ChronoXtract: A python library for time series feature extraction

Aman Kumar

Abstract

Contents

1 Introduction

Time series or sequential data are ubiquitous in modern applications, covering finance, healthcare, climate science, astrophysics, particle physics and the Internet of Things (IoT). Extracting meaningful features from such data is critical for anomaly detection, forecasting, and classification tasks. Such meaningful features are hard to identify that can summarise the long-term and short-term dynamics contained in the time series/ sequential data.

Statistical feature extraction serves as a foundational step in transforming raw time series data into structured, interpretable representations that reveal underlying patterns, trends, and anomalies. These features act as concise numerical summaries of complex temporal dynamics, enabling downstream tasks such as classification, forecasting, anomaly detection, and model training. For instance, basic statistical measures—such as the mean , median , and mode —capture central tendencies and distributional properties, while metrics like fractional variability (the ratio of the range to the mean) quantify volatility or irregularity in the data. Such metrics are indispensable for identifying outliers, assessing stability, or comparing datasets across domains. Beyond descriptive statistics, spectral features derived from methods like Fourier analysis provide insights into periodicity and cyclical behavior, which are critical for applications such as signal processing, climate modeling, and sensor data analysis. Fourier coefficients, for example, decompose time series into constituent frequencies, enabling the detection of hidden periodic patterns or noise components. Together, these statistical and spectral features form a versatile toolkit for distilling actionable insights from raw data, bridging the gap between raw observations and higher-level analytical tasks. In the next section, we describe the details about the statistical features included in the *ChronoXtract* library.

2 Statistical Features

Here, we discuss the list of statistical features extracted by the ChronoXtract package.

2.1 Mean

The mean (or average) represents the central tendency of the time series data. It provides a single value summarizing the overall magnitude of the dataset. Mean is given by:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

Algorithm:

1. Sum all elements in the time series vector.
2. Divide the sum by the length of the vector.

2.2 Median

The median is the middle value of a sorted dataset. It is robust to outliers and provides a measure of central tendency that is less sensitive to extreme values than the mean.

$$\text{Median} = \begin{cases} \frac{n+1}{2}, & \text{if } n \text{ is odd} \\ \frac{x_{\frac{n}{2}} + x_{\frac{n}{2}+1}}{2}, & \text{if } n \text{ is even} \end{cases} \tag{2}$$

Algorithm:

1. Sort the vector.
2. Compute the median based on whether the length of the vector is odd or even.

2.3 Mode

The mode is the most frequently occurring value in the dataset. It is useful for identifying dominant patterns or trends in categorical or discrete numerical data.

$$\text{Mode} = \text{argmax}\{\text{count}(x_i)\} \tag{3}$$

Algorithm:

1. Use a HashMap to count occurrences of each value in the time series.
2. Iterate through the HashMap to find the key with the maximum count.

2.4 Variance

Variance measures the spread or dispersion of the data around the mean. A higher variance indicates greater variability in the dataset.

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{4}$$

here μ is the mean of the dataset.

Algorithm:

1. Compute the mean of the time series.
2. Calculate the squared difference between each element and the mean.
3. Sum the squared differences and divide by the length of the vector.

2.5 Standard Deviation

Standard deviation is the square root of the variance. It provides a measure of the dispersion of the data around the mean. A higher standard deviation indicates greater variability in the dataset.

$$\sigma = \sqrt{\sigma^2} \tag{5}$$

Algorithm:

1. Compute the variance of the time series.
2. Take the square root of the variance.

2.6 Skewness

Skewness measures the asymmetry of the dataset. A positive skewness indicates a longer tail on the right side of the distribution, while a negative skewness indicates a longer tail on the left side.

$$\text{Skewness} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^3}{\sigma^3} \tag{6}$$

Algorithm:

1. Compute the mean and standard deviation of the time series.
2. Calculate the skewness using the formula above.

2.7 Kurtosis

Kurtosis measures the tailedness of the dataset. A higher kurtosis indicates heavier tails in the distribution, while a lower kurtosis indicates lighter tails.

$$\text{Kurtosis} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^4}{\sigma^4} \tag{7}$$

Algorithm:

1. Compute the mean and standard deviation of the time series.
2. Calculate the kurtosis using the formula above.

2.8 Min Max Range

The min-max range is the difference between the maximum and minimum values in the dataset. It provides a measure of the spread of the data.

$$\text{min-max range} = \max(x) - \min(x) \tag{8}$$

Algorithm:

1. Find the maximum and minimum values in the time series.
2. Calculate the difference between the maximum and minimum values.

2.9 Quantiles

Quantiles divide the dataset into equal parts. The median is the 50th percentile, while the quartiles divide the dataset into four equal parts. Quantiles provide a measure of the spread and distribution of the data.

Algorithm:

1. Sort the time series.
2. Calculate the quantiles based on the desired percentage.

2.10 Sum

The sum of the dataset provides a measure of the total magnitude of the data.

$$\text{Sum} = \sum_{i=1}^{N} x_i \tag{9}$$

Algorithm:

1. Sum all elements in the time series vector.

2.11 Absolute Energy

Absolute energy is the sum of the squared values in the dataset. It provides a measure of the total energy in the data.

$$\text{Absolute Energy} = \sum_{i=1}^{N} x_i^2 \tag{10}$$

Algorithm:

1. Square each element in the time series.
2. Sum the squared values.

2.12 Rolling Mean

The rolling mean (or moving average) is the average of a specified number of previous values in the dataset. It provides a smoothed version of the time series data.

$$\text{Rolling Mean} = \frac{1}{N} \sum_{i=t-N+1}^{t} x_i \tag{11}$$

Algorithm:

1. Initialize a window of size N.
2. For each time step t, calculate the mean of the previous N values.
3. Slide the window one step forward and repeat.
4. Continue until the end of the time series is reached.
5. Return the rolling mean values.

2.13 Rolling Variance

The rolling variance is the variance of a specified number of previous values in the dataset. It provides a measure of the variability of the data over time.

$$\text{Rolling Variance} = \frac{1}{N} \sum_{i=t-N+1}^{t} (x_i - \mu)^2 \tag{12}$$

Algorithm:

1. Initialize a window of size N.
2. For each time step t, calculate the variance of the previous N values.
3. Slide the window one step forward and repeat.
4. Continue until the end of the time series is reached.
5. Return the rolling variance values.

2.14 Expanding Sum

The expanding sum is the cumulative sum of the dataset up to the current time step. It provides a measure of the total magnitude of the data over time.

$$\text{Expanding Sum} = \sum_{i=1}^{t} x_i \tag{13}$$

Algorithm:

1. Initialize a variable to store the cumulative sum.
2. For each time step t, add the current value to the cumulative sum.
3. Continue until the end of the time series is reached.
4. Return the expanding sum values.

2.15 Exponential Moving Average

The exponential moving average (EMA) is a weighted average of the previous values in the dataset, where more recent values have a higher weight. It provides a smoothed version of the time series data that reacts more quickly to recent changes.

$$\text{EMA}_t = \alpha x_t + (1 - \alpha)\text{EMA}_{t-1} \tag{14}$$

where α is the smoothing factor, typically set to $\frac{2}{N+1}$, where N is the window size.

Algorithm:

1. Initialize the EMA with the first value of the time series.
2. For each subsequent time step t, calculate the EMA using the formula above.
3. Continue until the end of the time series is reached.
4. Return the EMA values.

2.16 Sliding Window Entropy

Sliding window entropy is a measure of the complexity or uncertainty of the dataset over a specified window size. It provides a measure of the information content in the data.

$$\text{Entropy} = -\sum_{i=1}^{N} p_i \log(p_i) \tag{15}$$

where p_i is the probability of each value in the dataset.

Algorithm:

1. Initialize an empty vector to store entropy values.
2. Validate inputs: return empty vector if window size is 0, window size exceeds series length, or bins is 0.

3. For each position i from 0 to (n - window):
   - (a) Extract a slice of the series from i to i + window.
   - (b) Find the minimum and maximum values in the window slice.
   - (c) Calculate the range (max_value - min_value).
   - (d) If the range equals 0, add 0.0 to the entropy vector and continue to the next iteration.
   - (e) Otherwise:
     - i. Initialize a histogram with 'bins' number of bins, each set to 0.
     - ii. For each value in the window:
       - A. Determine which bin the value belongs to using the formula: bin = floor((value - min_value) / range * bins).
       - B. Ensure the bin index doesn't exceed the number of bins.
       - C. Increment the count for that bin.
     - iii. Calculate entropy as follows:
       - A. For each bin with count > 0, calculate probability $p = count/window\_size$.
       - B. Sum up $(-p * \ln(p))$ for all bins with counts > 0.
     - iv. Add the calculated entropy value to the entropy vector.
4. Return the vector of entropy values.

2.17 Fractional Variability

Fractional variability is a measure of the variability of the dataset relative to its mean. It provides a normalized measure of the spread of the data.

$$\text{Fractional Variability} = \frac{\sigma}{\mu} \tag{16}$$

For real world data where there is an associated uncertainty with each data point, the fractional variability is given by:

$$Fvar = \sqrt{(S - \langle \sigma_{err} \rangle)}/\langle x \rangle \tag{17}$$

where σ_{err} is the uncertainty associated with each data point x_i. Algorithm:

1. Compute the mean and standard deviation of the time series.
2. Calculate the fractional variability using the formula above.
3. If the time series has associated uncertainties, compute the fractional variability using the second formula.

2.18 Minimum Variability Timescale

Minimum variability timescale is a measure of the shortest timescale over which the dataset exhibits significant variability. It provides insight into the temporal characteristics of the data.

The variability timescale quantifies how rapidly a signal changes over time, providing a characteristic timescale for variation in time series data.

The variability timescale (τ) is calculated from adjacent pairs of measurements using:

$$\tau = \frac{\Delta t}{|\ln(f_2/f_1)|} \tag{18}$$

where:

- f_1 and f_2 are the two adjacent measurements.
- Δt is the time difference between the two measurements.

The minimum variability timescale is the minimum value of τ calculated from all adjacent pairs of measurements in the time series.

Algorithm:

1. Validate input: ensure time, flux, and flux error arrays each contain at least 2 elements
2. For each adjacent pair of measurements (i, $i+1$):
   (a) Calculate time difference: $dt = time[i+1] - time[i]$
   (b) Skip if $dt \leq 0$ (ensuring time increases)
   (c) Skip if either flux value is non-positive (requirement for logarithm)
   (d) Calculate flux ratio: $\frac{f_2}{f_1}$
   (e) Calculate absolute logarithmic difference: $|\ln(f_2/f_1)|$
   (f) Skip if no variability is detected ($|\ln(f_2/f_1)| = 0$)
   (g) Calculate combined measurement error: $\sqrt{error_1^2 + error_2^2}$
   (h) Skip if flux difference is less than combined error (statistically insignificant)
   (i) Calculate $\tau = \frac{dt}{|\ln(f_2/f_1)|}$
   (j) Update minimum τ if this is the smallest value found
3. Return the minimum τ value as the characteristic variability timescale

Only flux changes that are statistically significant (i.e., the absolute difference between flux measurements exceeds the combined measurement error) are considered when calculating the variability timescale. This ensures that the calculated timescale reflects genuine signal variations rather than measurement noise.

3 Frequency Domain Features

Here, we discuss the list of frequency domain features extracted by the ChronoXtract package.

3.1 Fourier Transform

The Fourier Transform is a mathematical technique that transforms a time-domain signal into its frequency-domain representation. It decomposes a signal into its constituent frequencies, allowing us to analyze the frequency content of the signal. The Fourier Transform is widely used in various fields, including signal processing, image analysis, and data compression.

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt \tag{19}$$

Where:

- $X(f)$ is the Fourier Transform of the signal $x(t)$
- f is the frequency
- t is time
- j is the imaginary unit
- e is the base of the natural logarithm
- π is the mathematical constant pi
- dt is the differential element of time

The Fourier Transform can be computed using the Fast Fourier Transform (FFT) algorithm, which is an efficient way to calculate the Fourier Transform of a discrete signal. The FFT algorithm reduces the computational complexity from $O(N^2)$ to $O(N \log N)$, making it suitable for real-time applications.

FFT Algorithm:

1. Check if the length of the input sequence is a power of 2. If not, pad with zeros to the next power of 2.
2. Base case: If the input sequence has length 1, its FFT is itself.
3. Divide the input sequence into two subsequences:
    (a) Even-indexed elements: $x[0], x[2], x[4], \ldots$
    (b) Odd-indexed elements: $x[1], x[3], x[5], \ldots$
4. Recursively compute the FFT of the even-indexed subsequence.
5. Recursively compute the FFT of the odd-indexed subsequence.
6. Combine the results using the butterfly operation:
    (a) For $k = 0$ to $N/2 - 1$:
        i. $X[k] = E[k] + e^{-j2\pi k/N} \cdot O[k]$

   ii. $X[k + N/2] = E[k] - e^{-j2\pi k/N} \cdot O[k]$

7. Return the combined array X, which is the FFT of the original sequence.

Where $E[k]$ is the k-th element of the FFT of even-indexed elements, and $O[k]$ is the k-th element of the FFT of odd-indexed elements.

3.2 Lomb Scargle Periodogram

The Lomb-Scargle periodogram is a method for detecting periodic signals in unevenly sampled data. It is particularly useful in astronomy and other fields where data may be collected at irregular intervals. The Lomb-Scargle periodogram estimates the power spectral density of a signal, allowing us to identify the dominant frequencies present in the data. The Lomb-Scargle periodogram is defined as:

Given:

1. A set of observation times t_i
2. Corresponding measurements y_i
3. A range of frequencies f

The angular frequency is:

$$\omega = 2\pi f$$

Tau τ is:

$$\tau = \frac{1}{2\omega} \tan^{-1} \left(\frac{\sum_i \sin(2\omega t_i)}{\sum_i \cos(2\omega t_i)} \right)$$

Step 2: Compute sums for sine and cosine components

Define:

$$\phi_i = \omega(t_i - \tau)$$

Then compute:

$$\sum y_i \cos(\phi_i), \quad \sum y_i \sin(\phi_i), \quad \sum \cos^2(\phi_i), \quad \sum \sin^2(\phi_i)$$

Step 3: Compute the Lomb-Scargle power

The normalized power at frequency f is:

$$P(f) = \frac{1}{2} \left(\frac{\left(\sum y_i \cos(\phi_i)\right)^2}{\sum \cos^2(\phi_i)} + \frac{\left(\sum y_i \sin(\phi_i)\right)^2}{\sum \sin^2(\phi_i)} \right)$$

Lomb-Scargle assumes your data can be approximated by a sinusoidal model:

$$y(t) = A\cos(\omega t) + B\sin(\omega t) + \epsilon$$

Where:

- $y(t)$ is the observed data
- A and B are the amplitudes of the cosine and sine components
- ω is the angular frequency
- ϵ is the noise term

Rather than directly estimating A and B, it analytically calculates how well sine and cosine functions of each frequency explain the observed data.

The τ parameter corrects for phase shifts caused by uneven sampling, making the periodogram invariant to time translation. This ensures that the result depends only on the frequency content of the data, not on the arbitrary choice of time zero.

References