The Journal of Open Source Software

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

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

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*, 2015) with Python bindings facilitated through PyO3 (Project & Contributors, 2023). The feature extraction functions are implemented in Rust for even faster execution. To benchmark, the "1 billion row challenge" (Morling, 2024) was used. In this benchmark, TSFX took approx. 200 seconds (using 64 CPU cores) to extract features vs tsfresh which took 2000 seconds (using 80 CPU cores). This means 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*, 2021) configuration file (default name `.tsfx-config.toml`).

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

```python
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")
```

```python
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  ┆ val__me ┆ … ┆ value__  ┆ value__  ┆ value__  ┆ value__  │
│ --- ┆ ---    ┆ m_value  ┆ an      ┆   ┆ number_  ┆ number_  ┆ number_  ┆ number_  │
│ str ┆ u32    ┆ s        ┆ ---     ┆   ┆ peaks__  ┆ peaks__  ┆ peaks__  ┆ peaks__  │
│     ┆        ┆ ---      ┆ f64     ┆   ┆ n_3      ┆ n_5      ┆ n_10     ┆ n_50     │
│     ┆        ┆ 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      │
└─────┴────────┴──────────┴─────────┴───┴──────────┴──────────┴──────────┴──────────┘
```

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

Statement of need

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

Acknowledgements

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

References

Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series FeatuRe extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, *307*, 72–77. https://doi.org/https://doi.org/10.1016/j.neucom.2018.03.067

Morling, G. (2024). *The one billion row challenge*.https://www.morling.dev/blog/one-billion-row-challenge/

Project, P., & Contributors. (2023). *PyO3*. GitHub. https://github.com/PyO3

The rust programming language. (2015). https://rust-lang.org/.

TOML: Tom's obvious minimal language. (2021). https://toml.io/en/

The Journal of Open Source Software

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace. ISBN: 1441412697

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