
CERN

Transformation ML Framework

on basis of the XBox2 Data Set

Lorenz Fischl

Introduction

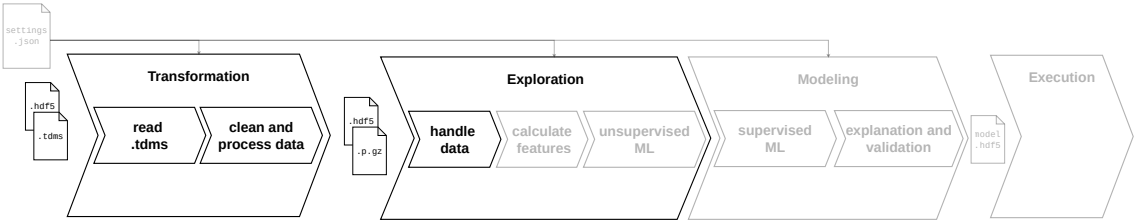
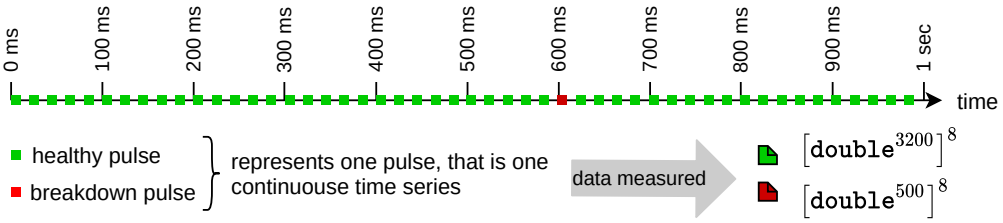


Table of Contents

XBox2 Data

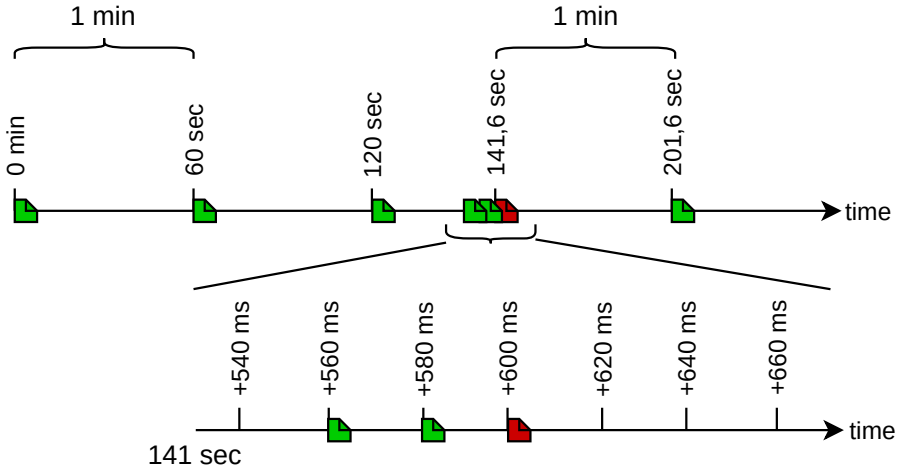
Every 20*ms* a pulse is sent into the RF cavity for particle acceleration. Sometimes an arc forms. Those events are called breakdown.

vspace1cm



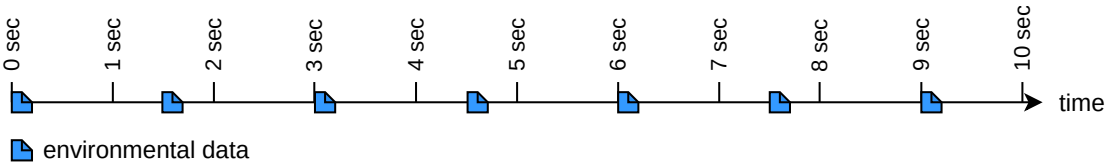
XBox2 EventData

A log group of one pulse is stored every minute.
When a breakdown happens the corresponding log group + the two prior log groups are stored.



XBox2 TrendData

35 values about the environmental conditions (that don't change rapidly) are stored roughly every 1,5 sec.
All TrendData of one day is stored in 1-2 groups.



Requirements

We need a data format that

- is compatible between systems (SWAN, htcondor, locally)
- can read data column wise fast
- can be Transformed and handled (sorted/filtered/processed) relatively easy
- can be transformed into `pandas` DataFrame
- outputs `numpy` arrays for machine learning

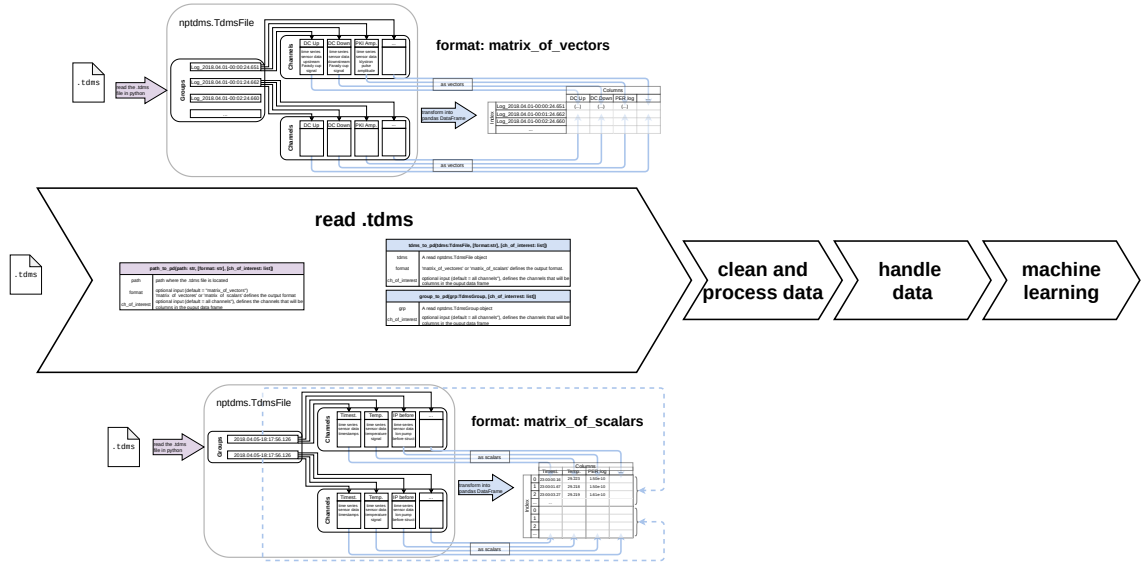
nptdms

- package `nptdms` can read and handle `.tdms` files
- very slow (ex.: read of a 100MB file can take >30sec)
- very space inefficient (ex. TrendData: 20,5 GB in .tdms $\rightarrow$ 2.8 GB of data)

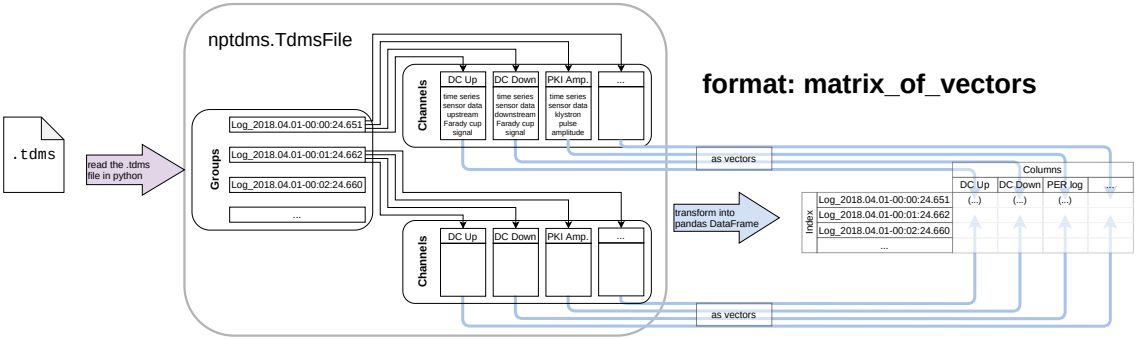
Transformation

`.tdms` $\longrightarrow$ `pd df/ dictionary` $\longrightarrow$ `pickle` $\longrightarrow$ `.gzip`

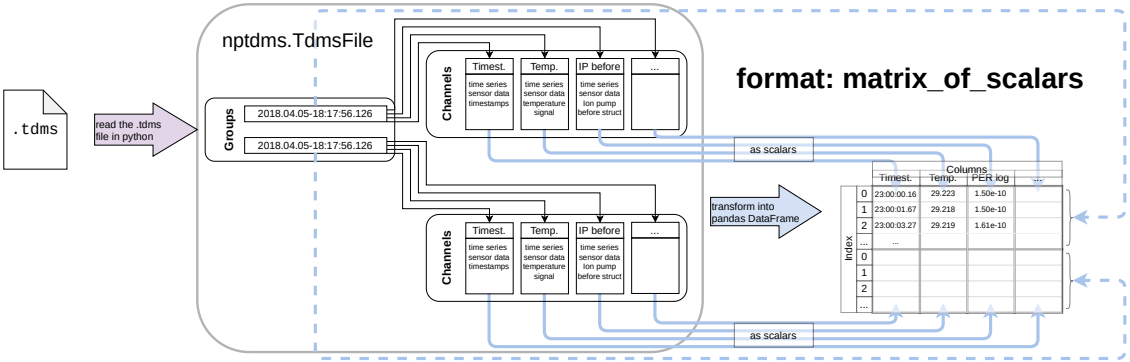
Transformation: read.tdms



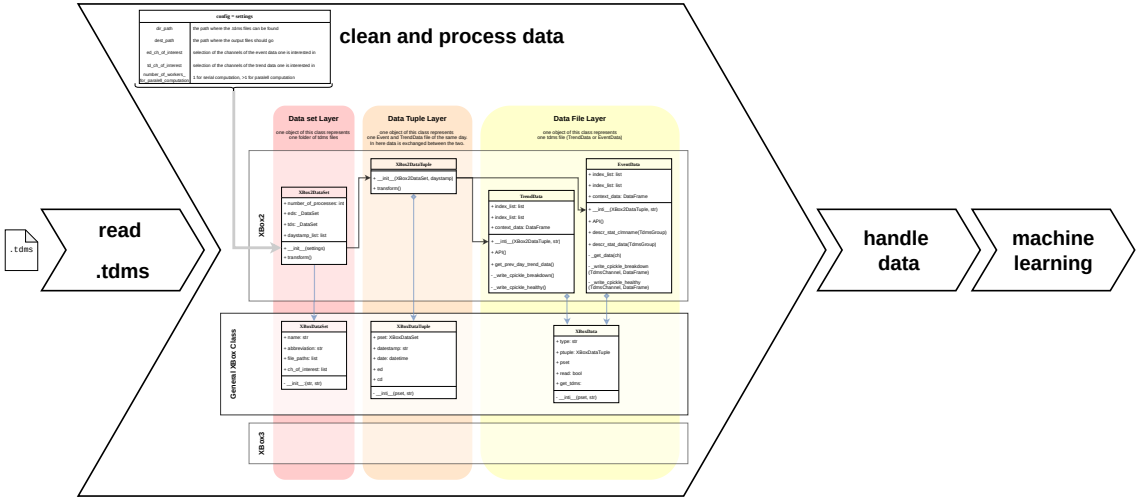
Transformation: read.tdms

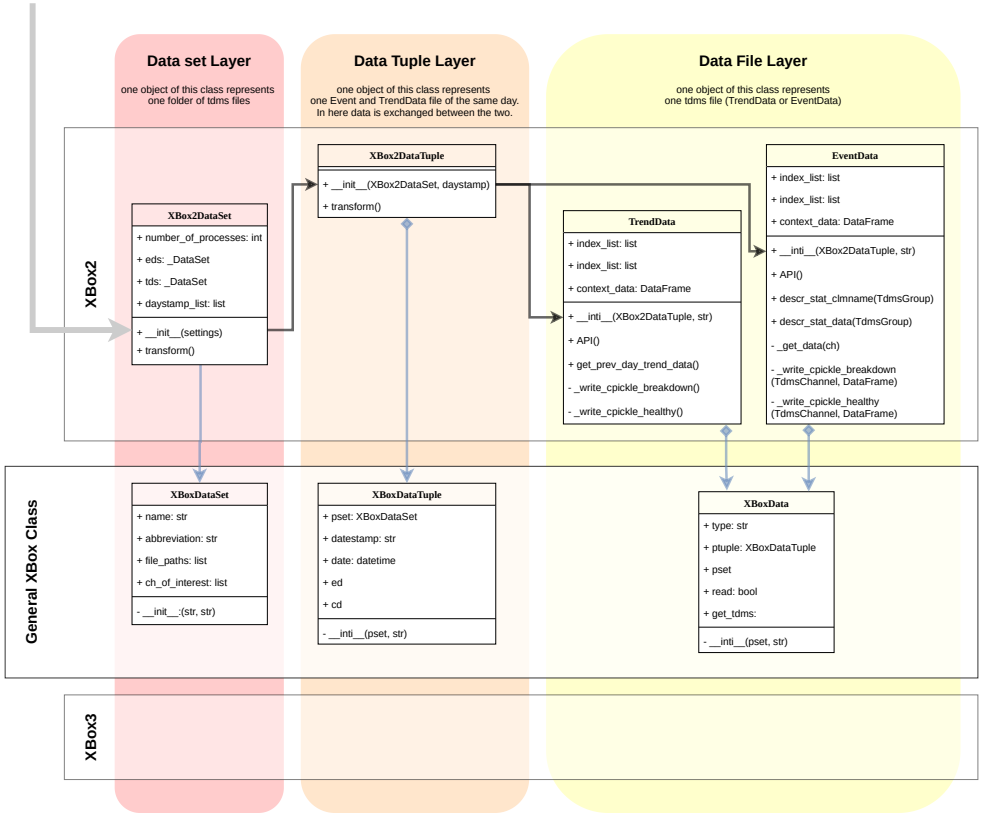


Transformation: read.tdms



Transformation: Clean and Process data with classes




Data set Layer
one object of this class represents one folder of tdms files
Data Tuple Layer
one object of this class represents one Event and TrendData file of the same day. In here data is exchanged between the two.
Data File Layer
one object of this class represents one tdms file (TrendData or EventData)
XBox2
XBox2DataSet
+ number_of_processes: int
+ eds: _DataSet
+ tds: _DataSet
+ daystamp_list: list
+ __init__(settings)
+ transform()
XBox2DataTuple
+ __init__(XBox2DataSet, daystamp)
+ transform()
TrendData
+ index_list: list
+ index_list: list
+ context_data: DataFrame
+ __inti__(XBox2DataTuple, str)
+ API()
+ get_prev_day_trend_data()
- _write_cpickle_breakdown()
- _write_cpickle_healthy()
EventData
+ index_list: list
+ index_list: list
+ context_data: DataFrame
+ __inti__(XBox2DataTuple, str)
+ API()
+ descr_stat_clmname(TdmsGroup)
+ descr_stat_data(TdmsGroup)
- _get_data(ch)
- _write_cpickle_breakdown (TdmsChannel, DataFrame)
- _write_cpickle_healthy (TdmsChannel, DataFrame)
General XBox Class
XBoxDataSet
+ name: str
+ abbreviation: str
+ file_paths: list
+ ch_of_interest: list
- __init__:(str, str)
XBoxDataTuple
+ pset: XBoxDataSet
+ datestamp: str
+ date: datetime
+ ed
+ cd
- __inti__(pset, str)
XBoxData
+ type: str
+ ptuple: XBoxDataTuple
+ pset
+ read: bool
+ get_tdms:
- __inti__(pset, str)
XBox3

Summary

- pandas DataFrame are easy to use in notebooks
- pickle speeds up reading time
- with compression takes up less space

- version issues with pickle protocol
- part of the process should not be in the Datahanlder instead of the Transformation
- EventData and TrenadData are stored differently
- channel properties are lost
- data was changed in place in notebooks in retrospect

.hdf5

- Hierarchical Data Format with a Unix file system like tree structure
- build for data management and storage
- keys work like pointers to the memory, channel wise reading possible
- in `nptdms` the function `to_hdf()` exists
- no data is lost in the Transformation, also properties are stored

Conclusion

- I implemented a generic class for converting `.tdms` files into `pd.df+cpickle`
- is there a better data format, maybe `.hdf5`?

	`nptdms`	`pd.df+cpickle`		`.hdf5`	
		w/o zip	w zip	w/o zip	w/ zip
space (GByte)	20.5GB	2.8GB	1GB	2.8GB	1GB
read (TD 1 channel)	$\sim$ 60min	4sec	12sec	0.5 sec	
read (TD 3 channels)	$\sim$ 60min	4sec	12sec	1 sec	
read (TD 15 channels)	$\sim$ 60min	4sec	12sec	4 sec	
feature calc. (ED)	$>$ 15min		7sec	8 sec	

home.cern