Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

TSFEL: Time Series Feature Extraction Library

Marília Barandas[a,*,1], Duarte Folgado[a,1], Letícia Fernandes[a], Sara Santos[a], Mariana Abreu[a], Patrícia Bota[a], Hui Liu[b], Tanja Schultz[b], Hugo Gamboa[a,c]

[a] Associação Fraunhofer Portugal Research, Rua Alfredo Allen 455/461, Porto, Portugal
[b] Cognitive Systems Lab, University of Bremen, Bremen, Germany
[c] Laboratório de Instrumentação, Engenharia Biomédica e Física da Radiação (LIBPhys-UNL), Departamento de Física, Faculdade de Ciê,ncias e Tecnologia da Universidade Nova de Lisboa, Monte da Caparica, Caparica 2892-516, Portugal

ARTICLE INFO

Article history:
Received 2 January 2020
Received in revised form 28 February 2020
Accepted 14 March 2020

Keywords:
Time series
Machine learning
Feature extraction
Python

ABSTRACT

Time series feature extraction is one of the preliminary steps of conventional machine learning pipelines. Quite often, this process ends being a time consuming and complex task as data scientists must consider a combination between a multitude of domain knowledge factors and coding implementation. We present in this paper a Python package entitled Time Series Feature Extraction Library (TSFEL), which computes over 60 different features extracted across temporal, statistical and spectral domains. User customisation is achieved using either an online interface or a conventional Python package for more flexibility and integration into real deployment scenarios. TSFEL is designed to support the process of fast exploratory data analysis and feature extraction on time series with computational cost evaluation.

Code metadata

Current code version	v0.1.2
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_1
Legal Code Licence	BSD 3-Clause Licence
Code versioning system used	Git
Software code languages, tools, and services used	Python 3.7
Compilation requirements, operating environments & dependencies	numpy(1.17.4), matplotlib(3.1.0), gspread(3.1.0), oauth2client(4.1.3), pandas(0.24.2), scipy(1.4.0), setuptools(41.0.1)
If available Link to developer documentation/manual	tsfel.readthedocs.io
Support email for questions	info@fraunhofer.pt

Software metadata

Current software version	v0.1.2
Permanent link to executables of this version	github.com/fraunhoferportugal/tsfel
Legal Software Licence	BSD 3-Clause Licence
Computing platforms/Operating Systems	Linux, OS X, Microsoft Windows
Installation requirements & dependencies	Python 3.7
If available, link to user manual–if formally published include a reference to the publication in the reference list	tsfel.readthedocs.io
Support email for questions	info@fraunhofer.pt

1. Motivation and significance

Over the last years, the technological breakthroughs motivated by the rise of Internet-of-Things led to the proliferation of sensors

[*] Corresponding author.
E-mail address: marilia.barandas@fraunhofer.pt (M. Barandas).
[1] These authors contributed equally.

https://doi.org/10.1016/j.softx.2020.100456

to measure a plethora of physical processes. Those observations often result in the creation of large quantities of data in the form of time series, which are described as sequences of numerical observations ordered in time. The process of time series feature extraction is one of the preliminary steps in conventional machine learning pipelines and aims to extract a set of properties to characterise time series. The feature extraction is a time-consuming and complex task, which poses challenges on such a significant and important step of the machine learning stack: "*At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used*" [1].

We present in this paper a Python package named Time Series Feature Extraction Library (`TSFEL`), which provides support for fast exploratory analysis supported by an automated process of feature extraction on multidimensional time series.

In literature, there exist related packages dedicated to feature extraction, such as `FATS` [2], `CESIUM` [3], `TSFRESH` [4] and `HCTSA` [5]. Those packages inspired the creation of `TSFEL` and in the future they may be combined. `TSFEL` extends their scope by integrating a more thorough analysis of the temporal complexity of the features. This fact is relevant in scenarios where feature extraction is computed in embedded devices such as wearables, with limited computational resources. `TSFEL` is being used to support feature extraction on inertial data acquired by smartphones and wearables in the contexts of Human Activity Recognition (HAR) [6–9], rehabilitation [10–12] and anomaly detection [13].

Users can interact with `TSFEL` in two forms: a backend built upon a Python package aimed for advance users; a frontend displayed in online spreadsheets aimed for beginners. For both cases, `TSFEL` also provides a crucial aspect for the deployment of machine learning algorithms in real scenarios - a comprehensive evaluation of the computational complexity of each feature.

2. Software description

2.1. Software architecture

`TSFEL` is written in Python 3. Most of the implemented feature extraction methods depend on Numpy and SciPy, which provide efficient numerical routines for multi-dimensional data. `TSFEL` output is standardised to be compatible with data science Python packages, namely Pandas and scikit-learn.

The frontend is implemented using the Google Sheets API to establish the interface to `TSFEL` backend. An online spreadsheet is used to configure the feature extraction. Users can choose the desired feature extraction methods, either by individual selection or filtering based on defined criteria (e.g. domain or computational complexity). A tutorial with an illustrative example which uses the online spreadsheet to configure `TSFEL` is available on a Google Colab.[2]

Fig. 1 summarises the `TSFEL` processing pipeline. Time series are passed as inputs for the main `TSFEL` extraction method either as arrays previously loaded in memory or stored in files on a dataset. Since `TSFEL` can handle multidimensional time series, a set of preprocessing methods is afterwards applied to ensure that not only the signal quality is adequate, but also, time series synchronisation, so that the window calculation process is properly achieved. After the feature extraction, the result is saved using a standard schema ready to be digested by most of the classification and data mining platforms. Each line corresponds to a window with the results of the feature extraction methods stored along the corresponding columns.

[2] https://github.com/fraunhoferportugal/tsfel.

2.2. Software functionalities

When deploying machine learning applications, a proper set of features can improve the performance of the algorithms and reduce the computational complexity. In the signal windowing step, time series are divided into user-defined fixed length time windows (which can optionally have some overlap), from which features are extracted.

2.2.1. Data ingestion and preprocessing

Before the feature extraction step, it is essential to ensure adequate data quality. The time series can be passed to the feature extraction method using two methods:

- `dataset_features_extractor`: receives a string containing the dataset `root_directory` and the configuration feature dictionary `feat_dict`. To ensure that unequal length time series become synchronised, time-based parameters must be defined, namely the sampling frequency in Hertz and the `time_unit` defining the temporal scale of the dataset. This method might optionally receive as input the `output_directory` where the extracted features will be saved in a delimited text file format and the `search_criteria`, which will be used to match and filter the files which are the ones more relevant to the user.
- `time_series_features_extractor`: receives a multidimensional structure with time series stored as variables in memory, the configuration feature dictionary `feat_dict` and the sampling frequency in Hertz.

Both methods receive additional parameters related to window metadata, defining the window parameters used for the feature extraction, such as window size in the number of samples and the overlap between windows, defined as a percentage value.

2.2.2. Feature extraction

The `TSFEL` features can be grouped into three categories according to the domain where they are calculated: temporal, statistical and spectral domain. The listing of the available features and detailed implementation containing mathematical formulation, pseudo-code and references are available at Appendix.

All features implemented are classified by their time complexity according to the following known models: $O(n^2)$, $O(n \log n)$, $O(n)$, $O(\log n)$ and $O(1)$. Firstly, a set of time series with incremental length are synthetically generated from a sinusoidal model. Secondly, the feature extraction runtime is calculated for each incremental length time series. This process allows creating a curve displaying the relationship between time series length and execution time for each feature. Thirdly, we use a non-linear least squares to fit the runtime results to the known models. For each feature, we assign the time complexity which minimises the χ^2 among all the time complexity models.

2.2.3. Unit tests

In order to properly maintain a significant number of different feature extraction methods, it is necessary to systematically verify the feature extraction implementation. A set of unit tests were created for each feature. Those tests compare the values obtained from `TSFEL` against a set of synthetic time series with a known distribution. Different time series synthesis parameters are used to cover a variety of properties (e.g. constant values, modulation of amplitude offsets, periodicity and noise addition). The contribution guidelines to incorporate new features in `TSFEL` require the introduction of at least one unit test for the new feature.

2.2.4. Personalised features

`TSFEL` provides flexibility for users to add their personal features to those already available on the library. Users are require to write the feature extraction method and declare its domain.

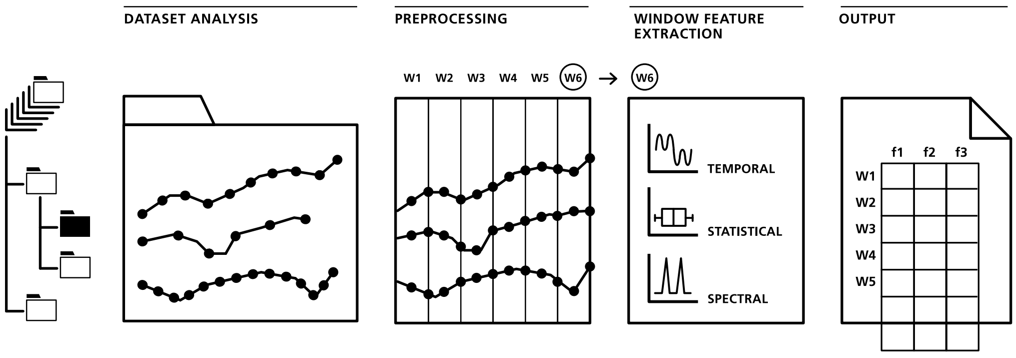

Fig. 1. TSFEL pipeline: dataset analysis, signal preprocessing, feature extraction and output.

3. Illustrative examples

We now provide an illustrative example with a typical pipeline to extract features in the context of HAR. We used a subset from the dataset collected by [12], composed of time series retrieved by accelerometer, gyroscope and goniometer sensors from two subjects performing four activities: stand, sit, stand-to-sit and sit-to-stand. Developers can accomplish the feature extraction using two distinct methods: conduct the feature extraction on a time series already loaded in memory or conduct the feature extraction across an entire time series database organised in directories. At line 4 of Fig. 2 the user loads a configuration setting which is available as a template. The `get_all_features()` template enables all the implemented features to be extracted. In line 7 a dataset composed of univariate time series from 3 different sensors is loaded. At line 8 the feature extraction process is accomplished. The method receives as inputs the configuration settings, the time series passed as a pandas' dataframe `df`, the sampling frequency in Hertz, the window size defined as number of samples and no overlap percentage between consecutive windows.

Fig. 3 summarises the interaction required to use TSFEL to extract features from a dataset stored in files across system directories. In line 4 we define the root directory of the dataset. The configuration file is defined on line 5. The main method is present on lines 7 to 13. The method receives as inputs the root dataset directory, the configuration file, the search criteria defined by keywords that will match the designated files for feature extraction, the resample parameters (which will configure the resampling methods to ensure the multivariate time series are synchronised prior dividing data into windows), and the parameters which define the window size and overlap.

Fig. 4 represents a set of the extracted features in the form of a horizon plot. The x-axis contains three repetitions of the following sequence of activities: sit, sit-to-stand, stand, stand-to-sit. The y-axis corresponds to the extracted features, and the range of y-axis to a third of the maximum value of each feature, represented by the darkest values. The blue and red colours correspond to positive and negative values, respectively. The goniometer and gyroscope sensors allow to discriminate between static activities (sit and stand) and transitions (sit-to-stand and stand-to-sit). The *acc_x_Upper_Min* and *acc_z_Upper_Min* clearly distinguish between sit and stand. Finally, the *goniometer_y_Slope* is an adequate feature to discriminate between sit-to-stand and stand-to-sit.

4. Impact

While working with the analysis of heterogeneous time series data from multiple sources, the process of feature engineering (i.e. transforming raw data into features to enter as inputs for machine learning models) is a time-consuming and complex task. TSFEL helps data scientists by creating an abstraction layer composed of a comprehensive list of time series feature extraction techniques. Those methods were aggregated from several scientific domains and previously validated on the context of HAR. The output of the TSFEL is a standardised file format ready to be digested by most of time series classification libraries like Orange, Weka or scikit-learn. TSFEL also provides a systematic methodology to record the inputs, the dataset metadata and the parameters of feature extraction experiments, promoting openness and reproducibility of the scientific method among researchers. TSFEL relies on a master configuration file which stores the metadata and can be easily edited thorough a spreadsheet configuration manager.

With the proliferation of Cyber–Physical Systems, more data and processes are being digitised across several sectors. Whilst machine learning holds the promise to uncover hidden patterns of big data to gain new insights on data-driven business, the deployment of such systems might still be limited by scalability concerns. For example, in the context of large manufacturing plants, the anomaly detection on their products based on sensors producing high data volume, requires a compromise between high accuracy and low latency. Therefore, the complete pipeline should be designed with a mindset of being able to run the inference step in near real-time. The successful implementation of this approach should be accomplished bottom-up, meaning to have design concerns on the initial stages of the machine learning stack. TSFEL helps mitigate these technology adoption risks by providing the user with a comprehensive list of feature extraction methods on time series with an associated estimate of computational complexity, enabling to have an idea of the computational cost of the feature extraction on the early stages of the machine learning stack.

5. Conclusions

We have developed a Python package entitled Time Series Feature Extraction Library, which provides a comprehensive list of feature extraction methods for time series. Over 60 different features are extracted across temporal, statistical and spectral domains. TSFEL includes unit tests for every implemented feature extraction method and proper code documentation. Besides these strongly oriented coding guidelines, this project has three main broad contributions to the scientific community:

```python
import tsfel
import pandas as pd

cfg = tsfel.get_features_by_domain()

for fn in ['Accelerometer.txt', 'Gyroscope.txt', 'Goniometer.txt']:
    df = pd.read_csv(fn)
    X = tsfel.time_series_features_extractor(cfg_file, df, fs=1000, window_size=250, overlap=0)
```

Fig. 2. Illustrative code example to extract features from a time series previously loaded in memory.

```python
import tsfel
import pandas as pd

main_directory = '../Dataset'
cfg = tsfel.get_features_by_domain()

tsfel.dataset_features_extractor(main_directory, cfg_file,
    output_directory='../tsfel_output',
    search_criteria=['Accelerometer.txt', 'Gyroscope.txt', 'Goniometer.txt'],
    time_unit=1e9,
    resample_rate=1000,
    window_size=250,
    overlap=0)
```

Fig. 3. Illustrative code example to extract features from a dataset organised in directories.

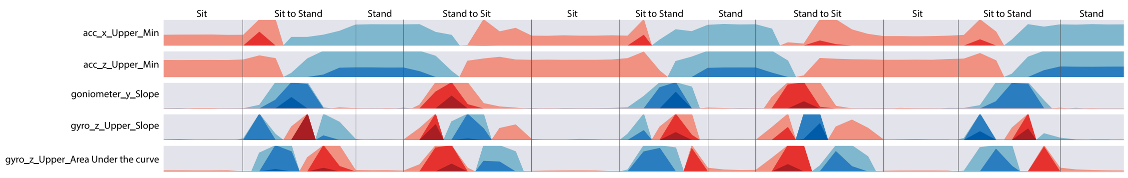

Fig. 4. Horizon plot representation from the most relevant features. The vertical lines correspond to groundtruth annotations to discriminate among different classes.

TSFEL can be used through a user interface built upon Google Sheets, requiring no installation whatsoever, allowing rapid prototyping and exploratory time series analysis, with reduced coding effort. More expert users can delve directly into TSFEL backend for more flexibility and integration with Python projects. A master configuration file is used to record the experiment parameters, emphasising traceability and reproducibility.

The feature extraction process is performed either on time series already stored in the memory environment or using a list of directories containing time series data files. The latter is achieved using a search criteria which filters the files relevant to the user. This process aims to reduce the time spent by the user in preparing data aggregation code for every application.

This design of TSFEL is oriented to have from the early stage of the machine learning pipeline an estimate regarding the computational complexity of the features being extracted, aligned with the "Edge artificial intelligence" strategy to empower the deployment of machine learning applications in large scale environments.

Although multiple features are extracted across three different domains, there is still room for improvement. Thus, as future work, features regarding new domains such as nonlinear features will be introduced.

TSFEL is open for contributions and invites users to extend the library with new time series feature extraction methods across different scientific disciplines.

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.

Acknowledgements

We would like to acknowledge the financial support obtained from the project Total Integrated and Predictive Manufacturing System Platform for Industry 4.0, co-funded by Portugal 2020, framed under the COMPETE 2020 (Operational Programme Competitiveness and Internationalisation) and European Regional Development Fund (ERDF) from European Union (EU), with operation code POCI-01-0247-FEDER-038436.

Appendix. Detailed implementation of available features

Let s represent the time series signal vector, Δs the discrete derivative of s, t the correspondent time vector, fs signal's sampling frequency, and N the length of s. Additionally, $P(x)$ states probability.

Domain transformations

Fast Fourier Transform

```
1: INPUT: $s, t \leftarrow s$ is the signal and $t$ the time
2: OUTPUT: $frequency, magnitude$
3: Computes the Fast Fourier Transform: $fft(t, s)$
4: Returns frequency and magnitude values
```

Wavelet transform

```
1: INPUT: $s, f, w$ ← $f$ is Ricker function (default), $w$ are the scale widths
2: OUTPUT: matrix with size $(len(w), len(s))$
3: Computes the discrete wavelet transform: $cwt(s, f, w)$
4: Returns a matrix with size $(len(w), len(s))$
```

Temporal domain

Autocorrelation: $\sum_{n \in Z} s(n)\overline{s(n-l)}$, where $\overline{s(n-l)}$ is the complex conjugate of $s(n)$, and l is a lag.

Centroid: $\frac{\sum_{i=0}^{N} t_i \times s_i^2}{\sum_i^N s_i^2}$

Mean absolute differences: $mean(|\Delta s|)$

Mean differences: $mean(\Delta s)$

Median absolute differences: $median(|\Delta s|)$

Median differences: $median(\Delta s)$

Distance: $\sum_{i=0}^{N-1} \sqrt{1 + \Delta s_i{}^2}$

Sum of absolute differences: $\sum_{i=0}^{N-1} |\Delta s_i|$

Total energy: $\frac{\sum_{i=0}^{N} s^2}{t_N - t_0}$

Entropy: $-\sum_{x \in s} P(x) \log_2 P(x)$

Peak to peak distance: $|max(s) - min(s)|$

Area under the curve: $\sum_{i=0}^{N} (t_i - t_{i-1}) \times \frac{s_i + s_{i-1}}{2}$

Absolute energy: $\sum_{i=0}^{N} s_i{}^2$

Maximum peaks:

```
1: INPUT: $s$
2: OUTPUT: $number\ of\ maximum\ peaks$
3: $\Delta s = s_{i+1} - s_i$
4: for each $i \in [0, N-1]$ do
5:     if $\Delta s_{i+1} < 0$ and $\Delta s_i > 0$ then
6:         Count one maximum peak
7:     else
8:         Go back to the next i
9:     end if
10: end for
```

Minimum peaks:

```
1: INPUT: $s$
2: OUTPUT: $number\ of\ minimum\ peaks$
3: $\Delta s = s_{i+1} - s_i$
4: for each $i \in [0, N-1]$ do
5:     if $\Delta s_i < 0$ and $\Delta s_{i+1} > 0$ then
6:         Count one minimum peak
7:     else
8:         Go back to the next i
9:     end if
10: end for
```

Slope:

```
1: INPUT: $s, t$
2: OUTPUT: $slope$
3: Fits a linear equation to the observed data: $y = mx + b$
4: Returns the m coefficient (slope)
```

Zero crossing rate:

```
1: INPUT: $s$
2: OUTPUT: $zero\ crossing\ rate$
3: $sign$ = list of zeros with size $N$
4: $zcr$ = list of zeros with size $N$
5: for each $i \in [0, N]$ do
6:     if $s_i > 0$ then
7:         $sign_i = 1$
8:     else if $s_i < 0$ then
9:         $sign_i = 0$
10:    end if
11:    Go back to the next i
12: end for
13: $\Delta sign = sign_{i+1} - sign_i$
14: for each $i \in [0, N]$ do
15:    if $\Delta sign_i > 0$ or $\Delta sign_i < 0$ then
16:        $zcr_i = 1$
17:    else if $\Delta sign_i = 0$ then
18:        $zcr_i = 0$
19:    end if
20:    Go back to the next i
21: end for
22: $zero\ crossing\ rate = \sum_i^N zcr_i$
```

Statistical domain

Histogram: $n = \sum_{i=1}^{k} m_i$, where m_i represents the histogram in which n is the total number of observations and k the total number of bins.

Interquartile range: $Q_3 - Q_1$, where Q_3 and Q_1 represent the first and third quartile, respectively.

Mean absolute deviation: $\frac{\sum_{i=1}^{N} |s_i^2 - mean(s)|}{N}$

Median absolute deviation: $median(|s - median(s)|)$

Root mean square: $\sqrt{\frac{1}{N} \sum_{i=1}^{N} s_i^2}$

Standard deviation: $\sqrt{var}$

Variance: $mean(|s - mean(s)|)^2$

ECDF percentile count:

```
1: INPUT: s, percentile value $(p)$
2: OUTPUT: Cumulative sum of samples that are less than the percentile.
3: $x$ = sorted $s$
4: $y$ = ECDF values
5: $percentilecount$ = length of $x$, where $y < p$
```

ECDF slope:

```
1: INPUT: s, two percentile values $(p_{init}, p_{end})$
2: OUTPUT: Slope of the ECDF between the two percentiles
3: $x_{init}, x_{end}$ = the $s$ values for the given percentiles
4: $slope = \frac{p_{end} - p_{init}}{x_{end} - x_{init}}$
```

The following features are implemented according to the following references: **Kurtosis** [14], **Skewness** [14], **Maximum** [15],

Minimum [15], **Mean** [15], **Median** [15] and **ECDF** [16], **ECDF Percentile** [16].

Spectral domain

FFT mean coefficient: $mean(spectrogram(s))$

Wavelet absolute mean: $|mean(wavelet(s))|$

Wavelet standard deviation: $|std(wavelet(s))|$

Wavelet variance: $|var(wavelet(s))|$

Spectral distance:

```
1: INPUT: $s, t$
2: OUTPUT: spectral distance
3: Computes the Fast Fourier Transform: $freq, fmag = fft(t, s)$
4: Cumulative sum of the magnitude ($cumsum_{fmag}$)
5: Linear regression of the $cumsum_{fmag}$ ($lr_{fmag}$)
6: Returns the spectral distance = $\sum_{i=0}^{N} lr_{fmag_i} - cumsum_{fmag_i}$
```

Fundamental frequency:

```
1: INPUT: $s, t$
2: OUTPUT: Fundamental Frequency ($ff$)
3: Computes the Fast Fourier Transform: $freq, fmag = fft(t, s)$
4: Finds the lowest frequency of vibration ($ff$)
```

Maximum frequency:

```
1: INPUT: $s, fs$
2: OUTPUT: Maximum Frequency ($mf$)
3: Computes the Fast Fourier Transform: $freq, fmag = fft(t, s)$
4: Cumulative sum of the magnitude ($cumsum_{fmag}$)
5: Returns the frequency with 95% of the $cumsum_{fmag}$ ($mf$)
```

Median frequency:

```
1: INPUT: $s, fs$
2: OUTPUT: Median Frequency ($medf$)
3: Computes the Fast Fourier Transform: $freq, fmag = fft(t, s)$
4: Cumulative sum of the magnitude ($cumsum_{fmag}$)
5: Returns the frequency with 50% of the $cumsum_{fmag}$ ($medf$)
```

Spectral maximum peaks:

```
1: INPUT: $s, fs$
2: OUTPUT: Number of maximum spectral peaks
3: Computes the Fast Fourier Transform: $freq, fmag = fft(t, s)$
4: Returns the maximum number of peaks of $fmag$
```

The following features are implemented according to the following references: **Maximum Power Spectrum** [17], **Spectral Centroid** [18], **Decrease** [18], **Kurtosis** [18], **Skewness** [18], **Spread** [18], **Slope** [18], **Variation** [18], **Spectral Roll-off** [7], **Roll-on** [7], **Human Range Energy**, [19], **MFCC** [20], **LPCC** [20], **Power Bandwidth** [21], **Spectral Entropy** [22], **Wavelet Entropy** [23] and **Wavelet Energy** [24].

References

[1] Domingos P. A few useful things to know about machine learning. Commun ACM 2012;55(10):78. http://dx.doi.org/10.1145/2347736.2347755, URL: http://dl.acm.org/citation.cfm?doid=2347736.2347755.

[2] Nun I, Protopapas P, Sim B, Zhu M, Dave R, Castro N, Pichara K. FATS: Feature analysis for time series. 2015, arXiv:1506.00010 [astro-ph], http://arxiv.org/abs/1506.00010, arXiv:1506.00010.

[3] Naul B, van der Walt S, Crellin-Quick A, Bloom JS, Pérez F. Cesium: Open-source platform for time-series inference. 2016, arXiv:1609.04504 [cs], http://arxiv.org/abs/1609.04504, arXiv:1609.04504.

[4] 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, URL https://linkinghub.elsevier.com/retrieve/pii/S0925231218304843.

[5] Fulcher B, Jones N. Hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. Cell Syst 2017;5:527–31. http://dx.doi.org/10.1016/j.cels.2017.10.001.

[6] Machado IP, Luísa Gomes A, Gamboa H, Paixão V, Costa RM. Human activity data discovery from triaxial accelerometer sensor: Non-supervised learning sensitivity to feature extraction parametrization. Inf Process Manage 2015;51(2):204–14. http://dx.doi.org/10.1016/j.ipm.2014.07.008, URL https://linkinghub.elsevier.com/retrieve/pii/S0306457314000685.

[7] Figueira C, Matias R, Gamboa H. Body location independent activity monitoring. In: Proceedings of the 9th international joint conference on biomedical engineering systems and technologies. Rome, Italy: SCITEPRESS - Science and and Technology Publications; 2016, p. 190–7. http://dx.doi.org/10.5220/0005699601900197, URL http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005699601900197.

[8] Abreu. M, Barandas. M, Leonardo. R, Gamboa. H. Detailed human activity recognition based on multiple HMM. In: Proceedings of the 12th international joint conference on biomedical engineering systems and technologies - Volume 4: BIOSIGNALS,. SciTePress; 2019, p. 171–8. http://dx.doi.org/10.5220/0007386901710178.

[9] Bota P, Silva J, Folgado D, Gamboa H. A semi-automatic annotation approach for human activity recognition. Sensors 2019;19(3):501. http://dx.doi.org/10.3390/s19030501, URL http://www.mdpi.com/1424-8220/19/3/501.

[10] Liu H, Schultz T. Ask: A framework for data acquisition and activity recognition. In: Proceedings of the 11th international joint conference on biomedical engineering systems and technologies - Volume 4 BIOSIGNALS: BIOSIGNALS. SciTePress; 2018, p. 262–8. http://dx.doi.org/10.5220/0006732902620268.

[11] Pereira A, Folgado D, Cotrim R, Sousa I. Physiotherapy exercises evaluation using a combined approach based on semg and wearable inertial sensors:. In: Proceedings of the 12th international joint conference on biomedical engineering systems and technologies. Prague, Czech Republic: SCITEPRESS - Science and Technology Publications; 2019, p. 73–82. http://dx.doi.org/10.5220/0007391300730082, URL http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0007391300730082.

[12] Liu H, Schultz T. A wearable real-time human activity recognition system using biosensors integrated into a knee bandage. In: Proceedings of the 12th international joint conference on biomedical engineering systems and technologies - Volume 3: BIODEVICES. SciTePress; 2019, p. 47–55. http://dx.doi.org/10.5220/0007398800470055.

[13] Varandas R, Folgado D, Gamboa H. Evaluation of spatial-temporal anomalies in the analysis of human movement:. In: Proceedings of the 12th international joint conference on biomedical engineering systems and technologies. Prague, Czech Republic: SCITEPRESS - Science and Technology Publications; 2019, p. 163–70. http://dx.doi.org/10.5220/0007386701630170, URL http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0007386701630170.

[14] Zwillinger D, Kokoska S. CRC standard probability and statistics tables and formulae. New York: Chapman & Hall; 2000.

[15] Oliphant TE. A guide to NumPy, Vol. 1. Trelgol Publishing USA; 2006.

[16] Raschka S. Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. J Open Source Softw 2018;3(24). http://dx.doi.org/10.21105/joss.00638, URL http://joss.theoj.org/papers/10.21105/joss.00638.

[17] Welch PD. The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. IEEE Trans Audio Electroacoust 1967;15(2):70–3. http://dx.doi.org/10.1109/TAU.1967.1161901.

[18] Peeters G, Giordano B, Susini P, Misdariis N, Mcadams S. The timbre toolbox: Extracting audio descriptors from musical signals. J Acoust Soc Am 2011;130:2902–16. http://dx.doi.org/10.1121/1.3642604.

[19] Fernandes LMS. Learning human behaviour patterns by trajectory and activity recognition (Master's thesis), FCT NOVA; 2019.

[20] Das O. Speaker Recognition. Tech. rep., 2016, URL https://ccrma.stanford.edu/~orchi/Documents/speaker_recognition_report.pdf.

[21] Henriksson U. Power Spectrum and Bandwidth. Tech. rep., Linköping University; 2003, URL https://www.commsys.isy.liu.se/TSDT45/Material/UlfsSpectrum2003.pdf.

[22] Pan Y, Chen J, Li X. Spectral entropy: A complementary index for rolling element bearing performance degradation assessment. Proceedings Inst Mech Eng C 2009;223:1223–31. http://dx.doi.org/10.1243/09544062JMES1224.

[23] Yan B, Miyamoto A, Brühwiler E. Wavelet transform-based modal parameter identification considering uncertainty. J Sound Vib 2006;291(1):285–301. http://dx.doi.org/10.1016/j.jsv.2005.06.005, URL http://www.sciencedirect.com/science/article/pii/S0022460X05003913.

[24] Kocaman Ç, Özdemir M. Comparison of statistical methods and wavelet energy coefficients for determining two common PQ disturbances: Sag and swell. In: IEEE International Conference on Electrical and Electronics Engineering. 2009, p. I–80–I–84. http://dx.doi.org/10.1109/ELECO.2009.5355235.