**env**:

Windows10

Python3.8.10

**Concepts**:

This content is based on a study conducted under the assumption of a strategic model that shows consistent and reliable results, represented by a steadily upward-sloping graph with a constant return rate.

# **1\_generate\_trade\_result.py**

This involves editing the provided sample data to save it by partitions, which are organized by the end-of-year partitioning. At this stage, a feature was added to allow each data set to be **resampled** to a **higher time frame** specified by the user, enabling the use of higher time frames. The reason for allowing the Higher Time Frame feature is to enable the exploration of diversified swing models.

# **2\_generate\_trade\_result.py**

**Points** that generate specific trading signals, as well as price information such as entry price, buying price, selling price, stop-loss price, and take-profit price, are provided through the **PriceBox**. Finally, the standardization of trading patterns using r**isk-to-reward ratios** and profitability ratios was specified.

Under the assumption that a consistent upward-sloping graph can be generated if a certain level of win rate is achieved based on specific risk-to-reward ratios in the market, we aimed to extract symbols that consistently possess such characteristics.

The compiled **Trade Result** is used to extract valid condition information, specifically combinations of Point PriceBox and the RR Ratio, which represents the risk-to-reward ratio.

# **3\_generate\_payload.py**

Another hypothesis used here is to split the input data, specifically the entire period of the time series data, into **Train, Validation, and Test Sets** at a fixed ratio. The symbols extracted through the training process are then input into the Validation Set, and only valid cases proceed to the Test phase.

**The evaluation of payload consistency used the R2 score based on simple and exponential returns, under the assumption that reliability arises from a consistent slope.**

# **4\_load\_payload.py & 5\_set\_payload.py**

The next step is to convert the extracted conditions into a payload format. This process requires several analytical tools. Therefore, temporary parameter values coded in the program have been organized (**set\_configurations()**). Using these values, analysis is conducted to extract symbols that exhibit specific patterns with consistent regularity.

**Performance metrics:**

1. r2-score

2. profit

**Assumption used**:

Year info, ex) 2024: The actual data will include the **year** attribute.

**Given additional time:**

1. Automating thresholding in pivoting.

2. Pay special attention to overnight positions.

3. Verifying results based on additional data:

- HTF (Higher Time Frame)

- Diversified strategy setting outcomes.