




TSFX: A Python package for time series feature extraction

Wilhelm Söderkvist Vermelin ^{1,2}

¹ RISE Research Institutes of Sweden ² Mälardalen University

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

TSFX is a Python ([Van Rossum & Drake, 2009](#)) library for extracting features from time series data. It is inspired by the tsfresh ([Christ et al., 2018](#)) Python package with a special focus on performance. In order to achieve good performance, it utilizes Polars ([Vink et al., 2023](#)) which is a high performance DataFrame library written in Rust ([The Rust Programming Language, n.d.](#)) with Python bindings created through PyO3 ([Project & Contributors, n.d.](#)). The feature extraction functions are also implemented in Rust for performance. Compared to tsfresh, TSFX offers a conservative estimate of 10x performance, using the same set of time series features.

TSFX can be installed using pip:

```
pip install tsfx
```

TSFX can also be configured using a TOML ([TOML, n.d.](#)) configuration file (default name `.tsfx-config.toml`).

Below is a simple example of extracting features from a time series dataset:

```
import polars as pl
from tsfx import (
    ExtractionSettings,
    FeatureSetting,
    extract_features,
)

df = pl.DataFrame(
    {
        "id": ["a", "a", "a", "b", "b", "b", "c", "c", "c"],
        "val": [1.0, 2.0, 3.0, 1.0, 2.0, 3.0, 1.0, 2.0, 3.0],
        "value": [4.0, 5.0, 6.0, 6.0, 5.0, 4.0, 4.0, 5.0, 6.0],
    },
).lazy()
settings = ExtractionSettings(
    grouping_cols=["id"],
    feature_setting=FeatureSetting.Efficient,
    value_cols=["val", "value"],
)
gdf = extract_features(df, settings)
gdf = gdf.sort(by="id")
with pl.Config(set_tbl_width_chars=80):
    print(gdf)
```

Running the code above generates a new DataFrame with the extracted features:

shape: (3, 314)

id	length	val__su m_value	val__me an	...	value__ number_ peaks__ n_3	value__ number_ peaks__ n_5	value__ number_ peaks__ n_10	value__ number_ peaks__ n_50
str	u32	s f64	f64		f64	f64	f64	f64
a	3	6.0	2.0	...	0.0	0.0	0.0	0.0
b	3	6.0	2.0	...	0.0	0.0	0.0	0.0
c	3	6.0	2.0	...	0.0	0.0	0.0	0.0

18 If the DataFrame has a time column, it is also possible to extract over a time win-
19 dow by passing DynamicGroupBySettings into the feature extraction settings, like so:
20 ExtractionSettings(..., dynamic_settings=DynamicGroupBySettings(...)).

21 Statement of need

22 Time series is a ubiquitous data modality, present in many domains such as finance, industry,
23 meteorology, and medicine, to mention a few. As hardware to collect and store time series
24 data is becoming increasingly affordable, the amount of available time series data is increasing
25 in many domains. A common preprocessing step when dealing with time series is feature
26 extraction where useful features, such as mean, variance, skewness, etc. are extracted from
27 time series to be used in downstream tasks such as classification, regression or clustering. For
28 large time series datasets, performance is important for enabling timely data preprocessing.
29 TSFX is made for this purpose: extracting features from large time series datasets.

30 Acknowledgements

31 The TSFX package was developed within the [Vinnova](#) projects [DFusion](#), [TolkAI](#), and [SIFT](#).
32 This research work has been funded by the Knowledge Foundation within the framework of
33 the INDTECH (Grant Number 20200132) and INDTECH+ Research School project (Grant
34 Number 20220132), participating companies and Mälardalen University.

35 References

- 36 Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series FeatuRe extrac-
37 tion on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*,
38 307, 72–77. <https://doi.org/https://doi.org/10.1016/j.neucom.2018.03.067>
- 39 Project, P., & Contributors. (n.d.). *PyO3*. GitHub. <https://github.com/PyO3>
- 40 *The rust programming language*. (n.d.). <https://rust-lang.org/>.
- 41 *TOML: Tom's obvious minimal language*. (n.d.). <https://toml.io/en/>
- 42 Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace.
43 ISBN: 1441412697
- 44 Vink, R., Gooijer, S. de, Beedie, A., Zundert, J. van, Hulselmans, G., Grinstead, C., Gorelli, M.
45 E., Santamaria, M., Heres, D., ibENPC, Leitao, J., Heerden, M. van, Jermain, C., Russell,
46 R., Pryer, C., Castellanos, A. G., Goh, J., Wilksch, M., illumination-k, ... Keller, J. (2023).
47 *Pola-rs/polars: Python polars 0.16.11* (py-0.16.11). Zenodo. <https://doi.org/10.5281/zenodo.7699984>