**Forecasting Non-Stationary Time Series**

[[](https://medium.com/@evertongomede?source=post_page-----03b638a7cd50--------------------------------)](https://medium.com/@evertongomede?source=post_page-----03b638a7cd50--------------------------------)

[Everton Gomede, PhD](https://medium.com/@evertongomede?source=post_page-----03b638a7cd50--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2Fb1a64eb107f0&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40evertongomede%2Fforecasting-non-stationary-time-series-03b638a7cd50&user=Everton+Gomede%2C+PhD&userId=b1a64eb107f0&source=post_page-b1a64eb107f0----03b638a7cd50---------------------post_header-----------)

6 min read

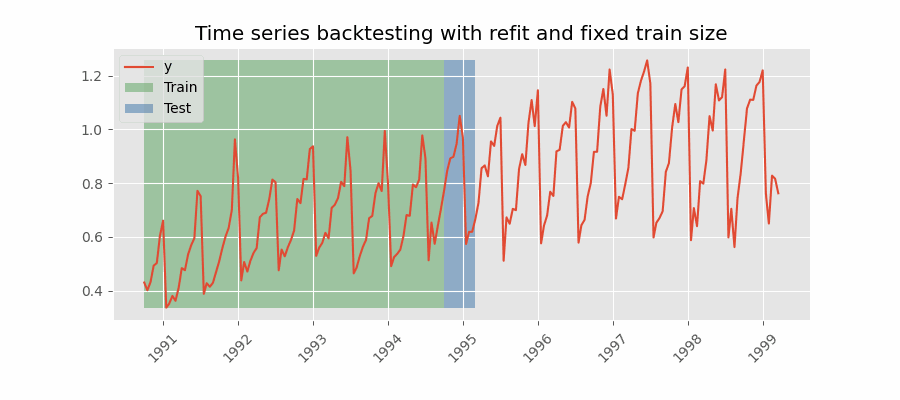
·

Oct 14

240

**Introduction**

Time series data, which records observations over time, is an essential component of various fields, including economics, finance, meteorology, and more. Accurate forecasting of time series data is critical for decision-making and planning. However, not all time series data is stationary, meaning its statistical properties change over time. Forecasting non-stationary time series is a challenging task that requires specialized techniques and models to handle the dynamic nature of the data. This essay explores the concept of non-stationary time series, the challenges it presents, and the methodologies used for forecasting in such scenarios.



*In the realm of data, forecasting non-stationary time series is like navigating a river’s ever-changing currents, where adaptability and precision are the compass and the map.*

**Understanding Stationarity**

Before delving into non-stationary time series, it’s crucial to grasp the concept of stationarity. A stationary time series is one whose statistical properties do not change over time. These properties include the mean, variance, and autocorrelation. In essence, a stationary time series has a constant mean and variance and exhibits consistent patterns and trends throughout its observations. Stationary data is easier to model and forecast since it follows stable statistical properties.

**Challenges of Non-Stationary Time Series**

Non-stationary time series, on the other hand, exhibit changing statistical properties over time. This makes them more challenging to work with and forecast. The main challenges associated with non-stationary time series are:

1. **Trend**: Non-stationary data often displays a trend, which is a long-term systematic movement in one direction, either upward (an increasing trend) or downward (a decreasing trend). Identifying and removing the trend is crucial for forecasting.
2. **Seasonality**: Seasonality is another common characteristic of non-stationary data. It represents periodic fluctuations that occur at regular intervals, such as daily, monthly, or yearly patterns. Accounting for seasonality is vital in making accurate forecasts.
3. **Heteroscedasticity**: Non-stationary time series often exhibit varying levels of volatility or variance over time. This heteroscedasticity can make it difficult to develop forecasting models that assume constant variance.

**Forecasting Techniques for Non-Stationary Time Series**

To address the challenges posed by non-stationary time series data, various forecasting techniques and models have been developed. Some of the commonly used approaches include:

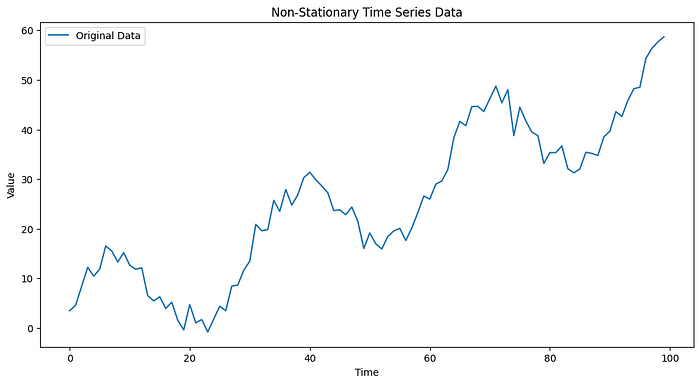
1. **Differencing**: One of the primary methods for dealing with non-stationarity is differencing. This involves computing the differences between consecutive observations, which can help in stabilizing the mean and eliminating trends and seasonality.
2. **Seasonal Decomposition**: Seasonal decomposition of time series (STL) is a method that separates a time series into its trend, seasonal, and residual components. This decomposition allows for modeling and forecasting each component separately, making it easier to handle non-stationarity.
3. **Exponential Smoothing**: Exponential smoothing methods, such as Holt-Winters, are effective for forecasting time series with trends and seasonality. These methods assign different weights to past observations, giving more importance to recent data points.
4. **ARIMA Models**: AutoRegressive Integrated Moving Average (ARIMA) models are widely used for non-stationary time series. ARIMA models involve differencing the data to make it stationary and then modeling it using autoregressive and moving average components.
5. **Seasonal ARIMA**: For time series with both trend and seasonality, Seasonal ARIMA models (SARIMA) extend the ARIMA framework by incorporating seasonal differencing and seasonal autoregressive and moving average components.
6. **State Space Models**: State space models, such as the Kalman filter, provide a flexible framework for modeling and forecasting non-stationary time series. These models can capture complex dynamics and structural changes in the data.

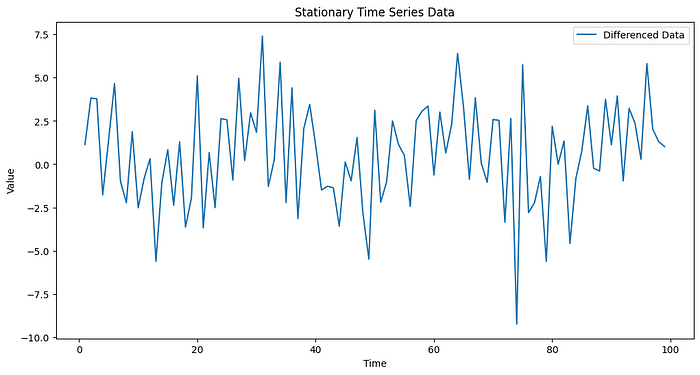
**Code**

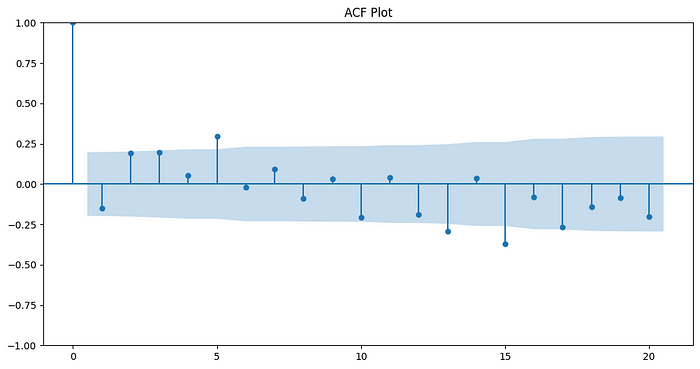
Forecasting non-stationary time series typically involves a series of steps, including data preprocessing, model selection, training, and plotting the results. Here’s a Python code example for forecasting a non-stationary time series using the ARIMA model with a dataset and plots:

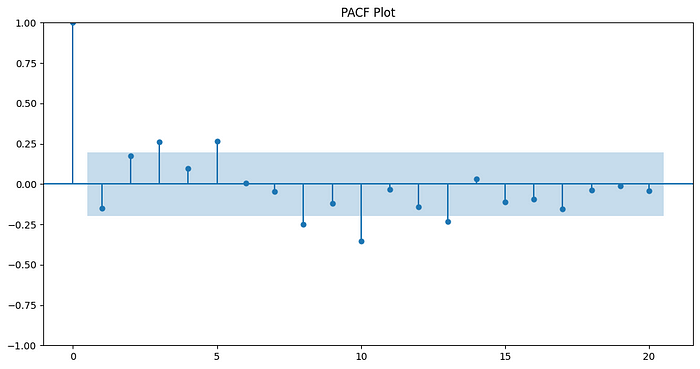
# Import necessary libraries  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from statsmodels.tsa.arima.model import ARIMA  
from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf  
from statsmodels.tsa.stattools import adfuller  
  
# Generate or load a non-stationary time series dataset  
# For demonstration, we'll generate a simple non-stationary dataset  
np.random.seed(42)  
  
# Generate a non-stationary time series with a trend and seasonality  
t = np.arange(1, 101)  
seasonal\_component = 10 \* np.sin(0.2 \* t)  
trend\_component = 0.5 \* t  
noise = np.random.normal(0, 2, 100)  
data = seasonal\_component + trend\_component + noise  
  
# Create a pandas DataFrame from the dataset  
df = pd.DataFrame({'Data': data})  
  
# Plot the original time series data  
plt.figure(figsize=(12, 6))  
plt.plot(df.index, df['Data'], label='Original Data')  
plt.xlabel('Time')  
plt.ylabel('Value')  
plt.title('Non-Stationary Time Series Data')  
plt.legend()  
plt.show()  
  
# Check for stationarity using Augmented Dickey-Fuller test  
def adf\_test(series):  
 result = adfuller(series)  
 print('ADF Statistic:', result[0])  
 print('p-value:', result[1])  
 print('Critical Values:', result[4])  
  
adf\_test(df['Data'])  
  
# Differencing to make the time series stationary  
df['Differenced\_Data'] = df['Data'] - df['Data'].shift(1)  
df = df.dropna()  
  
# Plot the differenced time series data  
plt.figure(figsize=(12, 6))  
plt.plot(df.index, df['Differenced\_Data'], label='Differenced Data')  
plt.xlabel('Time')  
plt.ylabel('Value')  
plt.title('Stationary Time Series Data')  
plt.legend()  
plt.show()  
  
# ACF and PACF plots for determining ARIMA orders  
plt.figure(figsize=(12, 6))  
plot\_acf(df['Differenced\_Data'], lags=20, ax=plt.gca())  
plt.title('ACF Plot')  
plt.show()  
  
plt.figure(figsize=(12, 6))  
plot\_pacf(df['Differenced\_Data'], lags=20, ax=plt.gca())  
plt.title('PACF Plot')  
plt.show()  
  
# Split the data into training and testing sets  
train\_size = int(0.8 \* len(df))  
train, test = df['Differenced\_Data'][:train\_size], df['Differenced\_Data'][train\_size:]  
  
# Fit an ARIMA model to the training data  
model = ARIMA(train, order=(1, 1, 1))  
model\_fit = model.fit()  
  
# Forecast the test data  
forecast = model\_fit.forecast(steps=len(test))  
  
# Calculate prediction intervals  
residuals = test - forecast  
prediction\_interval = 1.96 \* np.std(residuals) # 1.96 for a 95% prediction interval  
  
# Plot the forecasts and the actual values with prediction intervals  
plt.figure(figsize=(12, 6))  
plt.plot(df.index[train\_size:], test, label='Actual')  
plt.plot(df.index[train\_size:], forecast, label='Forecast', color='red')  
plt.fill\_between(df.index[train\_size:], forecast - prediction\_interval, forecast + prediction\_interval, color='pink', alpha=0.3)  
plt.xlabel('Time')  
plt.ylabel('Value')  
plt.title('ARIMA Forecasting with Prediction Intervals')  
plt.legend()  
plt.show()

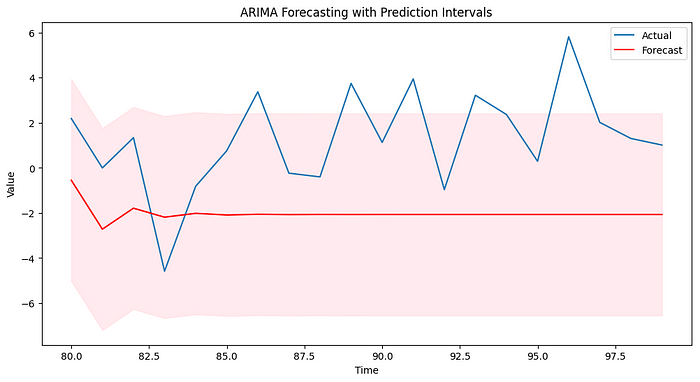
This code generates a non-stationary time series, converts it into a stationary series through differencing, determines ARIMA model orders using ACF and PACF plots, and then forecasts the test data. The results are plotted to visualize the forecasted values against the actual data.











Please note that in a real-world scenario, you would typically load your time series data from a file or database. Additionally, you may need to fine-tune the ARIMA model parameters based on the characteristics of your specific dataset.

**Conclusion**

Forecasting non-stationary time series is a complex and important task in various domains. Recognizing the challenges associated with non-stationarity, such as trends, seasonality, and heteroscedasticity, is the first step in developing accurate forecasts. To address these challenges, a variety of techniques and models, including differencing, seasonal decomposition, exponential smoothing, ARIMA models, and state space models, have been developed. These methods allow analysts to make more informed decisions and predictions in the face of non-stationary time series data, ultimately improving their ability to plan, allocate resources, and respond to changing conditions.