statistic is outside of the range of p-values available in
the
look-up table. The actual p-value is smaller than the p-value returned.

  kpss_res = kpss(y)
```

Figure 4: Testing Stationarity

3.4 How to apply Non-stationary -> Stationary Time Series conversion, and why is it not done here?

Non-stationary -> Stationary time series conversion methods:

- **Trend**
  - **Detrending**: Remove the trend component from the time series. This can be achieved by fitting a regression line or using techniques like moving averages.
  - **Differencing**: Take the difference between consecutive observations to remove the trend. This can be done once or multiple times until the data becomes stationary.
- **Seasonality**
  - **Seasonal Adjustment**: Use techniques such as seasonal decomposition of time series (e.g., STL decomposition) to separate the seasonal component from the data.
  - **Seasonal Differencing**: Take differences between observations at the same season of different years to remove seasonality.
- **Variance**
  - **Transformation**: Apply transformations such as logarithmic, square root, or Box-Cox transformation to stabilize the variance and make it more constant over time.
- **Auto-correlation**
  - **Differencing**: Besides removing trends, differencing can also help reduce autocorrelation by eliminating dependence between consecutive observations.
  - **Autoregressive Integrated Moving Average (ARIMA)**: Utilize ARIMA models, which incorporate differencing to handle autocorrelation.

However, **Time-varying Autoregression Model (TVAR) can handle non-stationary time series** because they allow the model parameters to change over time, adapting to the evolving nature of the data

- **Dynamic Coefficients**: TVAR models have coefficients that change over time, allowing them to capture the underlying dynamics of a non-stationary process. This flexibility means they can adapt to shifts in trends, seasonality, and other changes in the data.
- **Localized Fitting**: While the overall data may be non-stationary, TVAR models often assume local stationarity within short time windows. By fitting autoregressive models to these smaller segments, TVAR models can handle the variations in data behavior over longer periods.

3.5 Setting up the 3rd-order Time-Varying Autoregressive (TVAR (3)) model with Sliding Window Approach

Basic Autoregressive (AR) Model

A p-order autoregressive (AR) model can be represented as:

$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t$

where:

- y_t is the value of the time series at time t.
- ϕ_i (for i = 1, 2, ..., p) are the autoregressive coefficients.
- p is the order of the AR model.
- ϵ_t is the white noise error term with zero mean and constant variance.

Time-Varying Autoregressive (TVAR) Model

A p-order time-varying autoregressive (TVAR) model can be represented as:

$y_t = \phi_{1,t} y_{t-1} + \phi_{2,t} y_{t-2} + \cdots + \phi_{p,t} y_{t-p} + \epsilon_t$

where:

- y_t is the value of the time series at time t.
- $\phi_{i,t}$ (for i = 1, 2, ..., p) are the time-varying autoregressive coefficients.
- p is the order of the AR model.
- ϵ_t is the white noise error term with zero mean and constant variance.

Estimation of Time-Varying Coefficients

There are several methods to estimate the time-varying coefficients in a TVAR model, including:

- **Sliding Window Estimation**: Fit a separate AR model to each window of data and allow the coefficients to vary between windows.
- **Kalman Filtering**: Use state-space models and the Kalman filter to estimate the time-varying coefficients.
- **Smoothing Splines**: Use smoothing techniques to allow the coefficients to vary smoothly over time.

Sliding Window Approach

In the sliding window approach, the data is divided into overlapping windows, and a separate AR model is fitted to each window. The coefficients are then allowed to vary between windows. For example, with a window size w:

$\phi_{1,t}, \phi_{2,t}, \ldots, \phi_{p,t}$

are estimated for $t = w, w+1, \ldots, T$.

Kalman Filter Approach

Using a state-space representation, the TVAR model can be written as:

State Equation: $\phi_{t+1} = \phi_t + \eta_t$

Measurement Equation: $y_t = \phi_t^T y_{t-1} + \epsilon_t$

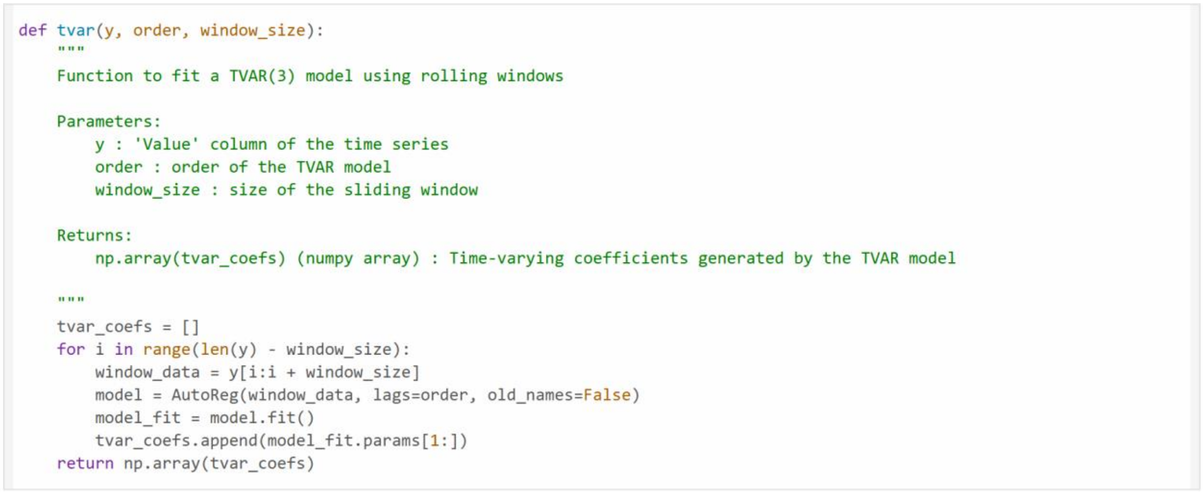
where ϕ_t is the state vector of time-varying coefficients and η_t is the process noise.

The Kalman filter recursively estimates the state vector ϕ_t from the observed data y_t.

Smoothing Splines Approach

In this approach, the coefficients are assumed to vary smoothly over time and are modeled using spline functions. This allows for smooth transitions in the coefficients and can be estimated using smoothing techniques.

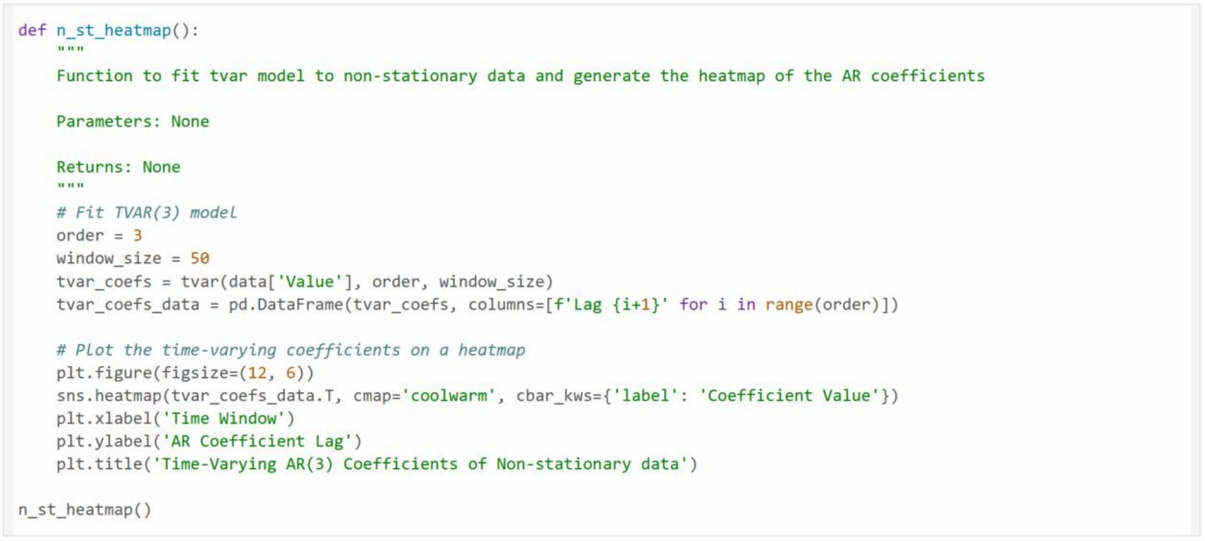
The **Sliding Window Approach** has been used here.



```python
def tvar(y, order, window_size):
    """
    Function to fit a TVAR(3) model using rolling windows

    Parameters:
        y : 'Value' column of the time series
        order : order of the TVAR model
        window_size : size of the sliding window

    Returns:
        np.array(tvar_coefs) (numpy array) : Time-varying coefficients generated by the TVAR model

    """
    tvar_coefs = []
    for i in range(len(y) - window_size):
        window_data = y[i:i + window_size]
        model = AutoReg(window_data, lags=order, old_names=False)
        model_fit = model.fit()
        tvar_coefs.append(model_fit.params[1:])
    return np.array(tvar_coefs)
```

Figure 5: Setting up the 3rd-order Time-Varying Autoregressive (TVAR (3)) model with Sliding Window Approach

3.6 Generating Heatmap of Time-Varying Autoregression (AR) Coefficients



```python
def n_st_heatmap():
    """
    Function to fit tvar model to non-stationary data and generate the heatmap of the AR coefficients

    Parameters: None

    Returns: None
    """
    # Fit TVAR(3) model
    order = 3
    window_size = 50
    tvar_coefs = tvar(data['Value'], order, window_size)
    tvar_coefs_data = pd.DataFrame(tvar_coefs, columns=[f'Lag {i+1}' for i in range(order)])

    # Plot the time-varying coefficients on a heatmap
    plt.figure(figsize=(12, 6))
    sns.heatmap(tvar_coefs_data.T, cmap='coolwarm', cbar_kws={'label': 'Coefficient Value'})
    plt.xlabel('Time Window')
    plt.ylabel('AR Coefficient Lag')
    plt.title('Time-Varying AR(3) Coefficients of Non-stationary data')

n_st_heatmap()
```

Figure 6: Generating Heatmap of Time-Varying Autoregression (AR) Coefficients

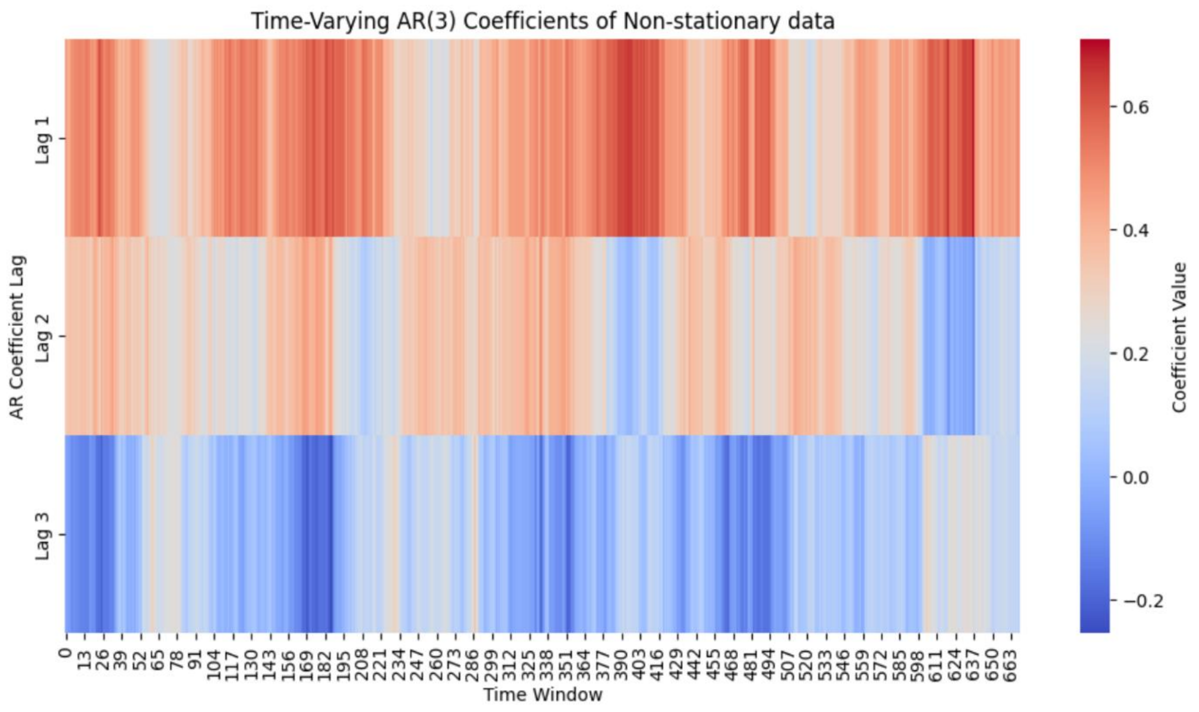


Figure 7: Heatmap of Time-Varying Autoregression (AR) Coefficients

3.7 Analyzing Heatmap

Key Points to Look For:

1. **Consistency of Coefficients Over Time**
   - Stationarity: If the coefficients remain relatively constant over time, it suggests that the time series may be stationary. In a stationary time series, the statistical properties, including the autoregressive coefficients, do not change significantly over time.
   - Non-Stationarity: If the coefficients change significantly over time, it indicates non-stationarity. This could be due to evolving trends, changing variances, or seasonal patterns that affect the time series.
2. **Patterns in Coefficients**
   - Trend/Drift: Look for patterns such as a gradual increase or decrease in the coefficients. This could indicate a trend or drift in the underlying time series.
   - Seasonality: Repeating patterns or periodic changes in the coefficients suggest the presence of seasonality. This is common in data with daily, monthly, or yearly cycles.
3. **Sudden Changes**
   - Structural Breaks: Sudden, large changes in the coefficients may indicate structural breaks in the time series. These breaks can result from significant events or regime changes affecting the data generation process.

Visual Inspection Using a Heatmap

A heatmap of AR coefficients can visually represent how these coefficients evolve over time. Here's an example of what to look for:

- **Stable Coefficients**: Uniform color bands indicating little change over time suggest stationarity.
- **Changing Coefficients**: Varied colors over time suggest non-stationarity.

4. Conclusion

From the heatmap of the time-varying AR (3) coefficients of non-stationary time-series, we observe:

1. **Non-stationarity**: The heatmap shows significant variations in the AR coefficients over time, which is significantly evident in Lag 1 and Lag 2, and lesser variation in Lag 3 considering the colors of the heatmap at the two ends. This indicates that the time series is non-stationary.
2. **Trend**: No specific trend was observed.
3. **Seasonality**: Seasonal patterns are present in the heatmap, which is significantly evident in Lag 1, Lag 2 and Lag 3 where they have high intensity peaks at around 15, 170, 325, 480 along the time window. Lag 1 and Lag 2 also has an additional peak at around 635.
4. **Structural Break and Noise**: In some periods of the heatmap, the TVAR coefficients might exhibit high variability with no obvious trend - which indicates periods of increased randomness or noise. The heatmap shows significant structural breaks in all 3 Lags –
    - in Lag 1 (around 416), coefficient value (color intensity) abruptly drops from above 0.6 to around 0.2
    - in Lag 2 (around 611), color abruptly changes from light red to light blue and vice versa abruptly i.e. between around 0.3 and below 0.0
    - in Lag 3 (around 182), coefficient value (color intensity) abruptly changes from below -0.2 to around 0.1
5. **Degree of Autocorrelation**: Autocorrelation tends to remain high over time. Lag 1 shows higher auto-correlation than the others since its coefficient values are closer to 1 than the others.
6. There are certain periods where the AR coefficients remain relatively stable, which suggests that the underlying time series is stationary in those periods.
    - in Lag 1, from around 143 to around 208
    - in Lag 2, from around 234 to around 325
    - in Lag 3, from around 507 to around 598