# The FreshPRINCE: A Simple Transformation Based Pipeline Time Series Classifier

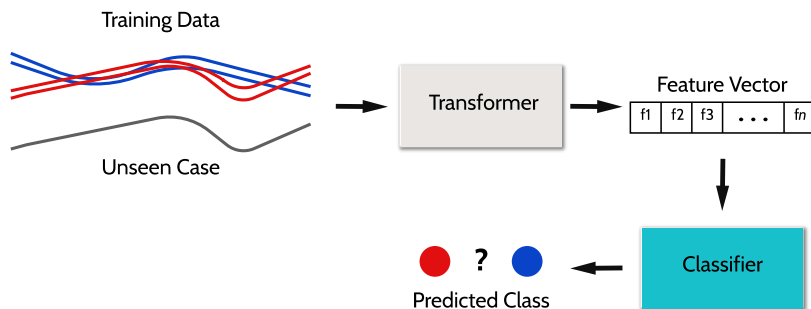Matthew Middlehurst and Anthony Bagnall

School of Computing Sciences, University of East Anglia, UK
M.Middlehurst@uea.ac.uk

**Abstract.** There have recently been significant advances in the accuracy of algorithms proposed for time series classification (TSC). However, a commonly asked question by real world practitioners and data scientists less familiar with the research topic, is whether the complexity of the algorithms considered state of the art is really necessary. Many times the first approach suggested is a simple pipeline of summary statistics or other time series feature extraction approaches such as TSFresh, which in itself is a sensible question; in publications on TSC algorithms generalised for multiple problem types, we rarely see these approaches considered or compared against. We experiment with basic feature extractors using vector based classifiers shown to be effective with continuous attributes in current state-of-the-art time series classifiers. We test these approaches on the UCR time series dataset archive, looking to see if TSC literature has overlooked the effectiveness of these approaches. We find that a pipeline of TSFresh followed by a rotation forest classifier, which we name FreshPRINCE, performs best. It is not state of the art, but it is significantly more accurate than nearest neighbour with dynamic time warping, and represents a reasonable benchmark for future comparison.

**Keywords:** Time series classification; transformation based classification; time series pipeline.

## 1 Introduction

A wide range of complex algorithms for time series classification (TSC) have been proposed. These include ensembles of deep neural networks [14], heterogeneous meta-ensembles build on different representations [22], homogeneous ensembles with embedded representations [26] and randomised kernels [10]. The majority of these algorithms rely on some form of transformation: features that in some way model the discriminatory time characteristics are extracted and used in the classification process. These features are often very complex, and usually embedded in the classifiers in complicated ways. For example, the Temporal Dictionary Ensemble (TDE) [19] is centred around the Symbolic Fourier Approximation (SFA) [25] transformation. The transform itself simply discretises the series into a set of words using a sliding window. However, just performing the transform does not lead to an algorithm that is competitive in accuracy. TDE also employs

**Fig. 1.** Visualisation of a simple pipeline algorithm for TSC. Could using standard transformers and vector based classifiers be as good as state of the art TSC algorithms?

a spacial pyramid, uses bi-gram frequency, a bespoke distance function and a Gaussian process based parameter setting mechanism. The complexity increases further if the data is multivariate, containing multiple time series per case.

Researchers not directly involved in TSC algorithm research, and data scientists in particular, often ask the not unreasonable question of whether these complicated representations are really necessary to get a good classifier. They wonder whether a simple pipeline using standard feature extractors, as illustrated in Figure 1 would not in fact be at least as good as complicated classifiers claiming to be state of the art? Clearly, the answer will not be the same for all problems, and the detailed answer depends on what level of accuracy is deemed sufficient for a particular application. However, we can address the hypothesis of whether, on average, a standard pipeline of transformation plus classifier performs as well as bespoke benchmarks and state of the art. Specifically, we compare a range of pipeline combinations of off the shelf unsupervised time series transformers with commonly used vector based classifiers to the current state of the art in TSC as described in [22]. In Section 2 we describe the transformers and classifiers used in our pipeline experiments, and give a brief overview of the state of the art in TSC. In Section 3 we describe our experimental structure, and in Section 4 we present our findings. Finally, in Section 5 we draw our conclusions and summarise what we have learnt from this study.

## 2 Background

TSC algorithms tend to follow one of three structures. the simplest involves single pipelines such as that described in Section 1, where the transformation is either supervised (e.g. Shapelet Transform Classifier [6]) or unsupervised (e.g. ROCKET [10]). These algorithms tend to involve an over-produce and select strategy: a huge number of features are created, and the classifier is left to determine which are most useful. The transform can remove time dependency, e.g.

by calculating summary features. We call this type series-to-vector transformations. Alternatively, they may be series-to-series, transforming into an alternative time series representation where we hope the task becomes more easily tractable (e.g. transforming to the frequency domain of the series).

The second transformation based design pattern involves ensembles of pipelines, where each base pipeline consists of making repeated, different, transforms and using a homogeneous base classifier (e.g. Canonical Interval Forest [21]). These ensembles can also be heterogeneous, collating the classifications from transformation pipelines and ensembles of differing representations of the time series (e.g. HIVE-COTE [22]).

The third common pattern involves transformations embedded inside a classifier structure. An example of this is a decision tree: where the data is transformed, or a distance measure is applied prior to any splitting criteria at each node (e.g. TS-CHIEF [26]).

## 2.1   State of the art for TSC

The state-of-the art for TSC consists of one classifier from each of the structures described, as well as a deep learning approach.

**The Random Convolutional Kernel Transform (ROCKET) [10]** is a transform designed for classification. It generates a large number of parameterised convolutional kernels, used as part of a pipeline alongside a linear classifier. Kernels are randomly initialised with respect to the following parameters: the kernel length; a vector of weights; a bias term added to the result of the convolution operation; the dilation to define the spread of the kernel weights over the input instance; and padding for the input series at the start and end. Each kernel is convoluted with an instance through a sliding window dot-product producing an output vector, extracting only two values: the max value and the proportion of positive values. These are concatenated into a feature vector for all kernels.

**The Time Series Combination of Heterogeneous and Integrated Embedding Forest (TS-CHIEF) [26]** is a homogeneous ensemble where hybrid features are embedded in tree nodes rather than modularised through separate classifiers. The trees in the TS-CHIEF ensemble embed distance measures, dictionary based histograms and spectral features. At each node, a number of splitting criteria from each of these representations are considered. These splits use randomly initialised parameters to help maintain diversity in the ensemble.

**InceptionTime [14]** is the only deep learning approach we are aware of which achieves state-of-the-art accuracy for TSC. InceptionTime builds on a residual network (ResNet), the prior best network for TSC [13]. The network is composed of two blocks of three Inception modules [27] each, as opposed to the three blocks of three traditional convolutional layers in ResNet. These blocks maintain residual connections, and are followed by global average pooling and softmax layers as before. InceptionTime creates an ensemble of networks with randomly initialised weightings.

The **Hierarchical Vote Collective of Transform Ensembles, HIVE-COTE 1.0 (HC1)** [**2**], alongside the three algorithms above, are not significantly different to each other in terms of accuracy. Additionally, all are significantly more accurate on average than the best performing algorithms from the bake off comparison of time series classifiers five years prior [3].

The second release of HIVE-COTE, **HIVE-COTE 2.0 (HC2)** [**22**] is a heterogeneous ensemble of four classifiers built on four different base representations. HC2 is the only algorithm we are aware of which performs significantly better than the four algorithms above. In HC2 three new classifiers are introduced, with only the Shapelet Transform Classifier (STC) [5] retained from HC1. TDE [19] replaces the Contractable Bag-of-SFA-Symbols (cBOSS) [20]. The Diverse Representation Canonical Interval Forest (DrCIF) replaces both Time Series Forest (TSF) [12] and the Random Interval Spectral Ensemble (RISE) [17] for the interval and frequency representations. An ensemble of ROCKET [10] classifiers called the Arsenal is introduced as a new convolutional/shapelet based approach. Estimation of test accuracy via cross-validation is replaced by an adapted form of out-of-bag error, although the final model is still built using all training data.

## 2.2 Unsupervised Time Series Transformations

**Time Series Feature Extraction based on Scalable Hypothesis Tests (TSFresh)** [**8**] is a collection of just under 800 features[1] extracted from time series data. TSFresh is very popular with the data science community, and is frequently proposed as a good transform for classification. The **Highly Comparative Time Series Analysis (*hctsa*)** [**15**] toolbox can create over 7700 features[2] for exploratory time series analysis. Alongside basic statistics of time series values, *hctsa* includes features based on linear correlations, trends and entropy. Features from various time series domains such as wavelets, information theory and forecasting among others are also present. Both TSFresh and *hctsa* cover similar domains, extracting masses of summary features from the time series. Some of these extracted features will be similar, with differently paramaterised variations of the same feature included if applicable.

**The Canonical Time Series Characteristics (catch22)** [**18**] are 22 features chosen to be the most discriminatory of the full *hctsa* [15] set. This was determined by an evaluation over the UCR datasets. The *hctsa* features were initially pruned, removing those which are sensitive to mean and variance and any which could not be calculated on over 80% of the UCR datasets. A feature evaluation was then performed based on predictive performance. Any features which performed below a threshold were removed. For the remaining features, a hierarchical clustering was performed on the correlation matrix to remove redundancy. From each of the 22 clusters formed, a single feature was selected, taking into account balanced accuracy, computational efficiency and

---

[1] `https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html`
[2] `https://hctsa-users.gitbook.io/hctsa-manual/list-of-included-code-files`

interpretability. Like the *hctsa* set it was extracted from, the catch22 features cover a wide range of feature concepts.

**Time Series Intervals** are used in the interval based representation of TSC algorithms. Classifiers from this representation extract multiple phase-dependent subseries to extract discriminatory features from. Classifiers from this representation include TSF [12] and the Canonical Interval Forest (CIF) [21]. Both of these algorithms select intervals with a random length and position, extracting summary features from the resulting subseries and concatenating the output of each. This interval selection and feature extraction process can itself be used as an unsupervised transformation.

**Generalised Signatures [23]** are a set of feature extraction techniques primarily for multivariate time series based on rough path theory. We specifically look at the generalised signature method [23] and the accompanying canonical signature pipeline. Signatures are collections of ordered cross-moments. The pipeline begins by applying two augmentations by default. The basepoint augmentation simply adds a zero at the beginning of the time series, making the signature sensitive to translations of the time series. The time augmentation adds the series timestamps as an extra coordinate to guarantee each signature is unique and obtain information about the parameterisation of the time series. A hierarchical dyadic window is run over the series, with the signature transform being applied to each window. The output for each window is then concatenated into a feature vector.

## 3  Experimental structure

We perform our experiments on 112 equal length datasets with no missing values from the UCR time series archive [9]. We resample each dataset randomly 30 times in a stratified manner, with the first resample being the original train-test split from the archive. Each algorithm and dataset resample are seeded using the fold index to ensure reproducibility.

The transformations used in our experiments can be found in the Python `sktime`[3] package. Each transformer was built and saved to file, with the process being timed for our timing experiments. The classification portion of our pipelines, and the TSC algorithms used in our comparison, were run using the Java `tsml`[4] toolkit implementations. An exception for this is the deep learning approach InceptionTime, which we use the `sktime` companion package `sktime-dl`[5] to run.

To compare our results for multiple classifiers over multiple datasets we use critical difference diagrams [11]. We replace the post-hoc Nemenyi test with a comparison of all classifiers using pairwise Wilcoxon signed-rank tests, and cliques formed using the Holm correction as recommended in [16,4].

---

[3] `https://github.com/alan-turing-institute/sktime`
[4] `https://github.com/uea-machine-learning/tsml`
[5] `https://github.com/sktime/sktime-dl`

We create pipelines primarily using the transformations described in section 2.2, with the exception of the *hctsa* feature set, which required too much processing time and memory to be run in our timeframe. In addition to these transformations, we also include two benchmark transformations: Principal Component Analysis (PCA) and seven basic summary statistics. The seven statistics we use are the mean, median, standard deviation, minimum, maximum and the quantiles at 25% and 75%. PCA and basic summary statistics are the simplest transformations available, and perhaps one of the simplest approaches one could take towards TSC, alongside building classifiers on the raw time series and one-nearest-neighbour classification with Euclidean distance.

Our random interval transformation experiments extract 100 randomly selected intervals per dataset. We form two random interval pipelines, one extracting our basic summary statistics from each interval and the other extracting the Catch22 features.

For the classifier portion of our pipelines, we test three different vector based classifiers. Rotation Forest (RotF) [24] is the classifier of choice for the STC pipeline, and has shown to be significantly better than other popular approaches on problems only containing continuous attributes [1]. Extreme Gradient Boosting (XGBoost) [7] of Kaggle fame is our second classifier option. Our third option is a ridge regression classifiers with cross-validation to select parameters (RidgeCV), the better performing linear classifier suggested for the ROCKET pipeline [10].
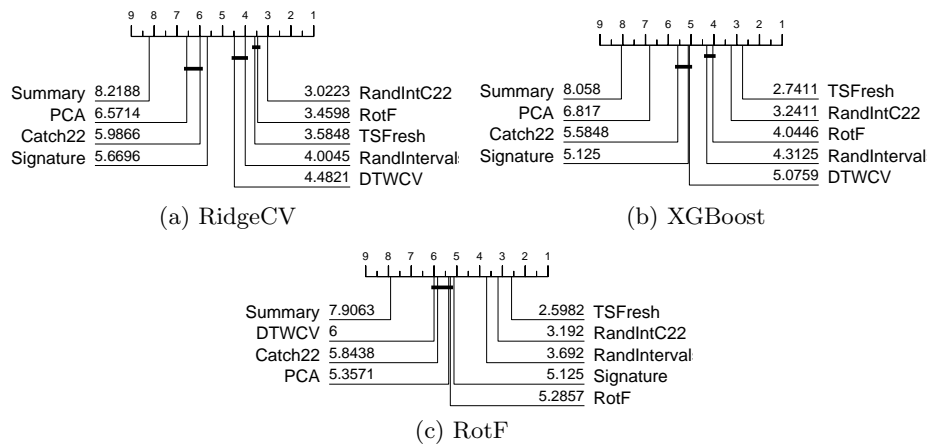
## 4 Results

We structure our results to answer three specific questions:

1. Which transformation is best given a specific classifier?
2. Which classifier is best, given a specific transform?
3. How do the pipeline classifiers compare to standard benchmarks?
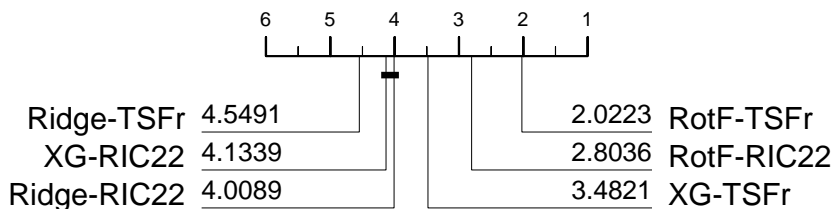4. How do the pipeline classifiers compare to state-of-the-art?

Figures 2 show the relative performance of difference transforms for our three base classifiers. We include for reference our two baseline classifiers, Rotation Forest (RotF) built on the raw time series and 1-nearest neighbour using dynamic time warping with a tuned window size (DTWCV).

The pattern of results is similar for all three classifiers: TSFresh and RandIntC22 are ranked top for all three classifiers. Both are significantly higher ranked than all the other transforms, and both baseline classifiers, except for the case of TSFresh with a ridge classifier. Summary statistics is always the worst approach and PCA, Catch22, Signatures are no better than, or worse than, the benchmark classifiers. RandomIntervals is significantly better than the benchmarks with rotation forest. There is an anomaly when drawing cliques (RotF and DTW are not always in the same clique despite there being no significant difference in all experiments), but the initial indications are clear: TSFresh and RandIntC22
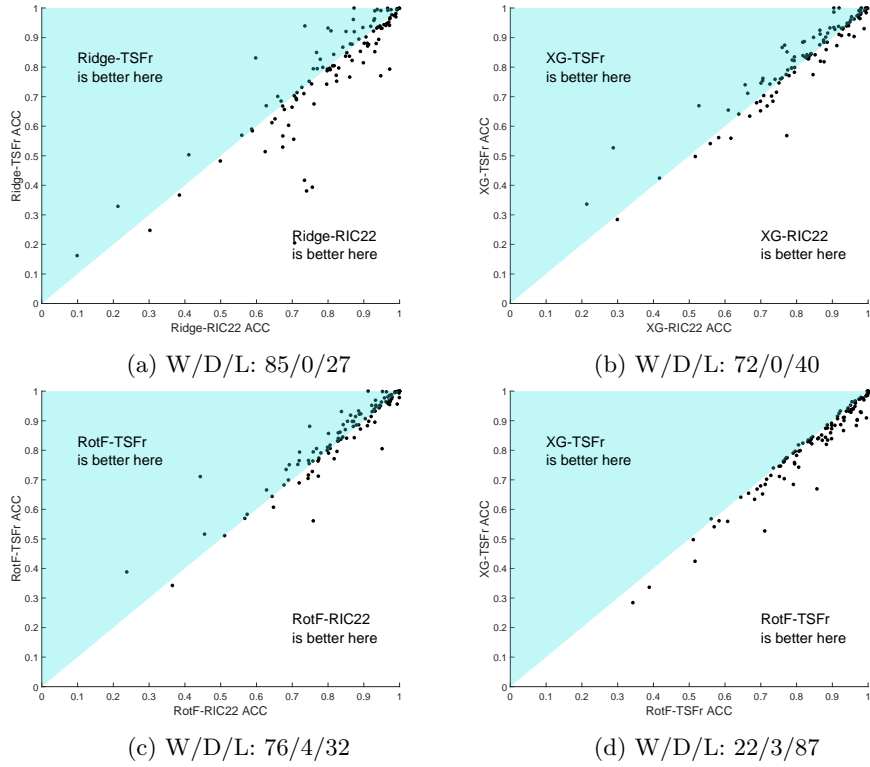
(a) RidgeCV      (b) XGBoost

(c) RotF

**Fig. 2.** Relative rank performance of seven transforms used in a simple pipeline with a linear ridge classifier (a), XGBoost (b) and rotation forest (c). TSFresh and RandInt22 are significantly better than all other transforms with most base classifiers.

are the best performing techniques and, with the possible exception of RandomIntervals, the others do not outperform the standard benchmarks, and are therefore of less interest. We investigate the relative performance of classifiers by comparing the two best transforms (TSFresh and RandIntC22) in combination with the three classifiers. Figure 3 shows that RotF is significantly better than RidgeCV and XGBoost for both transforms. This supports the argument made in [1] that rotation forest is the best classifier for problems with all continuous attributes. Figures 4 show the pairwise scatter plots for four pairs of pipelines. Figures (a), (b) and (c) show the difference in accuracies on the archive for both TSFresh and RandIntC22 using each of our base classifiers. Figure (d) compares our best performing pipeline, TSFresh with rotation forest, to the next best, TSFresh pipeline using XGBoost.
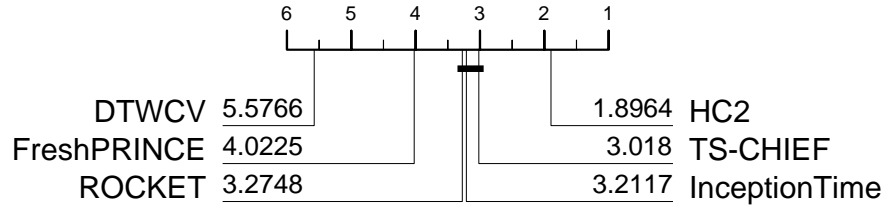


**Fig. 3.** Relative performance of three classifiers Rotation Forest, XBoost and RidgeCV (prefixes RotF, XG and Ridge) with two transforms TSFresh and RandIntCatch22 (suffix TSFr and RIC22). RotF is significantly better than the other classifiers, and RotF with TSFresh is the best overall combination.

**(a) W/D/L: 85/0/27**

**(b) W/D/L: 72/0/40**

**(c) W/D/L: 76/4/32**

**(d) W/D/L: 22/3/87**

**Fig. 4.** Pairwise scatter plots for TSFresh vs RandIntC22 with (a) RidgeCV, (b) XG-Boost and (c) rotation forest, and (d) the scatter plot of using TSFresh with XGBoost with TSFresh. (a), (b) and (c) demonstrate the superiority of TSFresh over Rand-IntC22. (d) shows that rotation forest significantly outperforms XGBoost.

Our primary finding is that the pipeline of TSFresh and rotation forest is, on average, the highest ranked and the most accurate simple pipeline approach for classifying data from the UCR archive. We feel the approach deserves a name better than RotF-TSFr. Hence, we call it the FreshPRINCE (Fresh Pipeline with RotatIoN forest Classifier). We investigate classification performance of the FreshPRINCE against the current and previous state of the art. Figure 5 shows FreshPRINCE against the very latest state of the art, HIVE-COTEv2.0 (HC2), the previously best performing algorithms, InceptionTime, TS-CHIEF and ROCKET and the popular benchmark, DTWCV. Figure 5 shows that Fresh-PRINCE does not achieve SOTA, but it does perform better than the popular benchmark 1-NN with DTW (DTWCV). Table 1 presents the summary performance measures averaged over all data. FreshPRINCE is approximately 6.5% more accurate than DTWCV, but on average 1.4% and 3.8% less accurate than ROCKET and HC2.

**Fig. 5.** Critical difference plot for FreshPRINCE against SOTA and DTW.

**Table 1.** summary performance statistics averaged over 112 UCR datasets. Test set accuracy (Acc), balanced accuracy (BalAcc), F1 statistic (F1), Area under the receiver operator curve (AUROC) and negative log likelihood (NLL).

| Classifier | Acc | BalAcc | F1 | AUROC | NLL |
|---|---|---|---|---|---|
| HC2 | 89.06% | 86.85% | 0.8575 | 0.9684 | 0.5245 |
| TS-CHIEF | 87.73% | 85.80% | 0.8475 | 0.9592 | 0.7479 |
| InceptionTime | 87.36% | 85.67% | 0.8443 | 0.9583 | 0.6104 |
| ROCKET | 86.61% | 84.58% | 0.8339 | 0.9536 | 1.5754 |
| FreshPRINCE | 85.22% | 82.98% | 0.8168 | 0.9565 | 0.7230 |
| DTWCV | 77.72% | 76.10% | 0.7449 | 0.7860 | 1.4796 |

Table 2 displays the run times for generating the results summarised in Figure 5 and Table 1. The FreshPRINCE is not as fast as the ROCKET classifier, but is still faster than then other SOTA TSC algorithms.

**Table 2.** RotF, Average (Minutes), Total (Hours), Max (Hours)

| | DTWCV | TSFresh | Rocket | InceptionTime | TS-CHIEF | HC2 |
|---|---|---|---|---|---|---|
| Average | 13.9545 | 10.5905 | 1.52939 | 46.3823 | 544.7552 | 182.844 |
| Total | 26.0485 | 19.7689 | 2.85461 | 86.5802 | 1016.8751 | 341.3084 |
| Max | 7.3248 | 3.856 | 0.4301 | 7.1093 | 166.7567 | 54.9177 |

We believe that, given the simplicity of the pipeline approach, the FreshPRINCE pipeline should be a benchmark against which new algorithms should be compared. If the claimed merits of an approach are primarily its accuracy, then we believe it should achieve significantly better accuracy than the simple approach of a TSFresh transform followed by a rotation forest classifier.

### 4.1 Implementation and Reproduction of Results

Given that we suggest FreshPRINCE as a benchmark classifier for new comparisons, we also provide resources for using it as such. We include our results for FreshPRINCE on the 112 UCR datasets used in this experiment on the time

series classification web page[6]. For experiments outside the UCR archive, we have implemented the pipeline in the Python *sktime* package. The most commonly used machine learning package for Python, *sklearn*, does not contain a rotation forest implementation. As such, we also include an implementation of the algorithm in *sktime*.

Listings 1.1 displays the process for running FreshPRINCE using the *sktime* package, loading data from its .ts file forma.

```python
from sktime.utils.data_io import
    load_from_tsfile_to_dataframe as load_ts
from sktime.classification.feature_based import FreshPRINCE

if __name__ == "__main__":
    # Load dataset
    trainX, trainY = load_ts("../Data/data_TRAIN.ts")
    testX, testY = load_ts("../Data/data_TEST.ts")

    # Create classifier and build on training data
    fresh_prince = FreshPRINCE()
    fresh_prince.fit(trainX, trainY)

    # Find accuracy on testing data
    accuracy = fresh_prince.score(testX, testY)
```

**Listing 1.1.** Running the FreshPRINCE pipeline in Python using *sktime*.

FreshPRINCE can also be run using a *sklearn* pipeline, using the *sktime* TSFresh transformer and rotation forest implementations as shown in Listing 1.2.

```python
    fresh_prince = Pipeline([
            (
                "transform",
                TSFreshFeatureExtractor(
                    default_fc_parameters="comprehensive"),
            ),
            ("classifier", RotationForest()),
        ])
```

**Listing 1.2.** Forming the FreshPRINCE pipeline in Python using *sktime* components and the *sklearn* Pipeline framework.

## 5 Conclusion

We have tested a commonly held belief that a simple pipeline of transformation and standard classifier is a useful approach for time series classification. We have found that there is some merit in this opinion: simple transformations such as PCA or summary stats are not effective, but more complex transformations such as TSFresh and random intervals with the Catch22 features do achieve a

---

[6] http://www.timeseriesclassification.com/results.php

respectable level of accuracy on average. They are significantly worse than state of the art in 2021 and 2020, but significantly better than the state from 10 years ago (DTWCV). We suggest the best performing pipeline, a combination of TSFresh and rotation forest we call FreshPRINCE for brevity, be used more commonly as a TSC benchmark.

## Acknowledgements

## References

1. Bagnall, A., Bostrom, A., Cawley, G., Flynn, M., Large, J., Lines, J.: Is rotation forest the best classifier for problems with continuous features? ArXiv e-prints arXiv:1809.06705 (2018), `http://arxiv.org/abs/1809.06705`
2. Bagnall, A., Flynn, M., Large, J., Lines, J., Middlehurst, M.: On the usage and performance of HIVE-COTE v1.0. In: proceedings of the 5th Workshop on Advances Analytics and Learning on Temporal Data. Lecture Notes in Artificial Intelligence, vol. 12588 (2020)
3. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery 31(3), 606–660 (2017)
4. Benavoli, A., Corani, G., Mangili, F.: Should we really use post-hoc tests based on mean-ranks? Journal of Machine Learning Research 17, 1–10 (2016)
5. Bostrom, A., Bagnall, A.: Binary shapelet transform for multiclass time series classification. proceedings of 17th International Conference on Big Data Analytics and Knowledge Discovery (2015)
6. Bostrom, A., Bagnall, A.: Binary shapelet transform for multiclass time series classification. Transactions on Large-Scale Data and Knowledge Centered Systems 32, 24–46 (2017)
7. Chen, T.: XGBoost: A Scalable Tree Boosting System. In: Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016)
8. Christ, M., Braun, N., Neuffer, J., Kempa-Liehr, A.W.: Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). Neurocomputing 307, 72–77 (2018)
9. Dau, H., Bagnall, A., Kamgar, K., Yeh, M., Zhu, Y., Gharghabi, S., Ratanamahatana, C., Chotirat, A., Keogh, E.: The UCR time series archive. IEEE/CAA Journal of Automatica Sinica 6(6), 1293–1305 (2019)
10. Dempster, A., Petitjean, F., Webb, G.: ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. Data Mining and Knowledge Discovery 34, 1454–1495 (2020)
11. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)

12. Deng, H., Runger, G., Tuv, E., Vladimir, M.: A time series forest for classification and feature extraction. Information Sciences 239, 142–153 (2013)
13. Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.: Deep learning for time series classification: a review. Data Mining and Knowledge Discovery 33(4), 917–963 (2019)
14. Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D., Weber, J., Webb, G., Idoumghar, L., Muller, P., Petitjean, F.: InceptionTime: finding AlexNet for time series classification. Data Mining and Knowledge Discovery 34(6), 1936–1962 (2020)
15. Fulcher, B., Jones, N.: hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. Cell Systems 5(5), 527–531 (2017)
16. García, S., Herrera, F.: An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. Journal of Machine Learning Research 9, 2677–2694 (2008)
17. Lines, J., Taylor, S., Bagnall, A.: Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles. ACM Transactions Knowledge Discovery from Data 12(5), 1–36 (2018)
18. Lubba, C., Sethi, S., Knaute, P., Schultz, S., Fulcher, B., Jones, N.: catch22: canonical time-series characteristics. Data Mining and Knowledge Discovery 33(6), 1821–1852 (2019)
19. Middlehurst, M., Large, J., Cawley, G., Bagnall, A.: The temporal dictionary ensemble (TDE) classifier for time series classification. In: proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. Lecture Notes in Computer Science, vol. 12457, pp. 660–676 (2020)
20. Middlehurst, M., Vickers, W., Bagnall, A.: Scalable dictionary classifiers for time series classification. In: proceedings of Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Science, vol. 11871, pp. 11–19 (2019)
21. Middlehurst, M., Large, J., Bagnall, A.: The canonical interval forest (CIF) classifier for time series classification. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 188–195. IEEE (2020)
22. Middlehurst, M., Large, J., Flynn, M., Lines, J., Bostrom, A., Bagnall, A.: Hive-cote 2.0: a new meta ensemble for time series classification. Machine Learning Online First, 1–33 (2021), http://link.springer.com/article/10.1007/s10994-021-06057-9
23. Morrill, J., Fermanian, A., Kidger, P., Lyons, T.: A generalised signature method for multivariate time series feature extraction. arXiv preprint arXiv:2006.00873 (2020)
24. Rodriguez, J., Kuncheva, L., Alonso, C.: Rotation forest: A new classifier ensemble method. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(10), 1619–1630 (2006)
25. Schäfer, P., Högqvist, M.: SFA: a symbolic Fourier approximation and index for similarity search in high dimensional datasets. In: proceedings of the 15th International Conference on Extending Database Technology. pp. 516–527 (2012)
26. Shifaz, A., Pelletier, C., Petitjean, F., Webb, G.I.: TS-CHIEF: a scalable and accurate forest algorithm for time series classification. Data Mining and Knowledge Discovery 34(3), 742–775 (2020)
27. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (June 2015)