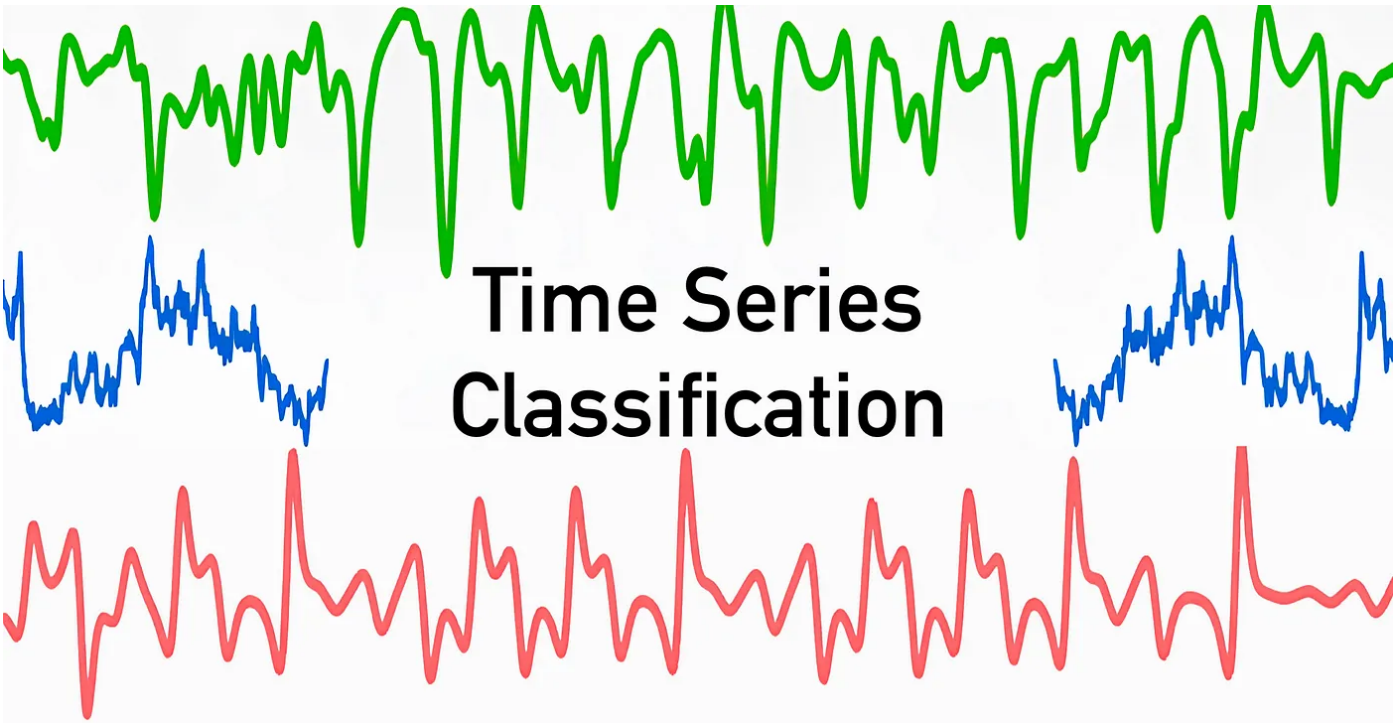
Member-only story

All 8 Types of Time Series Classification Methods

An exhaustive survey on algorithms for classifying time series

Jan Marcel Kezmann · Follow
Published in MLearning.ai
18 min read · Oct 27, 2022

Listen Share More



Classifying time series is one of the common tasks for applying machine and deep learning models.

This post will cover 8 types of time series classification methods and modeling techniques for classifying time series data. This ranges from a simple distance- or interval-based methods to the use of deep neural networks.

While this post will not cover any code, it will provide all the resources necessary for implementing any of the algorithms provided here. This post is intended to serve as a kind of reference article for all time series classification algorithms.

Even though the article is very long you can skip to any section by clicking on the respective link in the table of contents below. Most of the parts can be read without the necessity of reading the ones before it.

Table of contents

Time Series Definition

Before covering the various types of time series (TS) classification methods, a formal definition should be established. Thereby, TS can be divided either into univariate or multivariate TS.

- A univariate TS is an ordered set of (usually) real values.
- A multivariate TS is a set of univariate TS. Here, every single timestamp is a vector or array of real values.

Knowing that a data set of uni- or multivariate TS usually contains an (un-)ordered set of uni- or multivariate TS. Additionally, the dataset typically contains labels which are often represented by a one-hot encoded label vector, where its length represents the number of different classes.

The goal of TS classification is defined by training any type of model on the given dataset such that the model learns the probability distribution of the provided dataset. In other words, the model should learn to correctly assign a class label when given a TS.

Distance-Based Methods

Distance- or Nearest-Neighbor-based TS classification methods use various kinds of distance-based metrics to classify the given data. It is a supervised learning technique where the prediction result for a new TS depends on the information that is the labels of similar known time series.

The distance metric, which is the function that describes the distance between two or more time series, is decisive. Typical distance metrics are:

- p-norms (like Manhattan distance, Euclidean distance, etc.)
- Dynamic Time Warping (DTW)

Having decided on a metric one usually applies the k-nearest neighbors (KNN) algorithm, which measures the distance between a new TS and all TS in the trained

data set. Having calculated all distances, the k nearest is selected. Finally, the new TS is assigned the class to which most of the k nearest neighbors belong.

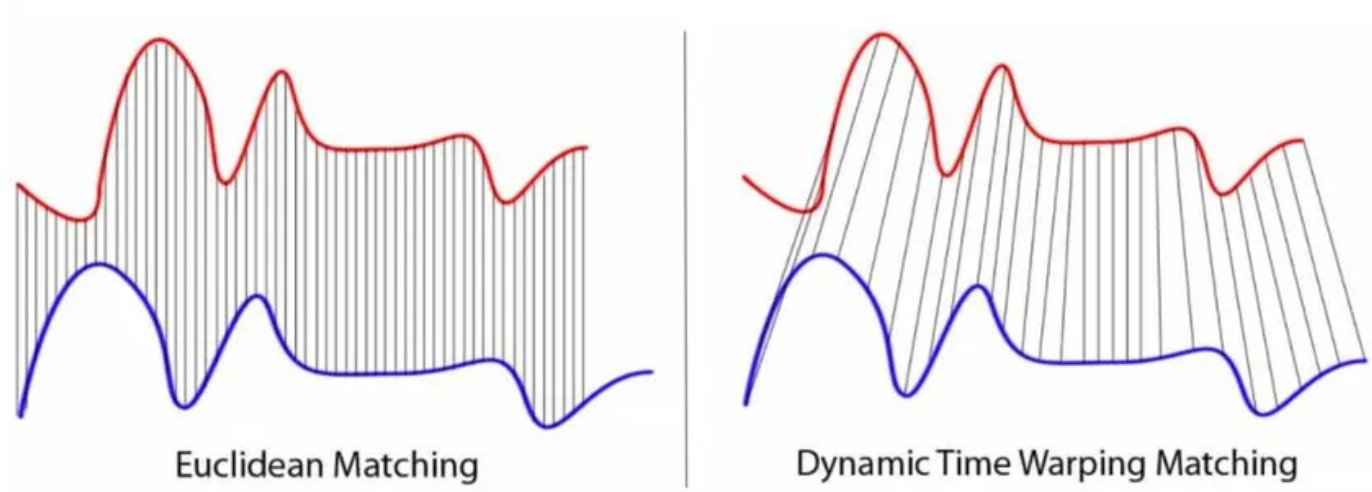
Choice of the distances: While the most popular norms are certainly the p-norms and especially the Euclidean Distance, they have two major drawbacks which make them not well suitable for the task of classifying TS.

- **First:** The norms are only defined for two TS of the same length, which in practice is not always given.
- **Second:** The norm only compares values of both TS at each time point independently, however, most values of TS are correlated with each other.

Solution: Dynamic Time Warping

DTW similarly to other metrics measures the distance between two given time series, however, it can address both limitations of the p-norms. This is done by taking into account slight shifts in the time series. The classical DTW minimizes the distance between two time series points with potentially different timestamps. This means that slightly shifted or distorted TS are still considered similar, which would not be the case with the other standards.

The image below visualizes the difference between the working of a p-norm based metric and DTW.



Source: ScanGAN360

In combination with a KNN DTW is considered as the baseline benchmarking algorithm for various benchmarking evaluations on TS classification.

Next to a KNN also decision tree-based approaches exist. For instance, the Proximity Forest algorithm models a decision tree forest that uses distance measures to partition the TS data.

Interval- and Frequency-Based Methods

Methods based on intervals typically split the TS into multiple distinct intervals. Then each subsequence is used to train an individual machine learning (ML) classifier. Thereby, an ensemble of classifiers is generated, each one acting on its own interval. Evaluating the most common classes among the individually classified subsequences returns the final label for the whole time series.

Time Series Forest

The best-known representative of the interval-based models is the time series forest (TSF). A TSF is an ensemble of decision trees built on random subsequences of the initial TS. Each tree is responsible for assigning a class to an interval.

This is done by computing summary features, typically the mean, standard deviation, and slope, to create a feature vector for each interval. Afterward, a decision tree is trained on the computed features and the prediction is received via a majority vote of all trees. The voting process is needed since every single tree only evaluates a certain subsequence of the initial TS.

Next to the TSF also other interval-based models exist. Variations of the TSF, for instance, use additional features like the median, interquartile range, min, and max of the respective subsequence.

In contrast to the classical TSF algorithms, there exist a rather sophisticated one called the Random Interval Spectral Ensemble (RISE) algorithm.

RISE

The RISE algorithm distinguishes itself in two ways from the classical TS forest.

- Use a single TS interval per individual tree
- It is trained by using spectral features extracted from the TS (instead of summary statistics as the mean, slope, etc.)

In the RISE technique, each decision tree is built on a distinct set of Fourier, autocorrelation, auto-regressive, and partial autocorrelation features. The algorithm then works as follows:

First random intervals of the TS are selected, and on these intervals, the above-mentioned features are computed. Then a new training set is created by combining the extracted features. On those, a decision tree classifier is trained. Lastly, these steps are repeated with different configurations to create an ensemble model, which is a random forest of individual decision tree classifiers.

Dictionary-Based Methods

Dictionary-based algorithms are another class of TS classifiers, which are based on the structures of dictionaries. They cover a large variety of different classifiers, that sometimes can be combined with the one mentioned above.

In case you want to skip to a specific one, here is the list of dictionary-based methods covered:

- Bag-of-Patterns (BOP)
- Symbolic Fourier Approximation (SFA)
- Individual BOSS
- BOSS Ensemble
- BOSS in Vector Space
- contractable BOSS

- Randomized BOSS
- WEASEL

Typically, this approach first transforms TS into sequences of symbols from which “words” are extracted via a sliding window. The final classification is then conducted by determining the distribution of the words, which often is done by counting and ranking them. The intuition behind this method is that times series are similar, that means of the same class if they contain similar words. The main process behind dictionary-based classifiers is usually the same.

1. Run a sliding window of a specific length over the TS
2. Transform each subsequence into a word (with a specific length and a fixed set of letters)
3. Create a histogram of these words

Below is a list of the most popular dictionary-based classifiers:

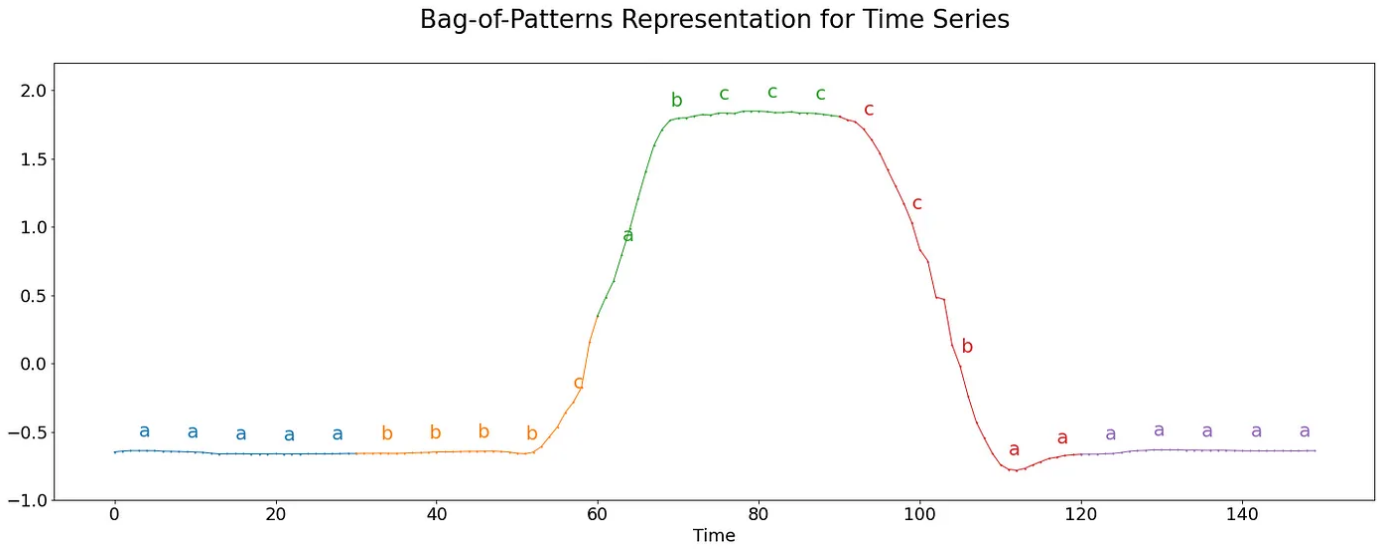
Bag-of-Patterns Algorithms

The Bag-of-Patterns (BOP) algorithm works similarly to Bag-of-Words algorithms used for the classification of text data. This algorithm counts the number of times a word occurs.

The most common technique to create words from numbers, here raw TS, is called Symbolic Aggregation approXimation (SAX). Thereby, you start by dividing the TS into different chunks as mentioned in 1. Usually, each chunk is getting standardized afterward, meaning its mean is 0 and the standard deviation is 1.

Often the length of a word is longer than the number of real values in a subsequence. Therefore one further applies binning on each chunk. Then the mean real value of each bin is computed which is then mapped to a letter. For instance, for all mean values below -1, the letter “a” is assigned, all values greater than -1 and smaller than 1 receive a “b” and all above 1 a “c”.

The figure below visualizes this procedure.



Here each segment contains 30 values which are binned into groups of 6, each bin is assigned one of the three possible letters making up a five-letter word.
Finally, the number of occurrences of each word can be summarized and used for the classification by plugging them into a nearest neighbor algorithm.

Next to the SAX method, also variations of it exist, like the SAX in Vector Space Model (SAX-VSM). Thereby, after applying the SAX algorithm the classes transformed into a Vector Space Model (VSM) using term frequencies (tf) and inverse document frequencies (idf).

Symbolic Fourier Approximation Algorithms

In contrast to the idea of BOP algorithms above, where the raw TS is taken and discretized into letters and then words, one can apply a similar methodology to Fourier coefficients of the TS.

The most known algorithm is the Symbolic Fourier Approximation (SFA), which in turn can be separated into 2 parts.

1. Compute the discrete Fourier transform of the TS, while keeping a subset of the computed coefficients.
   a) **Supervised Setting:** Normally, univariate feature selection is applied to select the higher-ranked coefficients based on statistics like the F-statistic or χ^2-statistic
   b) **Unsupervised Setting:** Usually a subset of the first coefficients is taken, which represents the trend of the TS

2. Every column of the resulting matrix is independently discretized, transforming the TS subsequence of a TS into a single word.
   a) **Supervised Setting:** Bin edges are computed such that an impurity criterion for instance entropy is minimized.
   b) **Unsupervised Setting:** The bin edges are calculated so that they are either based on the extreme values of the Fourier coefficients (bins are uniform) or that they are based on the quantiles of those (the number of coefficients in every bin is the same)

Based on the preprocessing explained above various kinds of algorithms exist that then further process the information to obtain predictions for TS.

BOSS Algorithm

The starting point makes the Bag-of-SFA-Symbols (BOSS) algorithm. The algorithm works as follows:

1. Extract subsequences of a TS via a sliding window mechanism
2. Apply the SFA transformation on each segment, returning an ordered set of words
3. Compute the frequencies of each word, which yields a histogram of words of the TS
4. In the end, classification is conducted by applying algorithms like the KNN combined with the custom BOSS metric, a slight variation of the Euclidean distance.

Variations of the BOSS algorithm

BOSS Ensemble

The BOSS ensemble algorithm is very often used to build multiple single BOSS models each differing from the others by the parameters: *word length, alphabet size, and window size*. The idea behind it is to capture various kinds of patterns of different lengths.

The multitude of models is obtained by conducting a grid search across the parameters and retaining only the best classifiers.

BOSS in Vector Space

The BOSS in Vector Space (BOSSVS) algorithm is a variation of the individual BOSS method which makes use of vector space models, similar to the Bag-of-Patterns

technique in SAX-VSM described above.

This method calculates for every class a single histogram coupled with the computation of the term frequency-inverse document frequency (TF-IDF) matrix. Then, by finding the class with the highest cosine similarity between the TF-IDF vector of each class and the histogram of the TS itself, the classification is obtained.

Contractable BOSS

Moving on to another variation of the BOSS ensemble algorithm, the contractable BOSS (cBOSS) algorithm. This method defines itself by being computationally significantly faster than the classic BOSS method.

This is achieved by not doing the grid search over the full parameter space but rather on randomly selected samples from it. Then cBOSS uses subsamples of the data for each base classifier. Next to that cBOSS improves the memory efficiency by only considering a fixed number of the best base classifiers in contrast to all classifiers above a certain performance threshold.

Randomized BOSS

The next variation of the BOSS algorithm presented here is the randomized BOSS (RBOSS). This method adds a random process to the choice of the length of sliding windows combined with clever ways of aggregating the predictions of the individual BOSS classifiers.

This similar to the cBOSS variant reduces the computation time while still maintaining benchmark performance.

WEASEL

By using sliding windows of various lengths inside the SFA transformation the Word Extraction for TS Classification (WEASEL) algorithm tries to enhance the performance of the standard BOSS method. It similarly to other BOSS variations transforms the TS into feature vectors using in that case various lengths of window sizes, which in turn are evaluated by a KNN classifier.

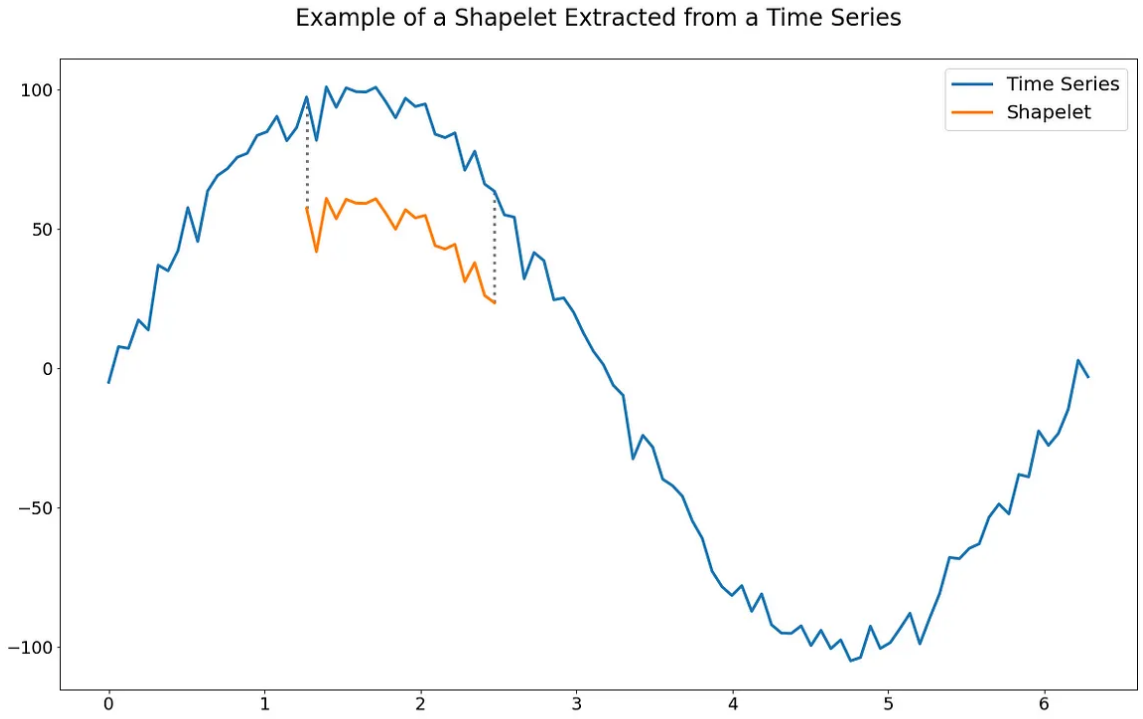
WEASEL distinguishes itself from other techniques by its specific method for deriving features, which is conducted by using only non-overlapping subsequences of each sliding window on which the χ^2-test is applied, filtering out the most relevant features.

Combining WEASEL with Multivariate Unsupervised Symbols and Derivatives (WEASEL+MUSE) makes it possible to extract and filter multivariate features from TS by encoding context information into each feature.

Shapelet-Based Methods

Shapelet-based methods are based on the idea of using subsequences, the shapelets, of the initial time series. The shapelets are chosen such that they can be used as a representative of a class, meaning that a shapelet contains the main characteristic of a class, which can be used to distinguish different classes from each other. In the optimal case, they can detect local similarities between TS within the same class independent of the phase of the TS.

The figure below gives an example of a shapelet. As you can see it is simply a subsequence of the whole TS.



A difficulty that arises with the use of shapelet-based algorithms is the problem of figuring out which shapelets to use. One can either pick the manually by crafting a set of shapelets, which however can be very hard. Another potentially better option is to automatically choose the shapelets with various kinds of algorithms.

There exist a handful of algorithms that rely on the use of shapelets. Two types of algorithms can thereby be distinguished from one another:

- Shapelet-extracting based algorithms
- Shapelet-learning based algorithms

Shapelet-Extracting Based Algorithms

Shapelet Transform, an algorithm proposed by Lines et al., is one of the most commonly used shapelet-extracting-based algorithms. Given a TS of n real-valued observations a shapelet is defined by a subset of that TS of length l.

Next, the minimum distance between the shapelet and the whole TS can be computed using the Euclidean distance— or any other distance metric — between the shapelet itself and all the shapelets of length l from the TS.

The algorithm then picks out the k best shapelets whose lengths belong to a certain range. This step can be viewed as some sort of univariate feature extraction in which every feature is defined by the distance between a shapelet and all the TS in the given dataset.

Afterward, the shapelets are ranked by some statistics. These are often the F-statistic or χ^2-statistic which rank the shapelets by their ability to distinguish the classes.

Having done the above steps one can apply any kind of ML algorithm to classify the new data set. Examples, in this case, are KNN-based classifiers, support vector machines, or random forests.

The search for the ideal shapelets is not the only difficulty that comes with the use of shapelet-based methods. Another problem can be the terrible time complexity that quadratically increases with the number of training samples and increases even to the fourth degree in the length of the shapelet.

Shapelet-Learning Based Algorithms

Shapelet-learning-based algorithms try to address the limitations of shapelet-transform-based ones. The idea is to learn a set of shapelets that are able to discriminate the classes instead of extracting them from the given data set directly.

Doing that has two major advantages:

- It can lead to shapelets that are not contained in the training set but are strongly discriminative of the classes.
- One may not run the algorithm over the whole data set which can reduce training time significantly

Anyway, this method comes with some drawbacks too that arise with the use of a differentiable minimization function and the classifier of choice.
Instead of the Euclidean distance, one has to rely on differentiable functions such that the shapelets can be learned via a gradient descent or backpropagation algorithm. Most commonly one relies on the LogSumExp function, which is a smooth approximation of the maximum by taking the logarithm of the sum of the exponentials of its arguments. This is paired with for instance the Logistic Regression as a simple ML classifier.

Since the LogSumExp function is not strictly convex, the optimization algorithm may not converge properly, meaning it can lead to bad local minima. Further, more often than not, one adds multiple hyperparameters to tune since the optimization process itself is the main component of the algorithm.

Nonetheless, the method can be very useful in practice leading to new insights into the discrimination of different classes.

Kernel-Based Algorithms

A slight variation to the shapelet-based algorithms is the kernel-based ones. The idea behind those is to learn and use random convolutional kernels, such as the ones most commonly known for computer vision algorithms, which extract features from a given TS.

The Random Convolutional Kernel Transform (ROCKET) algorithm was specifically designed for this purpose. It uses a large number of these kernels which all differ in length, weights, bias, dilation, and padding and are randomly created from fixed distributions.

On top of the extracted kernels, a classifier is put which is able to select the most relevant features for the discrimination of the classes. Therefore, the original paper used ridge regression, an L2 regularized variant of linear regression, to perform the prediction. Using it comes with two benefits, first its computational efficiency even for multi-class classification problems, and second the simplicity of fine-tuning the only regularization hyperparameter using cross-validation.

One of the core benefits of using kernel-based algorithms or more specific variations of the ROCKET algorithm is the comparable low computational cost of using them.

Feature-Based Methods

Feature-based methods in general can cover most of the algorithms that use some sort of feature extraction on given time series which are then taken by a classification algorithm to perform the prediction.

Regarding the features nearly everything is possible, ranging from simple statistical features to more complex fourier-based ones. A huge list of such features can be found in the hctsa repository, however, trying out and comparing every feature may be an exhaustive task, especially for larger data sets. Therefore the CAnonical Time-series CHaracteristics (catch22) algorithm has been developed.

catch22 algorithm

This method aims to infer a small set of TS features that not only exhibit strong classification performance but are further minimally redundant. Therefore catch22 selects 22 features in total from the hctsa library that offers more than 4000 features.

The developers of the method received the 22 features by training various models on 93 different data sets and evaluated the best-performing TS features on it, yielding a small

subset that still maintains great performance while being extremely fast. The classifier on top can be chosen freely which makes it another parameter to tune for.

Matrix Profile Classifier

Another feature-based method is the Matrix Profile (MP) classifier, which is an MP-based interpretable TS classifier that can maintain benchmark performance while providing interpretable results.

To accomplish this the designer adapted ideas from shapelet-based classifiers by extracting patterns that are possible to extract from a model named Matrix Profile. The idea behind the model is to represent all distances between subsequences of a TS with their nearest neighbor. With that, an MP is able to efficiently extract characterizing patterns of a TS such as motifs, which are subsequences of a TS that are very similar to each other, and discords, which in contrast describe sequences that are dissimilar to each other.

As a classification model in theory again any model can be used. The developer of this method, however, went for a Decision Tree Classifier.

Next to these two mentioned methods sktime offers a handful more feature-based TS classifiers. Of course, it is also possible to define additional manual features yourself, which may be ideally suited to the data.

Model Ensembles

Model ensembles themselves are no separate type of algorithm but rather a technique for combining various kinds of TS classifiers to create a better-combined prediction. The use of model ensembles makes sense for two reasons: The first is the reduction of variance that is achieved by combining multiple individual models, similar to the random forest using a multitude of decision trees. Second, using various types of different learning algorithms leads to a much broader and more diverse set of learned features which in turn may lead to a better discriminative power of the classes.

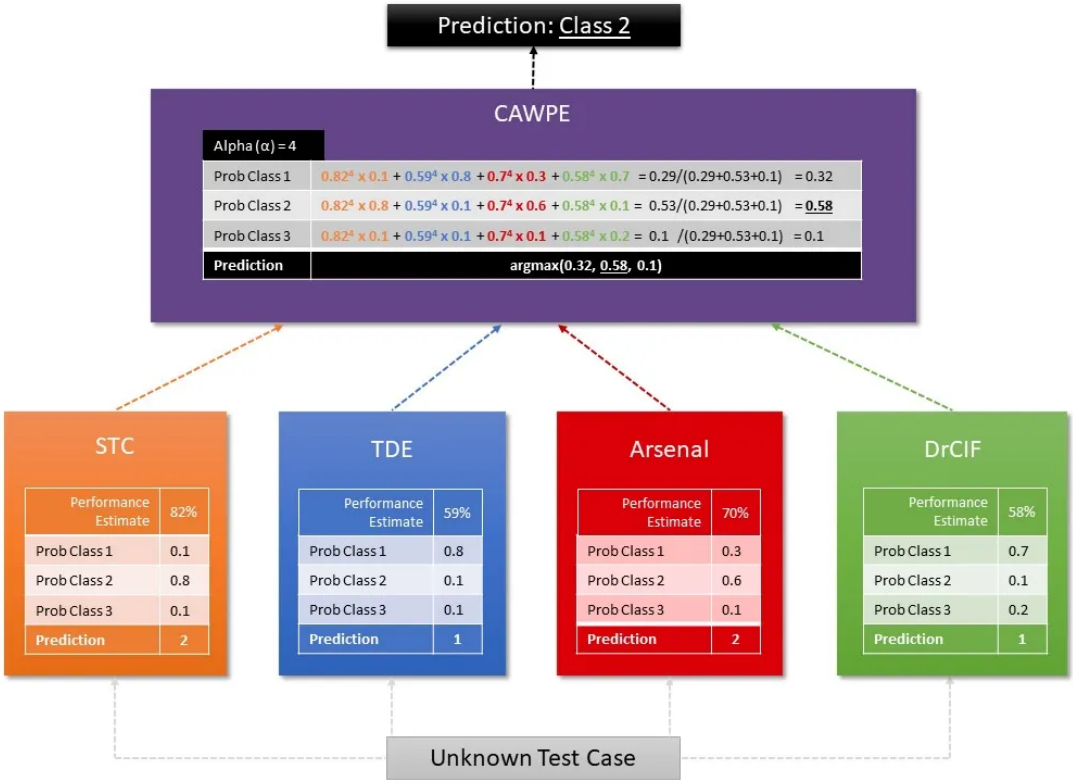
The most popular model ensembles are the Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE). There exist many different kinds of

similar versions of it where all have in common that they combine the information, that is the prediction, of different classifiers by using a weighted average over every single one of them.

Sktime, for instance, uses two different kinds of HIVE-COTE algorithms, where the first combines the probabilities of each estimator, which consist of a shapelet transform classifier (STC), a TS forest, a RISE, and a cBOSS. The second one is defined by the combination of STC, Diverse Canonical Interval Forest Classifier (DrCIF, a variation of the TS Forest), Arsenal (an ensemble of ROCKET models), and TDE (a variation of the BOSS algorithm).

The final prediction is obtained by the CAWPE algorithm, which assigns weights to each classifier, which are obtained by the relative estimate quality of a classifier found on the training data set.

The figure below is a common illustration used for visualizing the working structure of a HIVE-COTE algorithm:



Source: HIVE-COTE 2.0: a new meta ensemble for time series classification

Deep Learning Based Methods

Regarding deep learning based algorithms, one could probably write a very long post on its own explaining all the details about every architecture. Here, I just aim to provide some of the benchmark models and techniques commonly used for TS classification.

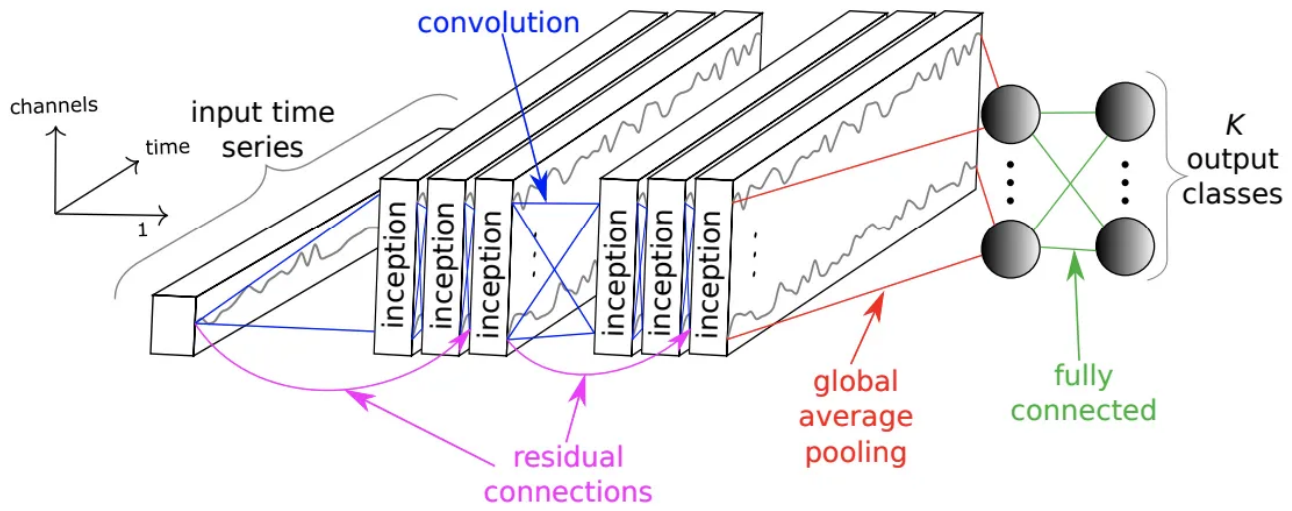
While deep learning based algorithms are extremely popular and widely studied in fields like computer vision and NLP, they are rather uncommon in the field of TS classification. Nonetheless, Fawaz et al. conducted on their paper on deep learning for TS classification, an exhaustive study of the currently existing methods. Thereby over 60 neural networks (NN) models with six types of architectures were studied:

- Multi-Layer Perceptron
- Fully Convolutional NN (CNN)
- Echo-State Networks (based on Recurrent NNs)
- Encoder
- Multi-Scale Deep CNN
- Time CNN

Most of the above-mentioned models were initially developed for different use cases, this paper provides, therefore, a strong baseline on how to use a neural network for TS classification.

In addition to those methods which are nowadays classical NNs, especially for computer vision tasks, in 2020 the InceptionTime network has been created. The InceptionTime is an ensemble of five deep learning models, where each one is created by cascading inception modules first proposed by Szegedy et al. These inception modules apply multiple filters of various lengths simultaneously to a TS while extracting relevant features and information from shorter as well as longer subsequences of the TS.

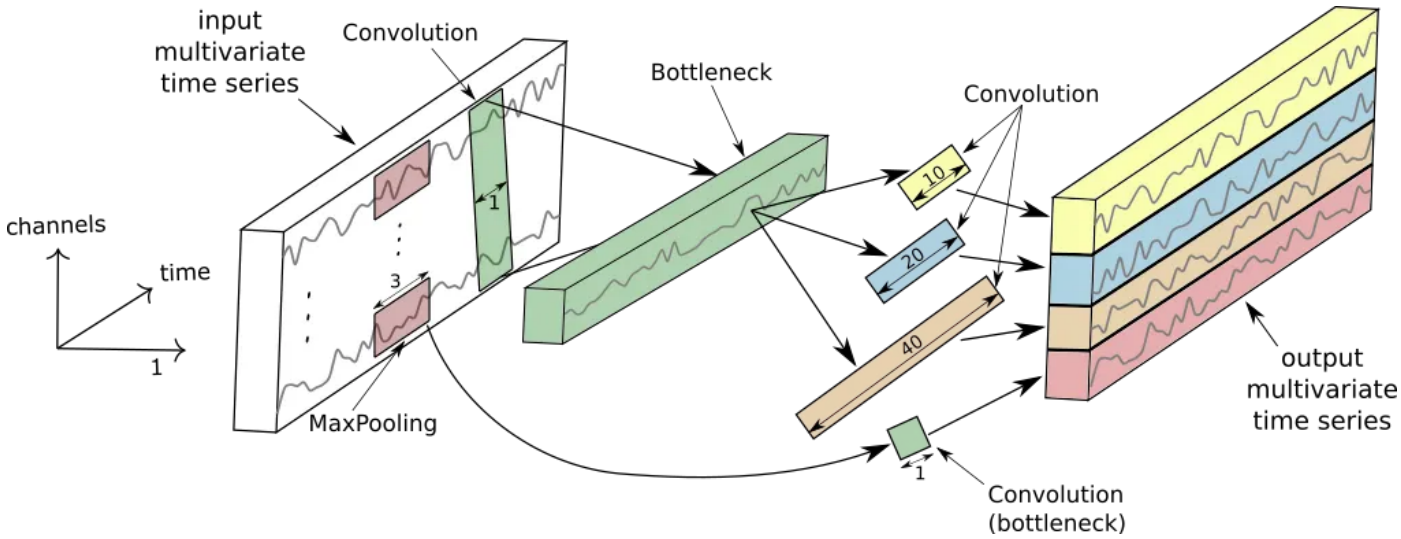
The figure below displays the modules of the InceptionTime network.



InceptionTime, Source: InceptionTime: Finding AlexNet for Time Series Classification

As one can see it consists of multiple inception modules stacked in a feedforward manner and additionally connected with residual connections. Finally, global average pooling combined with a simple fully connected NN produces the predictions.

The last figure displays the working of a single inception module.



Inception Module, Source: InceptionTime: Finding AlexNet for Time Series Classification

Final Notes

I hope that the large lists of algorithms, models, and techniques help you not only understand the vast field of time series classification methods but further provide you with a great overview in general.

Below you find some final sources for time series libraries and repositories as well as some additional references.

> *You can also take a look at my GitHub and other articles on Medium.*

Libraries and Repositories for TS classification

- pyts, a Python package for TS classification
- sktime, a unified framework for machine learning with time series
- sktime-dl, an extension package for deep learning with Keras for sktime
- tslearn, a Python package providing ML tools for the analysis of TS
- tsai, a state-of-the-art Deep Learning library for TS and Sequences

References

[1] Johann Faouzi. Time Series Classification: A review of Algorithms and Implementations. Ketan Kotecha . Machine Learning (Emerging Trends and Applications), Proud Pen, In press, 978–1–8381524- 1–3. ffhal-03558165

[2] Fawaz, Hassan Ismail, et al. “Deep learning for time series classification: a review”

[3] Dinger, Timothy R., et al. “What is time series classification?”

[4] Edin, Frederik, Time series classification — an overview

[5] Anion, Alexandra, A brief introduction to time series classification algorithms

Mlearning.ai Submission Suggestions

How to become a writer on Mlearning.ai

medium.com

Time Series | Classification | Time Series Model | Types Of Classifiers | Ml So Good

Follow

Written by Jan Marcel Kezmann

933 Followers · Writer for MLearning.ai

AI enthusiast, practitioner and writer. I write about AI, ML and Data Science in general. Join Medium with https://medium.com/@jan_marcel_kezmann/membership

More from Jan Marcel Kezmann and MLearning.ai

Jan Marcel Kezmann in MLearning.ai

10 Best Free-to-Use Text-to-Image Generators

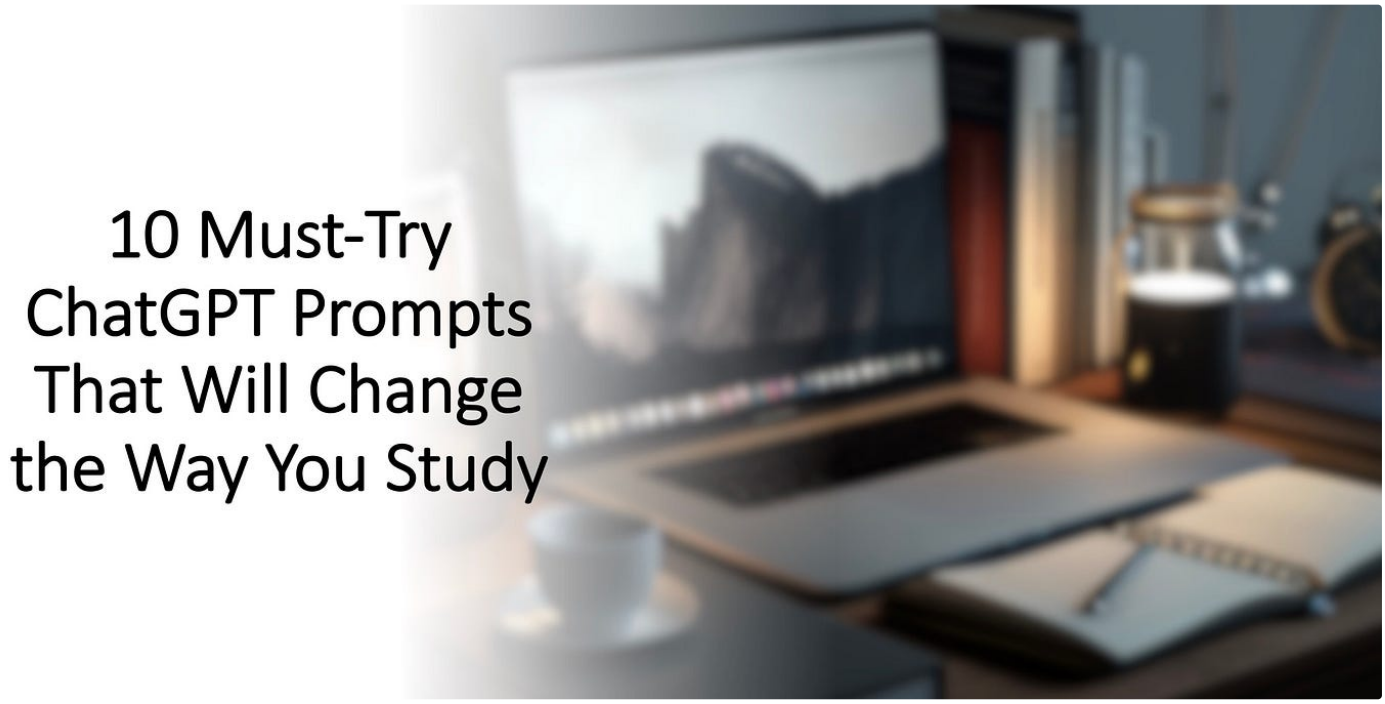
generating oil paintings, realistic photos, and more

· 5 min read · Oct 13, 2022

The ChatGPT list of lists: A collection of 3000+ prompts, examples, use-cases, tools, APIs...

Updated June 30, 2023. Superintelligence links added.

10 min read · Feb 7



Jan Marcel Kezmann in MLearning.ai

10 Must-Try ChatGPT Prompts That Will Change the Way You Study

Are you a student looking for ways to improve your learning, writing, and research skills? Or are you simply searching for a learning...

✨ · 8 min read · Feb 19

 29  

See all from Jan Marcel Kezmann

See all from MLearning.ai

Recommended from Medium

Coucou Camille in CodeX

Time Series Prediction Using LSTM in Python

Implementation of Machine Learning Algorithm for Time Series Data Prediction.

· 6 min read · Feb 10

Zain Baquar in Towards Data Science

Time Series Forecasting with Deep Learning in PyTorch (LSTM-RNN)

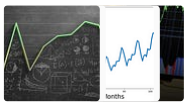
An in depth tutorial on forecasting a univariate time series using deep learning with PyTorch

· 12 min read · Feb 9

Lists

ChatGPT prompts

17 stories · 46 saves



Predictive Modeling w/ Python

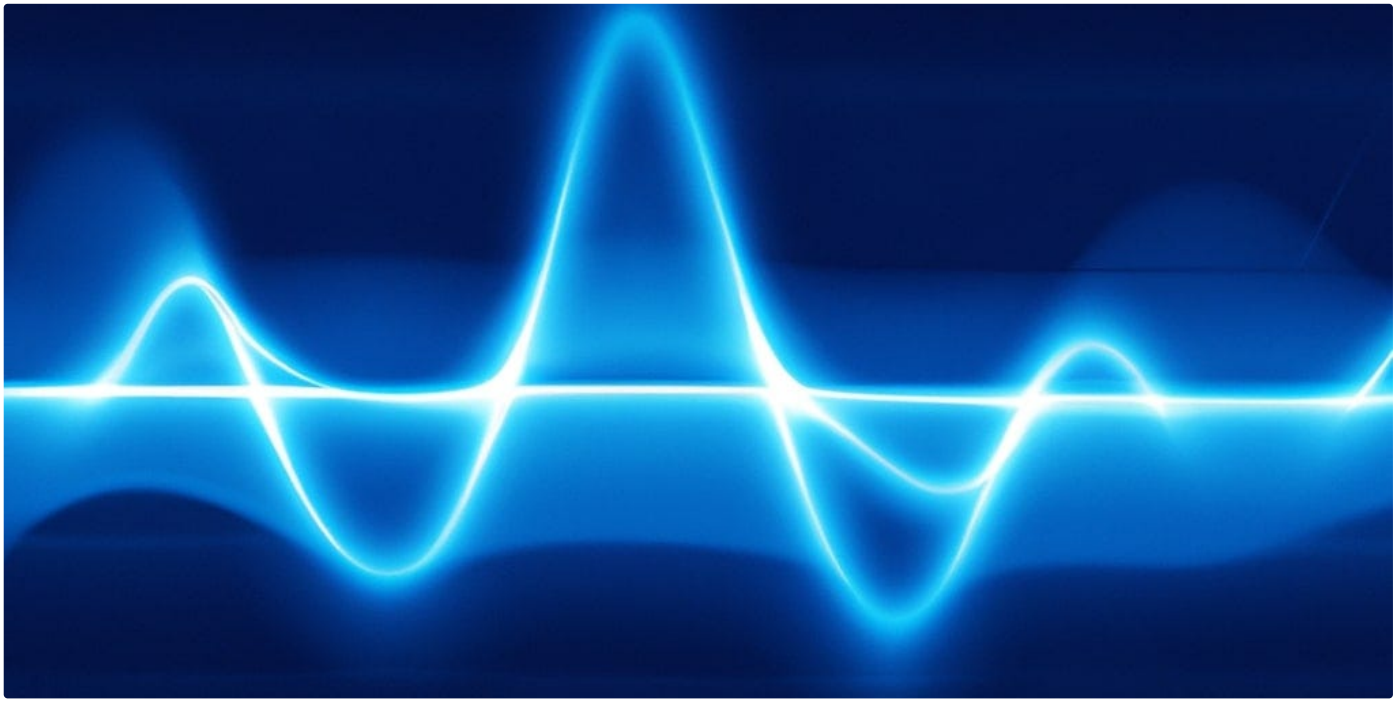
18 stories · 62 saves

Practical Guides to Machine Learning

10 stories · 76 saves

Tech & Tools

15 stories · 7 saves



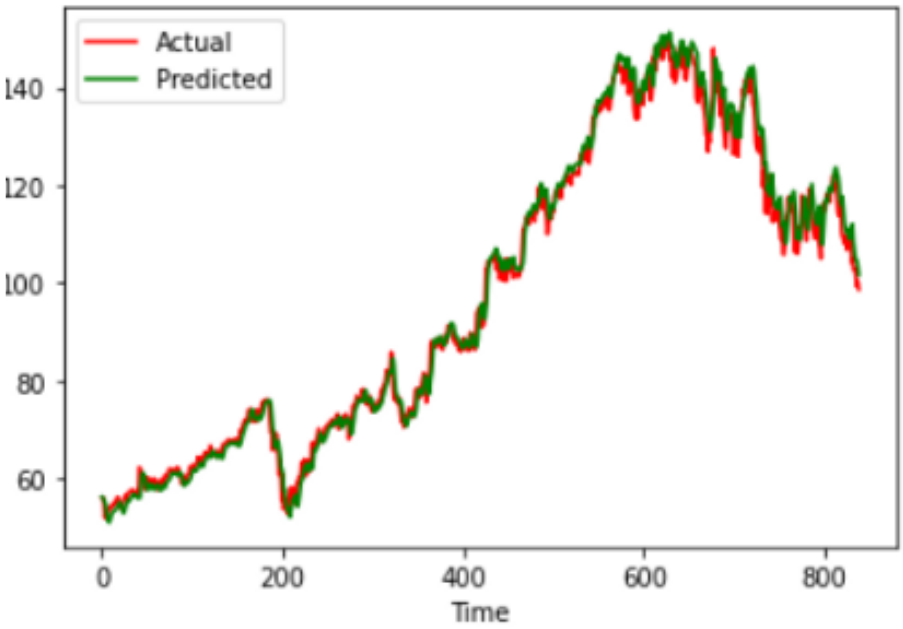
Nikos Kafritsas in Towards Data Science

Temporal Fusion Transformer: Time Series Forecasting with Deep Learning —Complete Tutorial

Create accurate & interpretable predictions

✨ · 12 min read · Nov 4, 2022

1.5K 18



Connor Roberts

Forecasting the stock market using LSTM; will it rise tomorrow.

Using a machine learning model, this tutorial will predict a stock's future value in real time with high accuracy on a stock exchange. To...

✨ · 6 min read · Jan 5

113 2

Marco Peixeiro in Towards Data Science

PatchTST: A Breakthrough in Time Series Forecasting

From theory to practice, understand the PatchTST algorithm and apply it in Python alongside N-BEATS and N-HiTS

· 10 min read · Jun 20

Isaac Godfried in Towards Data Science

Advances in Deep Learning for Time Series Forecasting and Classification: Winter 2023 Edition

The downfall of transformers for time series forecasting and the rise of time series embedding methods. Plus advances in anomaly detection...

· 16 min read · Jan 9