




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Published: unpublished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 on large time series datasets. To this end, it utilizes Polars ([Vink et al., 2023](#)) which is a fast DataFrame library written in Rust ([The Rust Programming Language, n.d.](#)) with Python bindings facilitated through PyO3 ([Project & Contributors, n.d.](#)). The feature extraction functions are implemented in Rust for even faster execution. To benchmark, the “1 billion row challenge” ([Morling, n.d.](#)) was used. The benchmark shows that compared to tsfresh, TSFX offers approximately 10 times higher 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)
```

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

shape: (3, 314)

id	length	val__su m_value s	val__me an f64	...	value__ number_ peaks__ n_3 f64	value__ number_ peaks__ n_5 f64	value__ number_ peaks__ n_10 f64	value__ number_ peaks__ n_50 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

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

23 Statement of need

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

33 Acknowledgements

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

38 References

- 39 Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series FeatuRe extrac-
 40 tion on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*,
 41 307, 72–77. <https://doi.org/https://doi.org/10.1016/j.neucom.2018.03.067>
- 42 Morling, G. (n.d.). *The one billion row challenge*. [https://www.morling.dev/blog/one-billion-](https://www.morling.dev/blog/one-billion-row-challenge/)
 43 [row-challenge/](https://www.morling.dev/blog/one-billion-row-challenge/)
- 44 Project, P., & Contributors. (n.d.). *PyO3*. GitHub. <https://github.com/PyO3>
- 45 *The rust programming language*. (n.d.). <https://rust-lang.org/>.
- 46 *TOML: Tom's obvious minimal language*. (n.d.). <https://toml.io/en/>
- 47 Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace.
 48 ISBN: 1441412697

49 Vink, R., Gooijer, S. de, Beedie, A., Zundert, J. van, Hulselmans, G., Grinstead, C., Gorelli, M.
50 E., Santamaria, M., Heres, D., ibENPC, Leitao, J., Heerden, M. van, Jermain, C., Russell,
51 R., Pryer, C., Castellanos, A. G., Goh, J., Wilksch, M., illumination-k, ... Keller, J. (2023).
52 *Pola-rs/polars: Python polars 0.16.11* (py-0.16.11). Zenodo. [https://doi.org/10.5281/](https://doi.org/10.5281/zenodo.7699984)
53 [zenodo.7699984](https://doi.org/10.5281/zenodo.7699984)

DRAFT