**eda\_autocorr.py**

The eda\_autocorr function is to measure the autocorrelation of the EDA signal.

"How similar is the signal now to how it was in the past?"

* **If the autocorrelation is high (close to +1)**, it means the signal is very predictable or rhythmic. The signal's pattern at one point in time is very similar to its pattern after a specific time delay (lag).
* **If the autocorrelation is low (close to 0)**, it means the signal is more random and less predictable. The signal's pattern now has very little relationship to its pattern in the past.
* **If the autocorrelation is negative (close to -1)**, it means the signal is anti-correlated. For example, a peak now corresponds to a trough after the time delay.

**Imports:**

**import pandas as pd**

**from ..signal import signal\_autocorr**

* **pandas as pd**: Used to handle table-like structures called DataFrames.
* **signal\_autocorr**: Performs the autocorrelation calculation.

Function Definition

def eda\_autocorr(eda\_cleaned, sampling\_rate=1000, lag=4):

* **eda\_autocorr -> eda\_cleaned**: The input signal. A cleaned EDA signal, meaning noise and artifacts have already been removed.
* **sampling\_rate=1000**: The number of data points recorded per second (Hz).
* lag=4: The time delay (in seconds) that you want to compare the signal against.

Docstring:

“””EDA Autocorrelation”””

* Docstring (user manual)

**Sanity Check & Formatting:**

**if isinstance(eda\_cleaned, pd.DataFrame):**

**columns = eda\_cleaned.columns.values**

**if len([i for i in columns if "EDA\_Clean" in i]) == 0:**

**raise ValueError(...)**

**else:**

**eda\_cleaned = eda\_cleaned["EDA\_Clean"]**

* This is the first sanity check and formatting step.
* It checks if you have provided a pandas DataFrame as input.
* If so, it searches for a column named EDA\_Clean to make sure it's working with the correct data.
* If it can't find the column, it stops and raises an error.
* If it finds the column, it extracts just that column to use for the calculation.

if isinstance(eda\_cleaned, pd.Series):

eda\_cleaned = eda\_cleaned.values

* If the input is a pandas Series (a single column), it converts it into a simple NumPy array, the standard format for numerical calculation.

**Autocorrelation Calculation**

lag\_samples = lag \* sampling\_rate```

\* The `signal\_autocorr` function needs to know the delay in terms of the number of data points (samples).

\* \*\*Example\*\*: If the `sampling\_rate` is 100 Hz (samples/second) and the `lag` is 4 seconds, then:

\* `lag\_samples = 4 \* 100 = 400 samples`

```python

if lag\_samples > len(eda\_cleaned):

raise ValueError(

"exceeds the duration of the eda signal."

)```

\* This is a sanity check. You can't shift the signal by a delay that is longer than the signal itself.

\* \*\*Example\*\*: You can't compare a 10-second signal to how it was 20 seconds ago.

```python

cor, \_ = signal\_autocorr(eda\_cleaned, lag=lag\_samples)

* This calls the signal\_autocorr function.
* It passes the eda\_cleaned signal and the calculated lag\_samples.
* This function performs the mathematical correlation and returns two values. We are only interested in the first one (the correlation value), so we store it in the cor variable. The \_ is a common Python convention for a variable whose value we want to ignore.

return cor

The function returns the single floating-point number (cor)