Efficient AI with Rust Lab
Rapid Time Series Datasets Library
RWTH Aachen University

Group 1

Marius Kaufmann[1] Amir Ali Aali[2] Kilian Fin Braun[1]

[1]Masters of Computer Science
[2]Masters of Data Science

19th Jul, 2025

Marius’s Part

Downsampling I

Goal: Reduce the number of data points in a time series dataset.

Downsampling I

Goal: Reduce the number of data points in a time series dataset.

Benefits:

- Reduces memory usage

Downsampling I

Goal: Reduce the number of data points in a time series dataset.

Benefits:

- Reduces memory usage
- Speeds up processing time

Downsampling I

Goal: Reduce the number of data points in a time series dataset.

Benefits:

- Reduces memory usage
- Speeds up processing time

Neccessary parameter when downsampling:

- Downsampling factor: How many data points to skip

Downsampling I

Goal: Reduce the number of data points in a time series dataset.

Benefits:

- Reduces memory usage
- Speeds up processing time

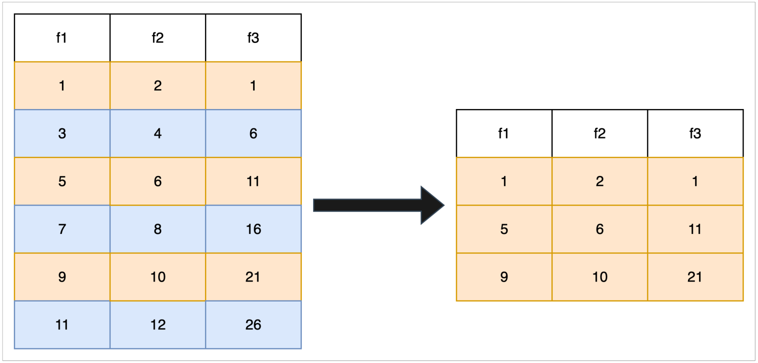
Neccessary parameter when downsampling:

- Downsampling factor: How many data points to skip

Example:

- Downsampling factor of 2: Every second data point is kept as shown in Figure 1

Downsampling II



f1	f2	f3
1	2	1
3	4	6
5	6	11
7	8	16
9	10	21
11	12	26

f1	f2	f3
1	2	1
5	6	11
9	10	21

Downsampling example with a factor of 2

Downsampling III

How it works:

- The downsampling function takes a time series dataset and a downsampling factor as input.

Downsampling III

How it works:

- The downsampling function takes a time series dataset and a downsampling factor as input.
- It iterates over the dataset and keeps every n-th data point, where n is the downsampling factor.

Downsampling III

How it works:

- The downsampling function takes a time series dataset and a downsampling factor as input.
- It iterates over the dataset and keeps every n-th data point, where n is the downsampling factor.

Bottleneck of passing the data by reference:

- Not possible. A copy is needed.

Downsampling III

How it works:

- The downsampling function takes a time series dataset and a downsampling factor as input.
- It iterates over the dataset and keeps every n-th data point, where n is the downsampling factor.

Bottleneck of passing the data by reference:

- Not possible. A copy is needed.
- Creating view only possible on contiguous data.

Downsampling III

How it works:

- The downsampling function takes a time series dataset and a downsampling factor as input.
- It iterates over the dataset and keeps every n-th data point, where n is the downsampling factor.

Bottleneck of passing the data by reference:

- Not possible. A copy is needed.
- Creating view only possible on contiguous data.
- Downsampling does not yield a contiguous data structure.

Splitting I

Goal: Split a time series dataset into three parts: training, validation, and test.

Splitting I

Goal: Split a time series dataset into three parts: training, validation, and test.

Different splitting strategies:

- Random split (Classification Data)

Splitting I

Goal: Split a time series dataset into three parts: training, validation, and test.

Different splitting strategies:

- Random split (Classification Data)
- In-Order split (Classification Data)

Splitting I

Goal: Split a time series dataset into three parts: training, validation, and test.

Different splitting strategies:

- Random split (Classification Data)
- In-Order split (Classification Data)
- Temporal split (Forecasting Data)

Splitting I

Goal: Split a time series dataset into three parts: training, validation, and test.

Different splitting strategies:

- Random split (Classification Data)
- In-Order split (Classification Data)
- Temporal split (Forecasting Data)

Neccessary parameter when splitting:

- Training set ratio

Splitting I

Goal: Split a time series dataset into three parts: training, validation, and test.

Different splitting strategies:

- Random split (Classification Data)
- In-Order split (Classification Data)
- Temporal split (Forecasting Data)

Neccessary parameter when splitting:

- Training set ratio
- Validation set ratio

Splitting I

Goal: Split a time series dataset into three parts: training, validation, and test.

Different splitting strategies:

- Random split (Classification Data)
- In-Order split (Classification Data)
- Temporal split (Forecasting Data)

Neccessary parameter when splitting:

- Training set ratio
- Validation set ratio
- Test set ratio

Splitting II (Random Split - Classification Data)

How it works:

1. Validate the proportions of train, validation, and test sets.

Splitting II (Random Split - Classification Data)

How it works:

1. Validate the proportions of train, validation, and test sets.
2. Shuffle the dataset randomly.

Splitting II (Random Split - Classification Data)

How it works:

1. Validate the proportions of train, validation, and test sets.
2. Shuffle the dataset randomly.
3. Compute the split offsets based on the proportions.

Splitting II (Random Split - Classification Data)

How it works:

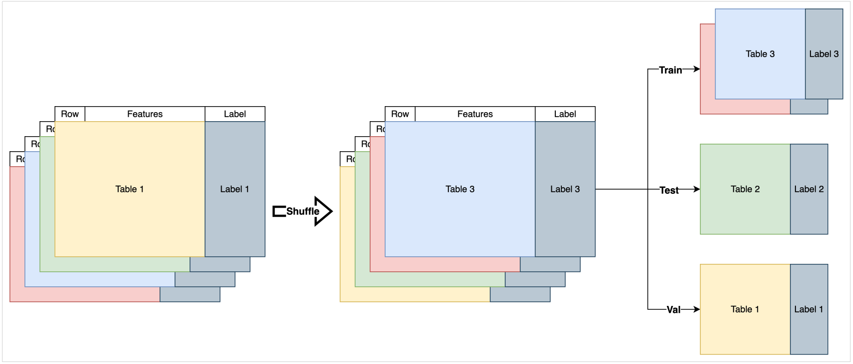
1. Validate the proportions of train, validation, and test sets.
2. Shuffle the dataset randomly.
3. Compute the split offsets based on the proportions.
4. Split the instances into three sets.

Splitting II (Random Split - Classification Data)

How it works:

1. Validate the proportions of train, validation, and test sets.
2. Shuffle the dataset randomly.
3. Compute the split offsets based on the proportions.
4. Split the instances into three sets.
5. Return the three sets as separate datasets.

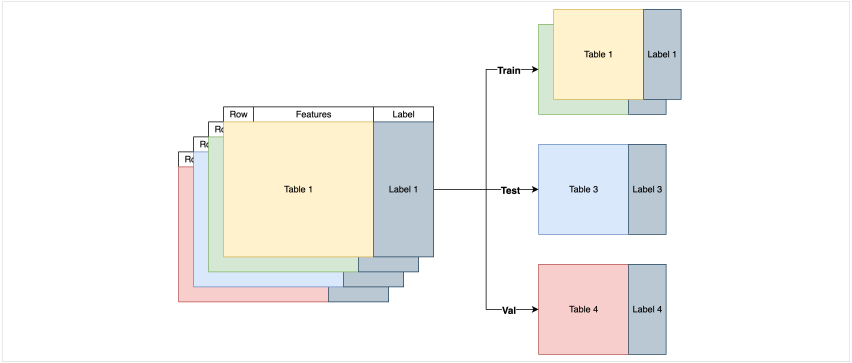
Splitting III (Random Split - Classification Data)



Random split example

Splitting IV (In-Order Split - Classification Data)

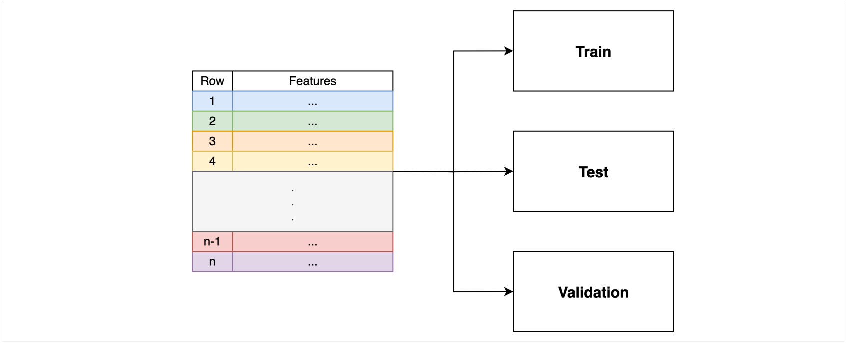
Works very similar to the random split, but it **doesn't shuffle** the dataset anymore.



In-Order split example

Splitting V (Temporal Split - Forecasting Data)

Similar to the in-order split, but this time we are dealing with forecasting data, which in most cases is only one instance and we split over **timestamps** and not isntances anymore.



Temporal split example

Kilian’s Part