Highly Integrated System Project

Assignment 1

Abu Saleh Md Nayem

Hardikkumar Ghanshyambhai Khunt

Mohsina Binte Asad

Patel Hemal Bharatkumar

Parthkumar Krushnakantbhai Patel

Zain Hanif

October 28, 2024

Introduction to Advanced Time Series Analysis with Python

This book is crafted for data scientists and machine learning (ML) engineers seeking advanced skills in time series analysis. Time series analysis, often bypassed in standard ML resources, is highly valuable for businesses where data is frequently recorded over time. Given its importance, and as time remains a fundamental dimension in our world, time series data is pervasive.

Although time series analysis originated in the 1920s and 1930s, the modern era's vast data generation has transformed the landscape. With the increasing data volume and ML interest, traditional statistical approaches like AutoRegressive Integrated Moving Average (ARIMA) have been expanded with cutting-edge ML techniques for richer analysis.

This chapter will cover the following foundational topics:

- **What is a time series?** – Defining the concept and structure of time series data.
- **Data-generating process (DGP)** – Understanding the underlying mechanisms that produce time series data.
- **What can we forecast?** – Exploring the scope and limitations of time series forecasting.
- **Forecasting terminology and notation** – Introducing essential terminology and notation for effective time series analysis.

Technical Requirements

To follow along with the code in this book, set up the **Anaconda** environment as per the instructions in the Preface. This setup provides all necessary packages and datasets required for the examples and exercises.

What is a Time Series?

A **time series** is a sequence of observations recorded at successive points in time. Each observation captures the same metric over time, such as monthly chocolate consumption or daily stock prices. By tracking these values consistently, time series analysis can uncover patterns, trends, and relationships over time.

Examples of Time Series

Examples of time series include:

- Weekly stock closing prices
- Daily rainfall measurements
- Hourly heart rate readings from a smartwatch

Types of Time Series

Time series data can be categorized into two main types:

- **Regular Time Series**: Observations occur at fixed intervals, such as hourly or monthly measurements.
- **Irregular Time Series**: Observations occur at inconsistent intervals, such as medical readings recorded only during patient visits.

Main Areas of Application for Time Series Analysis

Time series analysis has three primary areas of application:

- **Time Series Forecasting**: Predicting future values based on historical data. For instance, using past temperature data to forecast the next day's temperature.
- **Time Series Classification**: Identifying patterns or actions based on historical data. For example, using an electroencephalogram (EEG) or electrocardiogram (EKG) history to classify the result as normal or abnormal.
- **Interpretation and Causality**: Analyzing interrelationships within time series data to understand causes and underlying patterns, allowing for causal inference and deeper insights.

Data-Generating Process (DGP) and Modeling

Time series data is a sequence of observations taken over time, generated by various underlying factors, known as the **Data-Generating Process (DGP)**. For example, the daily shipments of chocolate are influenced by seasonal factors, raw material availability, and machinery uptime. Statistically, the DGP is often a stochastic process that can be approximated but not fully known.

Modeling the DGP

Since perfect knowledge of the DGP is unattainable, we aim to approximate it with models that capture key features, enabling practical forecasts or insights. Importantly, a model is an *approximation* of reality, not reality itself. This distinction is illustrated by comparing a map to an aerial view of Bengaluru: while the map helps with navigation (a specific purpose), it omits cultural and experiential aspects of the city.

Model Usefulness

The utility of a model depends on its application. For navigation, a map is valuable, but it is insufficient for understanding cultural nuances. Thus, the effectiveness of any model is context-dependent, serving as a simplified but purposeful representation of the DGP within defined limits.

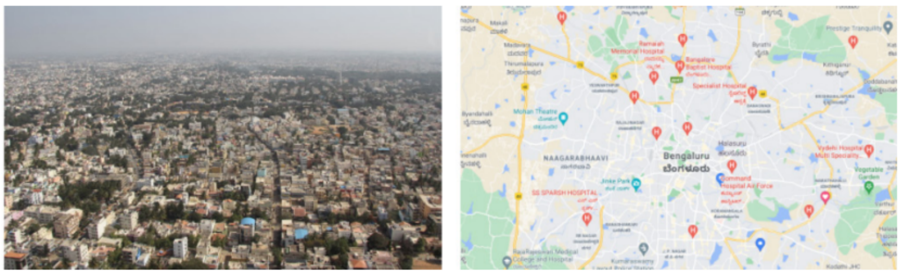

Figure 1: Comparison between an aerial view and a map of Bengaluru.

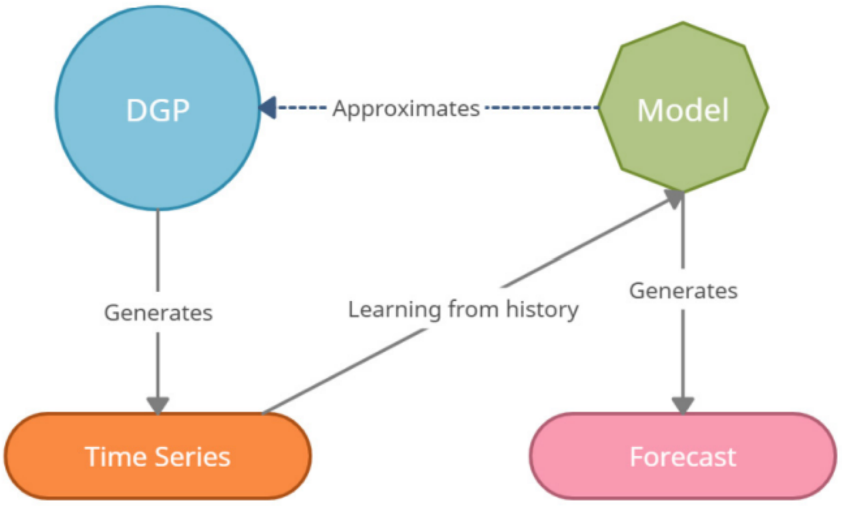

Figure 2: D
GP, model, and time series.

Generating Synthetic Time Series

Let's take a look at a few practical examples where we can generate a few time series using a set of fundamental building blocks. You can get creative and mix and match any of these components, or even add them together to generate a time series of arbitrary complexity.

White and Red Noise

A common stochastic process that generates time series data is **white noise**. White noise consists of a sequence of random values with a zero mean and constant standard deviation. It is frequently used as a baseline noise assumption in time series analysis.

Generating and Plotting White Noise

The following Python code demonstrates how to create a time series with white noise and plot it:

```
# Generate the time axis with sequential numbers up to 200
time = np.arange(200)
# Sample 200 random values
values = np.random.randn(200) * 100
plot_time_series(time, values, "White␣Noise")
```

The output plot is shown below:

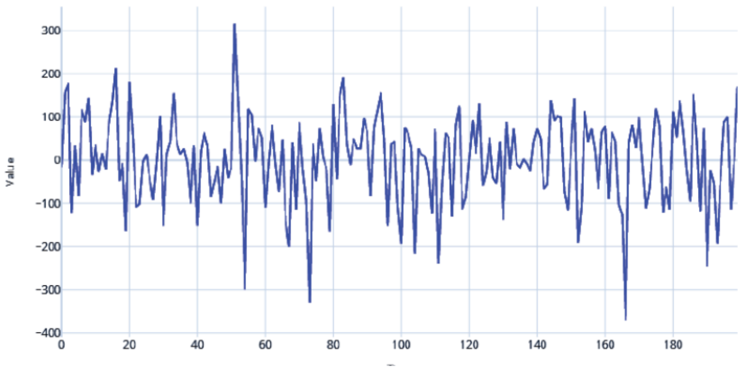

Figure 3: Generated white noise time series plot.

Red Noise Generation

Red noise, on the other hand, has zero mean and constant variance but is serially correlated in time. This serial correlation or redness is parameterized by a correlation coefficient r, such that:

$$X_{j+1} = r \cdot X_j + (1 - r^2)^3 \cdot w$$

where w is a random sample from a white noise distribution. Let's see how we can generate that, as follows:

```
# Setting the correlation coefficient
r = 0.4
# Generate the time axis
time = np.arange(200)
# Generate white noise
white_noise = np.random.randn(200) * 100
# Create Red Noise by introducing correlation between subsequent values
values = np.zeros(200)
for i, v in enumerate(white_noise):
    if i == 0:
        values[i] = v
    else:
        values[i] = r * values[i-1] + np.sqrt(1 - np.power(r, 2)) * v
plot_time_series(time, values, "Red Noise Process")
```

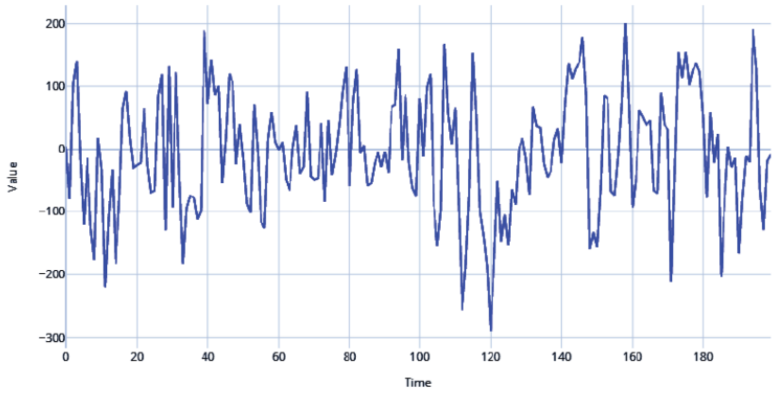

Figure 4: Generated white noise time series plot.

Cyclical or Seasonal Signals

Among the most common signals you see in time series are seasonal or cyclical signals. Therefore, you can introduce seasonality into your generated series in a few ways.

Python Code for Sinusoidal Signals

```
# Sinusoidal Signal with Amplitude=1.5 & Frequency=0.25
signal_1 = ts.signals.Sinusoidal(amplitude=1.5, frequency=0.25)


# Sinusoidal Signal with Amplitude=1 & Frequency=0.5
signal_2 = ts.signals.Sinusoidal(amplitude=1, frequency=0.5)
```

```
samples_1 , regular_time_samples , signals_1 , errors_1 =
generate_timeseries(signal=signal_1)

samples_2 , regular_time_samples , signals_2 , errors_2 =
generate_timeseries(signal=signal_2)

plot_time_series(regular_time_samples ,
                 [samples_1 , samples_2] ,
                 "Sinusoidal␣Waves",
                 legends=["Amplitude␣=␣1.5␣|␣Frequency␣=␣0.25",
                          "Amplitude␣=␣1␣|␣Frequency␣=␣0.5"])
```

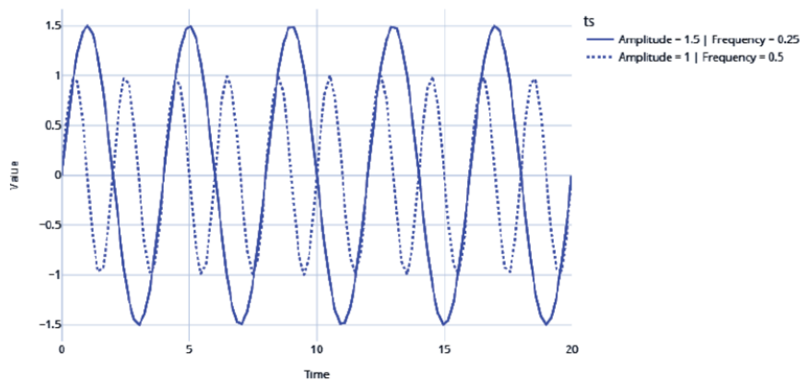

Figure 1.5 – Sinusoidal waves

Figure 5: Sinusoidal Signals.

Autoregressive Signals

Another very popular signal in the real world is an autoregressive (AR) signal. An AR signal refers to when the value of a time series for the current timestep depends on the values of the time series in the previous timesteps. This serial correlation is a key property of the AR signal and is characterized by the following parameters:

- **Order of serial correlation:** The number of previous timesteps the signal depends on.
- **Coefficients:** These coefficients combine the previous timesteps.

To illustrate how to generate an AR signal, consider the following example:

```
# Autoregressive signal with parameters 1.5 and −0.75
signal = ts.signals.AutoRegressive(ar_param=[1.5, −0.75])

# Generate Timeseries
samples, regular_time_samples, signals, errors =
generate_timeseries(signal=signal)

# Plot the time series
plot_time_series(regular_time_samples, samples, "Auto␣Regressive")
```

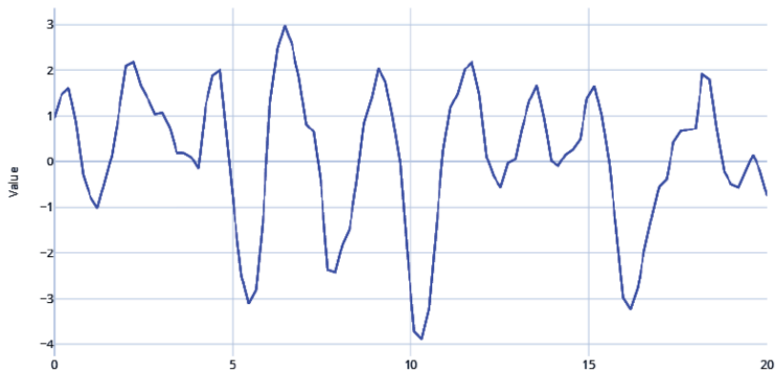

Figure 6: AR Signal

What Can We Forecast?

Before progressing, it is essential to understand the predictability of a time series in forecasting. The basic assumption in time series forecasting is that the future depends on the past; however, not all time series exhibit equal predictability.

We can rank examples of predictability from easiest to hardest:

- **High tide next Monday:** Easiest to predict due to its reliable nature.
- **The stock price of Tesla next Friday:** Difficult to predict, but not impossible.
- **Lottery numbers next Sunday:** Very hard to predict as they are largely random.

Three main factors influence the predictability of a time series:

- **Understanding the Data Generating Process (DGP):** Greater understanding leads to higher predictability.
- **Amount of Data:** More data generally improves predictability.
- **Adequately Repeating Patterns:** The presence of consistent patterns enhances predictability.

While this mental model of predictability is useful, more concrete methods for assessing predictability will be explored in Chapter 3, *Analyzing and Visualizing Time Series Data*. The key takeaway is that not all time series are equally predictable. To follow the upcoming discussions, we need to establish a standard notation and familiarize ourselves with terminology specific to time series analysis.

Forecasting Terminology

Understanding key terminologies in time series forecasting is essential for following this book and other literature. Here are some fundamental terms:

- **Forecasting:** The prediction of future values of a time series using known past values and/or related variables. This is similar to prediction in machine learning, where a model predicts unseen data.
- **Multivariate Forecasting:** Involves more than one time series variable that depends not only on its past values but also on other variables. For example, macroeconomic indicators like GDP and inflation form a multivariate time series. The goal is to model the interrelationships between variables and their pasts to forecast them together.
- **Explanatory Forecasting:** This type of forecasting incorporates additional information beyond the past values of a time series. For instance, predicting retail store sales may include data on promotional offers.
- **Backtesting:** A practice where a validation set is set aside from training data to evaluate models. In time series, backtesting uses historical data to assess a trained model's performance. Various validation methods will be discussed later in the book.
- **In-sample and Out-sample:** In-sample refers to training data, while out-sample pertains to unseen or testing data. In-sample metrics are calculated on training data, whereas out-sample metrics are derived from testing data.
- **Exogenous and Endogenous Variables:** Exogenous variables are external time series that aid in modeling the target time series but are not modeled for output. In contrast, endogenous variables are influenced by other variables in the system. The target variable can be viewed as endogenous, while explanatory regressors are considered exogenous.
- **Forecast Combination:** Similar to ensembling in machine learning, this process combines multiple forecasts using functions, either learned or heuristic-based, such as averaging multiple models.

While this is not an exhaustive list, familiarity with these terms will provide a solid foundation for understanding time series forecasting concepts discussed throughout the book.

Additional Resources

The associated code for Chapter 1 is available at:

https://github.com/PacktPublishing/Modern-Time-Series-Forecasting-with-Python-/tree/main/notebooks/Chapter01