SoftwareX 12 (2020) 100518

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

MVTS-Data Toolkit: A Python package for preprocessing multivariate time series data

Azim Ahmadzadeh *, Kankana Sinha, Berkay Aydin, Rafal A. Angryk

Computer Science Department, Georgia State University, Atlanta, GA 30302, United States of America

ARTICLE INFO

Article history:
Received 24 January 2020
Received in revised form 4 May 2020
Accepted 19 May 2020

Keywords:
Feature extraction
Data analysis
Multivariate time series
Class imbalance
Multi-class
Sampling

ABSTRACT

We developed a domain-independent Python package to facilitate the preprocessing routines required in preparation of any multi-class, multivariate time series data. It provides a comprehensive set of 48 statistical features for extracting the important characteristics of time series. The feature extraction process is automated in a sequential and parallel fashion, and is supplemented with an extensive summary report about the data. Using other modules, different data normalization methods and imputation are at users' disposal. To cater the class-imbalance issue, that is often intrinsic to real-world datasets, a set of generic but user-friendly, sampling methods are also developed.

Code metadata

C1	Current code version	v0.2.6
C2	Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_15
C3	Legal Code License	MIT license (MIT)
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	Python
C6	Compilation requirements, operating environments & dependencies	Python $\geq$ 3.6
C7	If available Link to developer documentation/manual	dmlab.cs.gsu.edu/docs/mvtsdata_toolkit/
C8	Support email for questions	aahmadzadeh1@cs.gsu.edu

Software metadata

S1	Current software version	v0.2.6.
S2	Permanent link to executables of this version	pypi.org/project/mvtsdatatoolkit/
S3	Legal Software License	MIT license (MIT)
S4	Computing platforms/Operating Systems	OS-agnostic
S5	Installation requirements & dependencies	See `requirements.txt`
S6	If available, link to user manual–if formally published include a reference to the publication in the reference list	bitbucket.org/gsudmlab/mvtsdata_toolkit/src/master/README.md
S7	Support email for questions	aahmadzadeh1@cs.gsu.edu

1. Motivation and significance

1.1. A research tool for multivariate time series

Time series is one of the first data types that has been introduced and heavily used even before the emergence of the digital world, in the form of sheets of numeric and categorical values. When several variables on the subject of study are observed and recorded simultaneously, the result essentially becomes multivariate time series data (hereafter abbreviated to 'mvts' data). A quick perusal of the literature reveals that most of the studies utilizing such a data type, share a set of preprocessing routines such as distribution analysis of the raw data [1], time series feature engineering [2], feature extraction [3], getting a set of summary statistics from the extracted features and visualization of the summary statistics, treatment of the missing and invalid

* Corresponding author.
E-mail address: aahmadzadeh1@cs.gsu.edu (A. Ahmadzadeh).
URL: http://azim-a.com (A. Ahmadzadeh).

https://doi.org/10.1016/j.softx.2020.100518

values, normalization over variables, and proper undersampling and oversampling of a given dataset. A careful implementation of these steps is often a time consuming endeavor and due to lack of a comprehensive and rigorous testing of the code, as it is often not in the capacity of individual researchers to create production-ready, fully-tested, and deployable software products, small errors in generating useful features can go unnoticed; leading to inconclusive or altogether incorrect interpretations of the results. To this end, we decided to expand the implementation of our recent research project [4] and share it as a domain-independent toolkit for a broader audience in the data-driven, research community.

1.2. Initial motivation

This project was initially developed for preprocessing of a specifically challenging mvts dataset, named Space Weather ANalytics for Solar Flares, in short SWAN-SF [5]. The dataset is created as a benchmark to be served for standardization of the ambitious task of flare forecasting. However, the challenges, for which we implemented the automated pre-processor, are not specific to any particular domain, task, or dataset. Therefore, we decided to expand it to a general-purpose, Python package, that is well-documented and properly tested, for extraction and analysis of mvts.

1.3. Comparison with existing tools

This toolkit is comparable with a few other open-source, Python packages that are already available. The most recent one of them is Time Series Feature Extraction Library (abbreviated to TSFEL) [6]. This package is solely dedicated to the automatic feature-extraction functionality for over 60 statistical, temporal, and spectral features appropriate for time series data. An interesting aspect of TSFEL is that it allows users to add personalized features to the existing ones using a JSON format. The current version of TSFEL, however, does not support parallel processing; an important capability that is often essential when dealing with large datasets of mvts and complex statistical features with expensive execution time. Another package that is more invested in the statistical and computational aspects of feature extraction is Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (in short, tsfresh) [3]. Although, it does not seem to support mvts,[1] it is equipped with both parallel and distributed computation capabilities, as well as customization of features by users. tsfresh allows extraction of many features based on 63 time series characteristics and also provides automatic hypothesis testing to help the feature selection process.

Comparing the above-mentioned packages with ours, MVTS-Data Toolkit provides both parallel and sequential, feature extraction, on multi-class, mvts datasets. It also gives statistical overview on each time series of the raw mvts dataset, as well as an automated analysis of the dataset of the extracted features. Moreover, two more preprocessing steps, namely normalization and sampling, are implemented in this toolkit. Although, we understand why other developers may find such steps out of scope of their packages, we found it very useful in our own research to be able to finalize our dataset using one single library before we start the learning phase. More specifically, since the program is already familiar with the data structure of the extracted features, it can seamlessly perform the other preprocessing steps, without any extra modification effort such as dealing with time-stamps, class-labels, and other non-numerical values in normalization.

There are other packages that are designed for analysis of time series but they have little or no intersection with our work (e.g., statsmodels [7]) and/or they are implemented in languages other than Python (e.g., hctsa [8] in MATLAB or MTS [9] in R). We exclude them from this comparison.

1.4. Limitations

It is worth noting that while MVTS-Data Toolkit provides a set of flexible and generic sampling techniques that can be easily called on the extracted features, more sophisticated approaches exist that may significantly improve the learning process of different models on datasets. Dealing with the class-imbalance issue is not particularly the objective of this package, and the implemented sampling methods are included only to provide quick remedies for this issue. In this regard, Imbalance-learn package [10] provides such methods and can be easily used on the `pandas.DataFrame` of the extracted features our toolkit produces.

At the current version of this package, normalization of the raw mvts data is not yet included. Although it is not difficult to implement the standard normalization methods on mvts data, the simplicity of such functionalities could result in misleading outcomes. To properly normalize a dataset, one would need to rigorously study the distribution of values and the outliers in each dimension of the data and make domain-specific and task-specific decisions regarding their normalization approach. For example, what is sometimes known as the "common practice", e.g., outlier removal based on the interquartile ranges, is often a naïve approach when it comes to the real-world datasets. To this end, we decided to not include any normalization module on the raw-data level. However, in the future releases, we may include tools that could pave the road for making decisions about normalization of any mvts dataset.

In the following sections, we present the design and application of MVTS-Data Toolkit. In Section 2 we first explain the architecture and the main components of the software, and then discuss the incorporated functionalities and the implementation details, such as the collection of statistical features, the sequential and parallel implementation of the mvts data analysis and the feature-extraction process, and other important preprocessing steps such as normalization and sampling of mvts datasets. In Section 3 we highlight the impact of this software from different angles. Finally, we conclude this work in Section 4.

2. Software description

MVTS-Data Toolkit is an open-source package implemented fully in Python, and is available at PyPI, a popular online repository called The Python Package Index, ready to be installed using the `pip install` command. This package is supplemented with detailed documentation that is embedded in the code and also available online (see Code metadata and Software metadata). The source code is publicly accessible in our Bitbucket repository in two permanent branches: the *master* branch that tracks the latest stable version of the software, and the *dev* branch where all the new developments, after they are implemented and tested in temporary branches, are merged into.

2.1. Software architecture

MVTS-Data Toolkit follows a simple architecture that gives priority to robustness, having a user-friendly interface for its classes and methods, and code extensibility. In the following sections, we highlight the important aspects of the architecture and design choices.

[1] see the conversation at https://github.com/blue-yonder/tsfresh/issues/566.

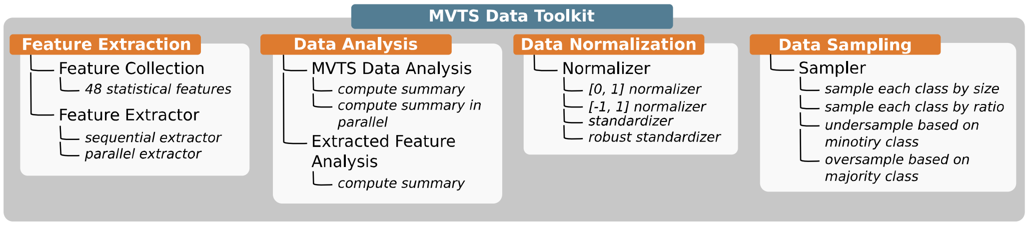

Fig. 1. The main components of MVTS-Data Toolkit package.

2.1.1. Main components

This package is composed of four main components that are illustrated in Fig. 1. The *Feature Extraction* component allows an easy and fast computation of a large set of statistical features on the dataset. The *Data Analysis* component gives high-level exploratory analyses for both the mvts (raw) datasets and the extracted features computed by the Feature Extraction component. The other two components, namely, *Data Normalization* and *Data Sampling*, as the names suggest, provide various preprocessing methods for data transformation, undersampling, and oversampling.

2.1.2. Scalability

The Feature Extraction and Data Analysis components are equipped with both the sequential and parallel execution capabilities. The incorporated parallelism, which comes without any additional complexity for the end users, is particularly useful when the dataset under study is considered to be 'large'.

The execution time of the parallel version is essentially linearly scalable with respect to the number of available processes. Moreover, the feature extraction process is designed to divide the input data into partitions, and then assign one partition to each process. Every process starts its job by receiving a list of file paths, instead of the actual files. As a result, the memory constraint is only determined by the size of the largest MVTS file. That is, as long as there is enough memory available for the largest MVTS to be loaded into, there will be no out-of-memory issue, regardless of the size of the entire dataset. This improves the robustness of the package's scalability.

The scalability mentioned above is only possible by the means of statistical techniques to obtain accurate estimations of the rank-based statistics the we would like to return. In other words, without loading the entire dataset into memory, it is only possible to estimate the percentiles of each time series. In this regard, we utilize an on-line algorithm, called t-digest [11], that accurately estimates the desired summary statistics by clustering real-valued samples and retaining the mean and the sample size. The clustering produces instances of a data structure called t-digest, which has a constant memory consumption as the algorithm proceeds iterations of estimations.

2.1.3. Dependencies of components

While the four components of the toolkit work independently, some components are designed to further process the output of the others, if desired. For example, MVTS Data Analysis package, from the Data Analysis component, gives general statistics about, and are based on, the raw MVTS data, and Feature Extractor package, from the Feature Extraction component, extracts a set of statistical features, also from the raw MVTS data. However, the Extracted Feature Analysis package, as the name suggests, analyzes the extracted features produced by the Feature Extraction component. The relation between these two components is illustrated in Fig. 2. Similarly, normalization and sampling provided by the other components also process the extracted features.

2.1.4. Global configuration

In order to provide a user-friendly package, a set of input arguments can be configured using `YAML` language prior to any execution. A `yml` configuration file can be stored at any directory in the host machine, as long as its path is passed to the class instance that needs such metadata, through a class-constructor. The keywords in this file are pre-defined, as described below:

- `PATH_TO_MVTS`: A relative or absolute path to where the mvts dataset is stored.
- `PATH_TO_EXTRACTED_FEATURES`: A relative or absolute path to where the extracted features will be stored (Feature Extraction component uses this path).
- `META_DATA_TAGS`: A list of tags based on which some pieces of information can be extracted from the file-names of the mvts. For example, if timestamps are encoded in the file-names, e.g., `_st[YYYY-MM-DD HH:MM:SS]`, then the string `st` (without brackets) is a tag that can be included in this list. In the feature extraction process, this adds what is wrapped in the square brackets in each filename, as an extra column to the data-frame of the extracted features. Generally, using this functionality, any extra metadata can be encoded in the file-names and consequently passed into the extracted features.
- `MVTS_PARAMETERS`: A list of parameter names that are used in the mvts dataset and their statistical features are of interest. These are, in other words, the column-names in the mvts files.
- `STATISTICAL_FEATURES`: A list of statistical features of interest to be extracted from the mvts. They must be chosen from the provided methods in the module `features.feature_collection.py`. For example, `get_min` is a valid feature-name as this method is implemented in the package.

The keywords documented in this manuscript may change as new releases of the package come out. We will update the online manual (the `README.md` file) in our repository based on the latest changes.

2.2. Software functionalities

MVTS-Data Toolkit provides an array of preprocessing routines applicable for any mvts dataset, to prepare them for further analyses, e.g., to be fed into machine learning algorithms. In the following sections, we give a high-level description of these functionalities.

2.2.1. MVTS statistical features

One of the primary contributions of MVTS-Data Toolkit lies in its comprehensive set of statistical features. The list of all statistical features that are available in this toolkit is given in Table 1, categorized in 9 groups. These features are collected from different domains of research, after a rigorous exploration over a large number of applied research on time series data.

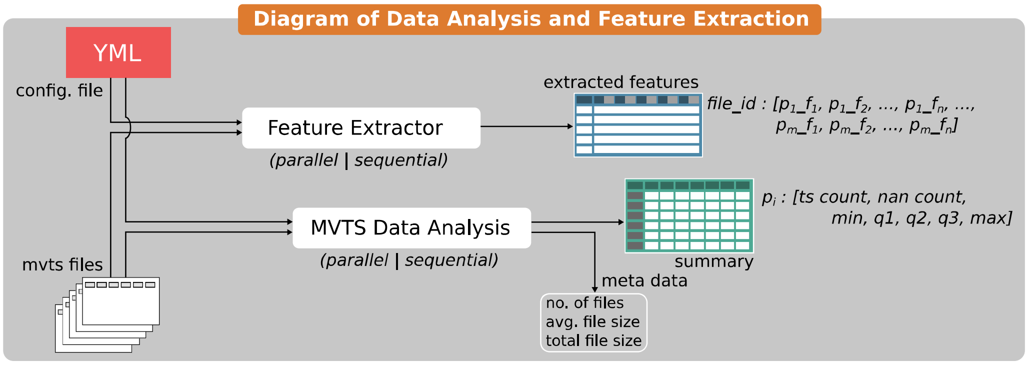

Fig. 2. Block diagram of the data analysis and feature extraction components.

Table 1
List of all statistical features available in MVTS-Data Toolkit package.

Group	Features	Description
1	$min(ts)$, $max(ts)$, $med(ts)$, $\mu(ts)$, $\sigma(ts)$, $sk(ts)$, $ku(ts)$	Descriptive statistics on the time series *ts*
2	$\lvert min(ts^{\dashv}) - min(ts^{\vdash})\rvert$, $\lvert max(ts^{\dashv}) - max(ts^{\vdash})\rvert$, $\lvert med(ts^{\dashv}) - med(ts^{\vdash})\rvert$, $\lvert\sigma(ts^{\dashv}) - \sigma(ts^{\vdash})\rvert$, $\lvert sk(ts^{\dashv}) - sk(ts^{\vdash})\rvert$, $\lvert ku(ts^{\dashv}) - ku(ts^{\vdash})\rvert$	Absolute differences between the descriptive statistics of the first and the second halves of the time series *ts*
3	$\lvert\{local\_minima(ts)\}\rvert$, $\lvert\{local\_maxima(ts)\}\rvert$, $\lvert\{local\_extrema(ts)\}\rvert$, $\lvert\{zero\_crossings(ts)\}\rvert$, $\mu\{local\_minima(ts)\}$, $\mu\{local\_maxima(ts)\}$, $\mu\{local\_maxima\_upsurges(ts)\}$, $\mu\{local\_minima\_downslides(ts)\}$	Several representations of time series in form of their extrema
4	$\mu(ts')$, $\sigma(ts')$, $sk(ts')$, $ku(ts')$	Descriptive statistics on derivative (i.e., windowing differences) of the time series
5	$\mu(\partial ts)$, $\sigma(\partial ts)$, $\sigma^2(\partial ts)$, $sk(\partial ts)$, $ku(\partial ts)$	Descriptive statistics on derivative (i.e., approximation of analytic gradient) of the time series
6	$lwa(ts)$, $qwa(ts)$, $\mu(\lvert ts'\rvert)$, $\mu(\lvert\partial ts\rvert)$	Linear and quadratic weighted average of times series, and changes of the derivatives
7	$\frac{\lvert\{p\in ts: p>0\}\rvert}{n}$, $\frac{\lvert\{p\in ts: p<0\}\rvert}{n}$	Positive and negative fractions of records in a times series of length n
8	$lv_{k=1}(ts)$, $\sum(lv_{k=10}(ts))$, $\mu(lv_{k=10}(ts))$	Description of time series in terms of their k last values ($lv_k(ts)$)
9	$longest\_positive\_run$, $longest\_negative\_run$, $longest\_monotonic\_increase$, $longest\_monotonic\_decrease$, $slope(longest\_monotonic\_increase)$, $slope(longest\_monotonic\_decrease)$, $\mu(\{slope(monotonic\_increases)\})$, $\mu(\{slope(monotonic\_decreases)\})$	Long-run trends of the time series *ts*

Notations. *ts*: time series, μ: mean, *med*: median, σ: standard deviation, *sk*: skewness, *ku*: kurtosis, $ts^{\dashv}$: first half of *ts*, $ts^{\vdash}$: second half of *ts*, $\{\cdot\}$: set, $\lvert\cdot\rvert$: absolute value(s), ts': difference derivative of *ts*, ∂ts: gradient derivative of *ts*, $\lvert\{\cdot\}\rvert$: set cardinality, lwa: linear weighted average, qwa: quadratic weighted average, lv_k: last k values.

They can, individually or in groups, describe many unique characteristics of time series parameters that have a reasonable degree of stochasticity. For more details on each of these features, we suggest the interested reader consult the documentation and open-sourced implementation of the toolkit (see the module `features.feature_collection.py`).

2.2.2. Data analysis

The toolkit provides a functionality to get some general insight about the dataset under study and the extracted features. The Data Analysis component gives summary on three different levels: (1) on the size and volume of the mvts files, (2) on the time series parameters; it gives the count of missing values, and the six-number summary, i.e., min, 1st quartile, mean, median, 3rd quartile, and max, for each parameter, and (3) on the extracted features using the similar statistics. The scalability in the design of this component, as discussed in Section 2.1, makes this a handy tool.

2.2.3. Normalization

The extracted-features data, calculated by the Feature Extraction component, often needs to go through a data transformation filter to be ready for further analyses or being fed into a machine learning model. The MVTS-Data Toolkit has a wrapper class around four data transformation methods implemented in the `sklearn.preprocessing` module in scikit-learn package. This wrapper assists the data normalization procedure such that users can normalize the extracted features without having to worry about the details such as dropping the non-numeric columns and appending them back to the data-frame after transformation of numeric columns; or protecting numeric columns that only preserve the uniqueness of each record (such as *id* or *index* columns) against such a transformation.

2.2.4. Sampling

A dataset is said to be *class imbalanced* when populations of the classes are of different sizes. In real-world problems the datasets are often imbalanced, i.e., the events of main interest occur significantly less frequently than others. This imbalance issue must be treated properly before it is passed into a machine learning model. One of the main resolutions is to impose a balance in populations of the classes, through means of random sampling. To this end, a sampler module is incorporated into this toolkit with several generic and flexible methods that together provide a large number of possibilities for tackling the class-imbalance issue. Fig. 3 illustrates some of these possible outcomes. For more details, see the documentation of the methods in the module `sampling.sampler.py`.

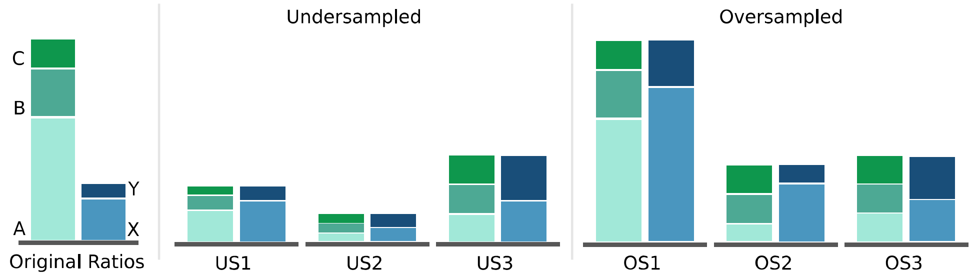

Fig. 3. An illustration of some of the possible sampling methodologies available through the Sampler module. In this example, it is assumed that there are five different classes in the dataset, where A, B, and C, together form the majority class (i.e., super-class ABC), and X and Y form the minority class (i.e., the super-class XY). US1 and OS1 result in a balance between ABC and XY while the ratios within the super-classes are preserved. OS2 is similar to OS1, with the additional constraint that it suppresses the largest class (i.e. here A). This is important as in some cases, this majority class might be too large such that its original size would dwarf other classes in subsampling. US2 and US3 impose a balance by taking X and Y classes, respectively, as a reference for undersampling other classes, and thus making a balance between the sub-classes as well. OS3 takes the class C as a reference for oversampling. Note how in each case, the sampled data is balanced with respect to the super-classes ABC and XY.

3. Impact

Any degree of success in discovering knowledge from time series datasets often goes hand in hand with the relevance of the chosen engineered features (unless automated feature-selection is utilized, that is often a product of Deep Neural Networks). We implemented a comprehensive collection of statistical features that gives researchers the opportunity to study various characteristics of their time series, those that might have previously gone unnoticed. Availability of such a collection encourages a truly data-driven research environment, which in the past decade has proven its importance in scientific domains of research.

MVTS-Data Toolkit is and will be progressively updated. We envision that its feature-collection module be extended continuously as we come across other time series features, in our ongoing research. In addition, contributors with relevant expertise can also add new features to this collection by submitting pull-requests to the repository. Therefore, this package serves as a framework for creation of out-of-the-box time series features.

Preprocessing of data often consists of several domain-independent routines that together bake the raw data for the actual analyses. Our generic implementation of some of these common routines, which caters mvts datasets, saves other researchers from re-implementing similar tasks for their specific research. Moreover, before each new release, the development pipeline passes the entire package through numerous test-cases to ensure that new changes do not yield erroneous or inaccurate outcomes. This necessary part adds a significant degree of reliability to the package.

The last but not least impact of this package is on flare-forecasting research, those of which that rely on time series data. To help advancing in this area of research, our lab at Georgia State University recently produced a new benchmark dataset, called Space Weather ANalytics for Solar Flares (SWAN-SF) [5]. It is made entirely of mvts, aiming to carry out an unbiased flare forecasting. So far, several studies have already utilized this package, or parts of it, [4,12,13]. With the official release of the benchmark dataset, we expect to see that other researchers show interest in using this package as a supplementary tool, next to the benchmark, that can help the community in their flare trend analysis and forecasting research.

4. Conclusions

We presented a Python package to help some of the preprocessing routines in working with mvts data. We discussed the main architecture of the package and the functionalities it provides. We also highlighted the several ways that this package contributes to the applied research on time series datasets, and how it can easily expand to have more tools and newer functionalities implemented. We hope that this package find its way into interdisciplinary research on time series data, as a useful toolkit.

CRediT authorship contribution statement

Azim Ahmadzadeh: Conceptualization, Methodology, Software, Visualization, Writing - original draft, Supervision. **Kankana Sinha:** Software, Investigation, Validation. **Berkay Aydin:** Validation, Data curation, Writing - review & editing. **Rafal A. Angryk:** Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by NASA, United States of America Grant Award [No. NNH14ZDA001N], and National Science Foundation, United States of America Grant Awards [No. AC1443061 and AC1931555]. The AC1443061 award has been supported by funding from the Division of Advanced Cyber infrastructure within the Directorate for Computer and Information Science and Engineering, United States of America, the Division of Astronomical Sciences within the Directorate for Mathematical and Physical Sciences, United States of America, and the Division of Atmospheric and Geospace Sciences within the Directorate for Geosciences, United States of America.

References

[1] Brockwell PJ, Davis RA. Introduction to time series and forecasting. Springer International Publishing; 2016, http://dx.doi.org/10.1007/978-3-319-29854-2.

[2] Fulcher BD. Feature-based time-series analysis. In: Feature engineering for machine learning and data analytics. CRC Press; 2018, p. 87–116.

[3] Christ M, Braun N, Neuffer J, Kempa-Liehr AW. Time series FeatuRe extraction on basis of scalable hypothesis tests (tsfresh – a Python package). Neurocomputing 2018;307:72–7. http://dx.doi.org/10.1016/j.neucom.2018.03.067.

[4] Ahmadzadeh A, Hostetter M, Aydin B, Georgoulis MK, Kempton DJ, Mahajan SS, et al. Challenges with extreme class-imbalance and temporal coherence: A study on solar flare data. In: IEEE international conference on big data. 2019, URL https://doi.org/10.1109/BigData47090.2019.9006505.

[5] Angryk RA, Martens PC, Aydin B, Kempton DJ, Mahajan SS, Basodi S, Ahmadzadeh A, Cai X, Boubrahimi SF, Hamdi SM, Schuh MA, Georgoulis MK. Multivariate time series dataset for space weather data analytics. Sci. Data 2019. Manuscript submitted for publication.

[6] Barandas M, Folgado D, Fernandes L, Santos S, Abreu M, Bota P, et al. TSFEL: Time series feature extraction library. SoftwareX 2020;11. 100456. http://dx.doi.org/10.1016/j.softx.2020.100456.
[7] Seabold S, Perktold J. Statsmodels: Econometric and statistical modeling with python. In: Proceedings of the 9th Python in science conference, vol. 57, Scipy; 2010, p. 61.
[8] Fulcher BD, Jones NS. Hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. Cell Syst 2017;5(5):527–31. http://dx.doi.org/10.1016/j.cels.2017.10.001.
[9] Tsay RS. MTS: All-purpose toolkit for analyzing multivariate time series (mts) and estimating multivariate volatility models. In: R package version, vol. 33, 2015.
[10] Lemaître G, Nogueira F, Aridas CK. Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. J Mach Learn Res 2017;18(17):1–5, URL http://jmlr.org/papers/v18/16-365.
[11] Dunning T, Ertl O. Computing extremely accurate quantiles using t-digests. 2019, arXiv preprint arXiv:1902.04023.
[12] Hostetter M, Ahmadzadeh A, Aydin B, Georgoulis MK, Kempton DJ, Angryk RA. Towards understanding the impact of statisticaltime series features for flare prediction analysis. In: IEEE international conference on big data. 2019, URL https://doi.org/10.1109/BigData47090.2019.9006116.
[13] Ahmadzadeh A, Aydin B, Kempton DJ, Georgoulis MK, Mahajan SS, Hostetter M, et al. Rare-event time series prediction: A case study of solar flare forecasting. In: 18th IEEE international conference on machine learning and applications. 2019.