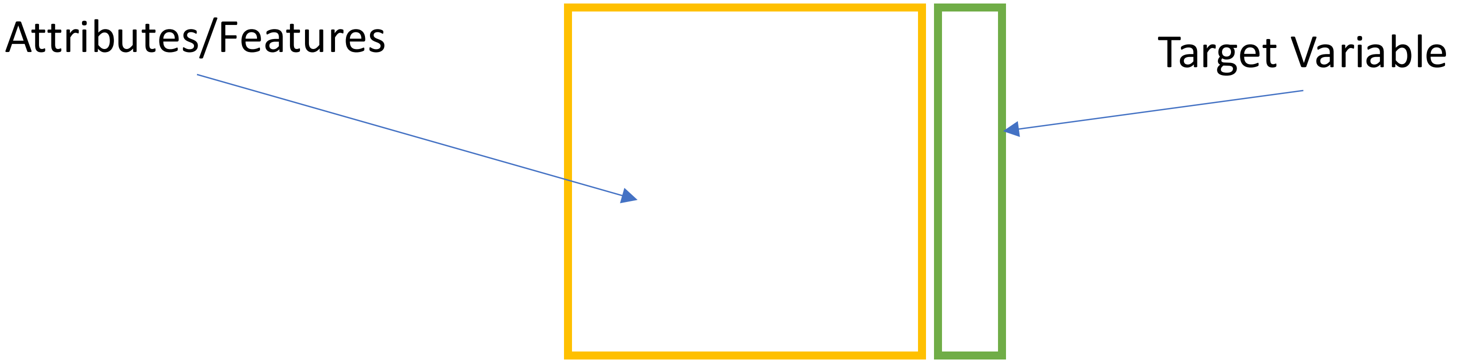
DATA MINING 2
Time Series Classification

Riccardo Guidotti

a.a. 2024/2025

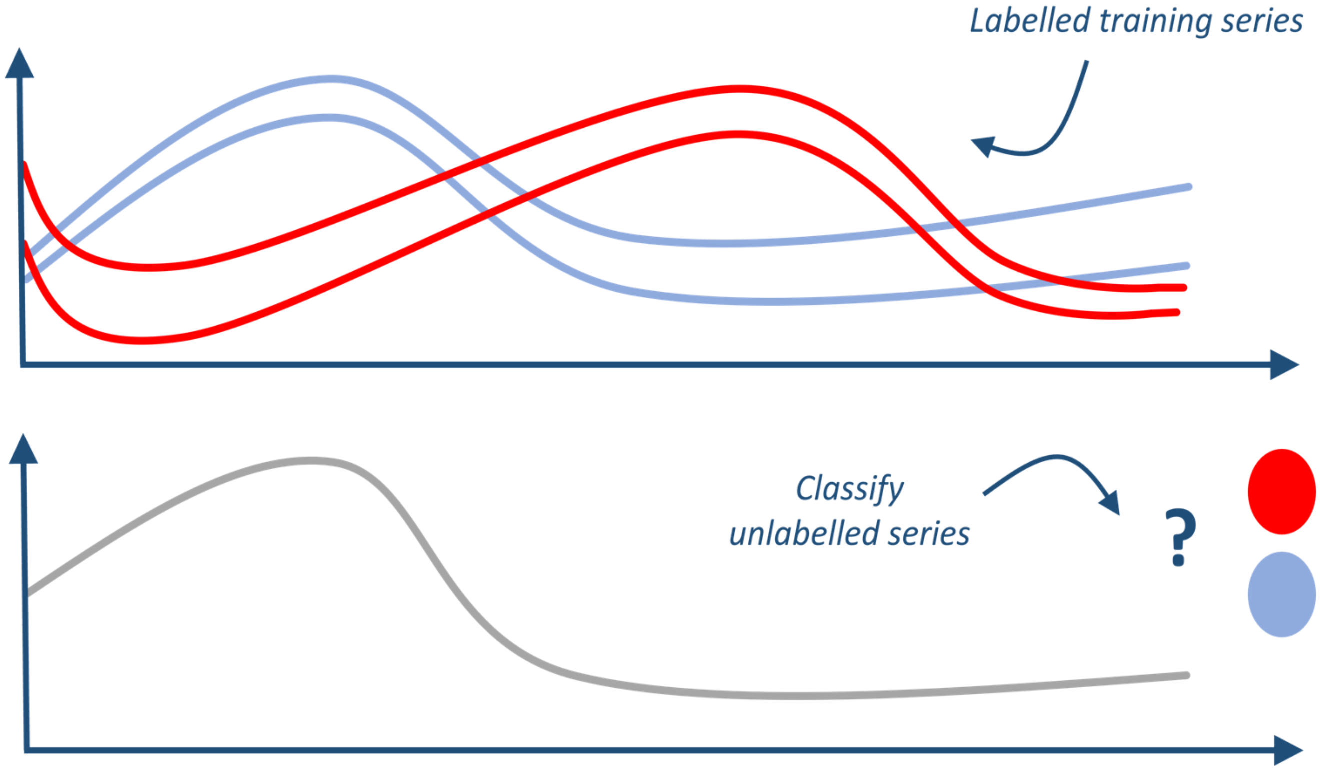
Supervised Learning

- Supervised learning refers to problems where the value of a target attribute should be predicted based on the values of other attributes.
- Problems with a ***categorical target*** attribute are called **classification**, problems with a ***numerical target*** attribute are called **regression**.



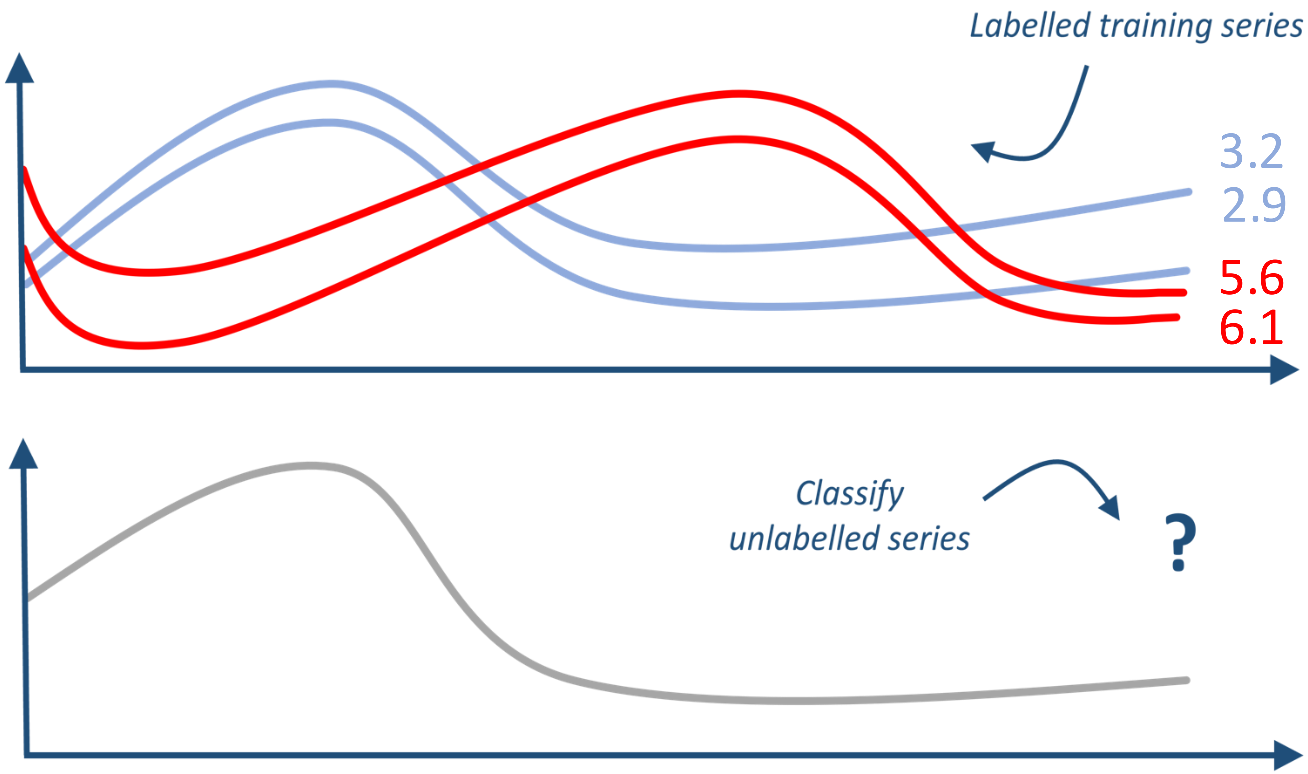
Time Series Classification - TSC

- Given a dataset $X = \{\boldsymbol{T_1}, \ldots \boldsymbol{T_n}\}$, TSC is the task of training a model f to predict an exogenous <u>categorical</u> output y for each time series T, i.e., $f(T) = y$.



Time Series Extrinsic Regression - TSER

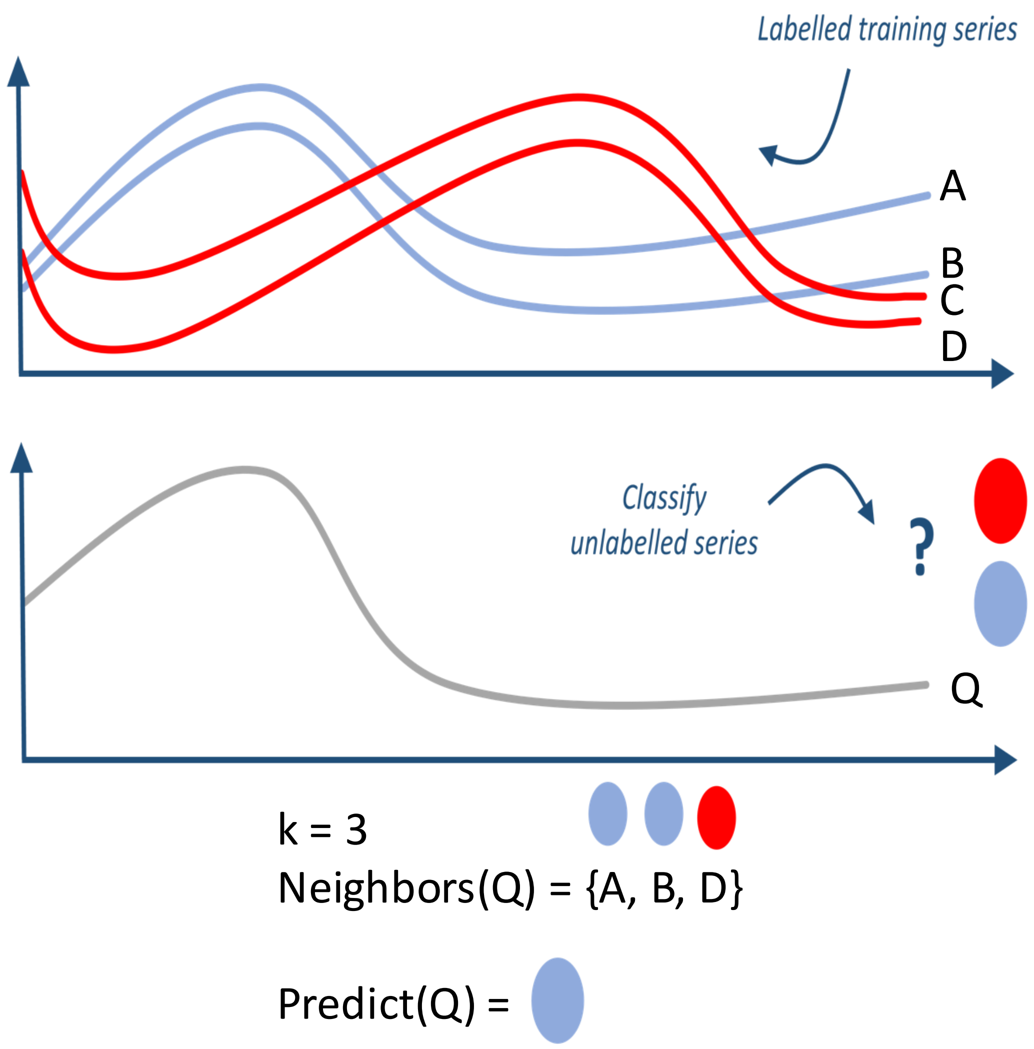
- Given a dataset $X = \{\boldsymbol{T_1}, \ldots\ \boldsymbol{T_n}\}$, TSER is the task of training a model f to predict an exogenous continuous output y for each time series T, i.e., $f(T) = y$.



Instance-based Models

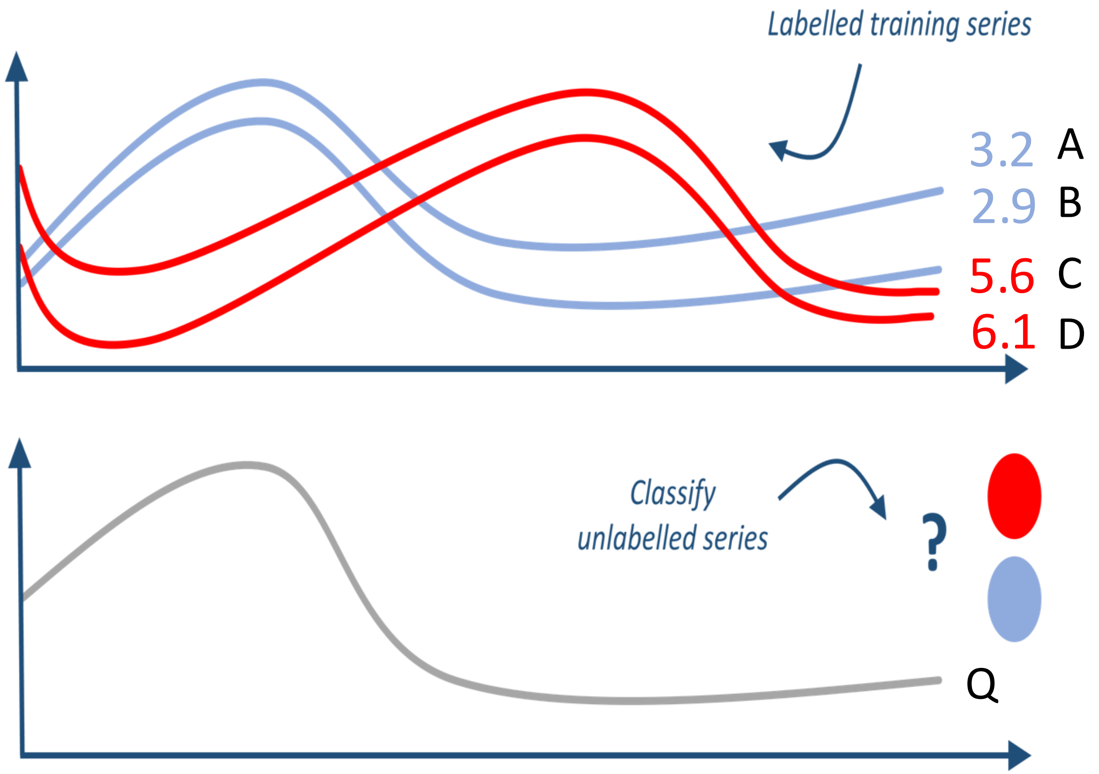
Nearest-Neighbor Classifier (K-NN)

- Basic idea: If it walks like a duck, quacks like a duck, then it is probably a duck.
- Given a set of training records, and a test record:
    1. Compute the distances from the test to the training records.
    2. Identify the k "nearest" records.
    3. Use class labels of nearest neighbors to determine the class label of test record (e.g., by taking majority vote).



Nearest-Neighbor Classifier (K-NN)

- Basic idea: If it walks like a duck, quacks like a duck, then it is probably a duck.
- Given a set of training records, and a test record:
  1. Compute the distances from the test to the training records.
  2. Identify the k "nearest" records.
  3. Use target values of nearest neighbors to determine the target value of test record (e.g., by making the average).



k = 3 3.2 2.9 6.1

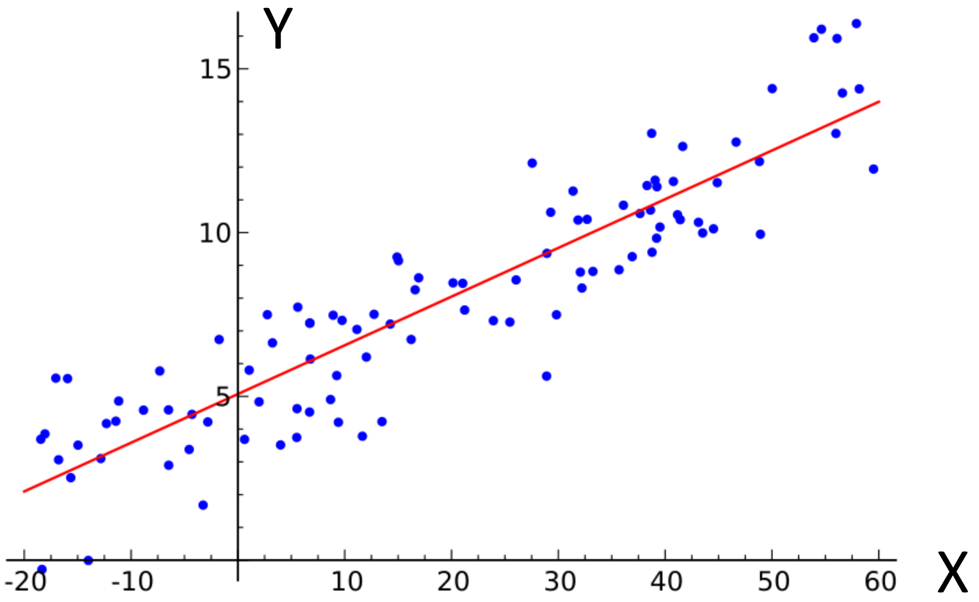
Neighbors(Q) = {A, B, D}

Predict(Q) = 3.8

Linear Models

Linear Regression

- **Linear regression** is a linear approach to modeling the relationship between a *dependent variable Y* and one or more *independent* (explanatory) variables *X*.
- The case of *one* explanatory variable is called **simple linear regression**.
- For *more than one explanatory variable*, the process is called **multiple linear regression**.
- For *multiple correlated dependent variables*, the process is called **multivariate linear regression**.



Simple Linear Regression

Dependent Variable   Independent Variable

Linear Model: $$Y = mX + b \qquad Y = \beta_1 X + \beta_0$$

Slope   Intercept (bias)

- Such linear relationship may not hold exactly for all the population.
- We call the deviations from *Y* errors or residuals, i.e., $y_i - f(x_i)$
- The objective of linear regression is to find values for the parameters *m* and *b* which would provide the “best fit” for the observed points.

Tree-based Models

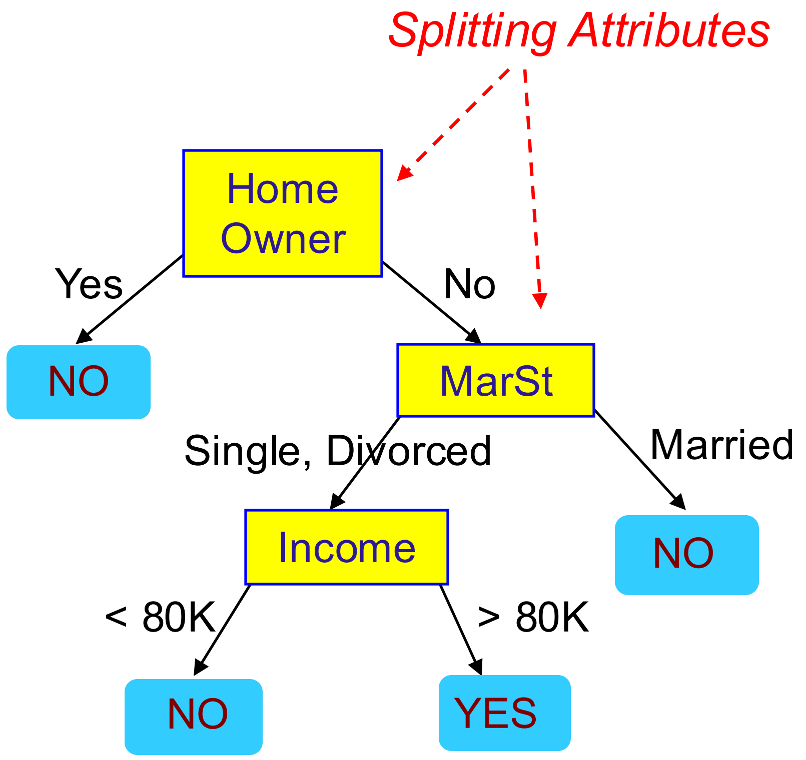
Example of a Decision Tree

Consider the problem of predicting whether a loan borrower will repay the loan or default on the loan payments.

categorical categorical continuous class

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data

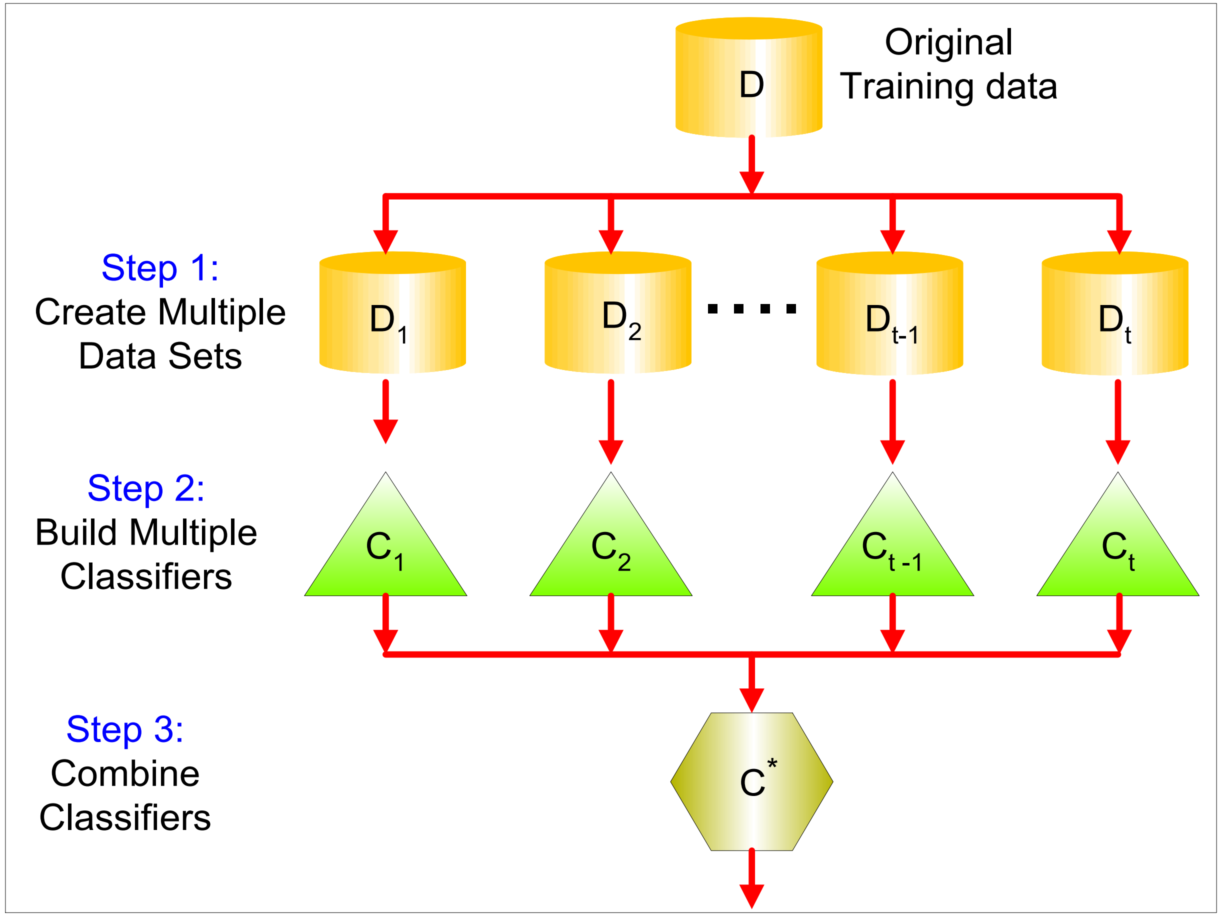


Model: Decision Tree

Trees Ensemble Models

Ensemble Methods

- Improves the accuracy by **aggregating the predictions** of multiple classifiers.
- Construct a set of **base models** from the training data.
- Make the prediction of test records by combining the predictions made by multiple base models.
  - Majority for classification
  - Average for regression

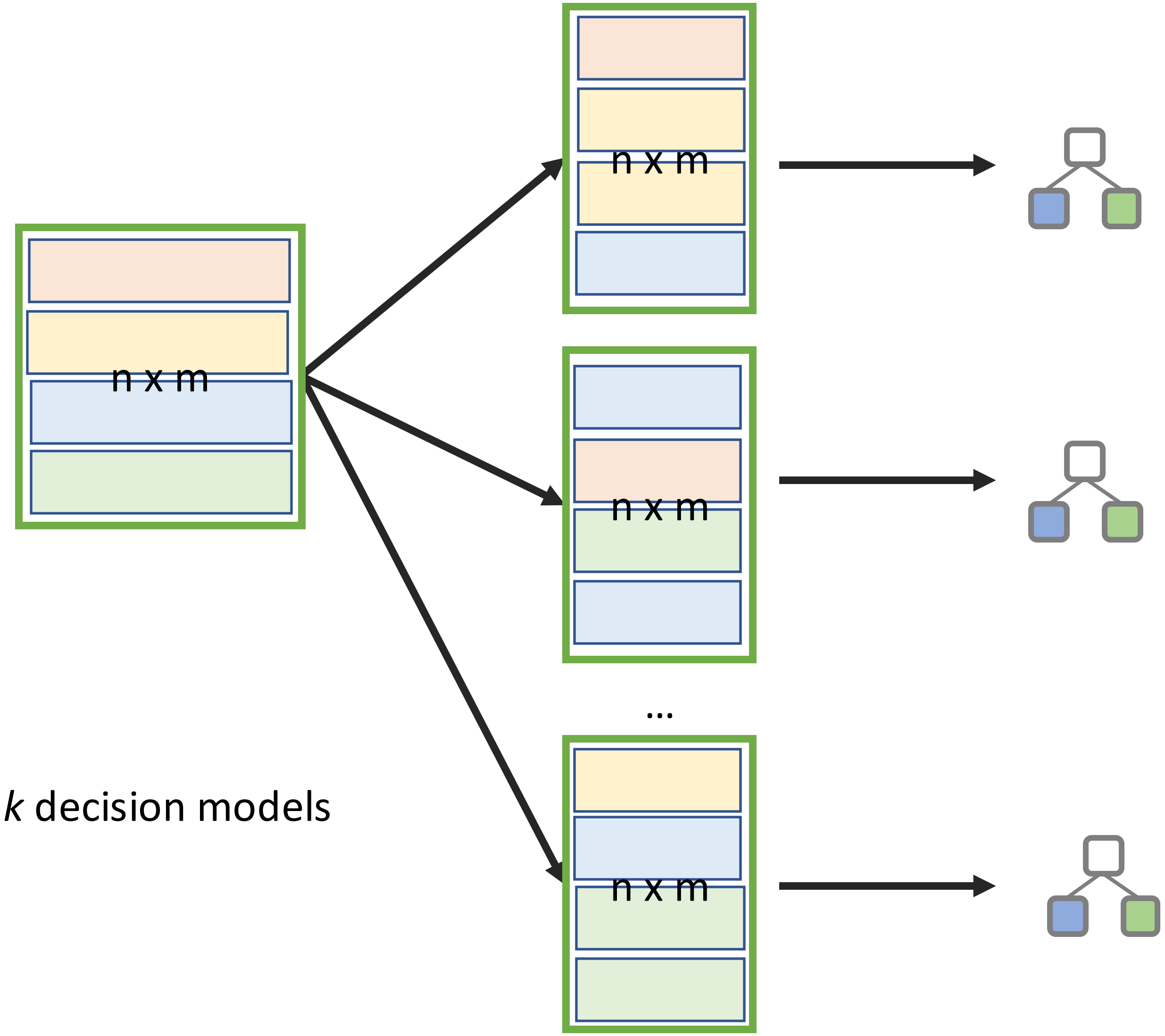


Types of Ensemble Methods

- Manipulate data distribution
  - Bagging
  - Boosting

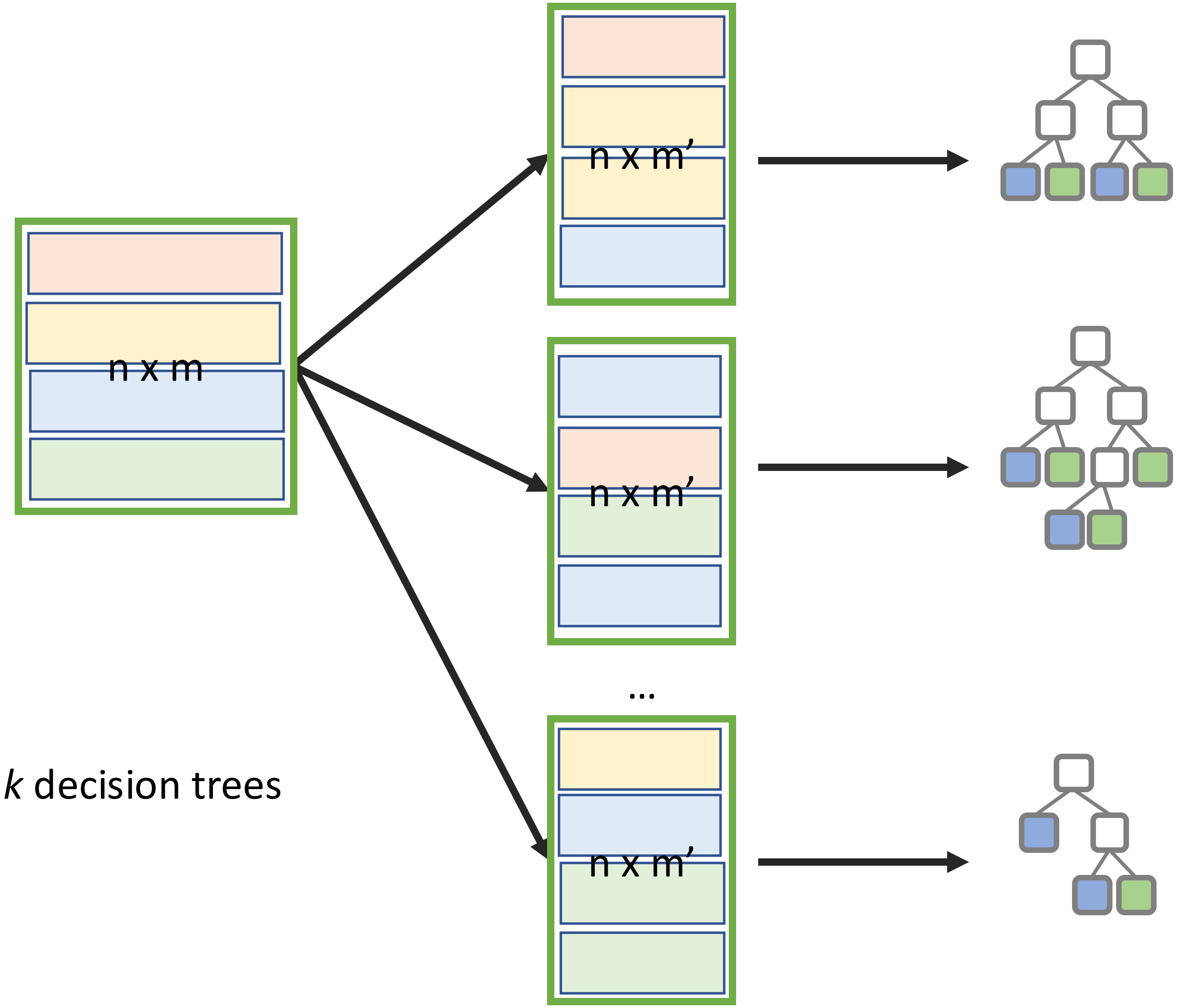
- Manipulate input features
  - Random Forests

Bagging (a.k.a. Bootstrap AGGregatING)



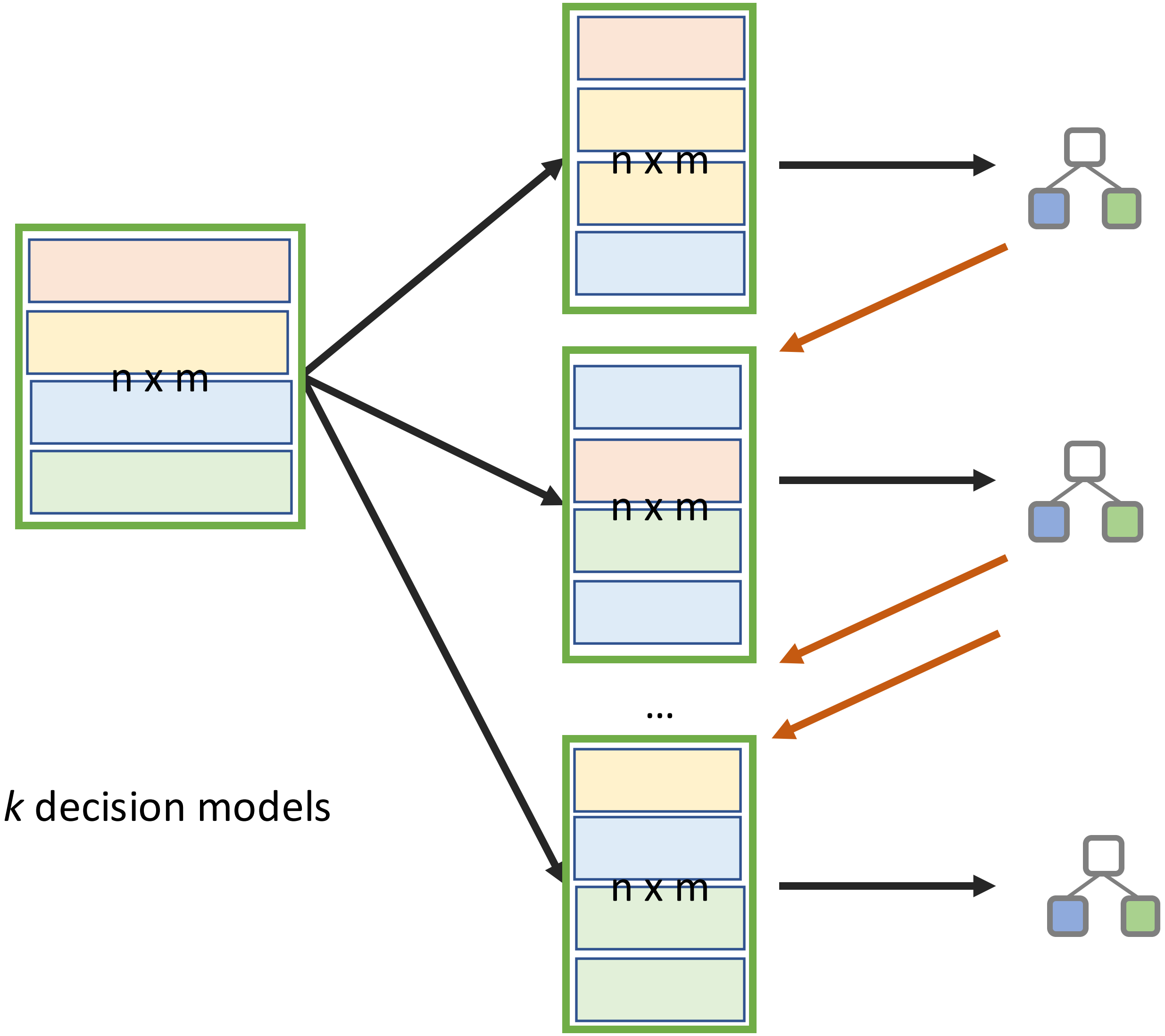
k decision models

Random Forest



k decision trees

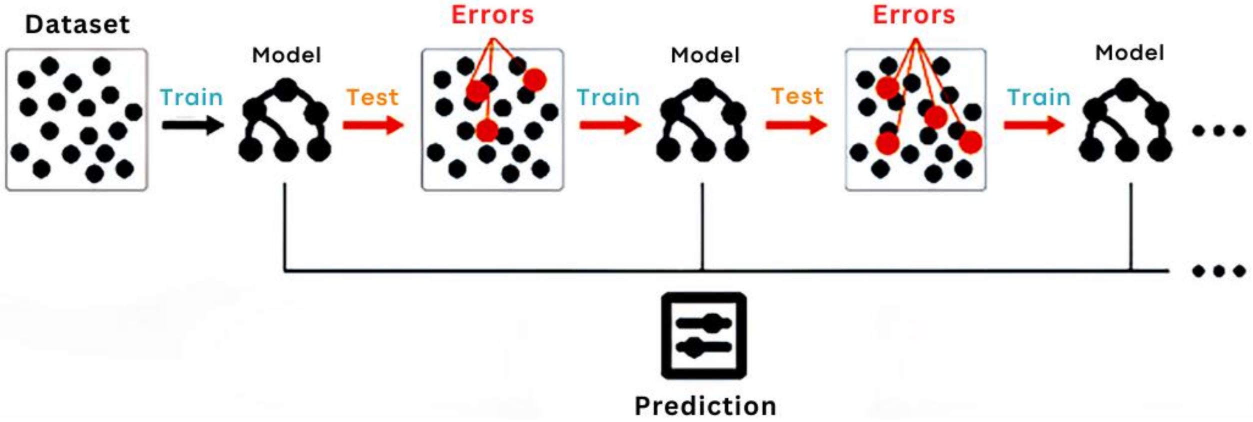
Boosting



k decision models

Gradient Boosting Machines

- First builds a naïve model F_0 considering the entire training set as mode for classification, mean for regression
- Iteration *i*:
  - Applies F_i on the training set and calculate the prediction (pseudo)-residual as the difference between the real value and the predicted one.
  - Then train a decision tree regressor DTR_i to predict the (pseudo)-residual
  - Creates a model as $F_{i+1} = F_i$ + LearningRate * DTR_i
  - Repeats iterations a predefined number of times or until the sum of the (pseudo)-residual is smaller than a certain threshold.

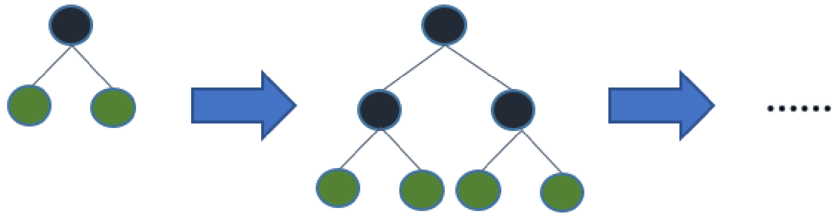


Gradient Boosting Machines Improved

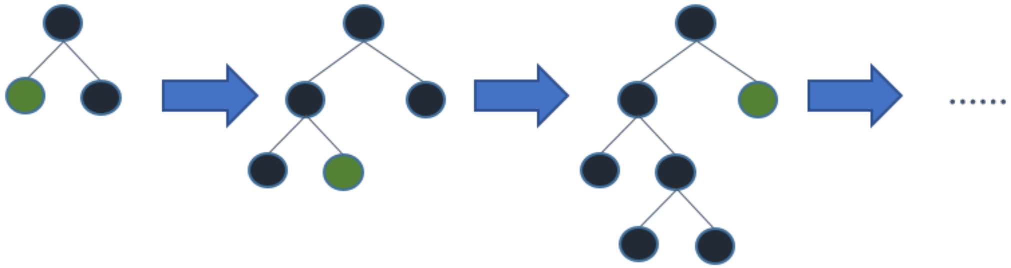
- XGBoost (Extreme Gradient Boost)
- LightGBM (Light Gradient Boosting Machine)
- CatBoost (Categorical Boosting)
- All designed for large complex datasets

XGBoost vs LightGBM vs CatBoost

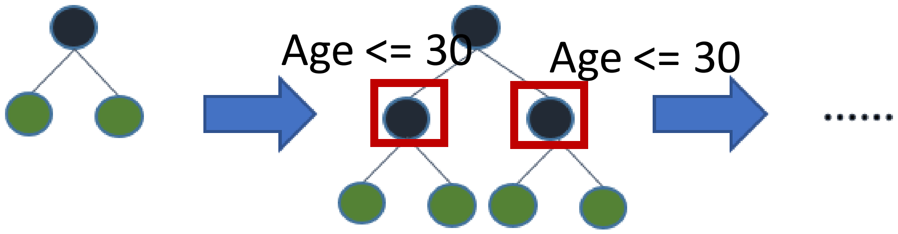
- XGBoost: level-wise (horizontal) growth
- XGBoost: novel splitting function



- LightGBM: out leaf-wise (vertical) growth
- LightGBM significantly faster than XGBoost with almost equivalent performance



- CatBoost: symmetric decision trees
- CatBoost: designed for categorical attributes

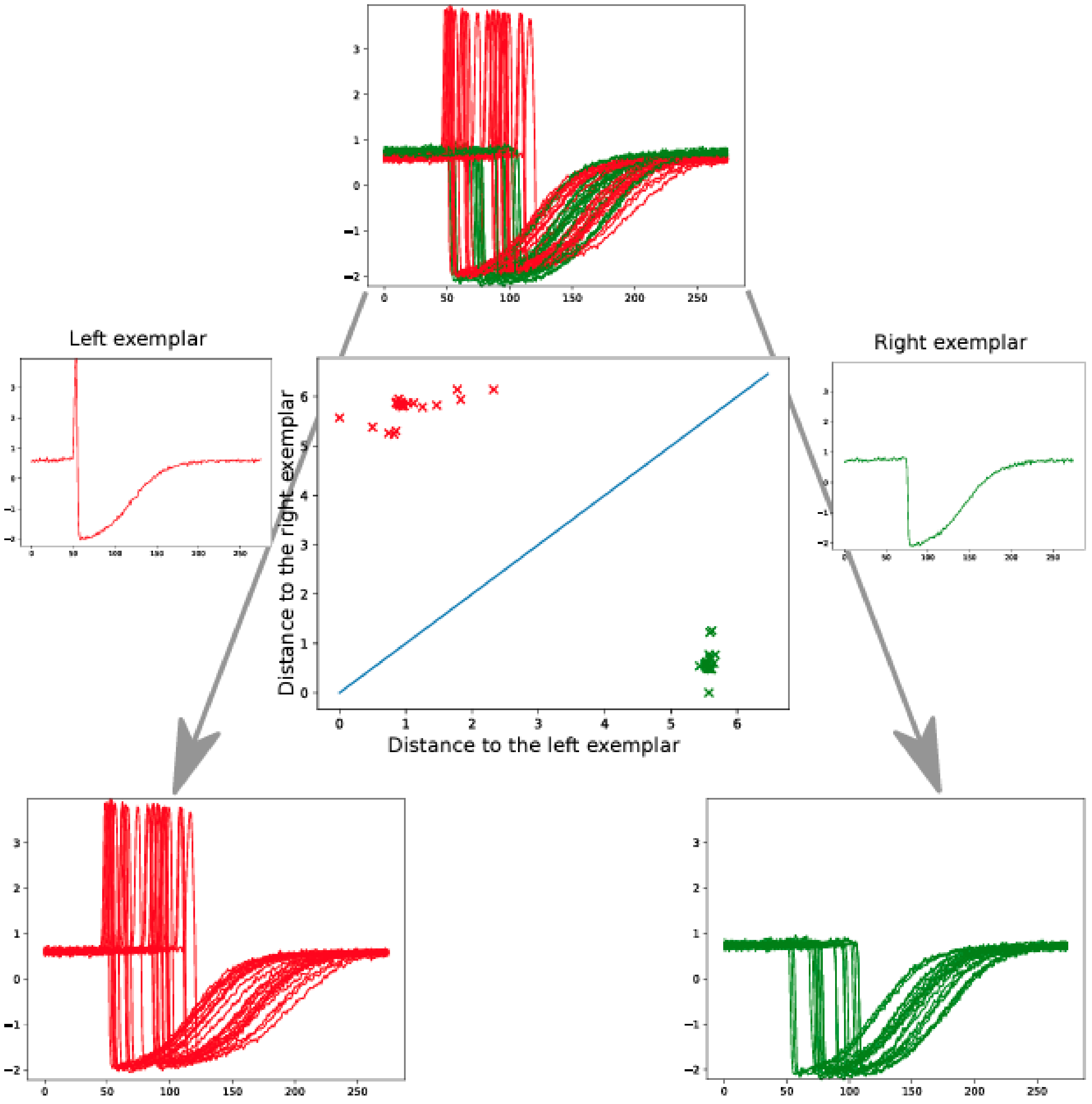


Problems with Tree-based Models in TS

- Decision trees make a split on the value of an attribute.
- Treating the values of a raw TS at each time stamp as belonging to a single attribute independent from the others does not work well on TS as the values in the time stamps are not independent as analyzed by the trees.
- Furthermore, the resulting tree is only limitedly interpretable vanishing the structure of the model.
- Thus, it is advisable to use tree-based models on TS only after having represented them in forms of independent features such as using global structural features or time-independent approximations.

Proximity Forest

- A Proximity Forest is an ensemble of *k* Proximity Trees.
- Each branch of an internal node has an associated exemplar TS.
- A test TS follows the branch corresponding to the exemplar to which it is closest according to a parameterized similarity measure.
- *R* sets of candidate exemplars are randomly selected for each split.
- Among the *R* candidate exemplars are selected those with the highest Information Gain (if *R=1* the choice is completely random).

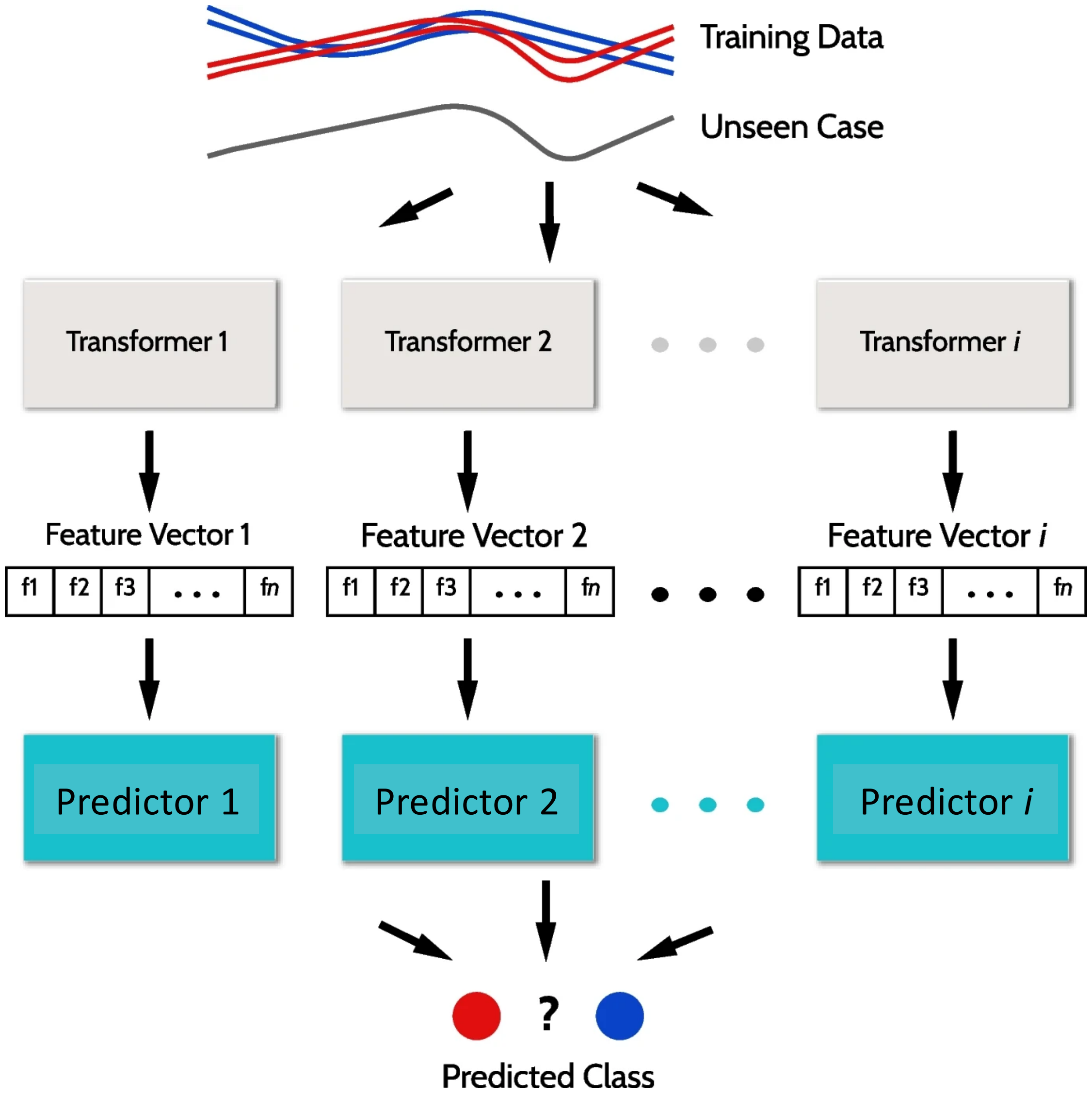


Proximity Forest Distances

1. Euclidean Distance (ED);
2. Dynamic Time Warping using the full window (DTW);
3. Dynamic Time Warping with a restricted warping window (DTW-R);
4. Weighted Dynamic Time Warping (WDTW);
5. Derivative Dynamic Time Warping using the full window (DDTW);
6. Derivative Dynamic Time Warping with a restricted warping window (DDTW-R);
7. Weighted Derivative Dynamic Time Warping (WDDTW);
8. Longest Common Subsequence (LCSS);
9. Edit Distance with Real Penalty (ERP);
10. Time Warp Edit Distance (TWE)

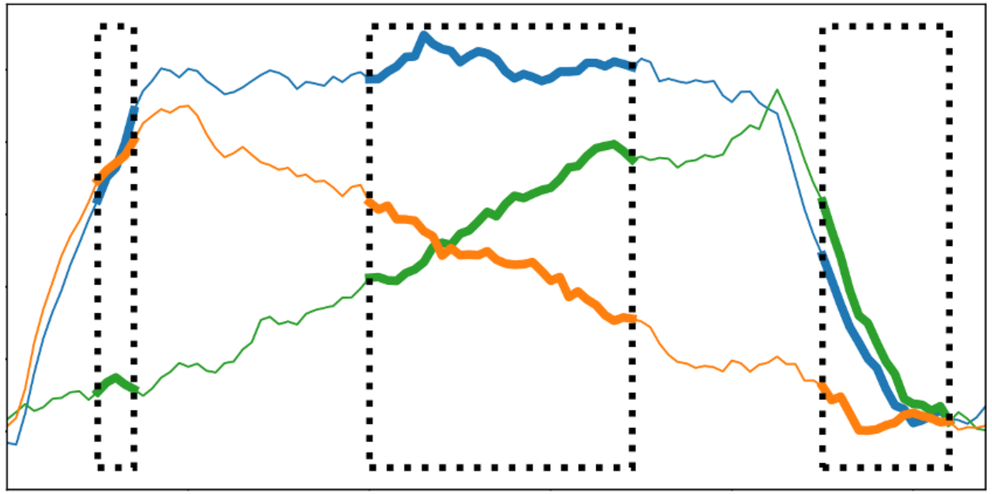
Interval-based Models

Ensemble of Interval-based Models



Interval-based Approaches

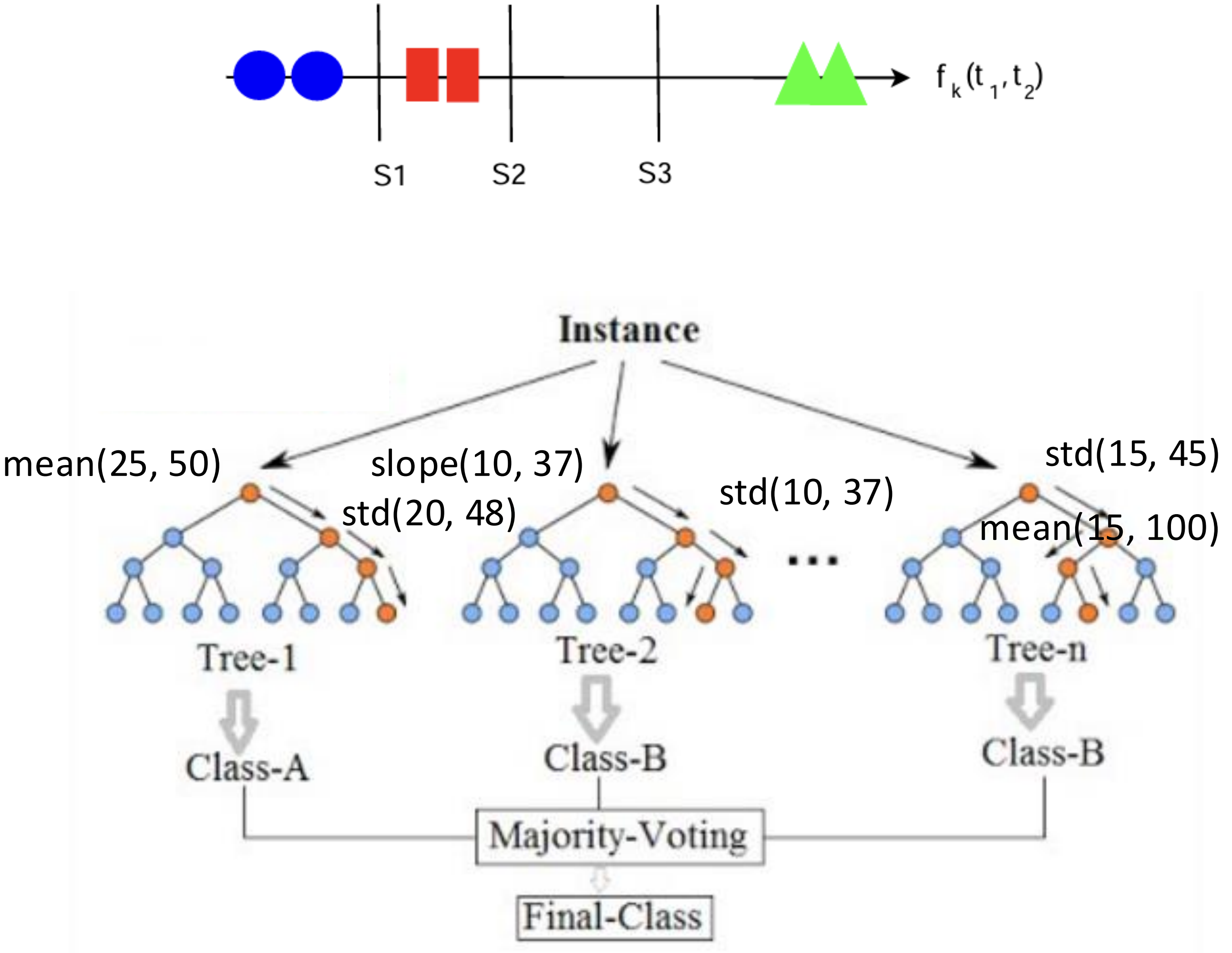
- Interval-based approaches look at phase dependent intervals of the entire TS, calculating summary statistics from selected subsequences to be used in prediction.
- Existing interval-based approaches
  - Time Series Forest (TSF)
  - Random Interval Spectral Ensemble (RISE)
  - Supervised Time Series Forest (STSF)
  - Canonical Interval Forest (CIF)
  - Diverse Representation CIF (DrCIF)



	Interval #1 mean, std, ..., cov	Interval #2 mean, std, ..., cov	Interval #3 mean, std, ..., cov
TS_1			
TS_2			
TS_3			

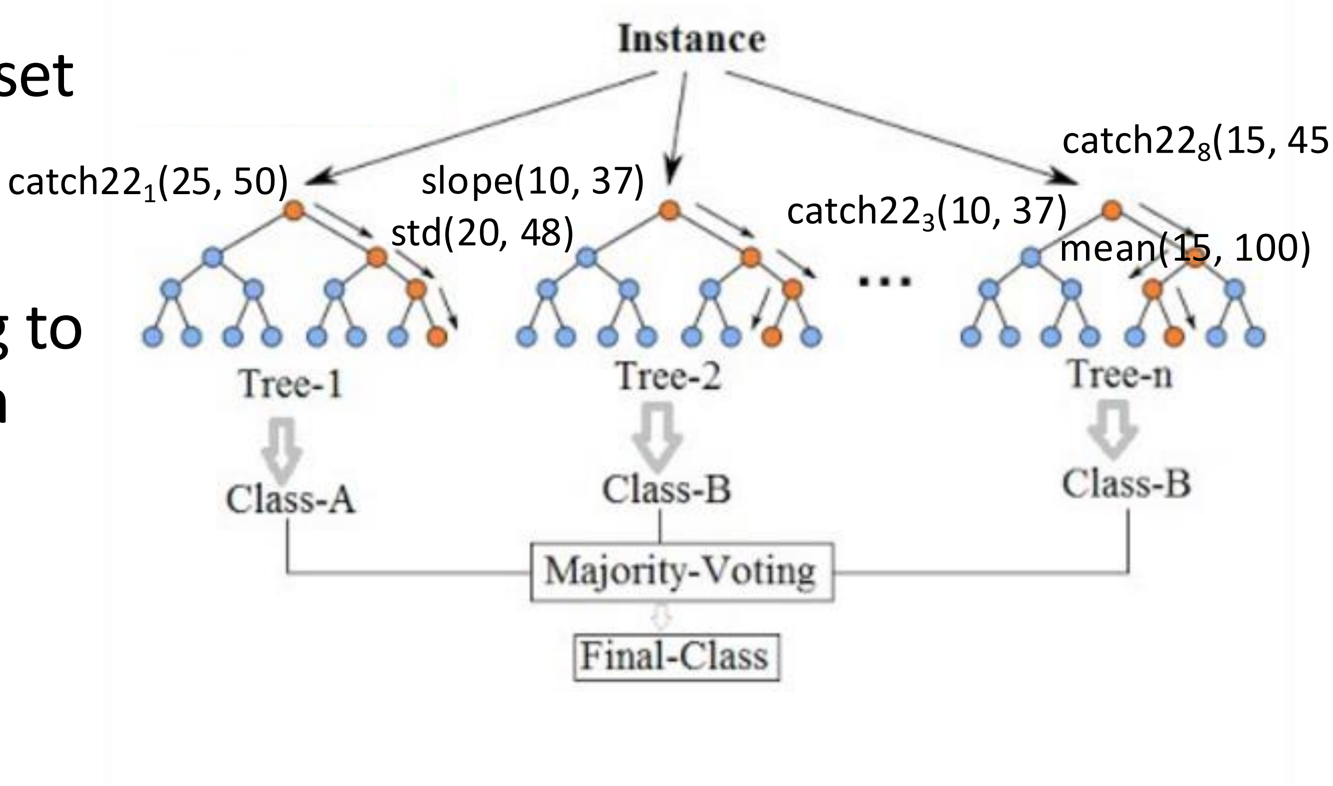
Time Series Forest (TSF)

- A TSF is an ensemble TS trees.
- TS trees select the best split by employing Entrance (Entropy and margin distance) gain to identify high-quality splits, i.e.,
- *Entrance = △Entropy + α·Margin*
- Interval features f_k: mean, standard deviation, slope.
- Randomly selected intervals for each TS tree.



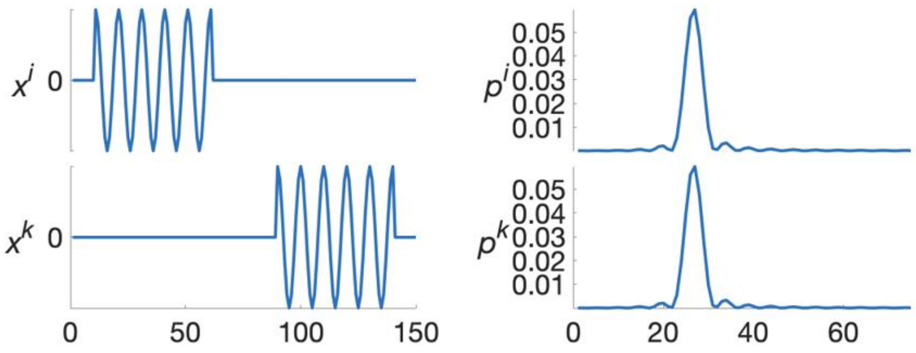
Canonical Interval Forest (CIF)

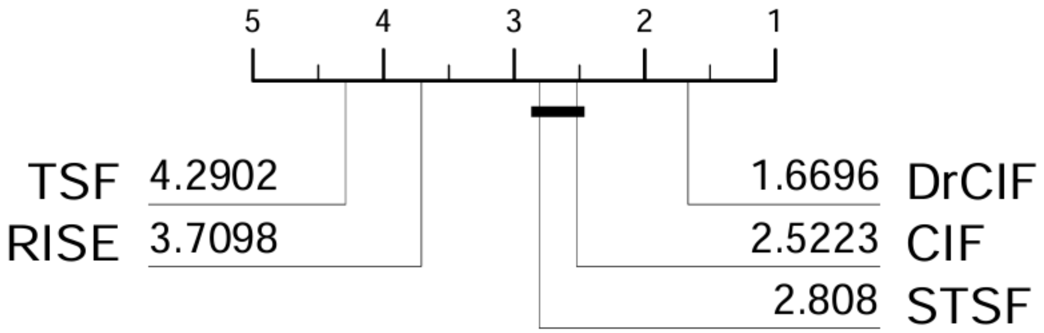
- CIF is an ensemble of TS trees.
- CIF extends TSF by augmenting the set of interval features mean, standard deviation, slope with the set of features catch22.
- To speedup the calculus the two features DN_OutlierInclude looking to outliers above and below the mean are calculated on normalized intervals, while all the others on unnormalized intervals.



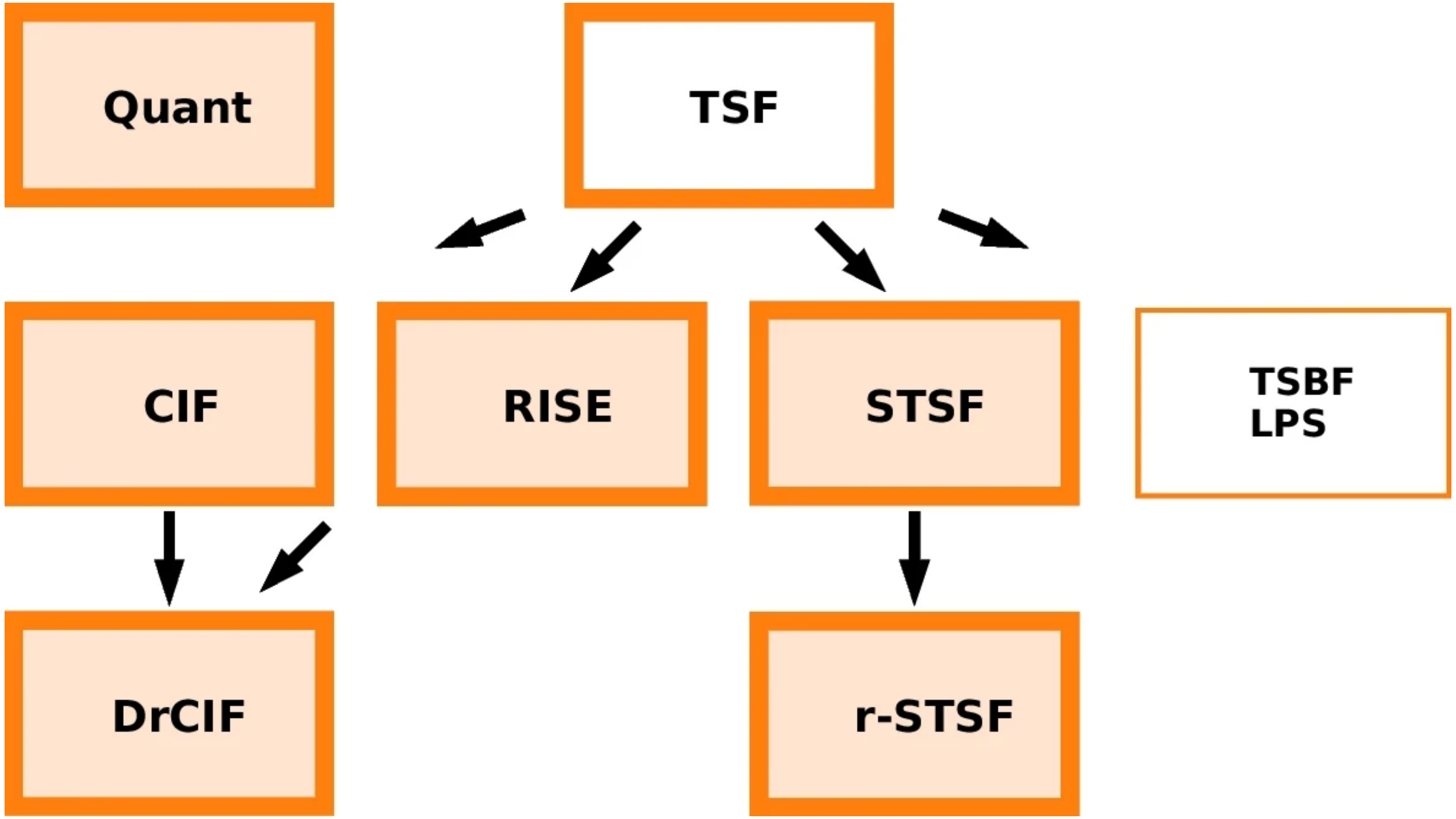
Diverse Representation CIF (DrCIF)

- DrCIF extends CIF using alternative data representations.
- DrCIF extends RISE and STSF using catch22 features.
- DrCIF selects multiple intervals taken from
  - the raw TS
  - the differencing of the TS
  - the periodograms of the TS, i.e., its DFT





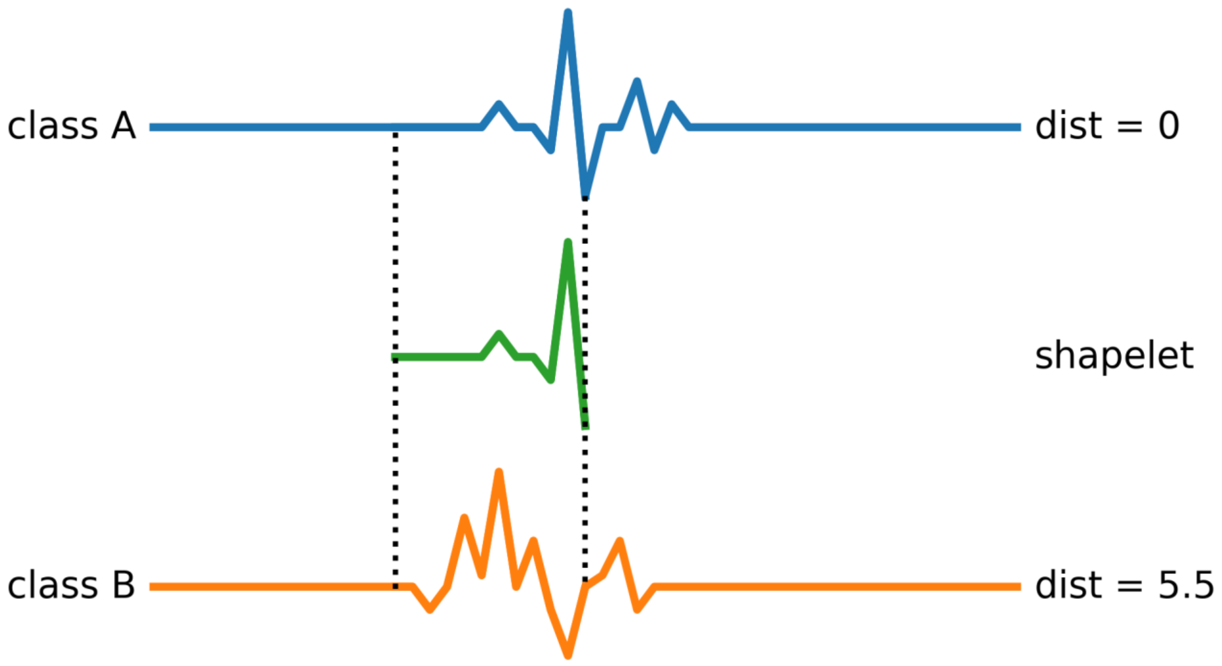
Overview of Interval-based Models and Relationships

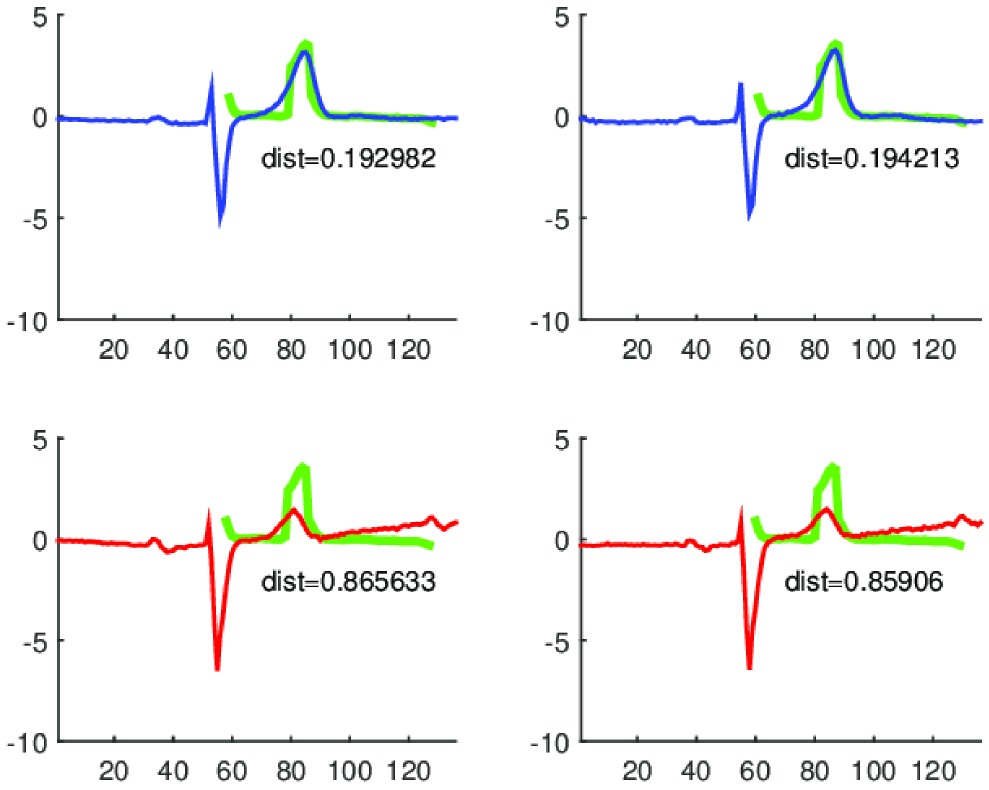


Shapelet-based Models

Shapelets

- Shapelets are TS subsequences which are maximally representative of a class or maximally discriminative between a class and another





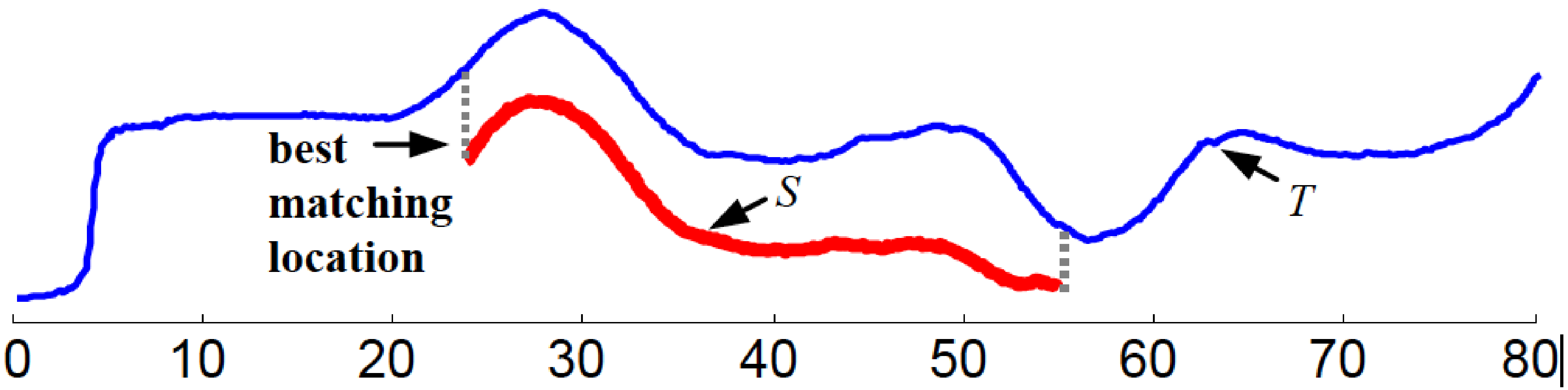
Distance with a Subsequence

- The distance between a TS and a subsequence *subsequenceDist(T, S)* is a function that takes *T* and *S* as inputs and returns a nonnegative value *d*, which is the distance from *T* to *S* as

$$subsequenceDist(T, S) = min(Dist(S, S')), \; for \; S' \in S_T^{|S|}$$

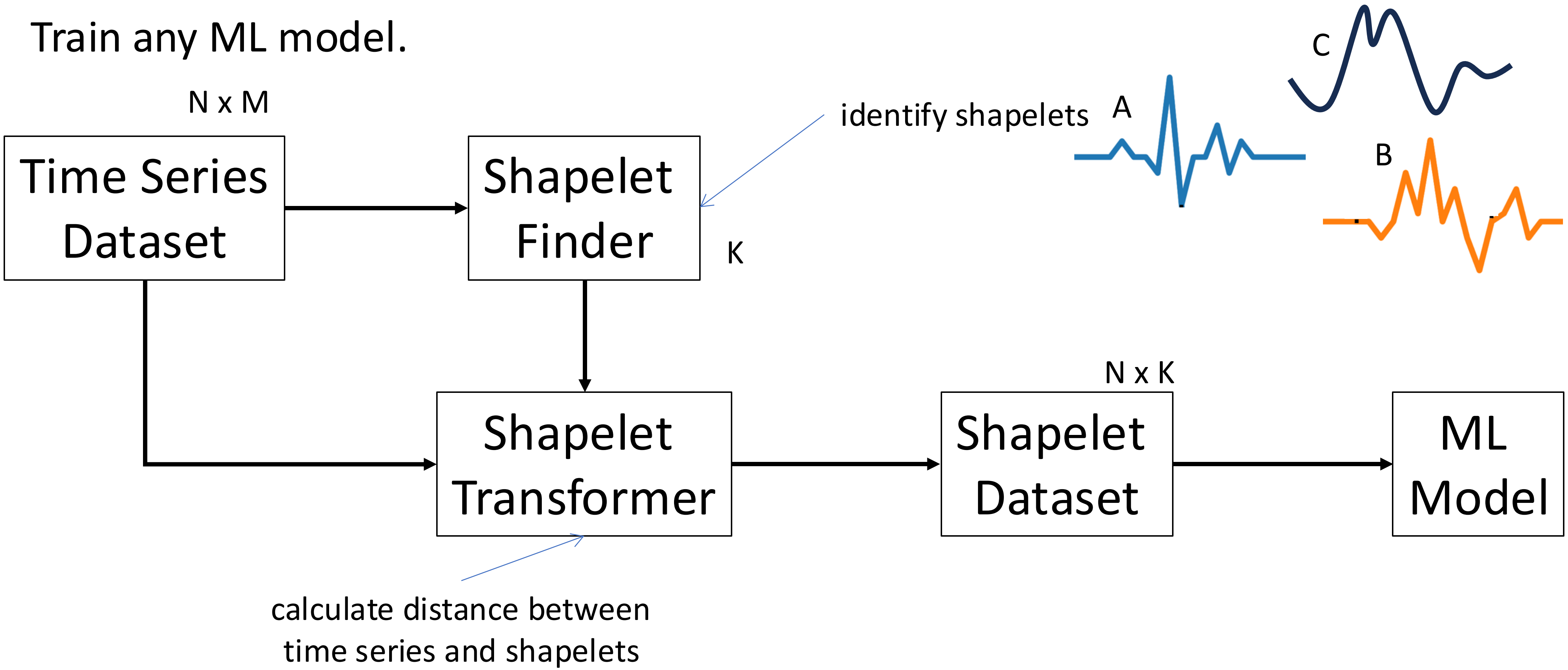
where $S_T^{|S|}$ is the set of all possible subsequences of *T*

- Intuitively, *subsequenceDist(T, S)* it is the distance between *S* and its best matching location in *T*.

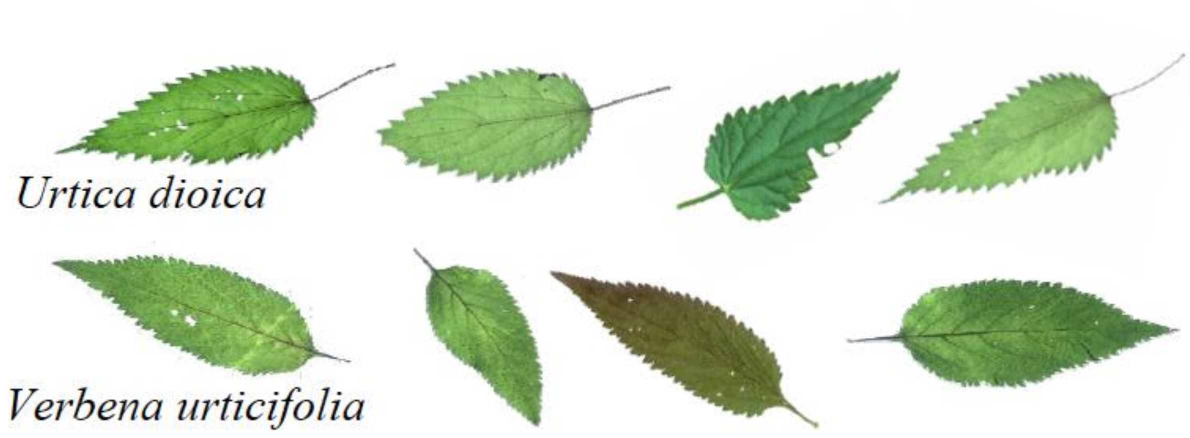


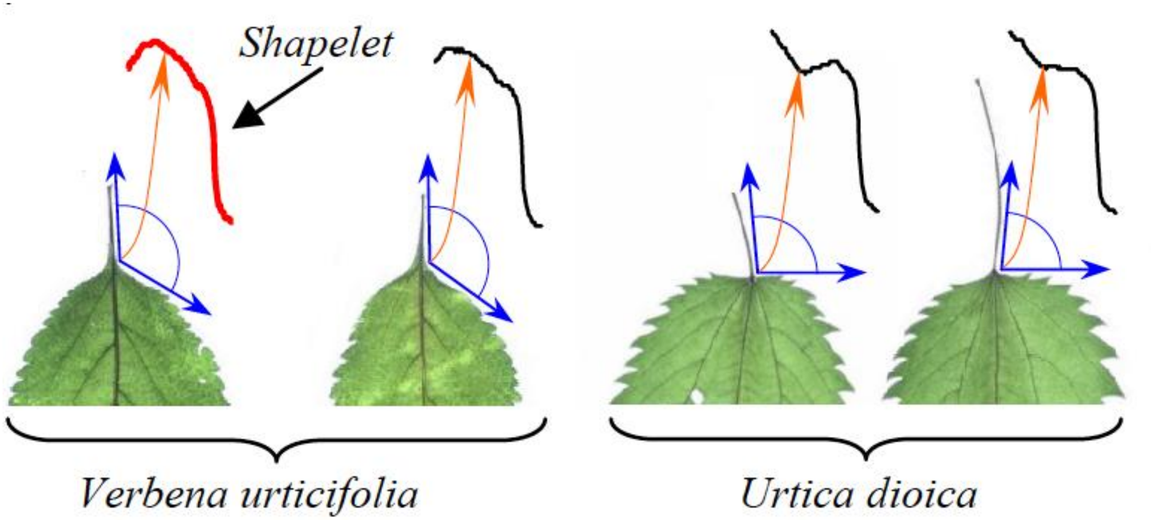
Shapelet-based Model

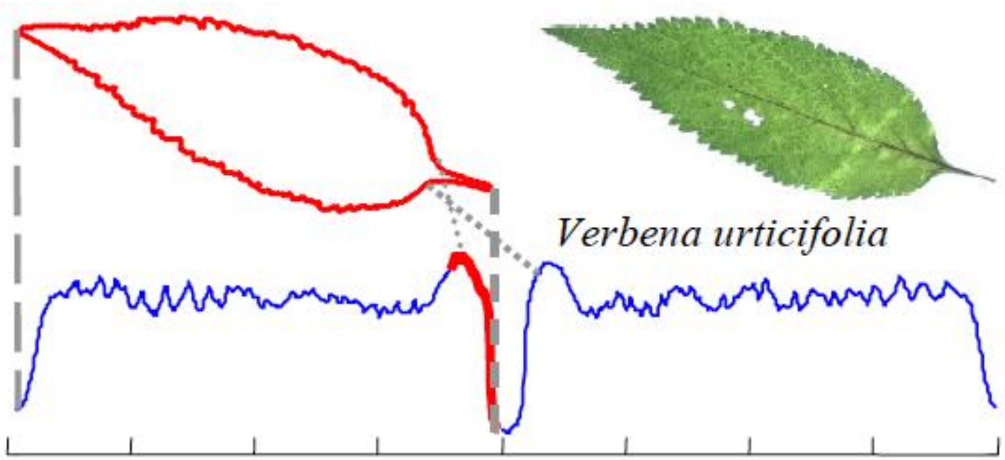
1. Given a TS dataset for classification, extract a set of K highly discriminative shapelets.
2. Transform each TS as a vector of distances with the K shapelets.
3. Train any ML model.

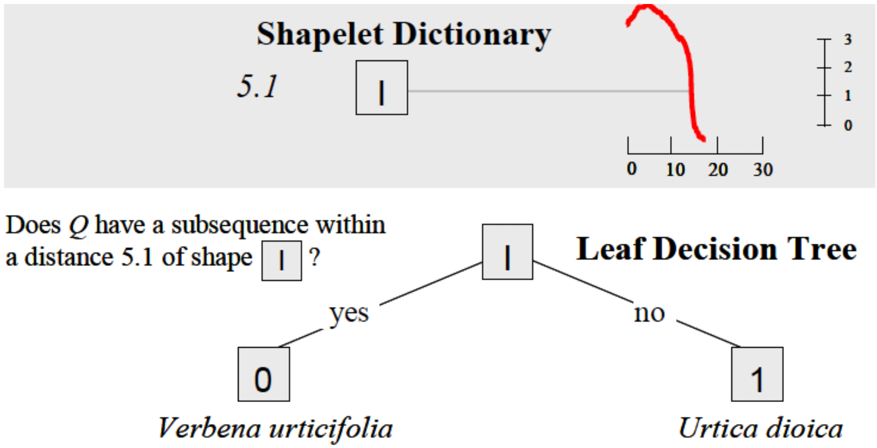


Shapelets





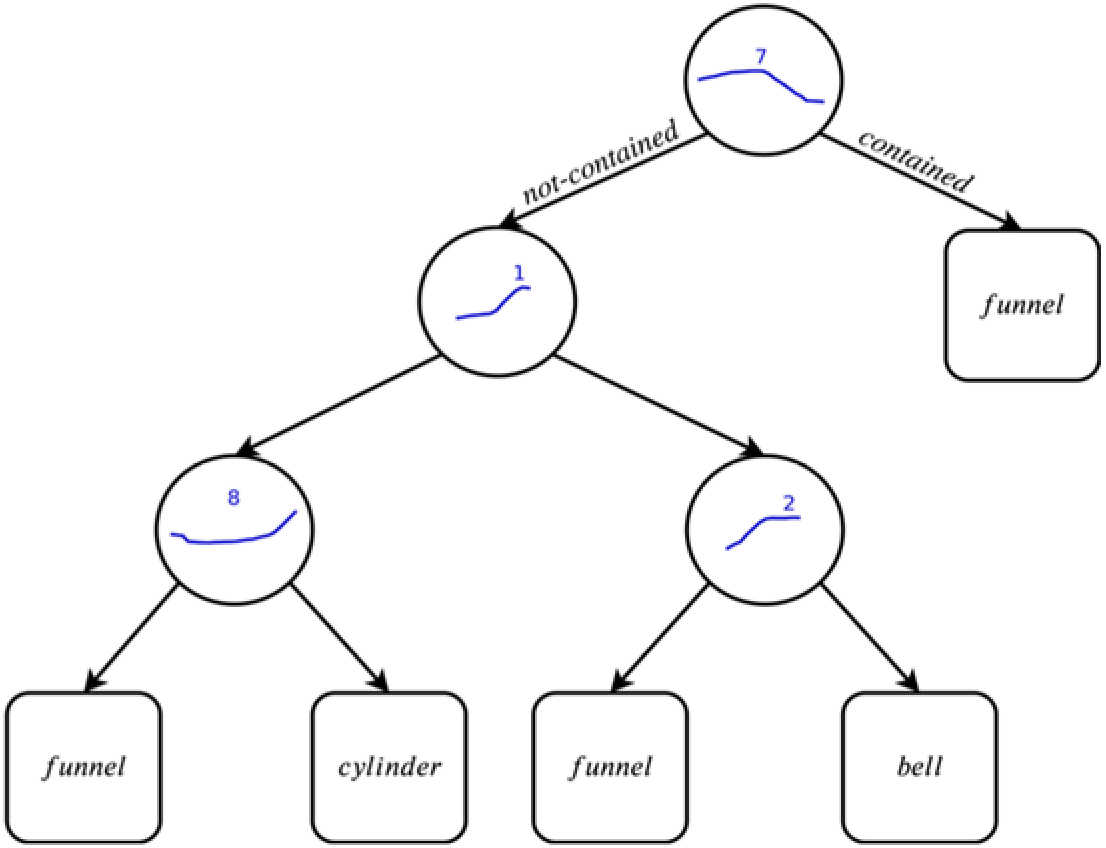


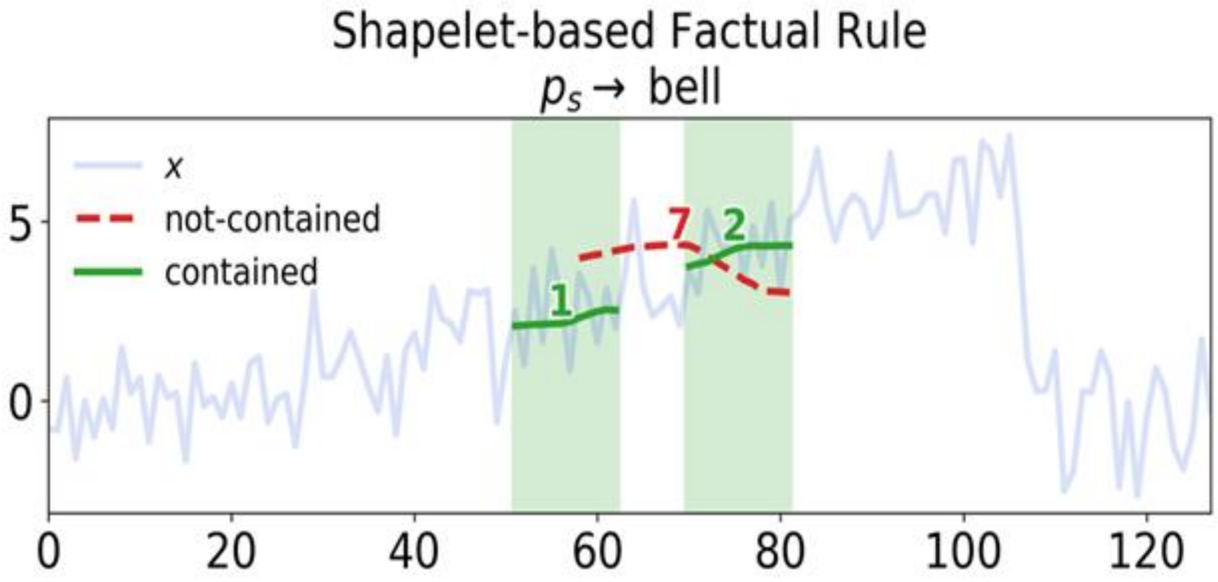


3.2	8.7
1.4	7.9
6.7	4.2
9.2	3.4

Shapelet-based Classifier

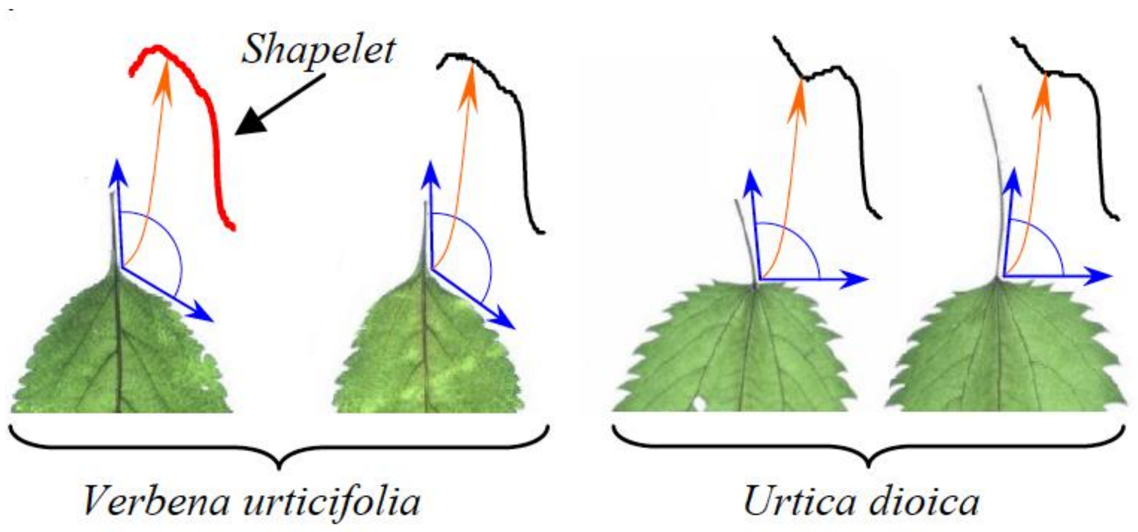
- The Shapelet-transformed TS dataset can be paired with any ML model like Decision Tree or kNN.





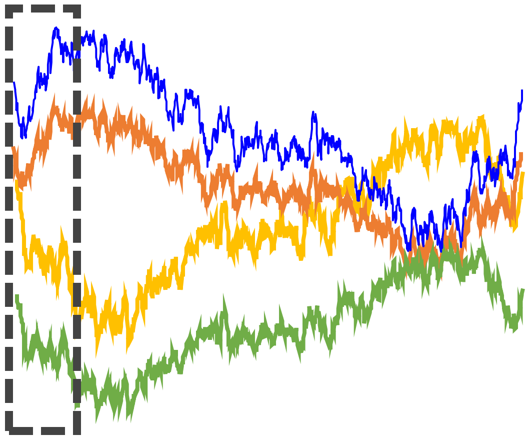
How to Extract Shapelets?

- Brute Force
- Random
- Gradient-based
- Genetic-based



Brute Force Shapelet Extraction

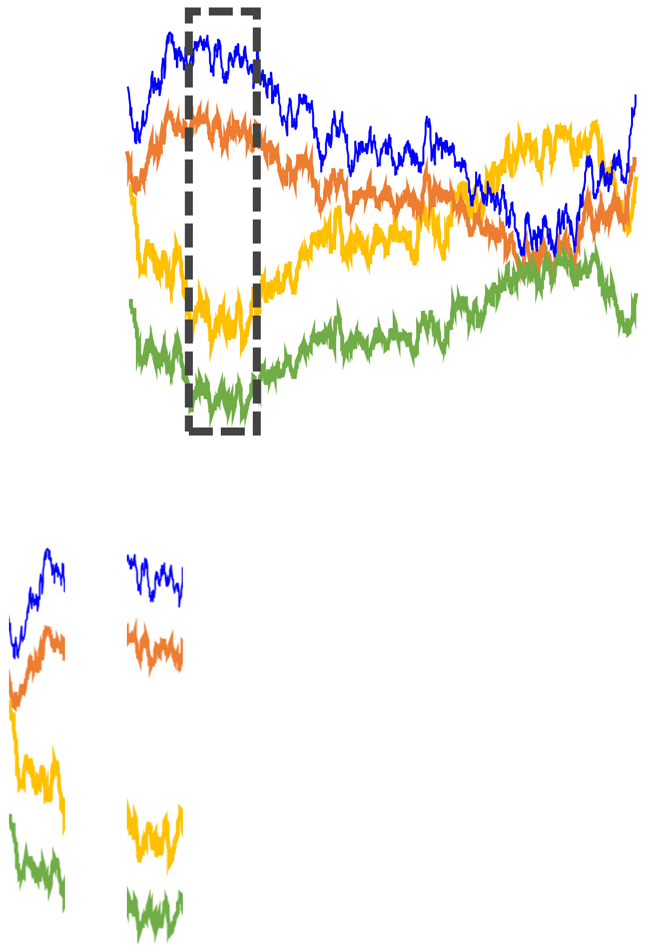
- Given a set of time windows $w_1, w_2, \ldots, w_l$ with different lengths and slices $s_1, s_2, \ldots, s_l$
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.



candidate shapelets

Brute Force Shapelet Extraction

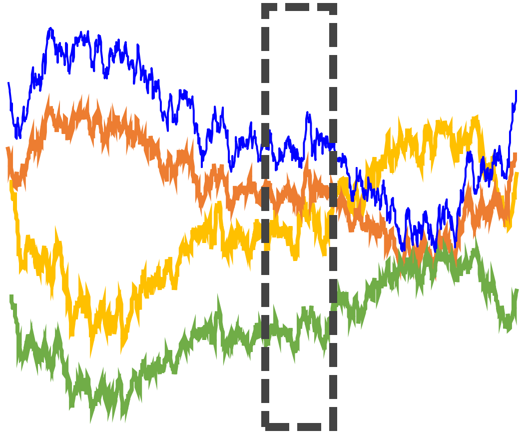
- Given a set of time windows $w_1, w_2, \ldots, w_l$ with different lengths and slices $s_1, s_2, \ldots, s_l$
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.

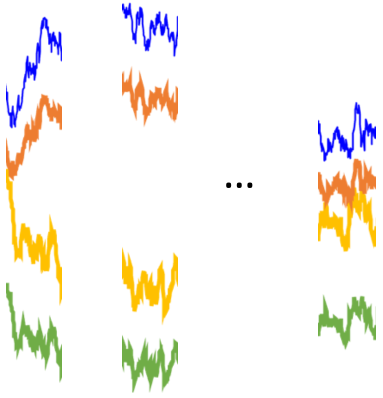


candidate shapelets

Brute Force Shapelet Extraction

- Given a set of time windows w_1, w_2, …, w_l with different lengths and slices s_1, s_2, …, s_l
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.

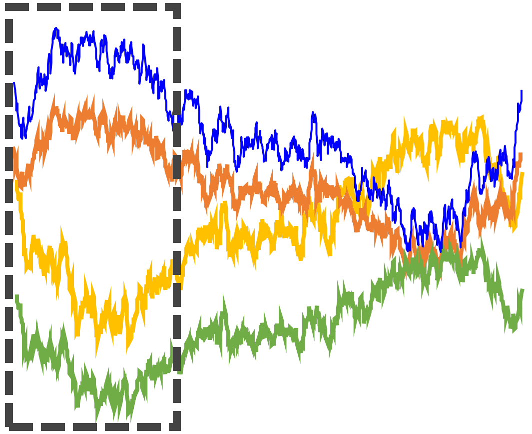


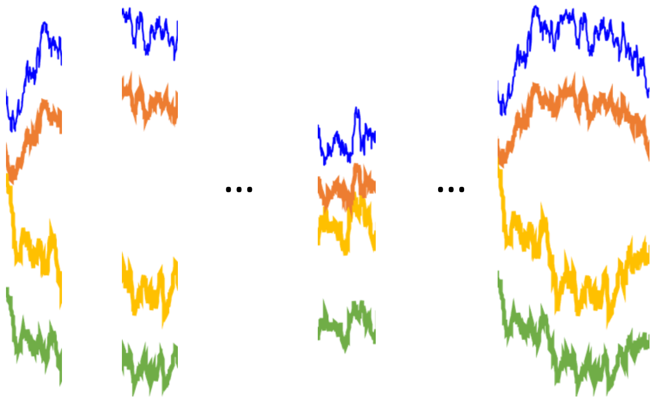


candidate shapelets

Brute Force Shapelet Extraction

- Given a set of time windows $w_1, w_2, \ldots, w_l$ with different lengths and slices $s_1, s_2, \ldots, s_l$
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.

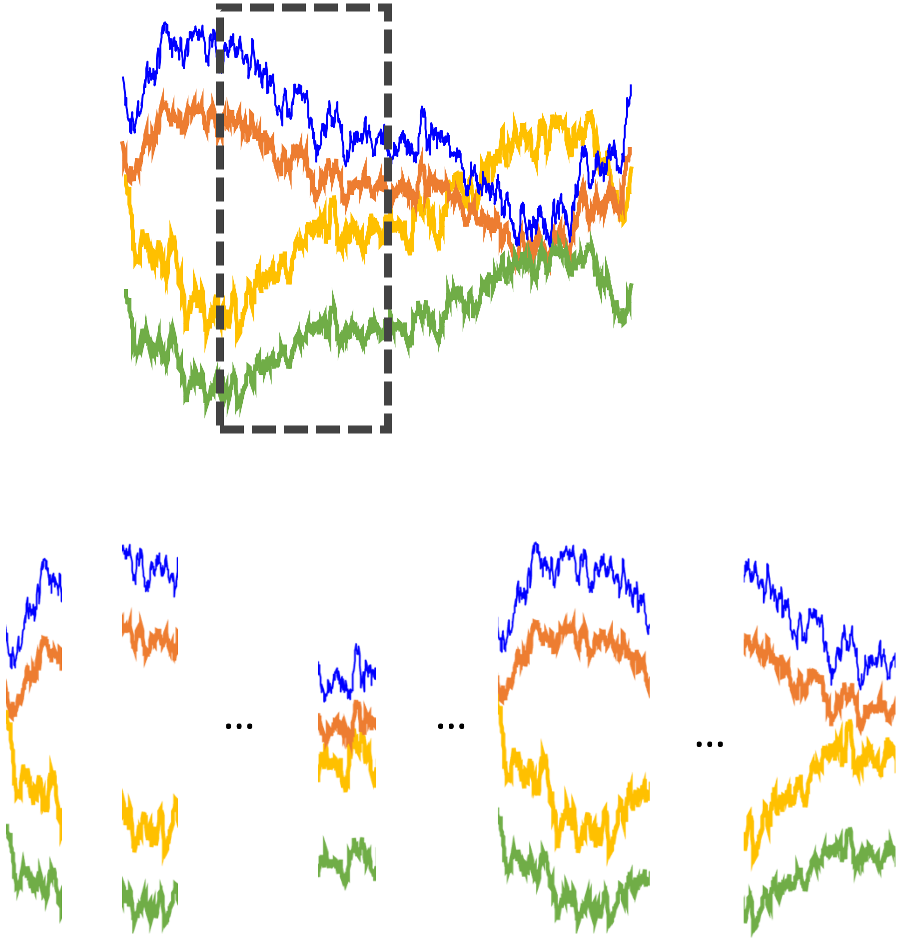




candidate shapelets

Brute Force Shapelet Extraction

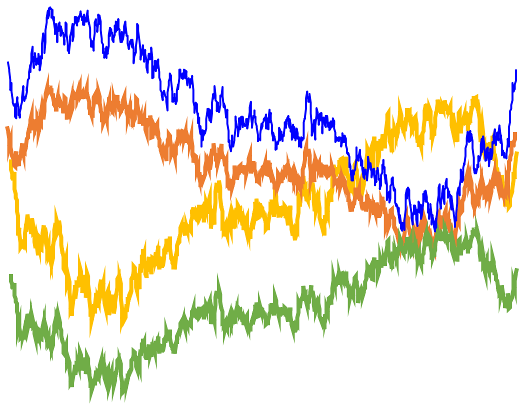
- Given a set of time windows $w_1, w_2, \ldots, w_l$ with different lengths and slices $s_1, s_2, \ldots, s_l$
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.

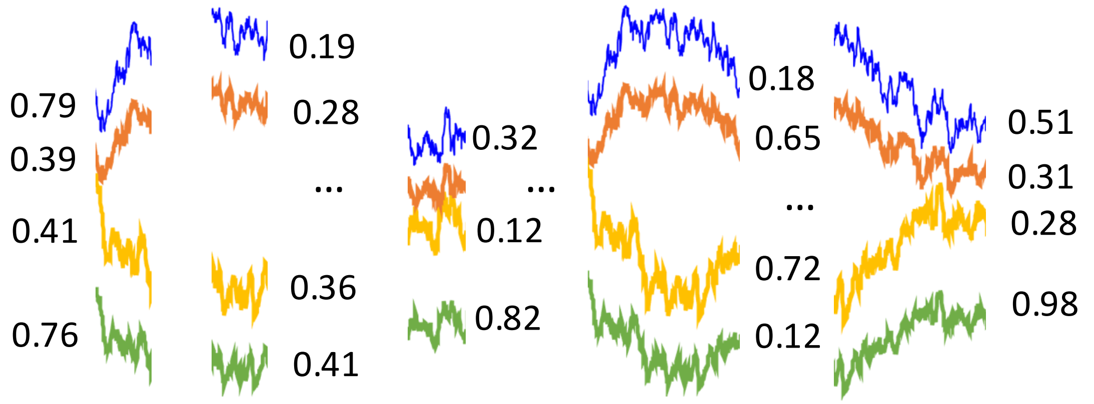


candidate shapelets

Brute Force Shapelet Extraction

- Given a set of time windows $w_1, w_2, \ldots, w_l$ with different lengths and slices $s_1, s_2, \ldots, s_l$
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.

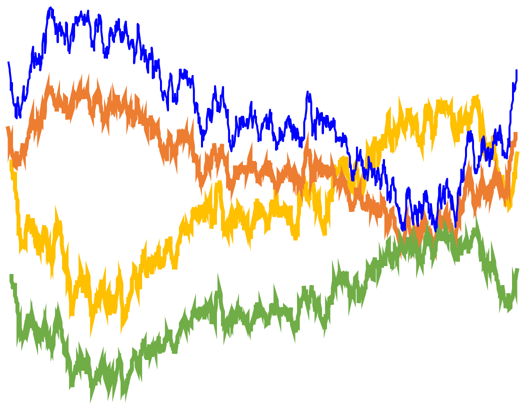


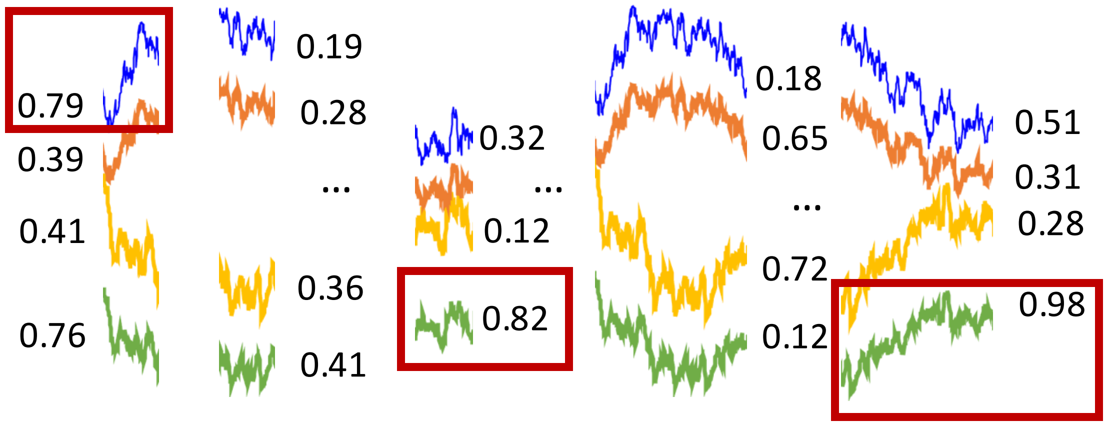


candidate shapelets

Brute Force Shapelet Extraction

- Given a set of time windows $w_1, w_2, \ldots, w_l$ with different lengths and slices $s_1, s_2, \ldots, s_l$
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.

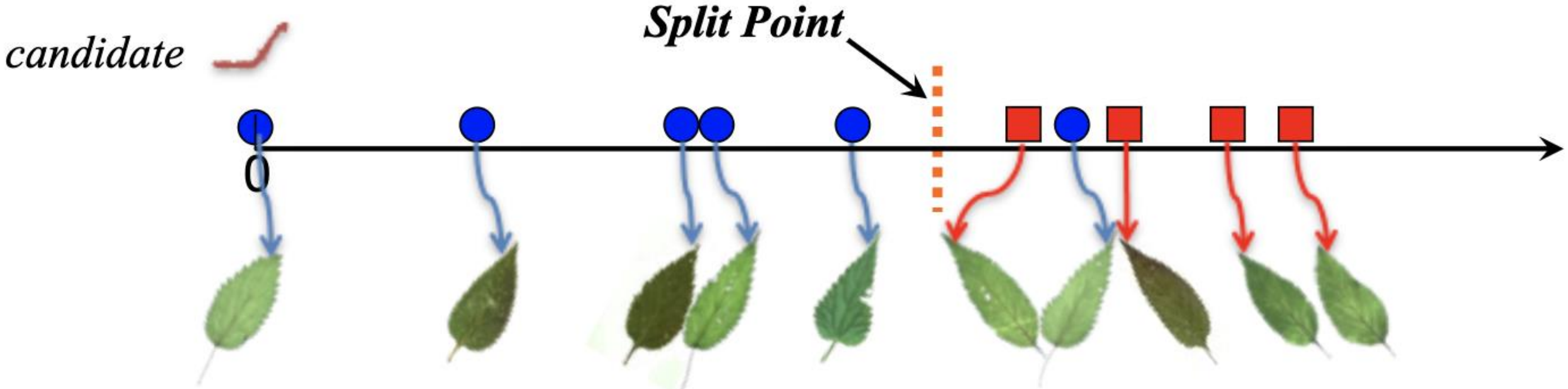




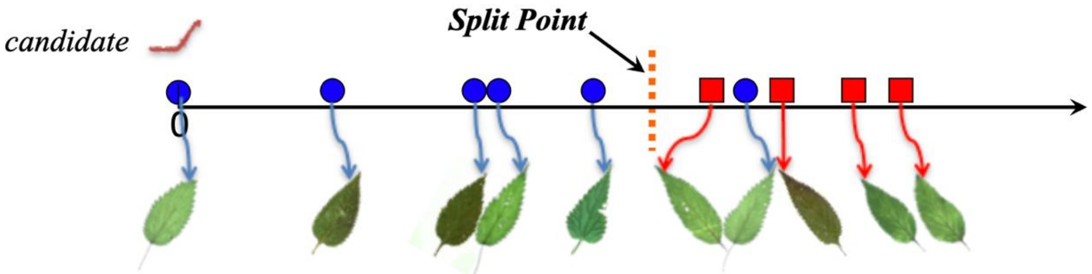
candidate shapelets

Testing The Utility of a Candidate Shapelet

- Arrange the TSs in the dataset *D* based on the distance from the candidate.
- Find the optimal split point that maximizes the information gain (same as for Decision Tree classifiers)
- Pick the candidate achieving best utility as the shapelet

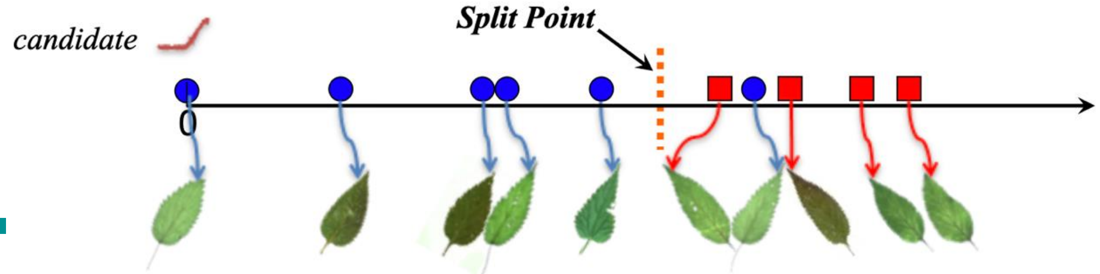


Entropy



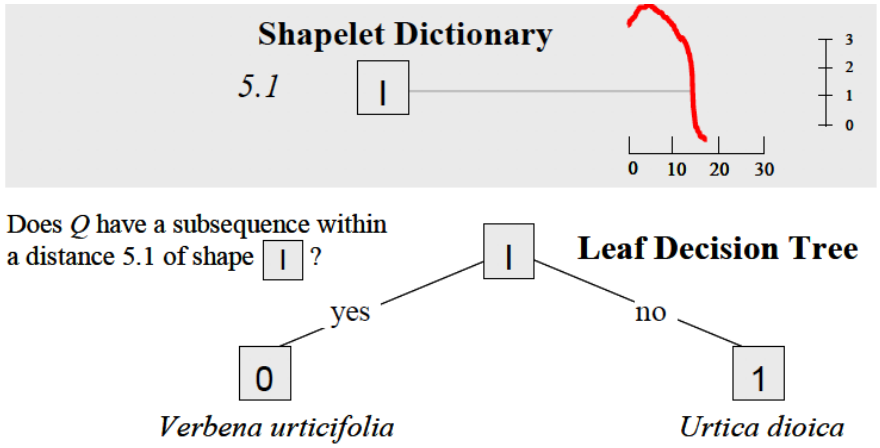
- A TS dataset D consists of two classes, A and B.
- Given that the proportion of objects in class A is $p(A)$ and the proportion of objects in class B is $p(B)$,
- The **Entropy** of D is: $I(D) = -p(A)\log(p(A)) - p(B)\log(p(B))$.
- Given a strategy that divides the D into two subsets D_1 and D_2, the information remaining in the dataset after splitting is defined by the weighted average entropy of each subset.
- If the fraction of objects in D_1 is $f(D_1)$ and in D_2 is $f(D_2)$,
- The total entropy of D after splitting is $\hat{I}(D) = f(D_1)I(D_1) + f(D_2)I(D_2)$.

Information Gain



Split point distance from shapelet = 5.1

- Given a certain split strategy *sp* which divides *D* into two subsets D_1 and D_2, the entropy before and after splitting is $I(D)$ and $\hat{I}(D)$.
- The **information gain** for this splitting rule is:
- $Gain(sp) = I(D) - \hat{I}(D) =$
- $= I(D) - f(D_1)I(D_1) + f(D_2)I(D_2)$.
- We use the distance from *T* to a shapelet *S* as the splitting rule *sp*.



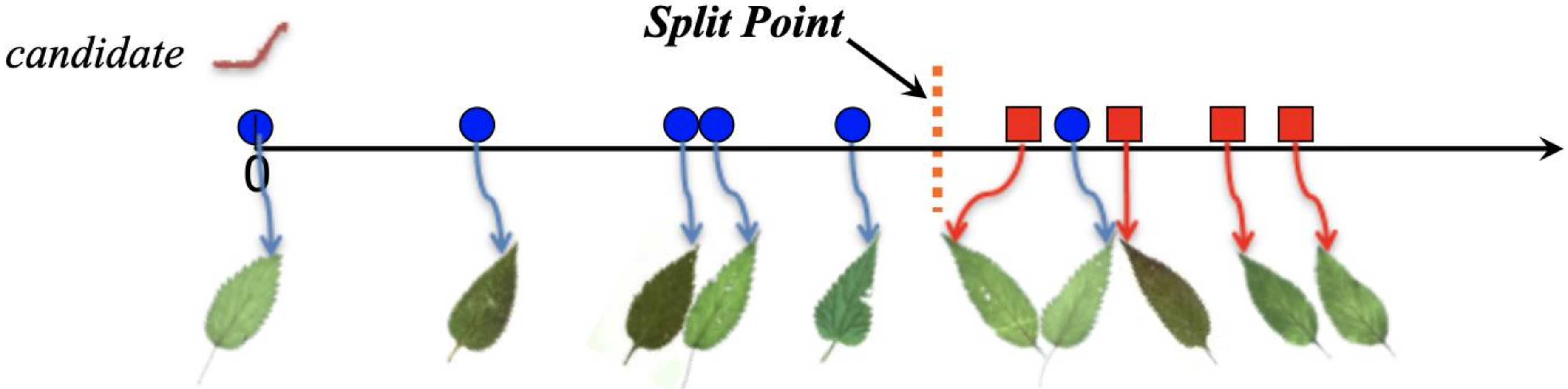
Problem with Brute Force Shapelet

- The total number of candidate is $$\sum_{l=MINLEN}^{MAXLEN} \sum_{T_i \in D} (|T_i| - l + 1)$$

- For each candidate you have to compute the distance between this candidate and each training sample
- For instance
  - 200 instances with length 275
  - 7,480,200 shapelet candidates

Speedup

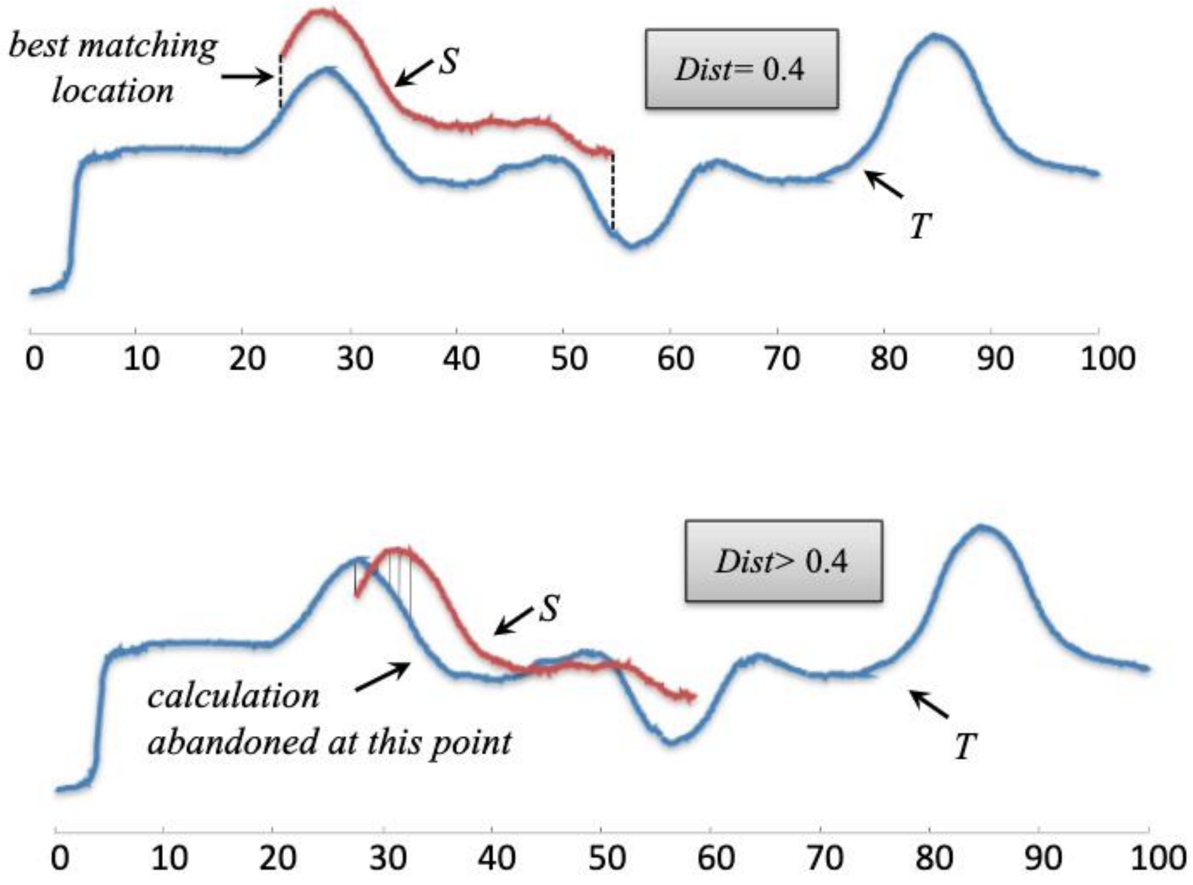
- Distance calculations form TSs to shapelet candidates is expensive.
- Reduce the time in two ways
- Distance Early Abandon
  - reduce the distance computation time between two TS
- Admissible Entropy Pruning
  - reduce the number of distance calculations



Distance Early Abandon

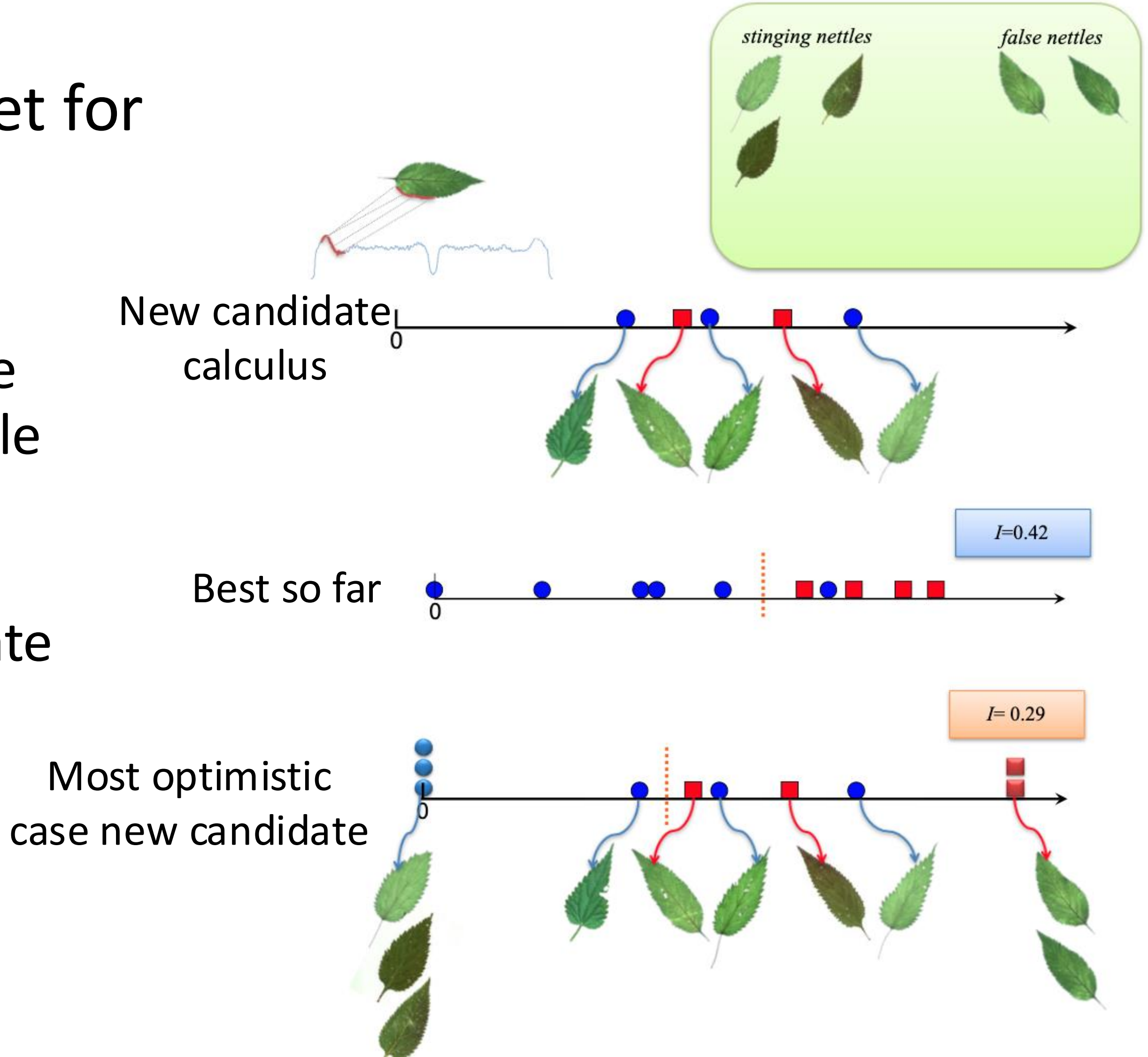
- We only need the minimum distance.

- Method
  - Keep the best-so-far distance
  - Abandon the calculation if the current distance is larger than best-so-far.



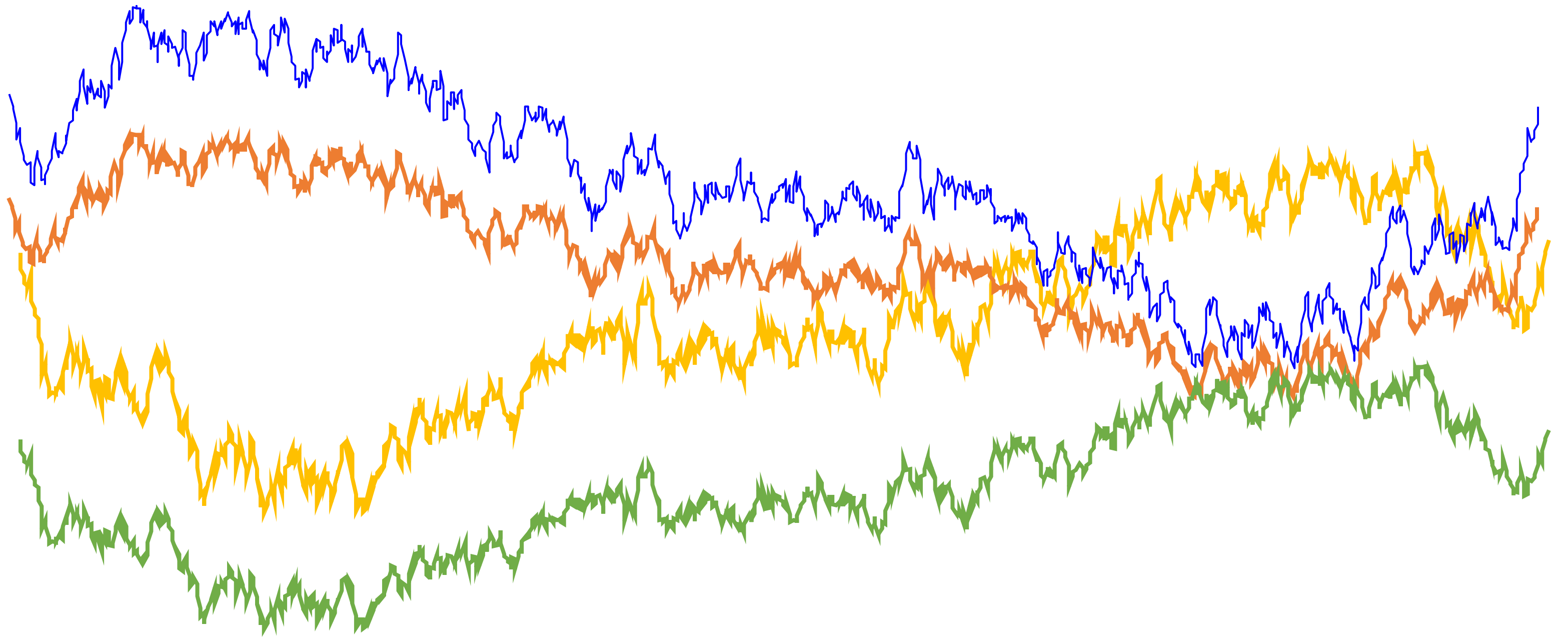
Admissible Entropy Pruning

- We only need the best shapelet for each class
- For a candidate shapelet
  - We do not need to calculate the distance for each training sample
  - After calculating some training samples, the upper bound of information gain < best candidate shapelet
  - Stop calculation
  - Try next candidate



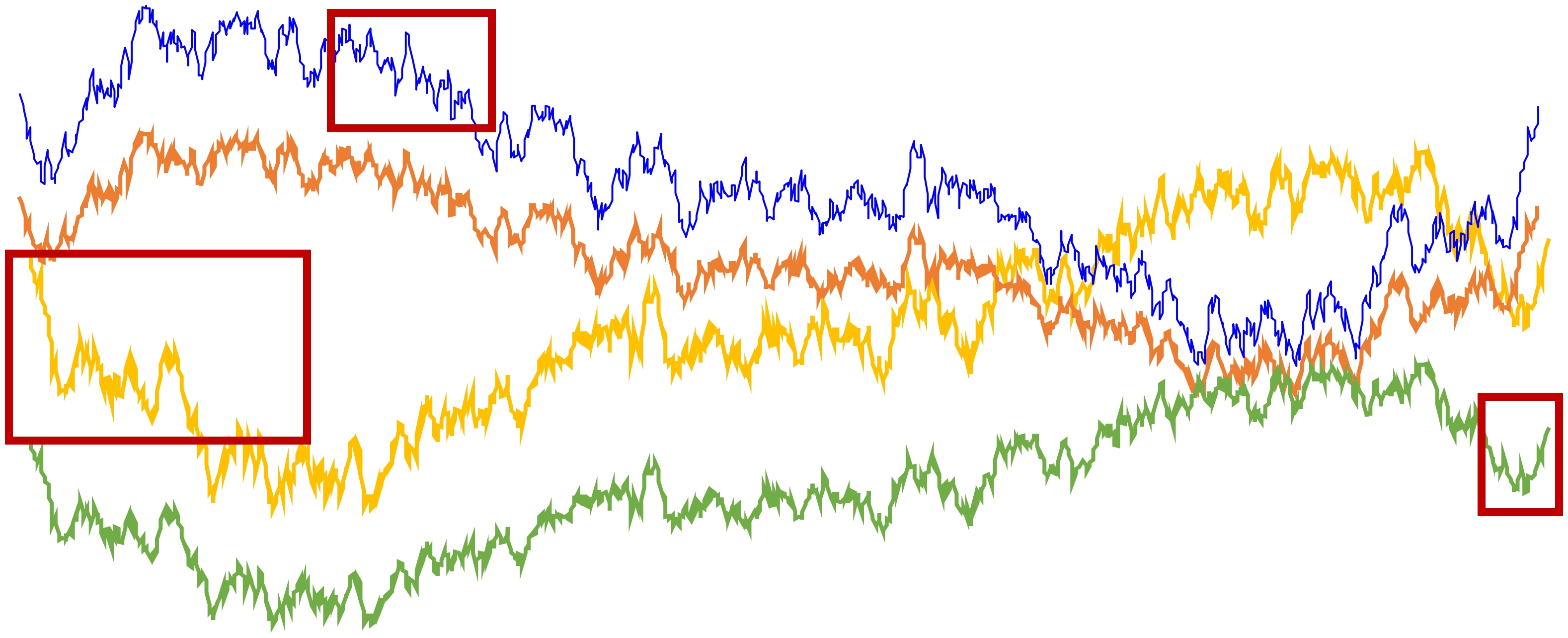
Random Shapelet Extraction

- Given a set of time windows $w_1, w_2, \ldots, w_l$ and a dataset X of time series randomly select K subsequences from the time series in X to be used as shapelets.



Random Shapelet Extraction

- Given a set of time windows $w_1, w_2, \ldots, w_l$ and a dataset X of time series randomly select K subsequences from the time series in X to be used as shapelets.



$$\mathcal{F}_i = \mathcal{L}(Y_i, \hat{Y}_i) + \frac{\lambda_W}{I} \sum_{k=1}^{K} {W_k}^2$$

$$\hat{Y}_i = W_0 + \sum_{k=1}^{K} M_{i,k} W_k, \quad \forall i \in \{1, \ldots, I\}$$

$$\mathcal{L}(Y, \hat{Y}) = -Y \ln \sigma(\hat{Y}) - (1 - Y) \ln \left(1 - \sigma(\hat{Y})\right)$$

Gradient-based Shapelet

- Learn optimal shapelets without exploring all possible candidates.
- Step 1: start with rough initial guesses for the shapelet
- Step 2: iteratively learn/optimize the shapelets by minimizing a loss function by using a predictive model that is differentiable with respect to shapelets.
- Shapelets can be updated in a stochastic gradient descent optimization fashion by taking steps towards the minimum of the classification loss function

Algorithm 1 Learning Time-Series Shapelets

Require: $T \in \mathbb{R}^{I \times Q}$, Number of Shapelets K, Length of a shapelet L, Regularization λ_W, Learning Rate η, Number of iterations: maxIter

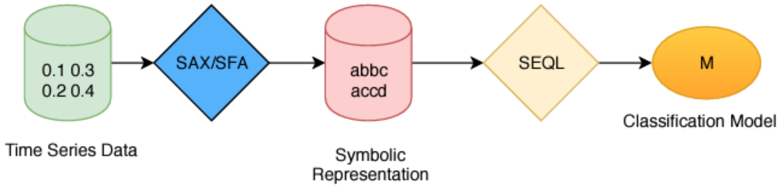
Ensure: Shapelets $S \in \mathbb{R}^{K \times L}$, Classification weights $W \in \mathbb{R}^K$, Bias $W_0 \in \mathbb{R}$

for iteration=$\mathbb{N}_1^{\text{maxIter}}$ **do**
  for $i = 1, \ldots, I$ **do**
    for $k = 1, \ldots, K$ **do**
      $W_k \leftarrow W_k - \eta \frac{\partial \mathcal{F}_i}{\partial W_k}$
      for $L = 1, \ldots, L$ **do**
        $S_{k,l} \leftarrow S_{k,l} - \eta \frac{\partial \mathcal{F}_i}{\partial S_{k,l}}$
      end for
    end for
    $W_0 \leftarrow W_0 - \eta \frac{\partial \mathcal{F}_i}{\partial W_0}$
  end for
end for
return S, W, W_0

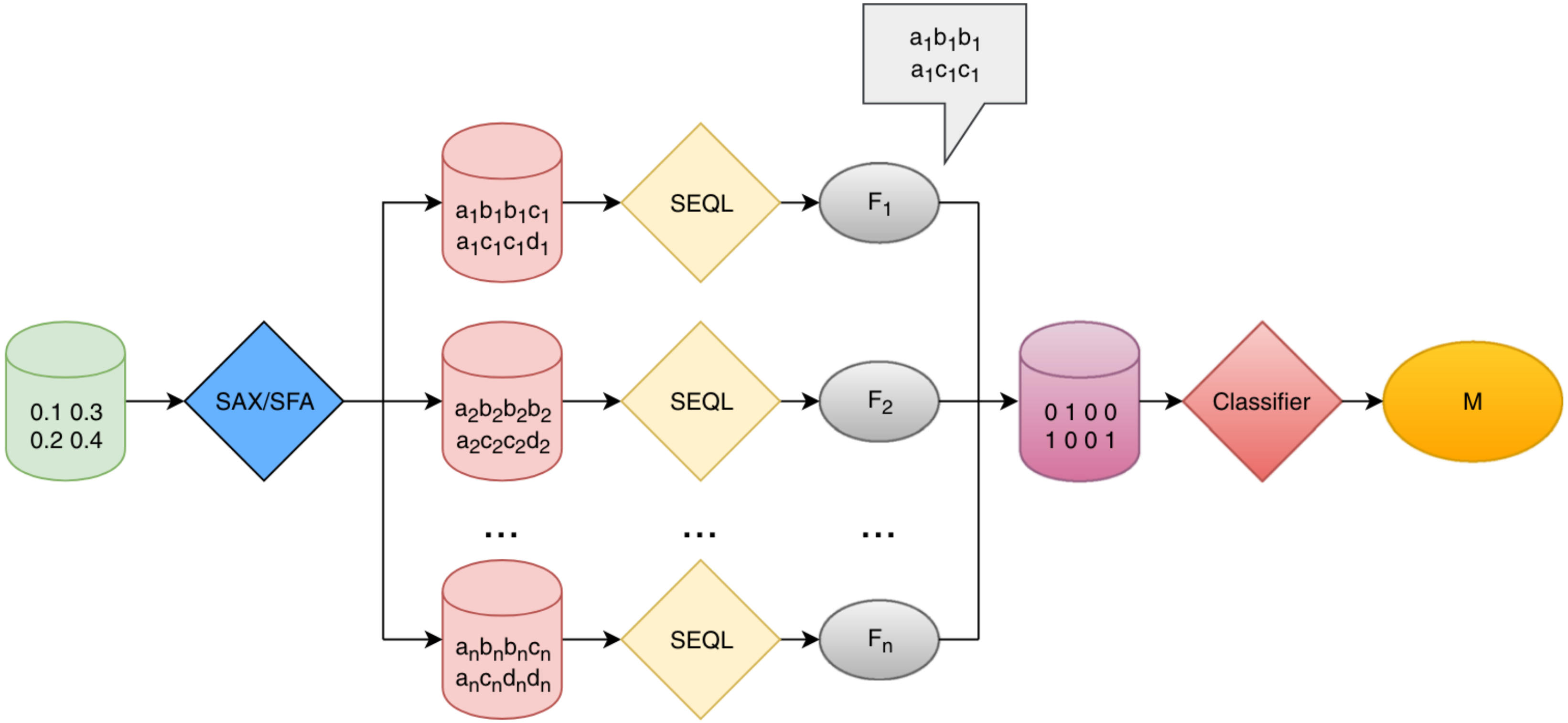
Shapelets Remarks

- A shapelet is a pattern/subsequence which is maximally representative of a class/maximally discriminative between a class and the rest with respect to a given dataset of TS.
- Shapelets can be significantly more accurate/robust then global structural features because as they are *local features* based on distances

MrSEQL – Multi Resolution SQL

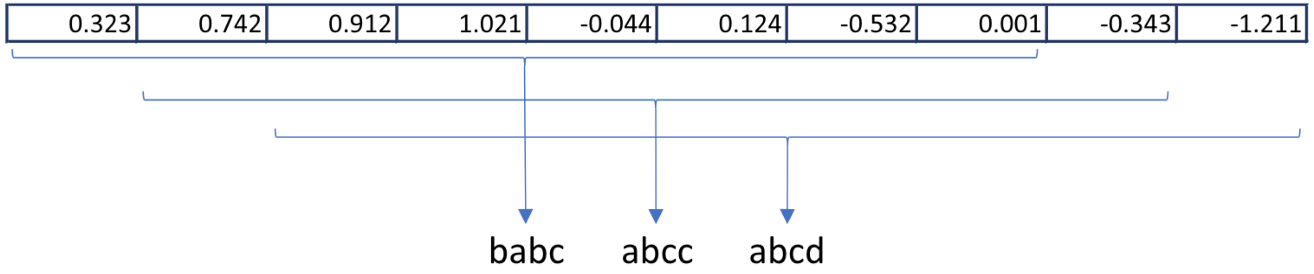


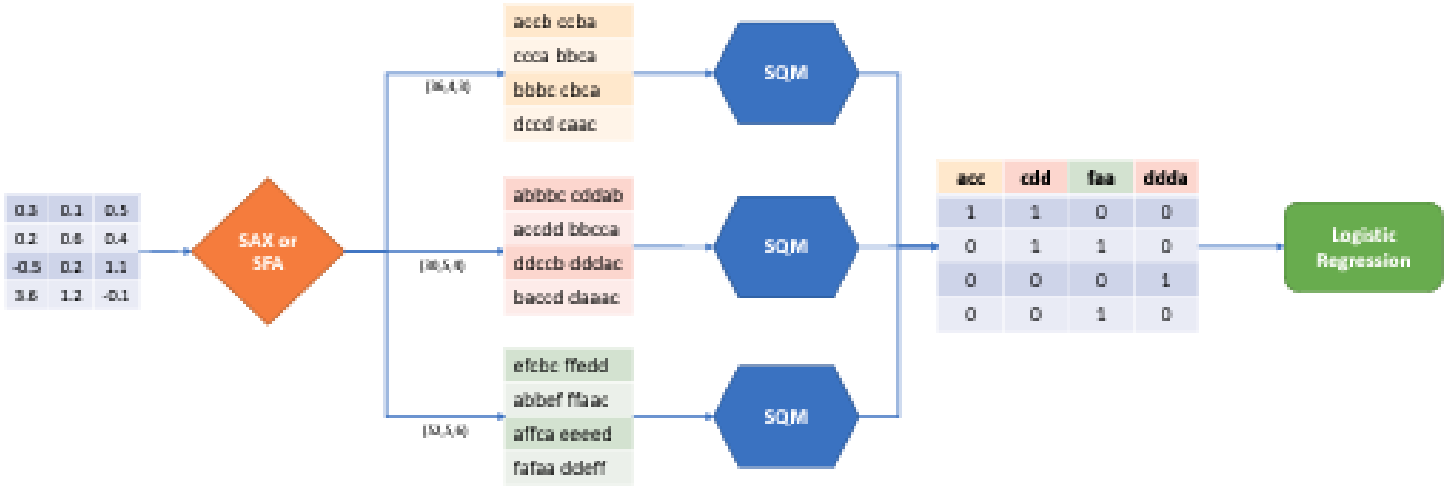
- MrSEQL is an ensemble of SEQL algorithms.
- SEQL (SEQuence Learner) selects a set of discriminative subsequences.
- MrSEQL produce *k* SEQL models from different SAX or SFA representations.



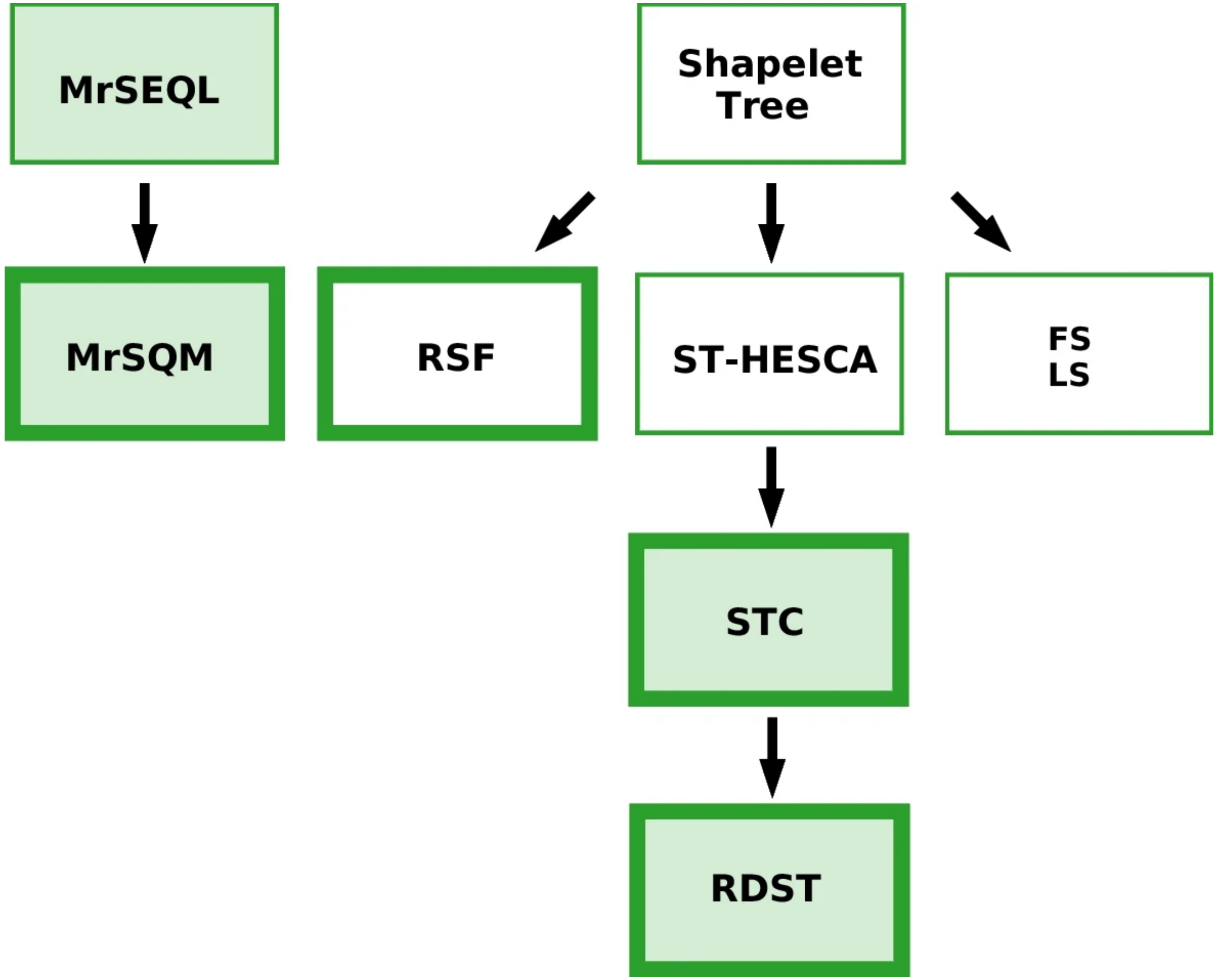
MrSQM - Multiple Representations Sequence Miner

- Extends MrSEQL with a sampling strategy for reducing the number of features generated in the BOP.





Overview of Shapelet-based Models and Relationships



Dictionary-based Models

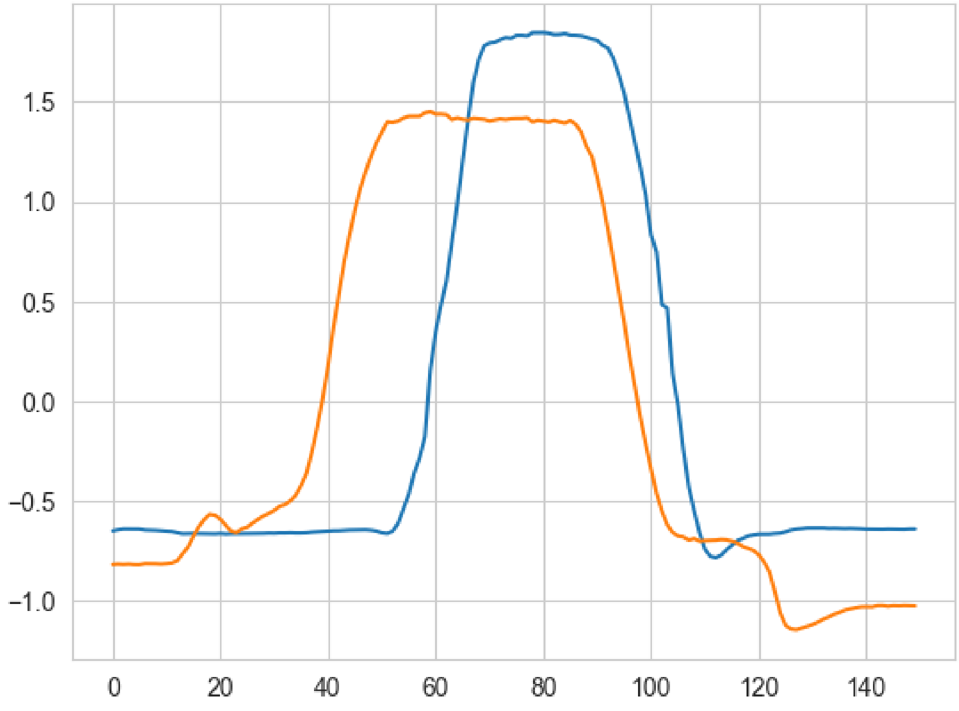
Bag of Words

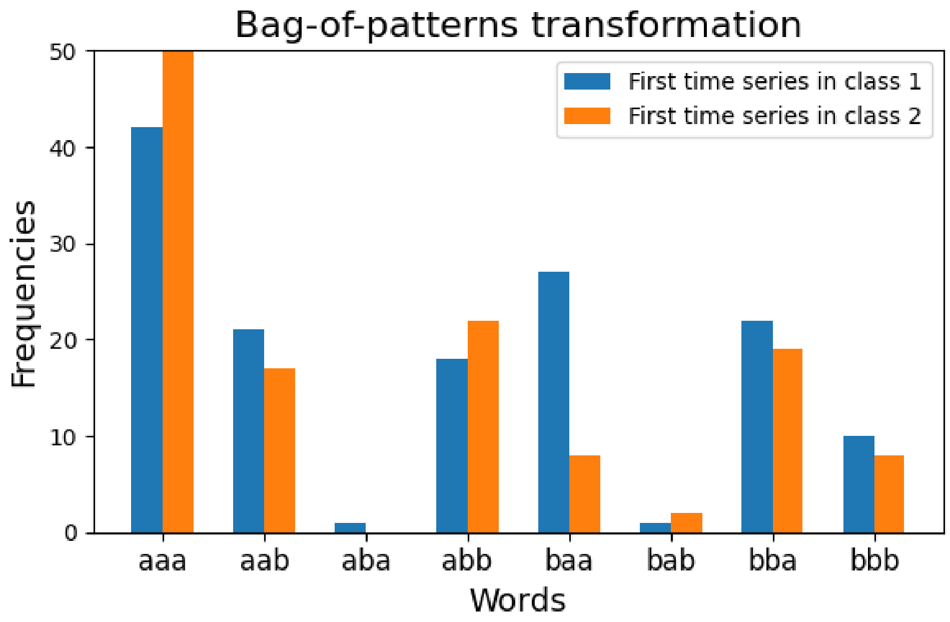
- Typically used in Natural Language Processing (NLP) and Information Retrieval (IR) but common also in TSA.
- A bag-of-words transformation turns a text into an unordered collection (or “bag”) of words typically accounting for multiplicity.

- Example: *John likes to watch movies. Mary likes movies too.*
- BoW: "John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1

Bag of Words in TSA

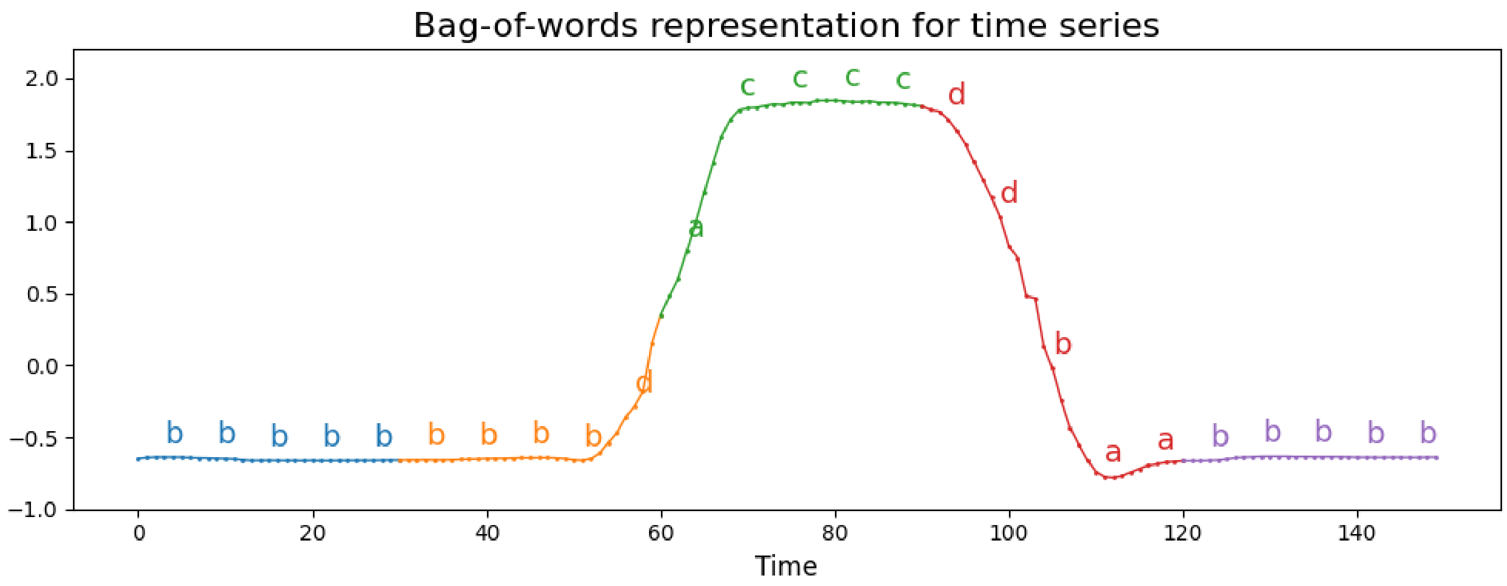
- Given a TS *T*, Bag of Words extracts subseries using a sliding window *w*, then normalize each subseries and transforms it into a **word** using an approximation approach such as:
- SAX - Symbolic Aggregate approXimation
- SFA - Symbolic Fourier Approximation





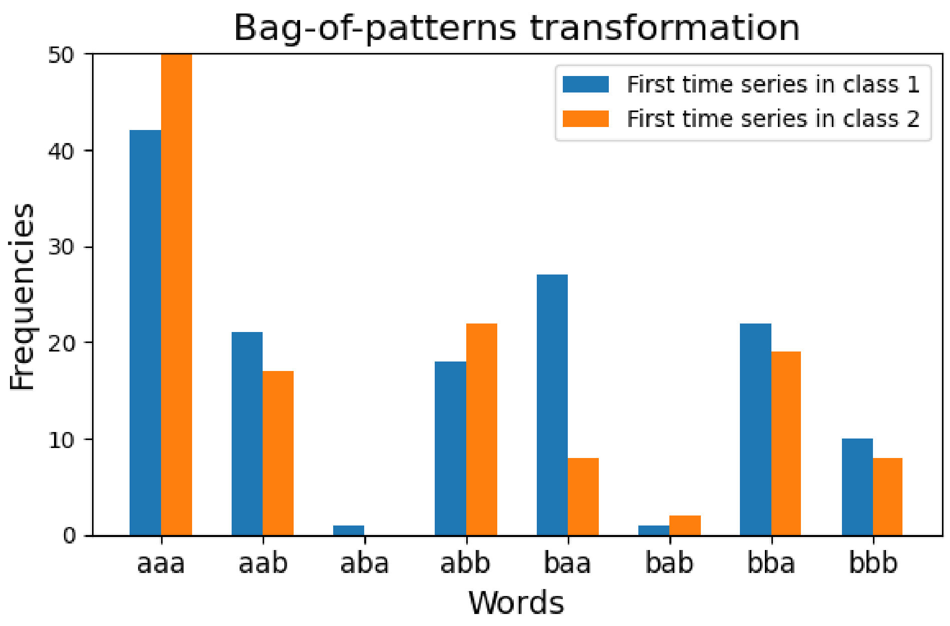
Bag of Words in TSA Hyperparameters

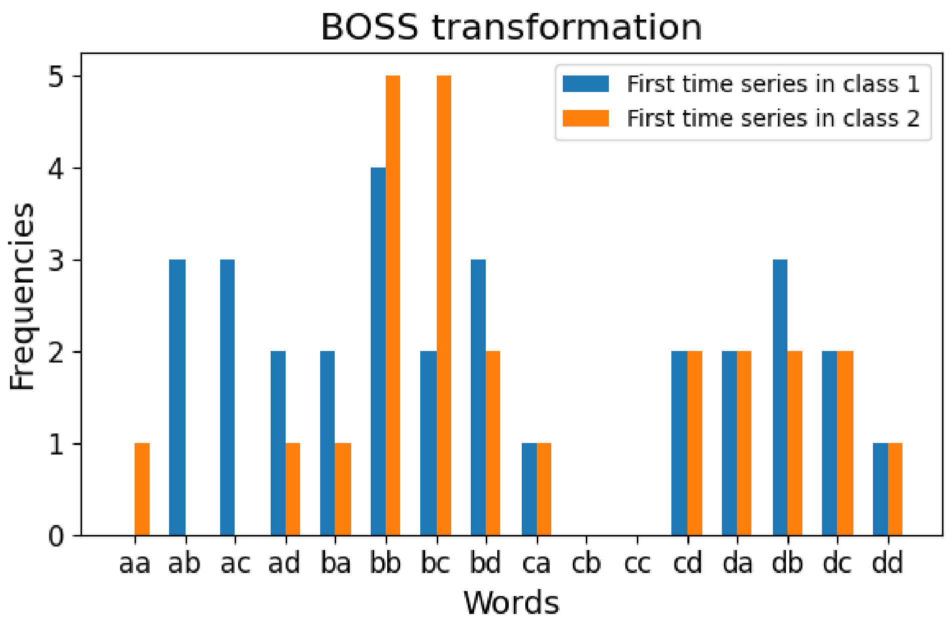
- window_size: length of the sliding window w
- window_step: step of the sliding window w (default 1)
- word_size: length of the words, i.e., number of characters
- n_bins: size of the alphabet, i.e., number of distinct characters



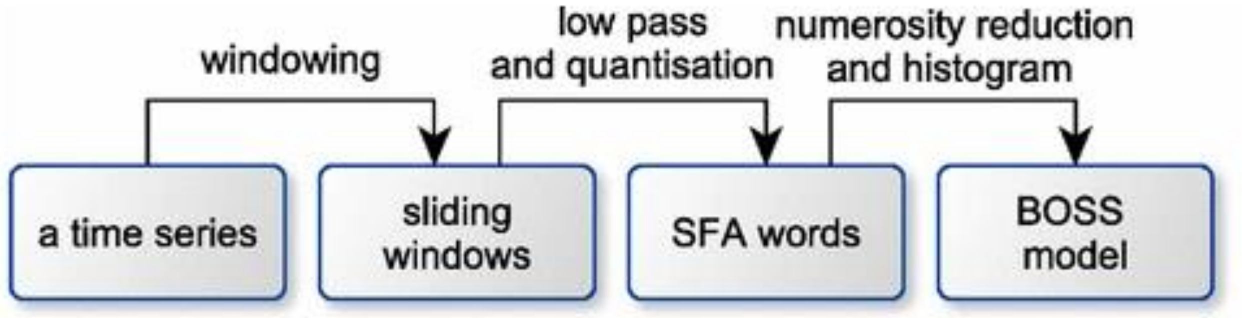
Bag of Words Algorithms

- Bag of Patterns (BOP) transforms each subsequence into a word using SAX*
- Bag-of-SFA Symbols (BOSS) transforms each subsequence into a word using SFA
- * Sometimes also BOSS transformations are named BOP.

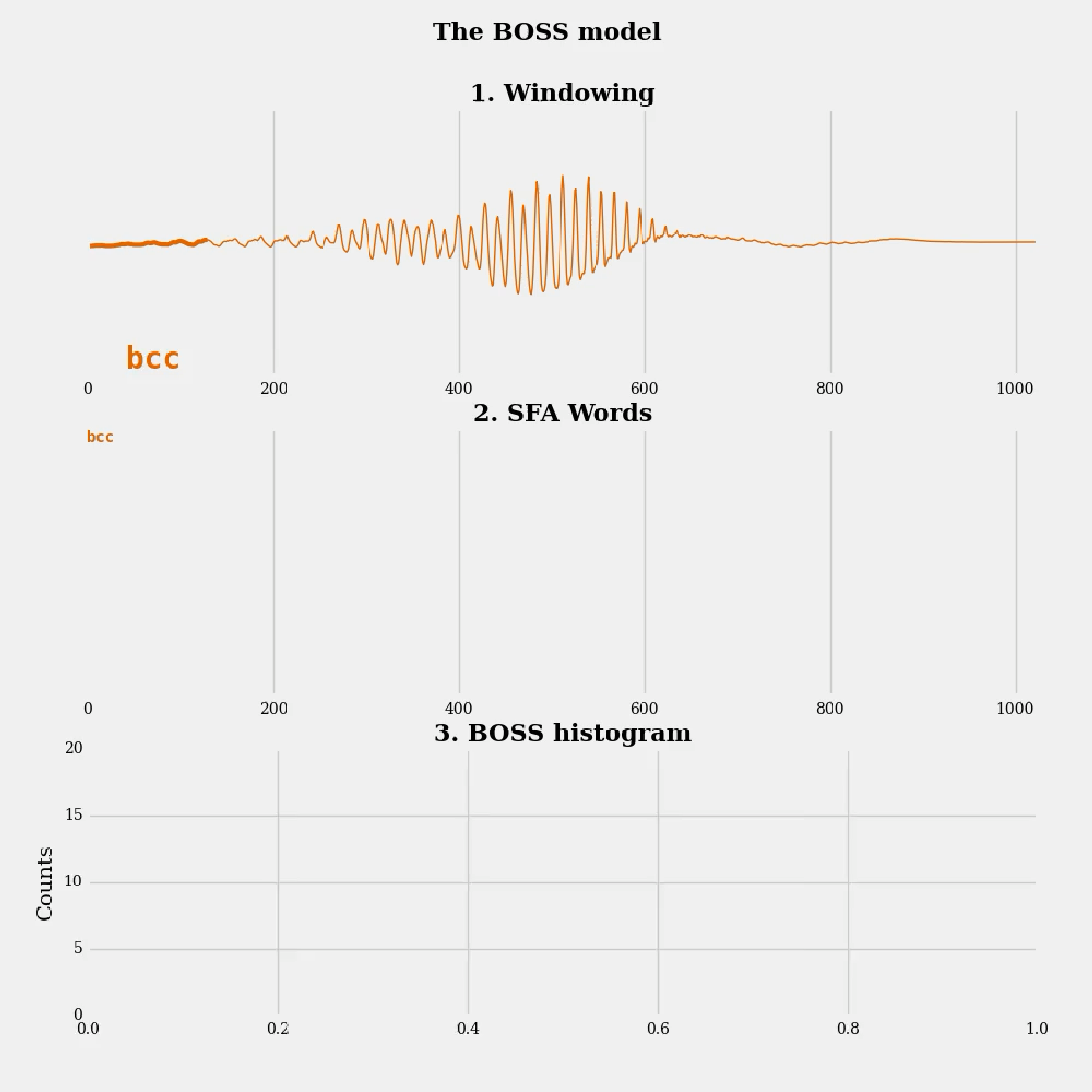


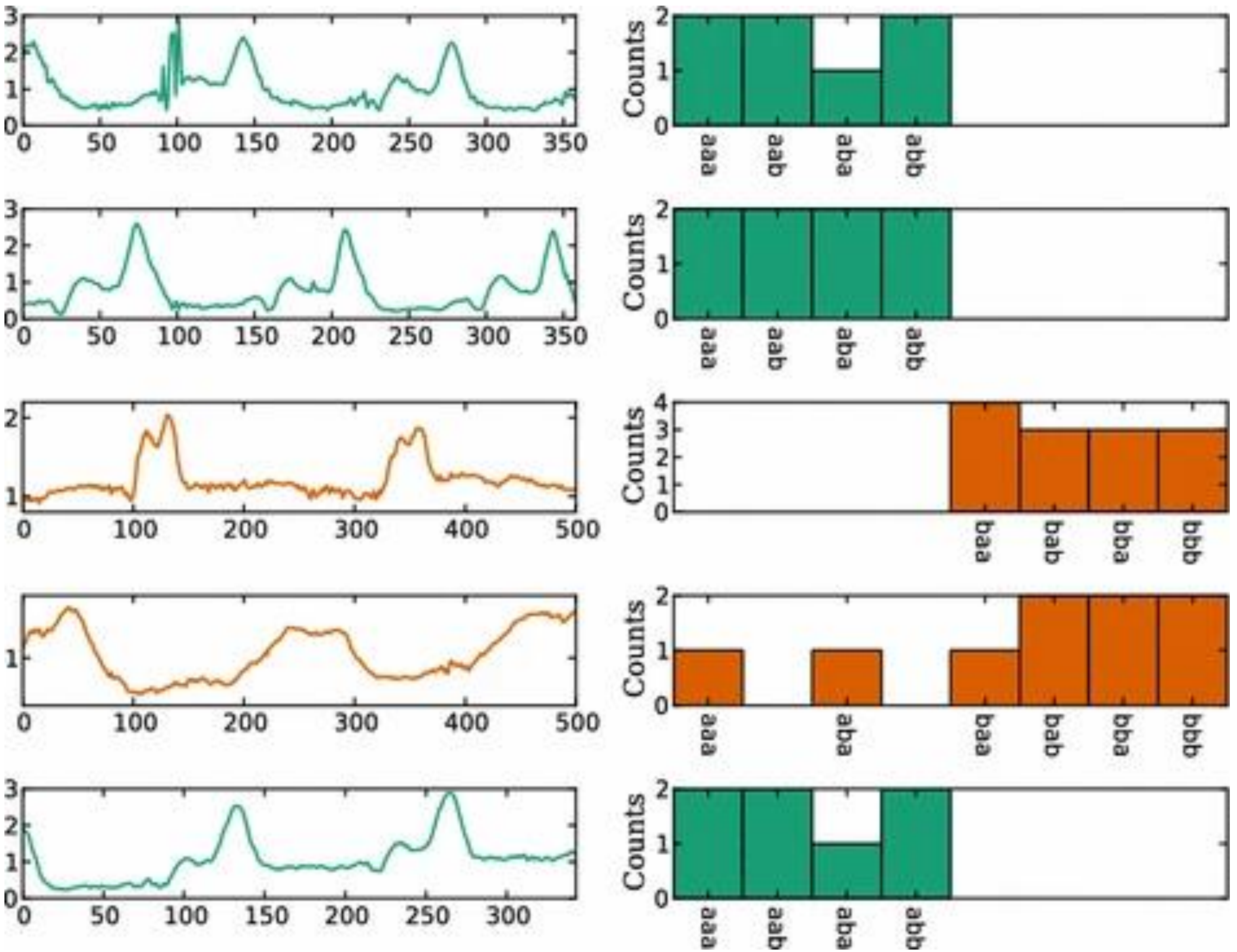


The BOSS Model



- First, sliding windows of length are extracted from a TS.
- Next, each sliding window is normalized to have a standard deviation of 1 to obtain amplitude invariance.
- SFA transformation is applied to each real valued sliding window transforming a time series into an *unordered* set of SFA words.
- Using an unordered set provides invariance to the horizontal alignment of each substructure within the TS (phase shift invariance).
- The first occurrence of an SFA word is registered and all duplicates are ignored.
- From these SFA words a histogram is constructed, which counts the occurrences of the SFA words

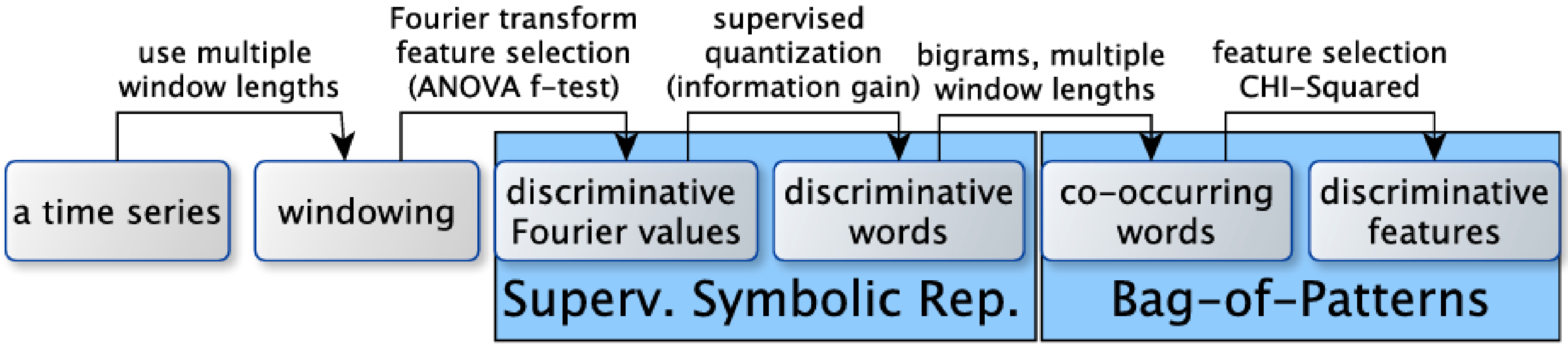

The BOSS model
1. Windowing
bcc
0
200
400
600
800
1000
2. SFA Words
bcc
0
200
400
600
800
1000
3. BOSS histogram
20
15
10
5
0
Counts
0.0
0.2
0.4
0.6
0.8
1.0



Counts
aaa
aab
aba
abb
baa
bab
bba
bbb

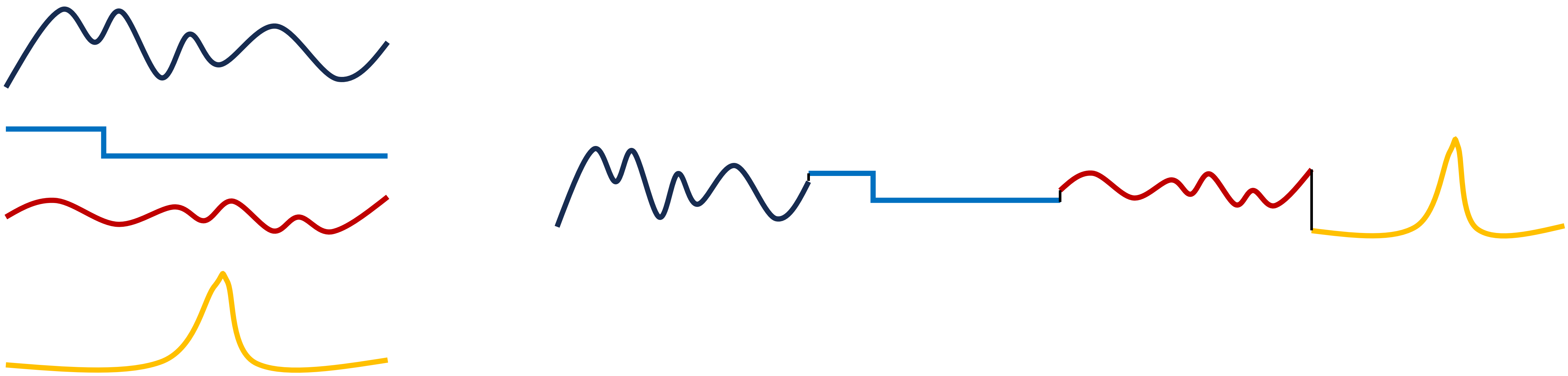
WEASEL - Word ExtrAction for time Series cLassification

- WEASEL extracts normalized windows of different lengths from a time series.
- Each window is approximated using the SFA, and those Fourier coefficients are kept that best separate TS from different classes using the ANOVA F-test.
- The remaining coefficients are discretized into a word using information gain binning.
- A bag-of-patterns is built from the words (unigrams) and neighboring words (bigrams), also including windows of variable lengths.
- The ChiSquared test is applied to filter out irrelevant words.
- A logistic regression classifier is applied.



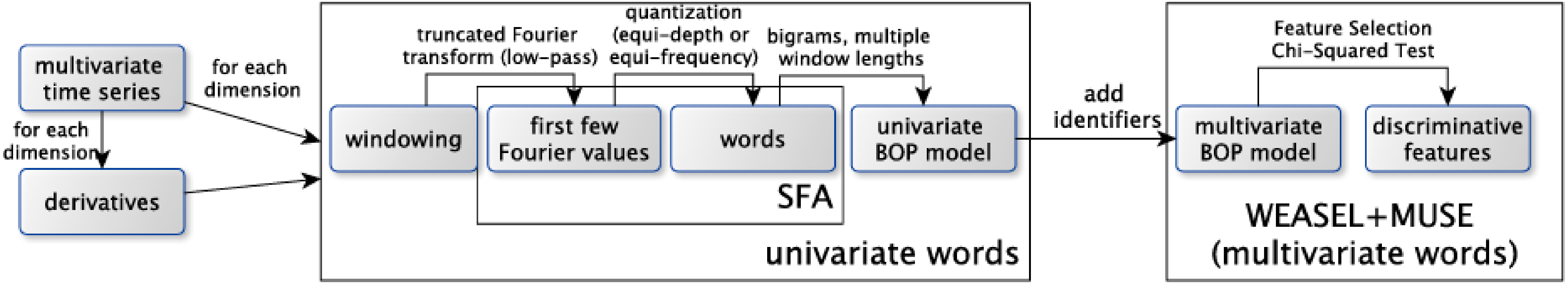
How to Deal with Multivariate TS

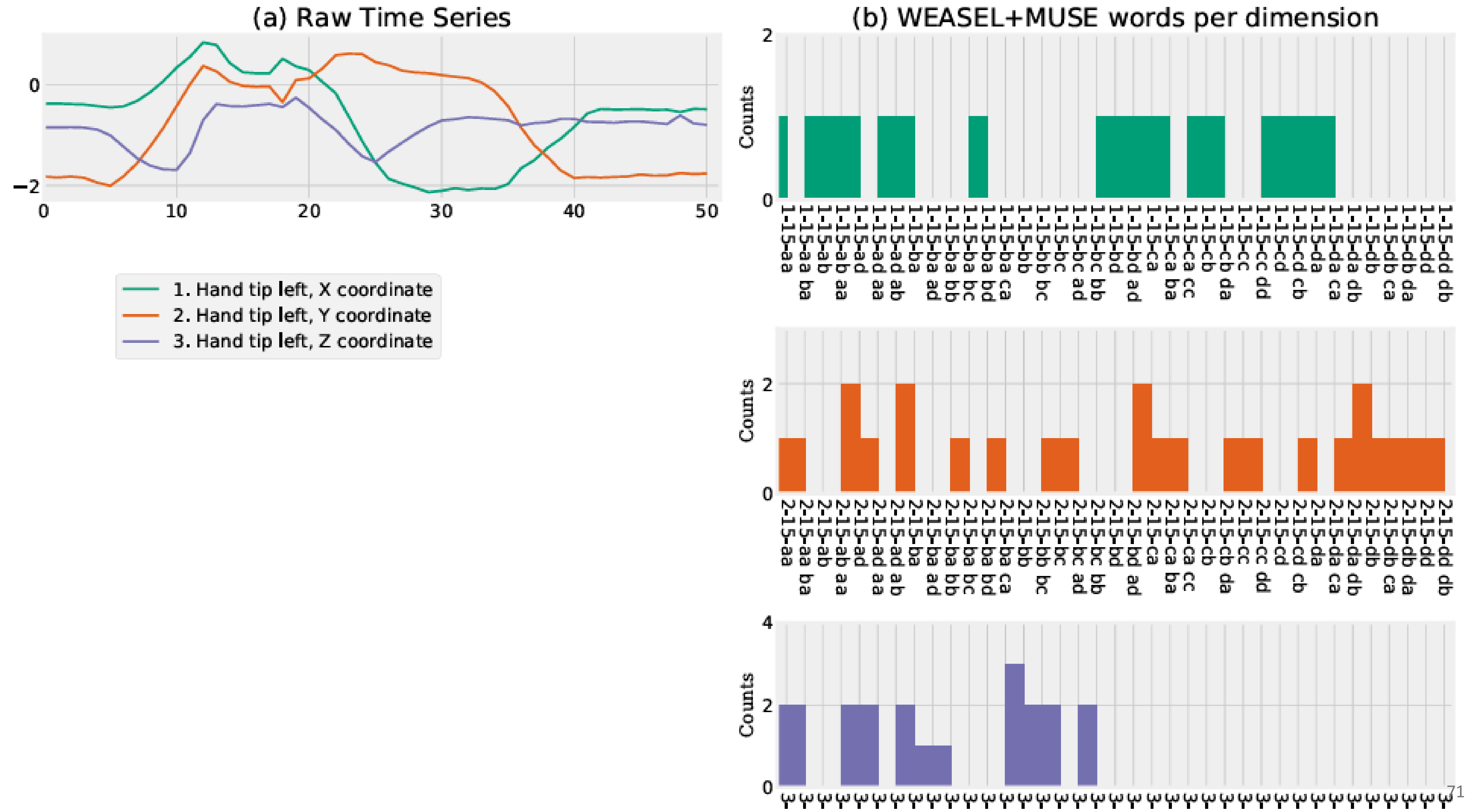
- We can assume that the various signals are independent
- A trivial way to adopt all the models analyzed is to concatenate the different signals to obtain a (long) univariate time series.



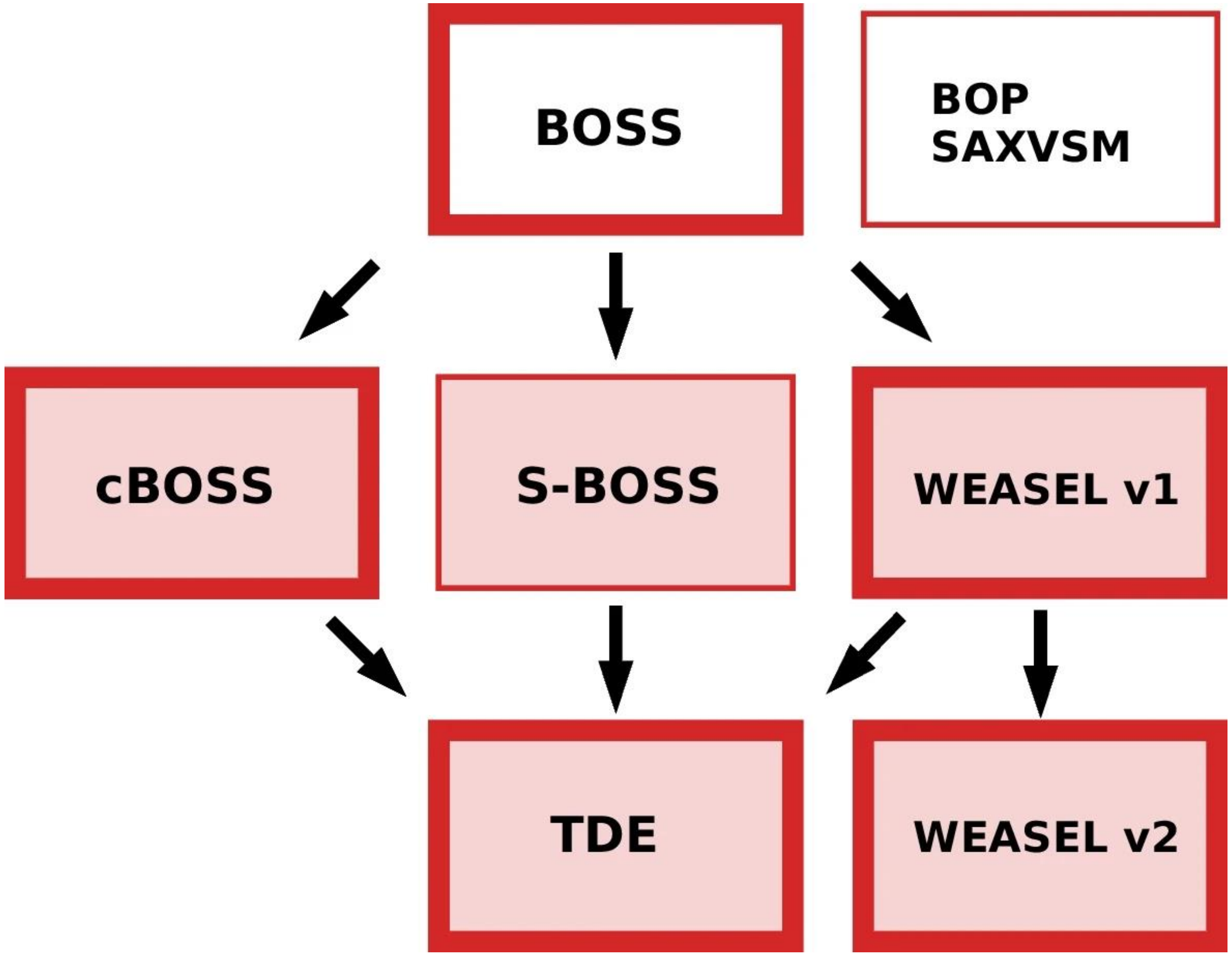
MUSE - Multivariate Unsupervised Symbols and dErivatives

- Also named WEASEL+MUSE, extends WEASEL by considering multivariate words.
- Multivariate words are obtained from the univariate words by concatenating each word with an identifier representing the sensor and the window size.



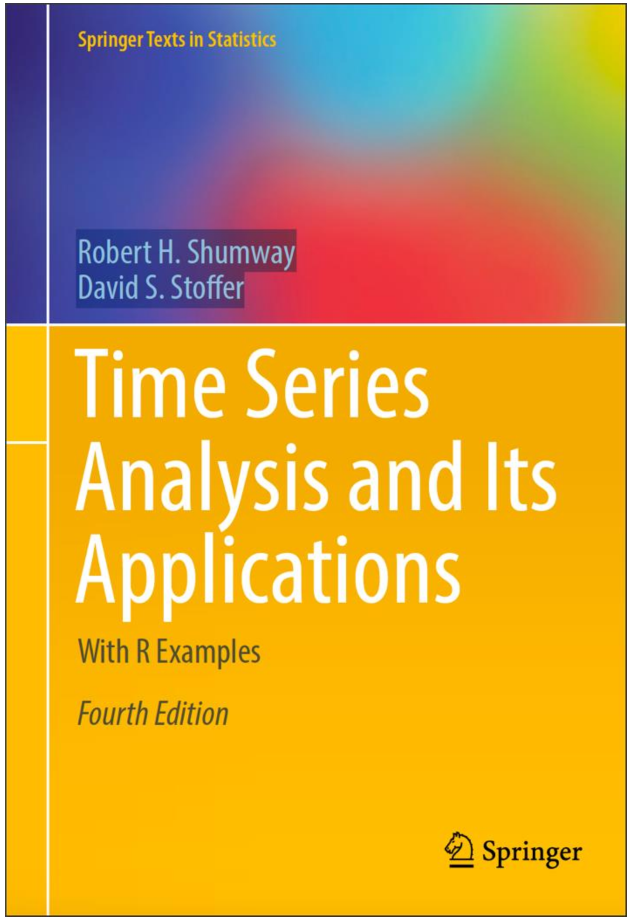

(a) Raw Time Series
1. Hand tip left, X coordinate
2. Hand tip left, Y coordinate
3. Hand tip left, Z coordinate
(b) WEASEL+MUSE words per dimension
Counts
1-15-aa
1-15-aa ba
1-15-ab
1-15-ab aa
1-15-ad
1-15-ad aa
1-15-ad ab
1-15-ba
1-15-ba ad
1-15-ba bb
1-15-ba bc
1-15-ba bd
1-15-ba ca
1-15-bb
1-15-bb bc
1-15-bc
1-15-bc ad
1-15-bc bb
1-15-bd
1-15-bd ad
1-15-ca
1-15-ca ba
1-15-ca cc
1-15-cb
1-15-cb da
1-15-cc
1-15-cc dd
1-15-cd
1-15-cd cb
1-15-da
1-15-da ca
1-15-da db
1-15-db
1-15-db ca
1-15-db da
1-15-dd
1-15-dd db
2-15-aa
2-15-aa ba
2-15-ab
2-15-ab aa
2-15-ad
2-15-ad aa
2-15-ad ab
2-15-ba
2-15-ba ad
2-15-ba bb
2-15-ba bc
2-15-ba bd
2-15-ba ca
2-15-bb
2-15-bb bc
2-15-bc
2-15-bc ad
2-15-bc bb
2-15-bd
2-15-bd ad
2-15-ca
2-15-ca ba
2-15-ca cc
2-15-cb
2-15-cb da
2-15-cc
2-15-cc dd
2-15-cd
2-15-cd cb
2-15-da
2-15-da ca
2-15-da db
2-15-db
2-15-db ca
2-15-db da
2-15-dd
2-15-dd db

Overview of Dictionary-based Models and Relationships



References

- Selective review of offline change point detection methods. Truong, C., Oudre, L., & Vayatis, N. (2020). Signal Processing, 167, 107299.
- Time Series Analysis and Its Applications. Robert H. Shumway and David S. Stoffer. 4th edition.(https://www.stat.pitt.edu/stoffer/tsa4/tsa4.pdf)
- Mining Time Series Data. Chotirat Ann Ratanamahatana et al. 2010. (https://www.researchgate.net/publication/227001229_Mining_Time_Series_Data)
- Dynamic Programming Algorithm Optimization for Spoken Word Recognition. Hiroaki Sakode et al. 1978.
- Experiencing SAX: a Novel Symbolic Representation of Time Series. Jessica Line et al. 2009
- Compression-based data mining of sequential data. Eamonn Keogh et al. 2007.



References

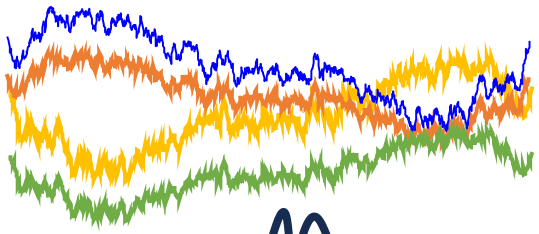
- catch22: CAnonical Time-series CHaracteristics selected through highly comparative time-series analysis Carl H Lubba et al. 2019.
- Proximity Forest An effective and scalable distance-based classifier for time series. Benjamin Lucas et al. 2018.
- A time series forest for classification and feature extraction. Houtao Deng. 2013.
- The Canonical Interval Forest (CIF) Classifier for Time Series Classification. Matthew Middlehurst et al. 2020.
- Diverse Representation Canonical Interval Forest. (HIVE-COTE 2.0: a new meta ensemble for time series classification). Matthew Middlehurst et al. 2021.
- Time series shapelets: a new primitive for data mining. L. Ye and E. Keogh. 2009.
- Classification of time series by shapelet transformation. Jon Hills et al. 2014.
- Learning time-series shapelets. Josif Grabocka et al. 2014.
- Binary Shapelet Transform for Multiclass Time Series Classification. A. Bostrom et al. 2017
- The BOSS is concerned with time series classification in the presence of noise. Patrick Schäfer 2014.
- Interpretable Time Series Classification using Linear Models and Multi-resolution Multi-domain Symbolic Representations. Thach Le Nguyen et al. 2020.
- MrSQM: Fast Time Series Classification with Symbolic Representations. Thach Le Nguyen et al. 2022.
- Fast and Accurate Time Series Classification with WEASEL. Patrick Schäfer et al. 2017.
- Bake off redux: a review and experimental evaluation of recent time series classification algorithm. Matthew Middlehurst et al. 2024

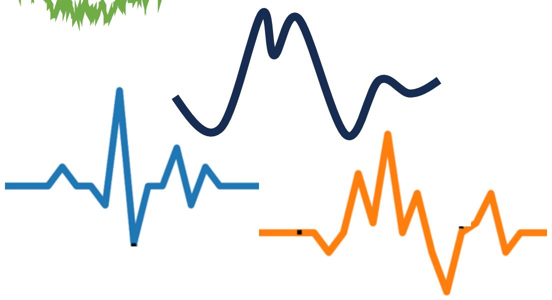
Deep-learning Models

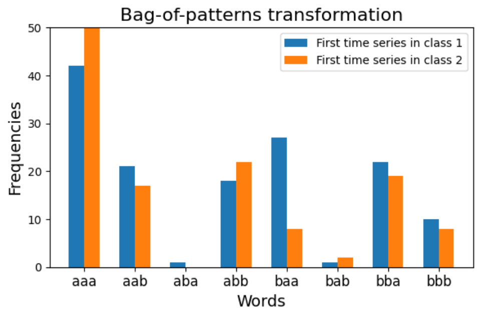
DNN for TS

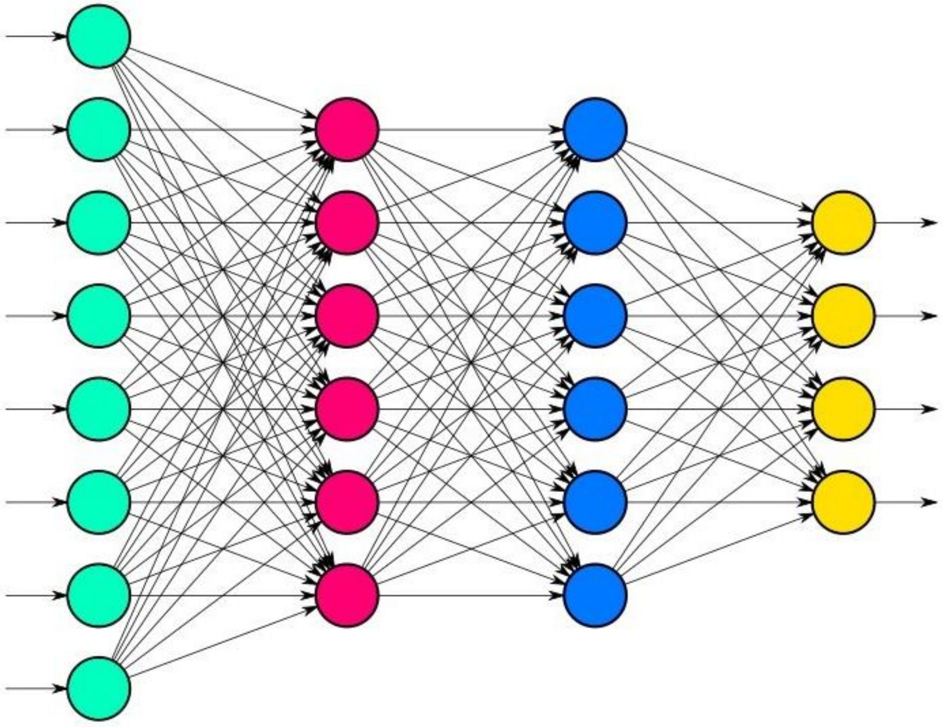
DNN, i.e., Multilayer Neural Networks can be used as they are on any data representation discussed so far calculated on on any domain:

- Raw TS
- Shapelets
- Intervals
- Distances
- Bag of Patterns

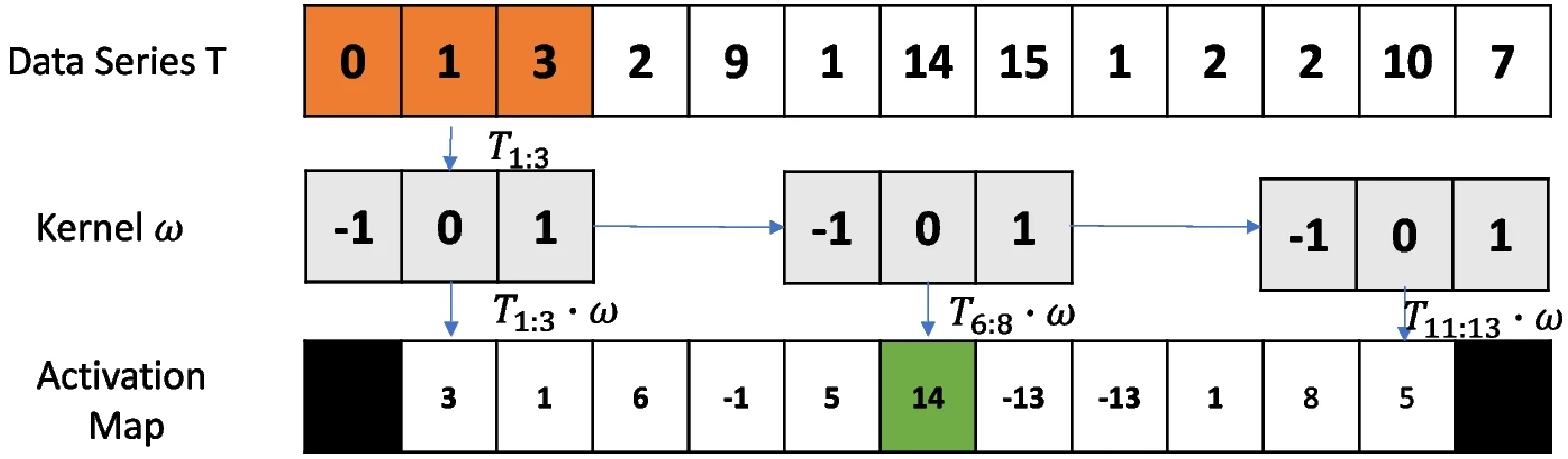








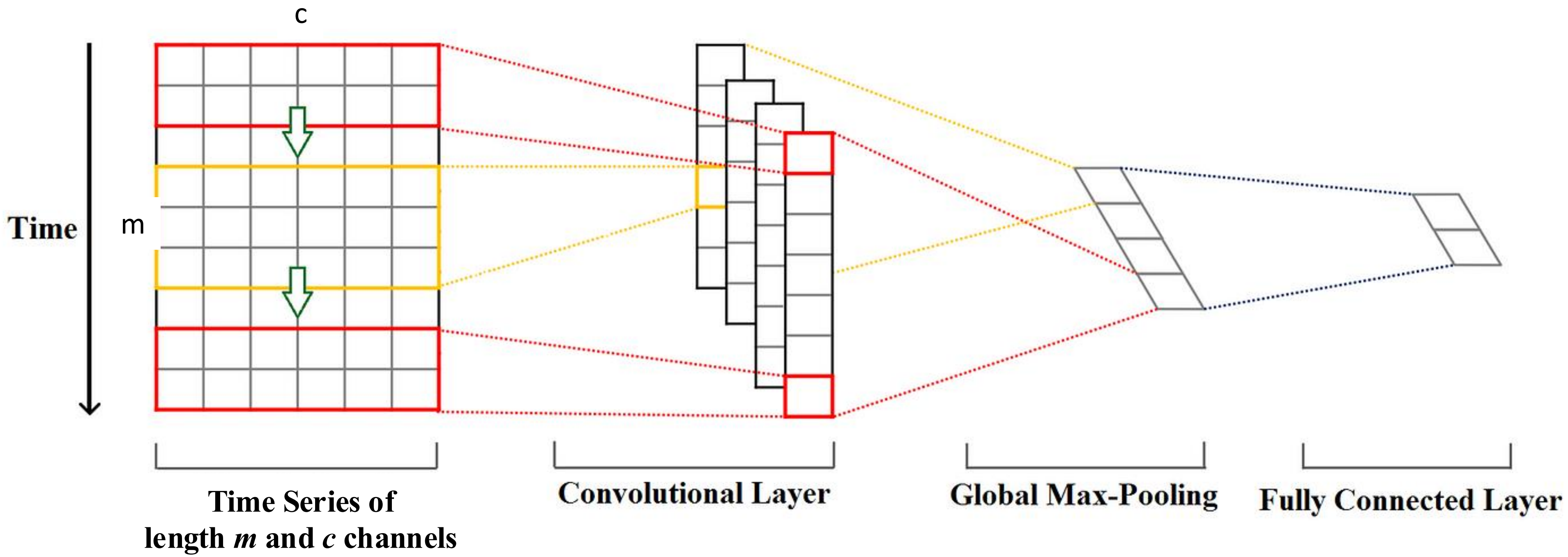
Convolutions for TS



Max-Pooling: 14
PPV-Pooling: 8/11

CNN for TS

- Convolution and pooling operations are alternatively used to generate deep features of the raw data.
- Then the features are connected to a multilayer perceptron (MLP) to perform classification.

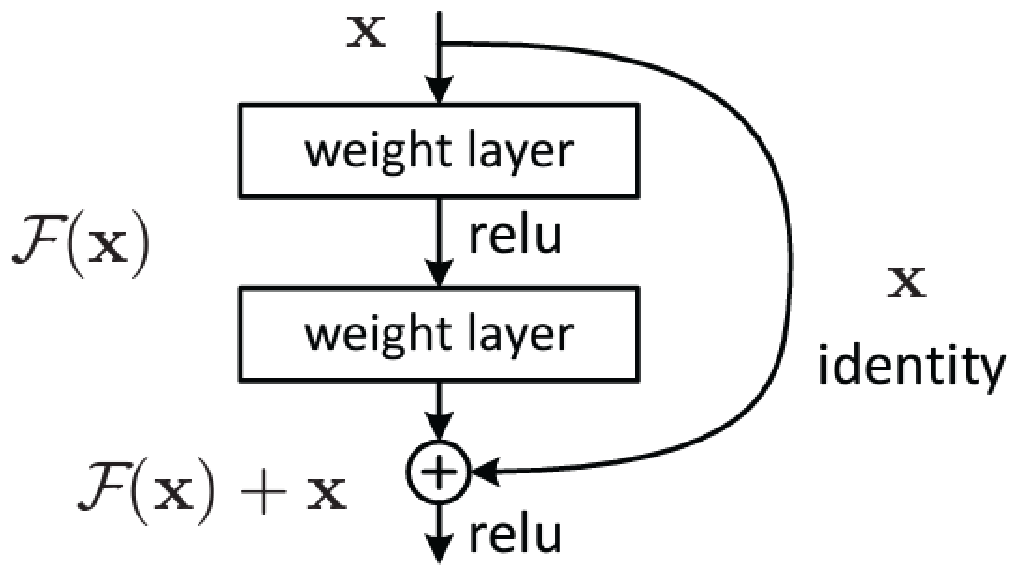


Problems with DNN

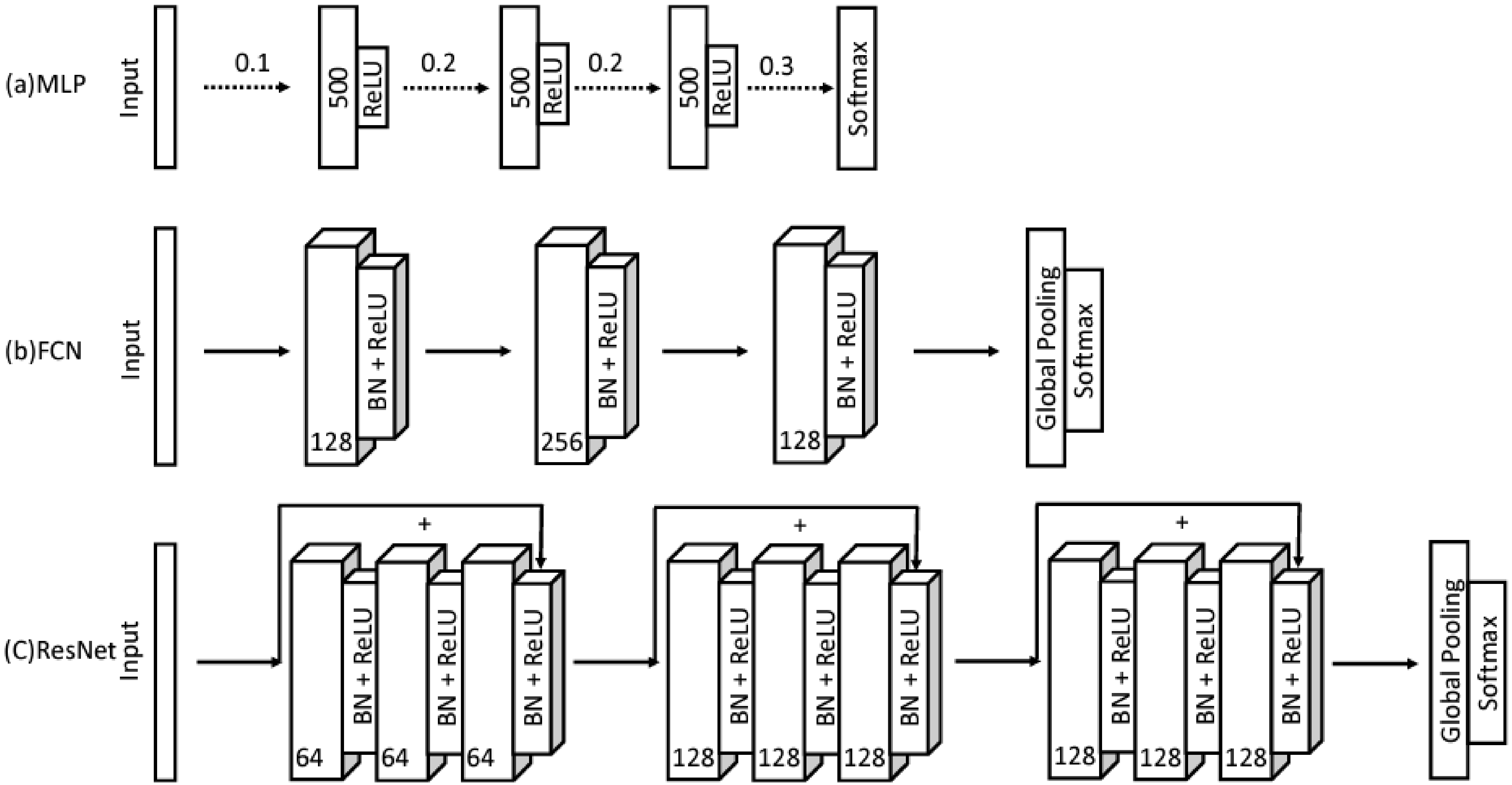
- When deeper networks are able to start converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated and then degrades rapidly.
- Such degradation is not caused by overfitting, and adding more layers to a suitably DNN leads to higher training error.

ResNet

- Deep Residual Learning framework.
- Denoting the desired underlying mapping as H(x), we let the stacked nonlinear layers fit another mapping of F(x) = H(x) − x.
- The original mapping is recast into F(x) + x.
- F(x) + x can be realized with a **shortcut connection**, i.e., skipping one or more layers by simply performing identity mapping and adding the outputs of the stacked layers.

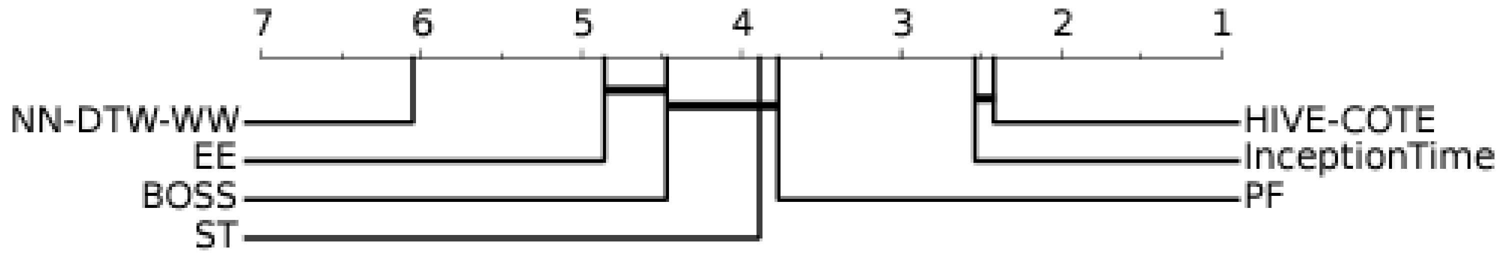


MLP vs CNN vs ResNet

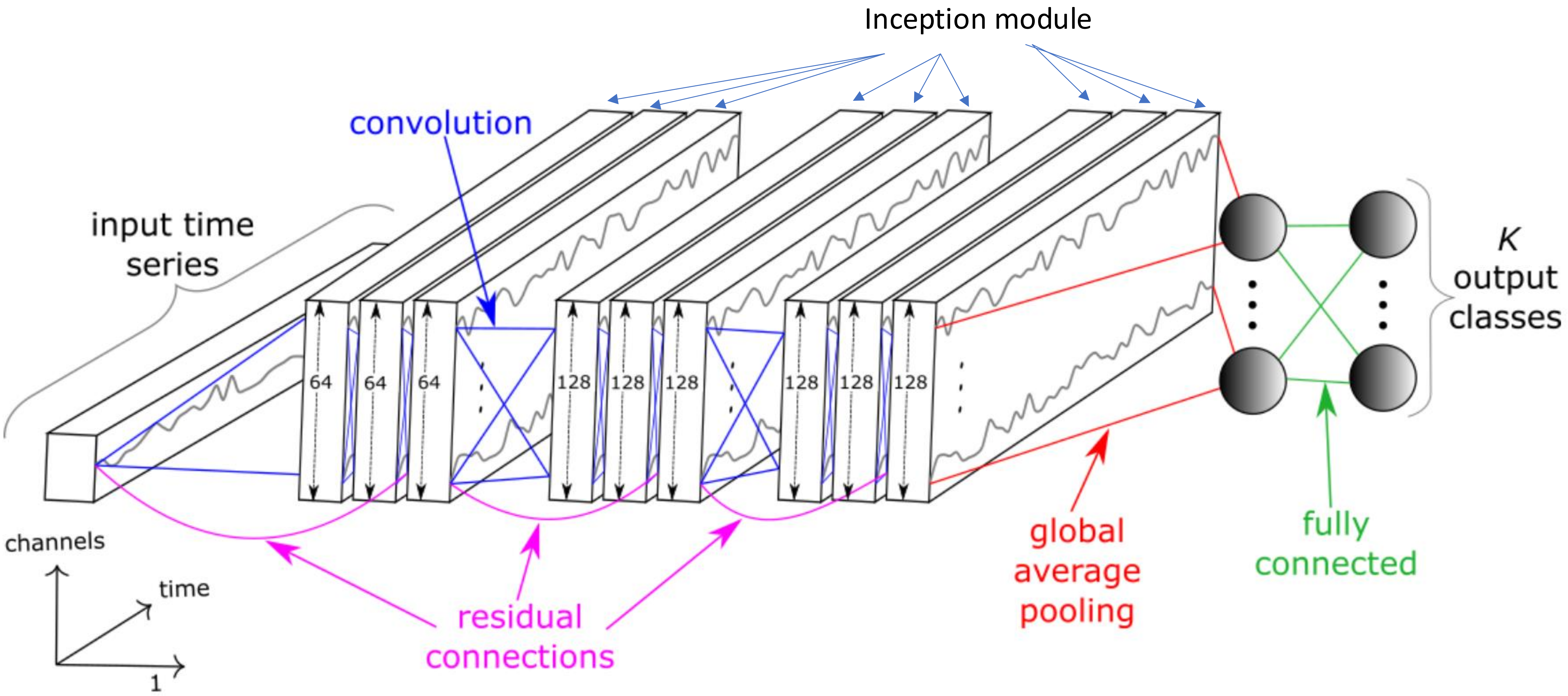


InceptionTime

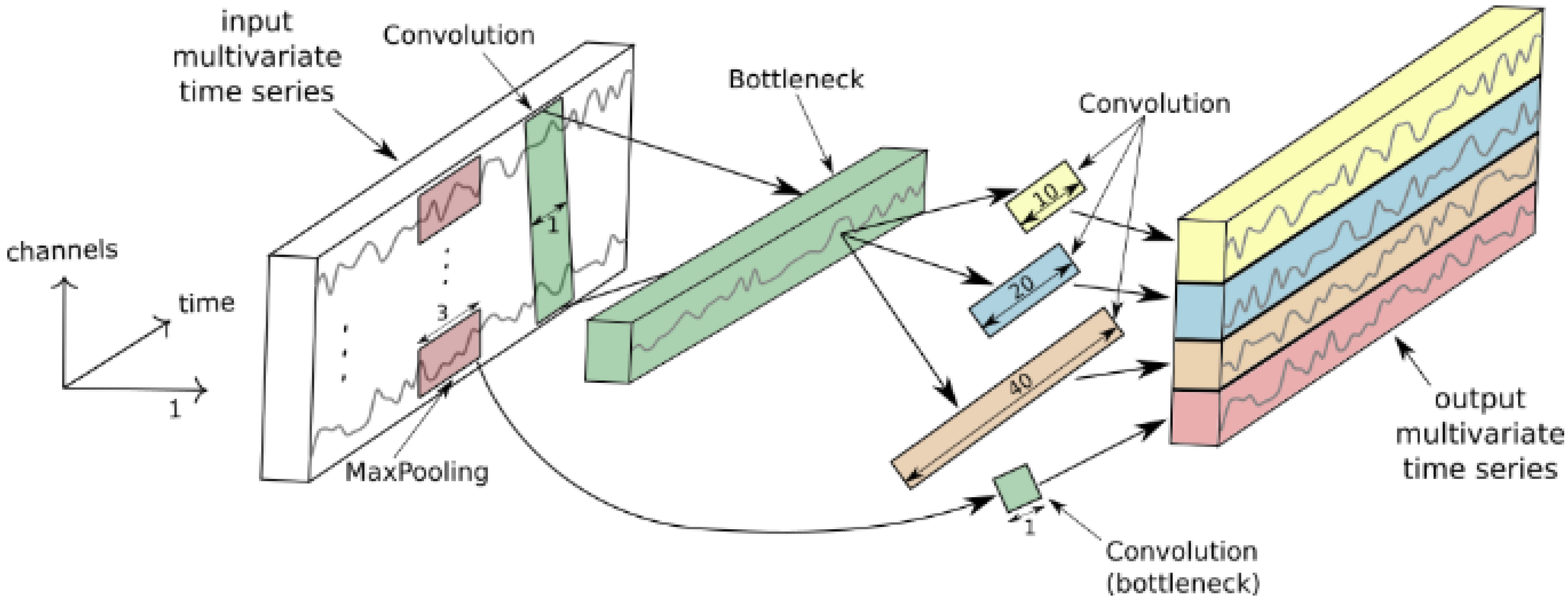
- Ensemble of CNN consisting of five Inception Networks.
- For each Inception Network:
  - 3 Inception Modules (6 blocks by default)
  - Global Averaging Pooling
  - Fully-Connected layer with the softmax activation function.
- Each Inception module consists of convolutions with kernels of several sizes followed by batch normalization and RELU activation function.



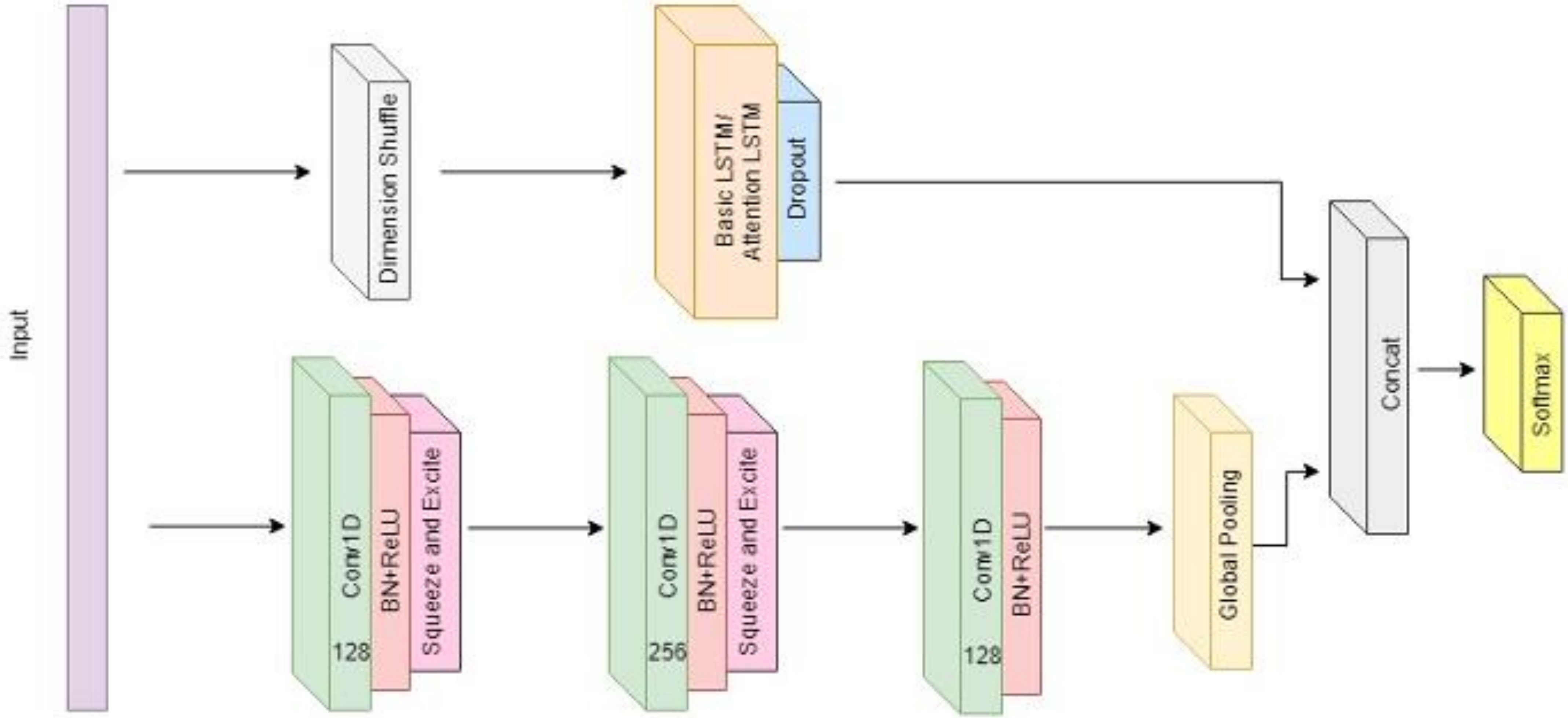
Inception Network



Inception Module



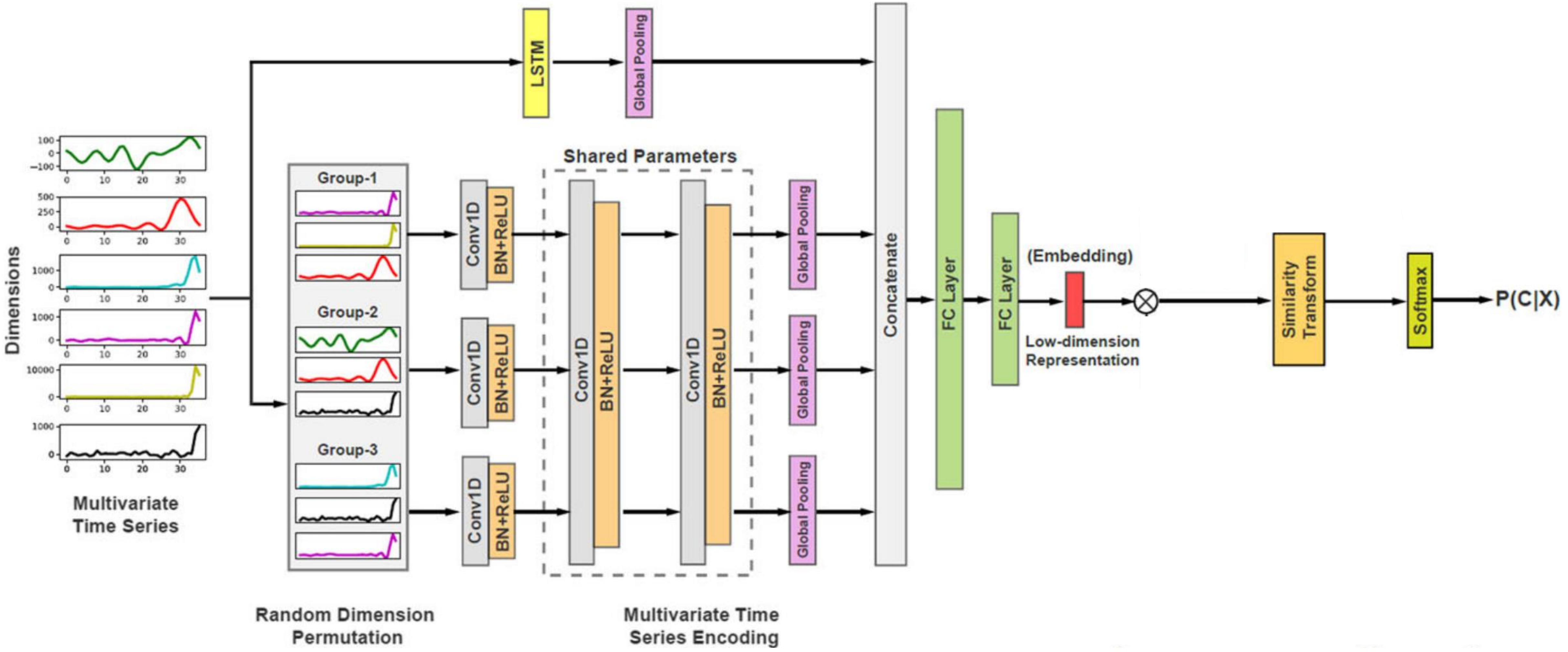
Multivariate LSTM-FCN



TapNet

- Draws on the strengths of both traditional and deep learning approaches:
  - **Deep learning Approaches:** excel at learning low dimensional features without the need for embedded domain knowledge, whereas
  - **Traditional Approaches:** work well on small datasets.
- Three distinct modules:
  - Random Dimension Permutation: produce groups of randomly selected dimensions with the intention of increasing the likelihood of learning how combinations of dimension values effect class value.
  - Multivariate Time Series Encoding:
    - 3 sets of 1d Convolutional layers followed by Batch Normalisation
    - Raw data is also passed through an LSTM and Global Pooling Layer
  - Attentional Prototype Learning: used for unlabelled data

TapNet

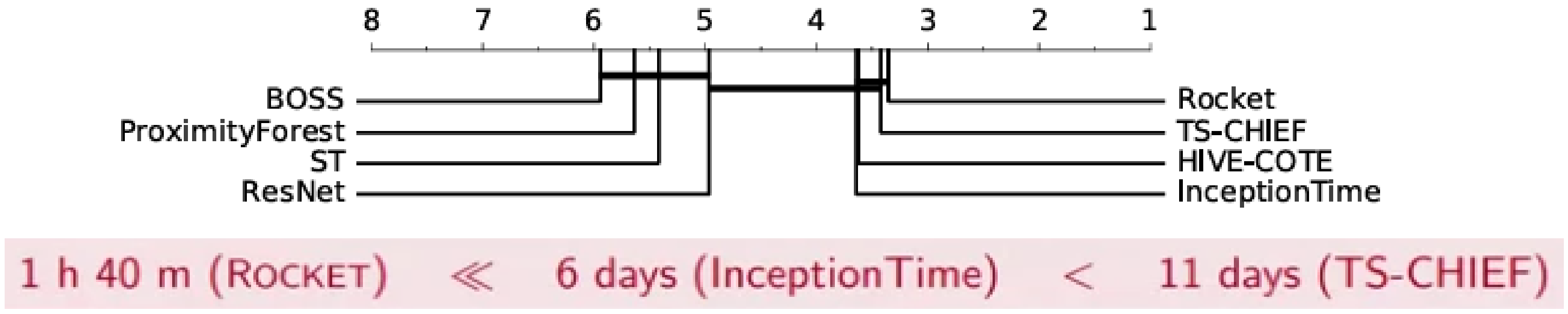


LSTM
Global Pooling
Shared Parameters
Group-1
Group-2
Group-3
Conv1D
BN+ReLU
Global Pooling
Concatenate
FC Layer
FC Layer
(Embedding)
Low-dimension Representation
Similarity Transform
Softmax
P(C|X)
Dimensions
Multivariate Time Series
Random Dimension Permutation
Multivariate Time Series Encoding

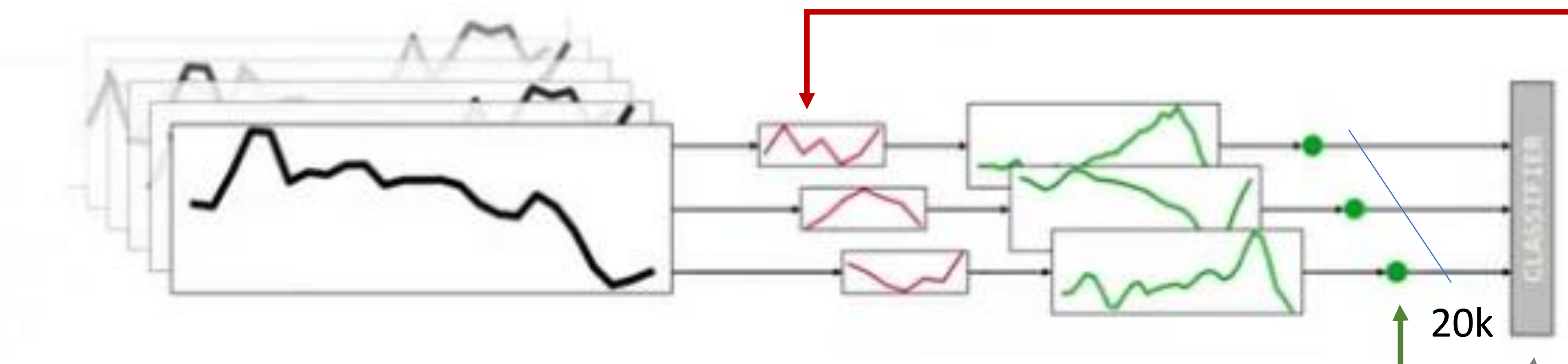
Kernel-based Models

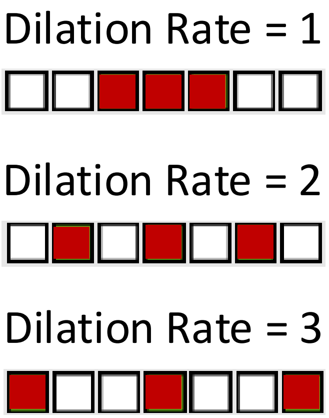
ROCKET - RandOm Convolutional KErnel Transform

- ROCEKT transforms TS using random convolutional kernels.
- Then uses the transformed features to train a linear classifier.
- It is accurate, fast and scalable.
- Much faster than other methos of comparable accuracy.



ROCKET - Core Aspects



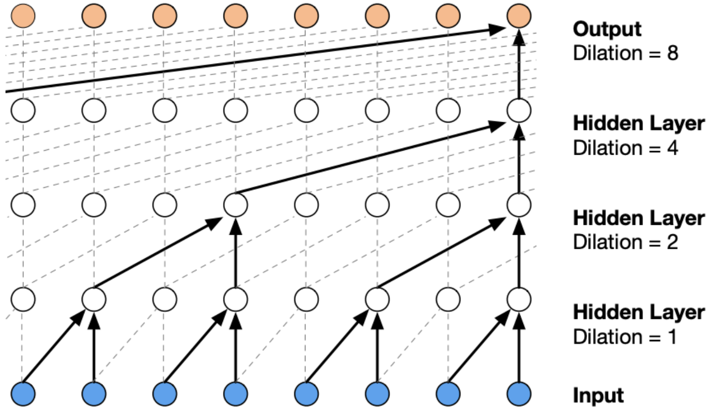


- 10k Random Convolutional Kernels with Dilation
- Two Pooling Strategies:
  - Max Value
  - PPV - Proportion of Positive Values, how much of the input matches the kernel
- Predictive Model:
  - Ridge Regression
  - Logistic Regression + SGD

Produces 20k Features

ROCKET vs. CNN

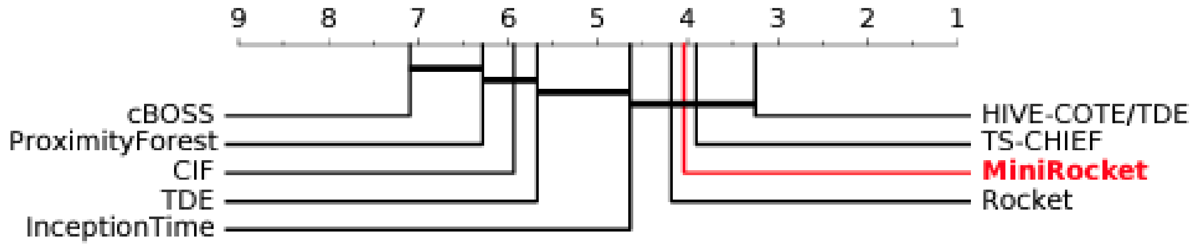
- CNNs use trainable kernels optimized by SGD to find patterns in the input data.
- ROCKET uses a single layer containing a very large number of random kernels.
- ROCKET uses a large variety of kernels: each kernel has random length, dilation, and padding, weights and biases.
- In CNNs kernel dilation increases exponentially with depth.
- ROCKET sample dilation randomly for each kernel.
- CNN uses Global Max Pooling
- ROCKET uses the Max value and the PPV.
- CNN hyperparameters are learnign rates, and network architecture
- ROCKET only hyperparameter is the number of kernels that handles the trade-off between classification accuracy and computation time.



MINIROCKET - MINImally ROCKET

- Like ROCKET, MINIROCKET transforms input TS using convolutional kernels, and uses the transformed features to train a linear classifier.
- MINIROCKET maintains dilation and PPV.
- Unlike ROCKET, MINIROCKET uses a small, fixed set of kernels, dose not use Max value pooling, and is almost entirely deterministic.
- MINIROCKET is up to 75 times faster than ROCKET

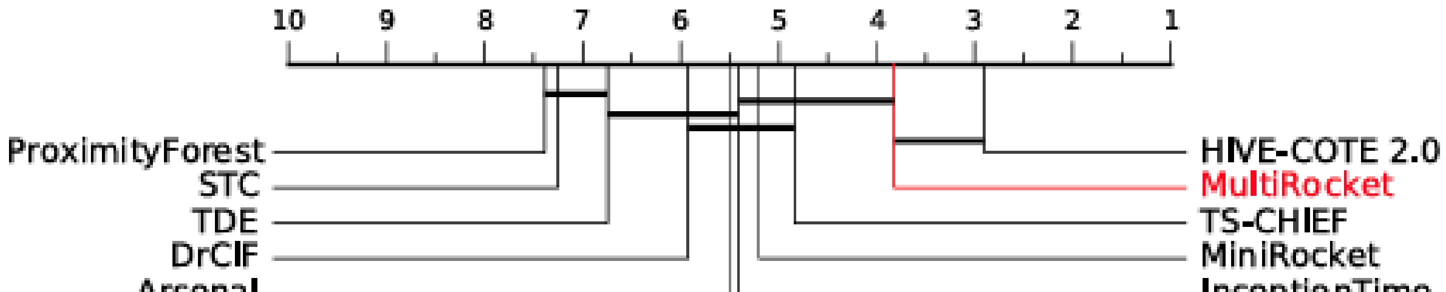
	ROCKET	MINIROCKET
length	$\{7, 9, 11\}$	9
weights	$\mathcal{N}(0, 1)$	$\{-1, 2\}$
bias	$\mathcal{U}(-1, 1)$	from convolution output
dilation	random	fixed (rel. to input length)
padding	random	fixed
features	PPV + max	PPV
num. features	20K	10K



MultiROCKET

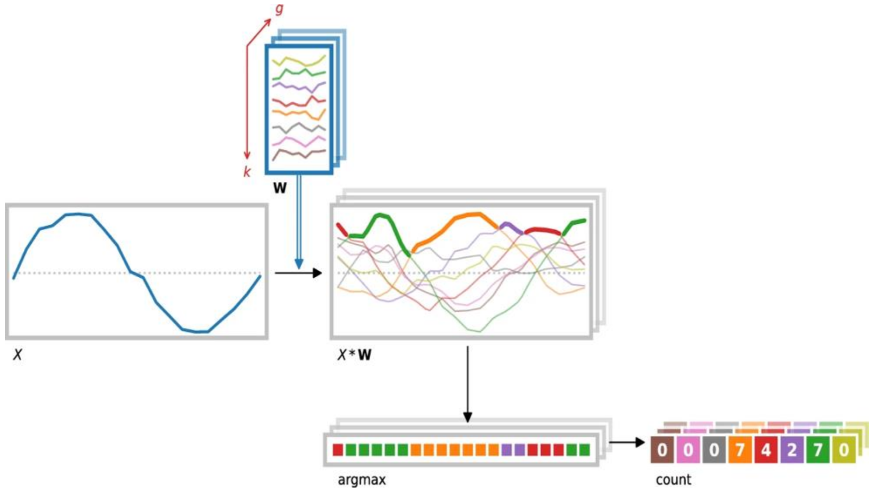
"Multi" does not refer to Multivariate TS

- MultiROCKET transforms a TS into its first order difference.
- Then both the original and the first order difference TS are convolved with the 84 MINIROCKET kernels.
- A different set of dilations and biases is used for each representation because both representations have different and range of
- Besides PPV, MultiROCKET adds 3 additional pooling operators
- By default, MultiROCKET produces approximately 50,000 features per TS.
- The transformed features are used to train a linear classifier.

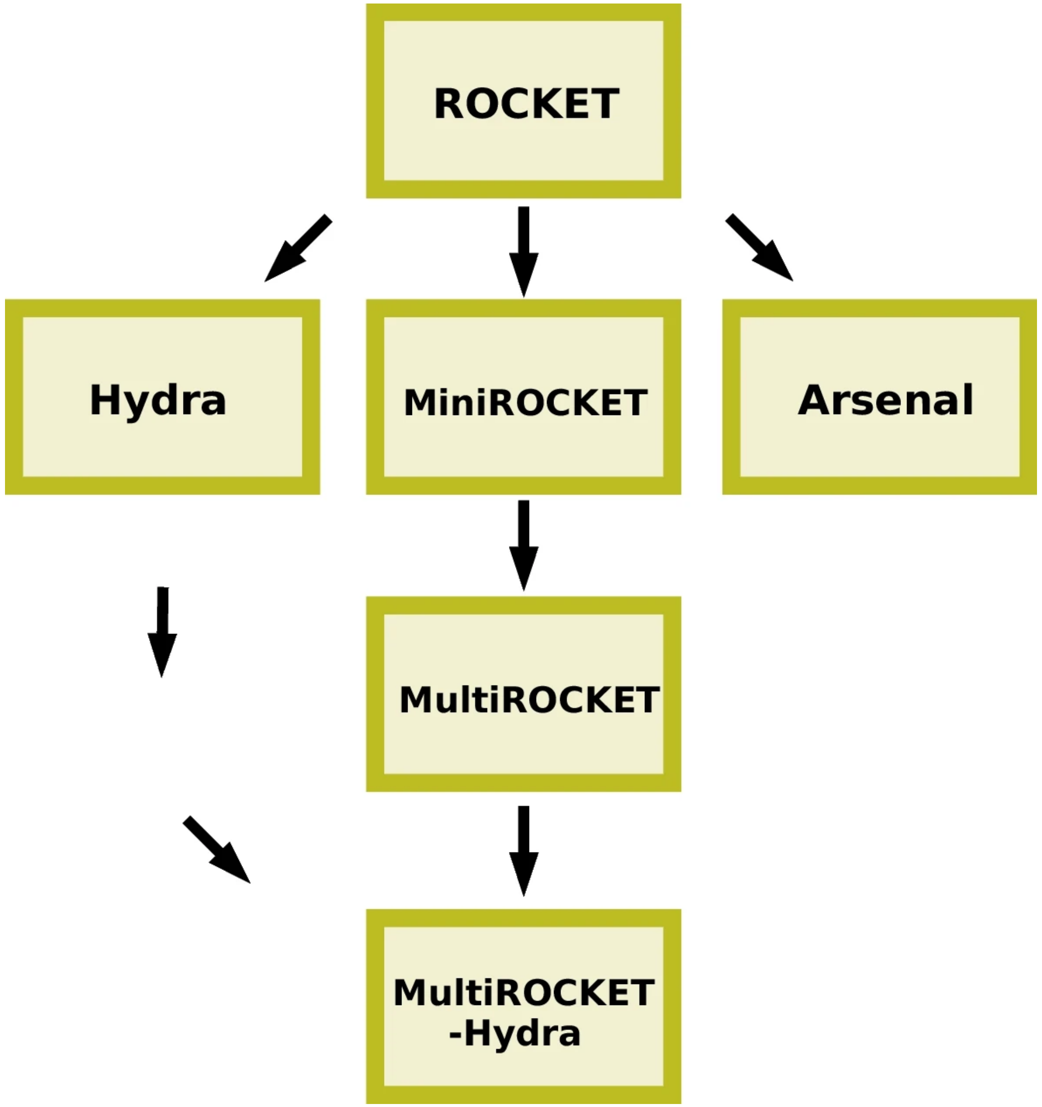


Hydra & MultiROCKET-Hydra

- Hydra: HYbrid Dictionary-ROCKET Architecture combines dictionary-based and convolution-based models.
- It starts with *g* groups of *k* random convolutional kernels each to calculate the activation of time series.
- In each group, is calculated the activation of a kernel with the time series and it is recorded how frequently this kernel is the best match (counts the highest activation).
- This results in a *k*-dimensional count vector for each of the *g* groups, resulting in a total of *g x k* features. Default *g = 64* and *k = 8.*
- Hydra is applied to both the time series and its first-order differences
- MultiROCKET-Hydra concatenates features from MultiROCKET and Hydra.



Overview of Kernel-based Models and Relationships



Hybrid Models

HIVE-COTE - Hierarchical Vote Collective of Transformation-based Ensembles

- Heterogeneous meta ensemble for TSC.
- Five ensembles working on features from four different data transformation:

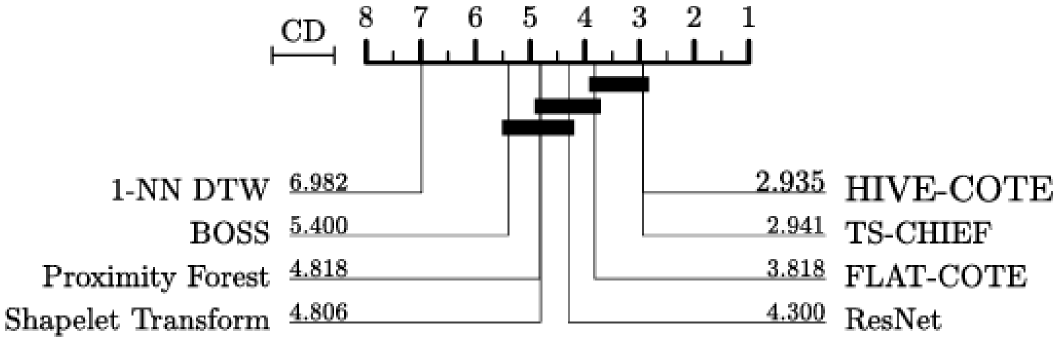
- Elastic Ensemble
- Shapelet Transform Classifier
- Time Series Forest
- Bag of Symbolic-Fourier-Approximation Symbols
- Random Interval Spectral Ensemble

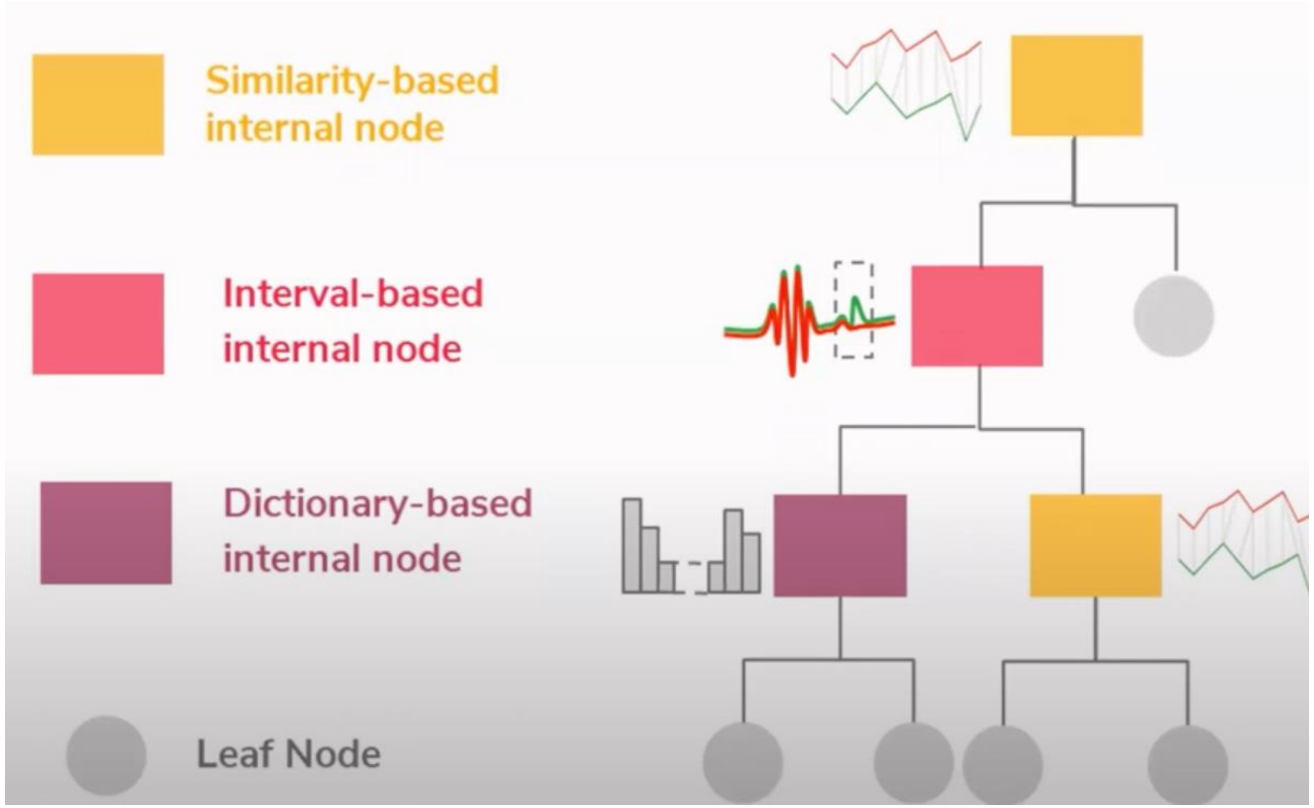
- Raw TS
- Shapelet-Tranformed TS
- Autocorrelation Features
- Power Spectrum Features

- Each ensemble is trained on the train data independently of the others.
- For new data, each ensemble passes an estimate of class probabilities to the control unit, which combines them to form a single prediction.
- It does this by weighting the probabilities of each module by an estimate of its testing accuracy formed from the training data.

TS-CHIEF - Time Series Combination of Heterogeneous and Integrated Embeddings Forest

- Tree-based ensemble for TSC using heterogeneous splits.
- Extends the Proximity Forest with trees considering splits w.r.t. dictionary-based and interval-based features.

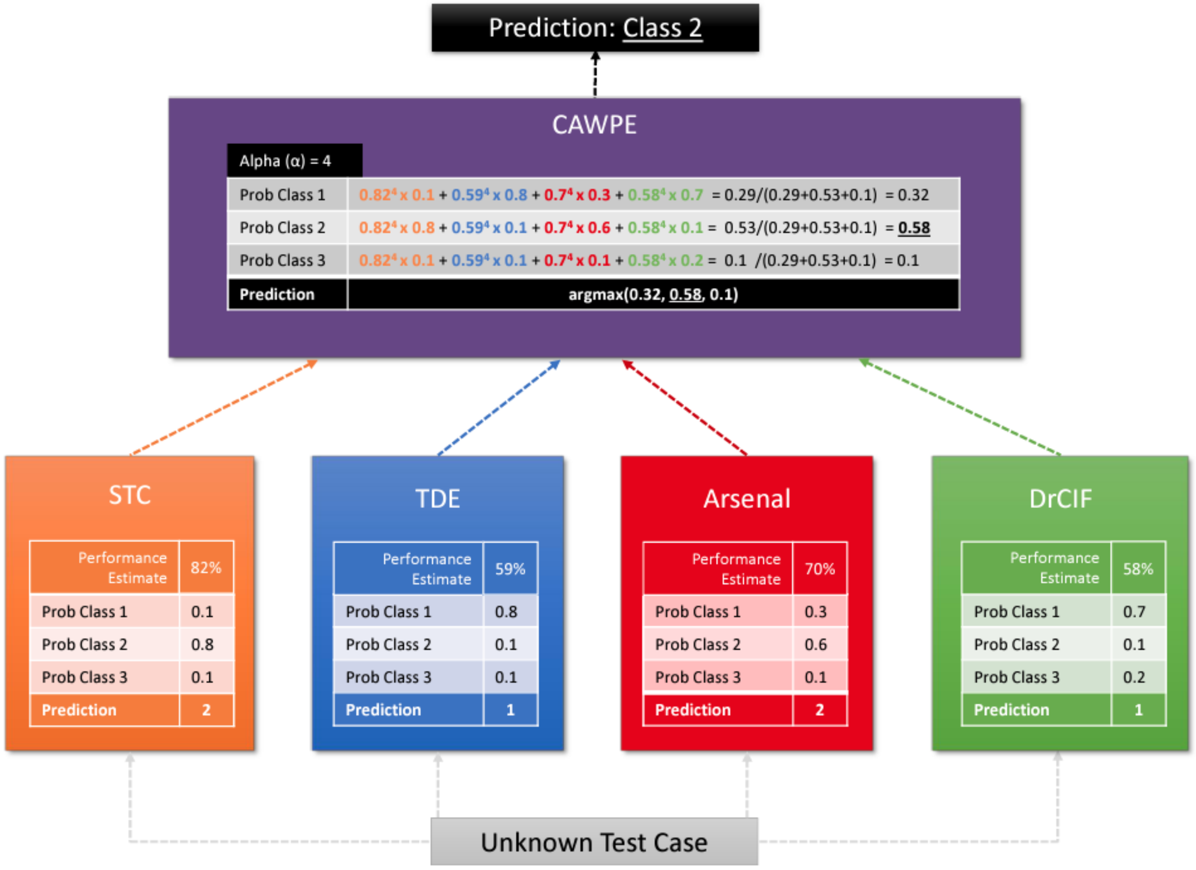


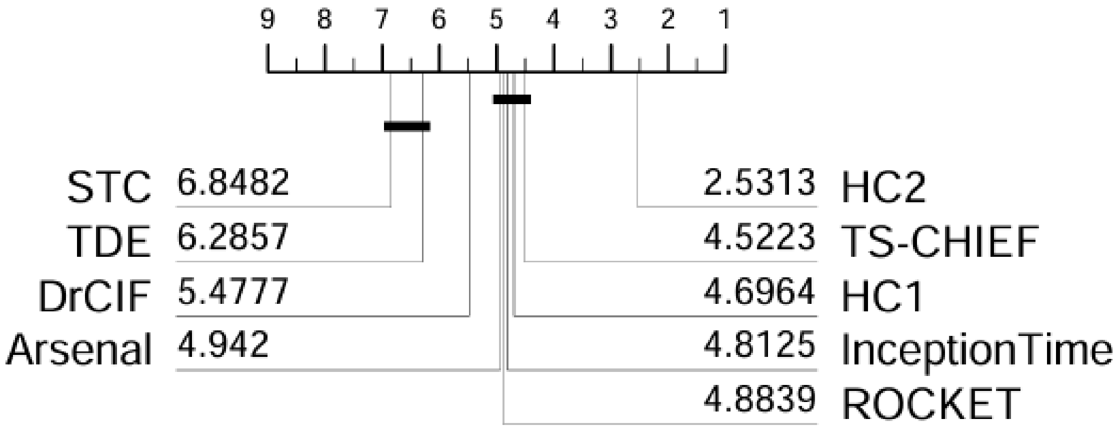


HIVE-COTE 2.0

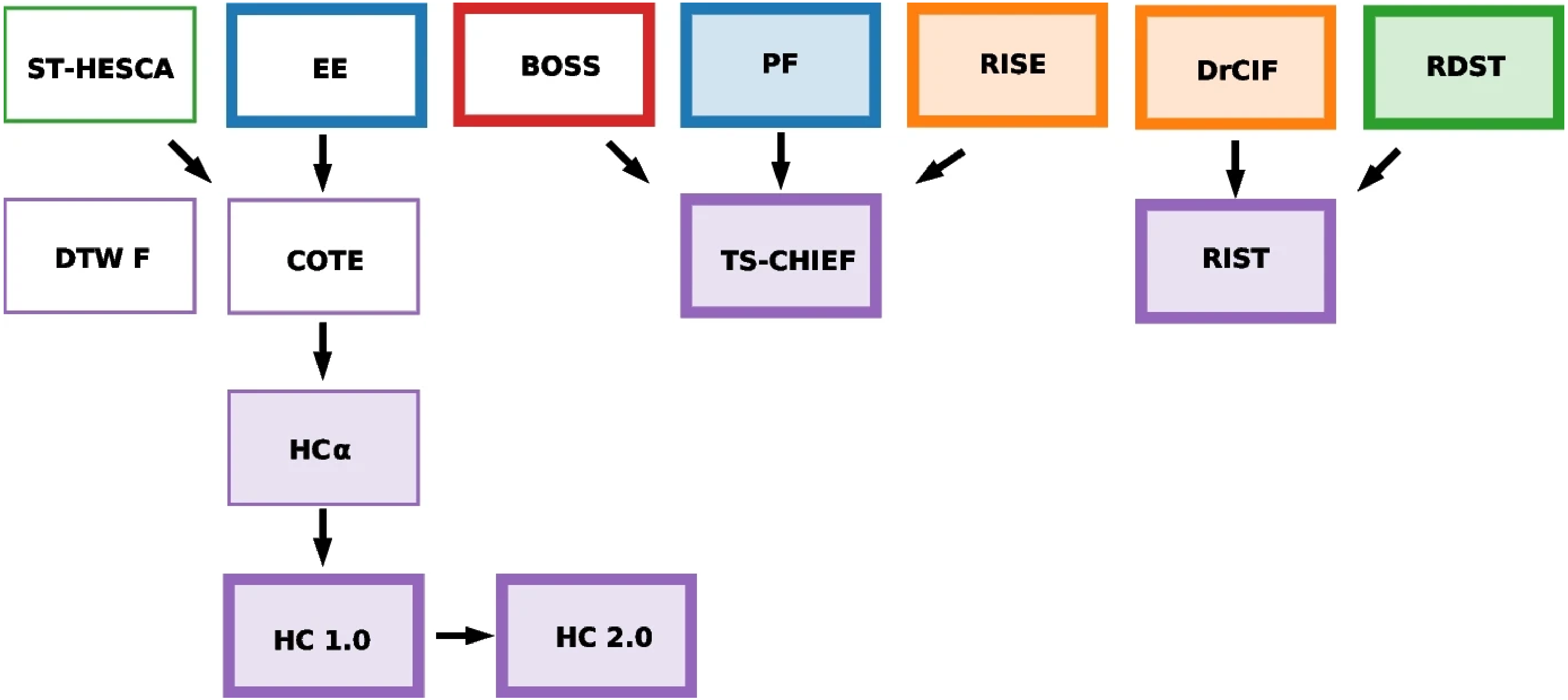
Adopts the following ensembles:

- Shapelet Transform Classifier.
- A convolution-based ensemble of ROCKET named Arsenal.
- The dictionary-based Temporal Dictionary Ensemble, i.e., a fast version of BOSS.
- The interval-based DrCIF.

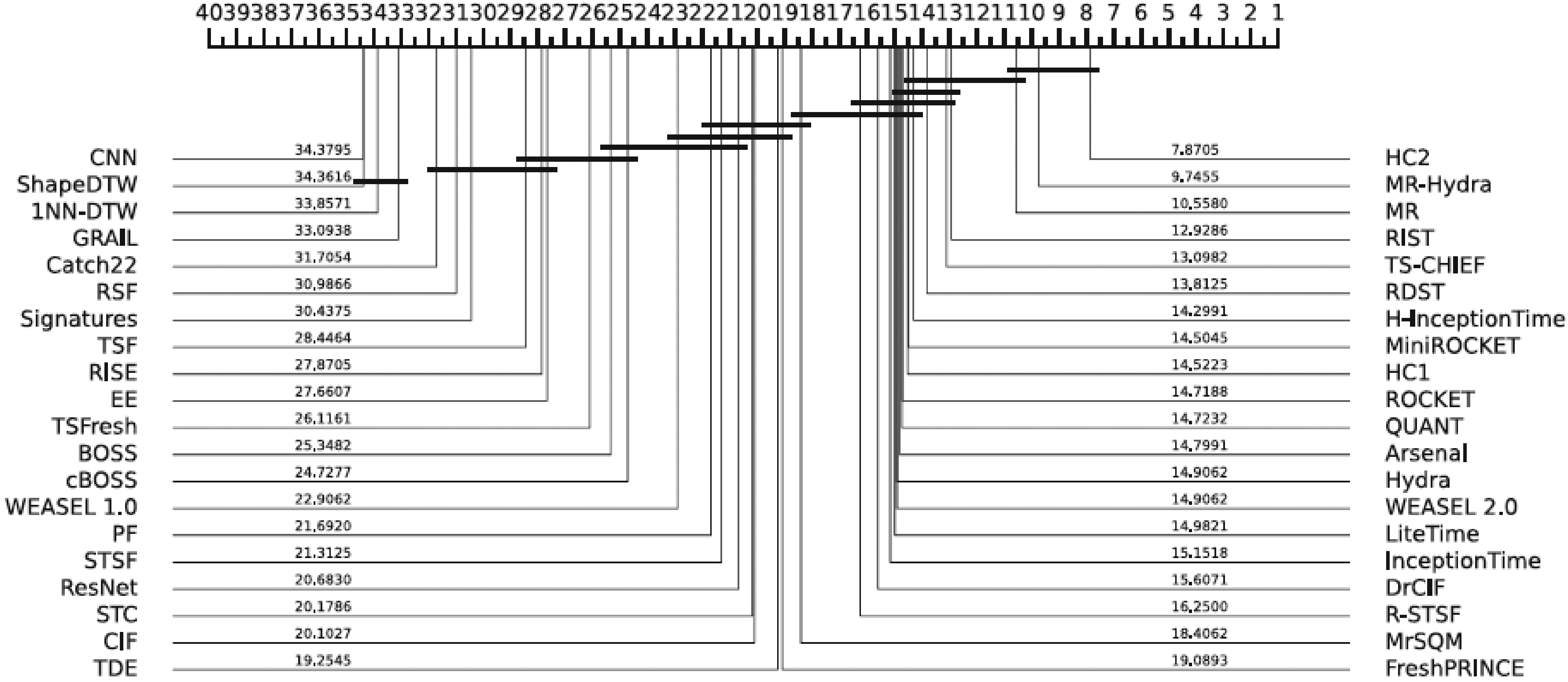




Overview of Hybrid Models and Relationships



TSC Methods Comparison



TSC Methods Comparison

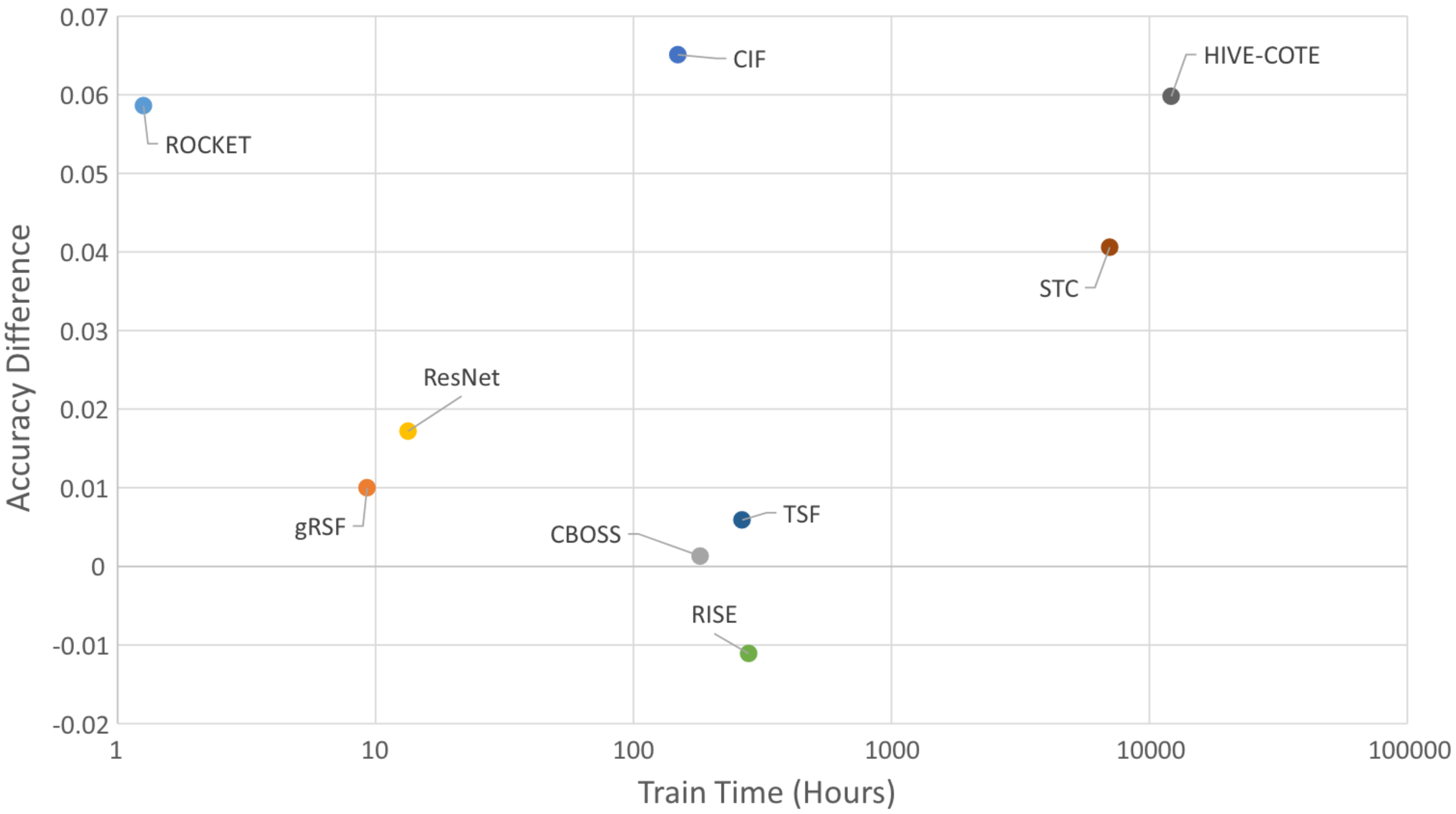


Fig. 10 Average difference in accuracy to DTW_D versus train time for 9 MTSC algorithms

XAI for TSA

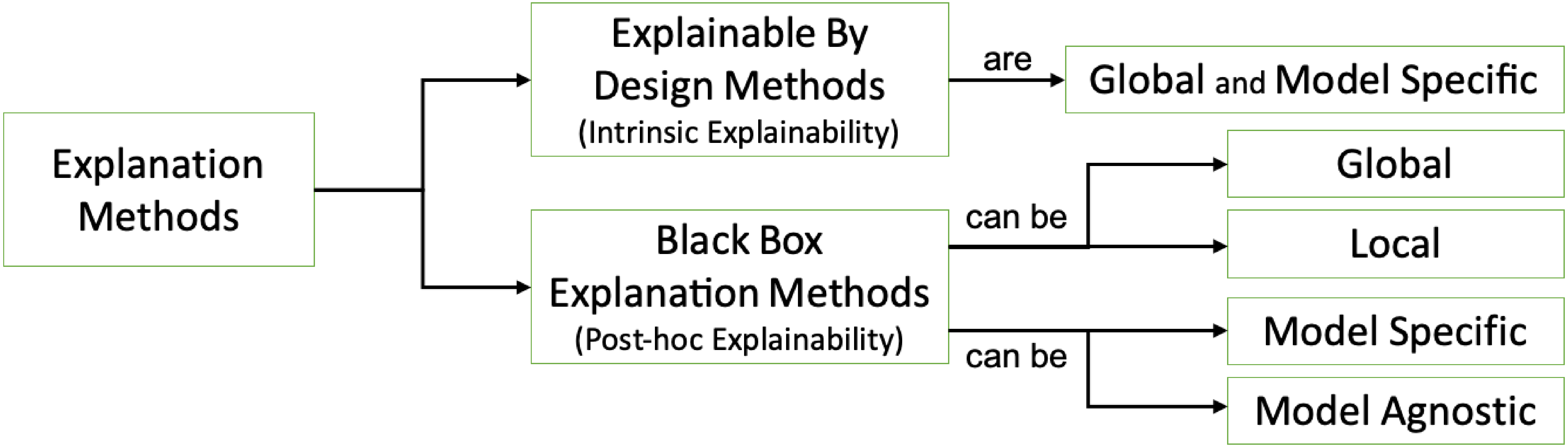
What is a Black Box Model?

- A ***black box*** is a model, whose internals are either unknown to the observer or they are known but uninterpretable by humans.

Example:

- DNN
- Ensembles
- ROCKET
- HIVE-COTE

XAI Taxonomy of Explanation Methods



Local Post-hoc Explanations Types

- Outlook = Sunny
- Temp = Hot
- Humidity = Normal
- Wind = Weak

- Black Box Prediction:
- Play Tennis = Yes

- Logic-based
  - Rule-based
  - Decision Tree
- Score-based
  - Features Importance
  - Saliency Maps
  - Attributions
- Instance-based
  - Prototypes
  - Counter-exemplars

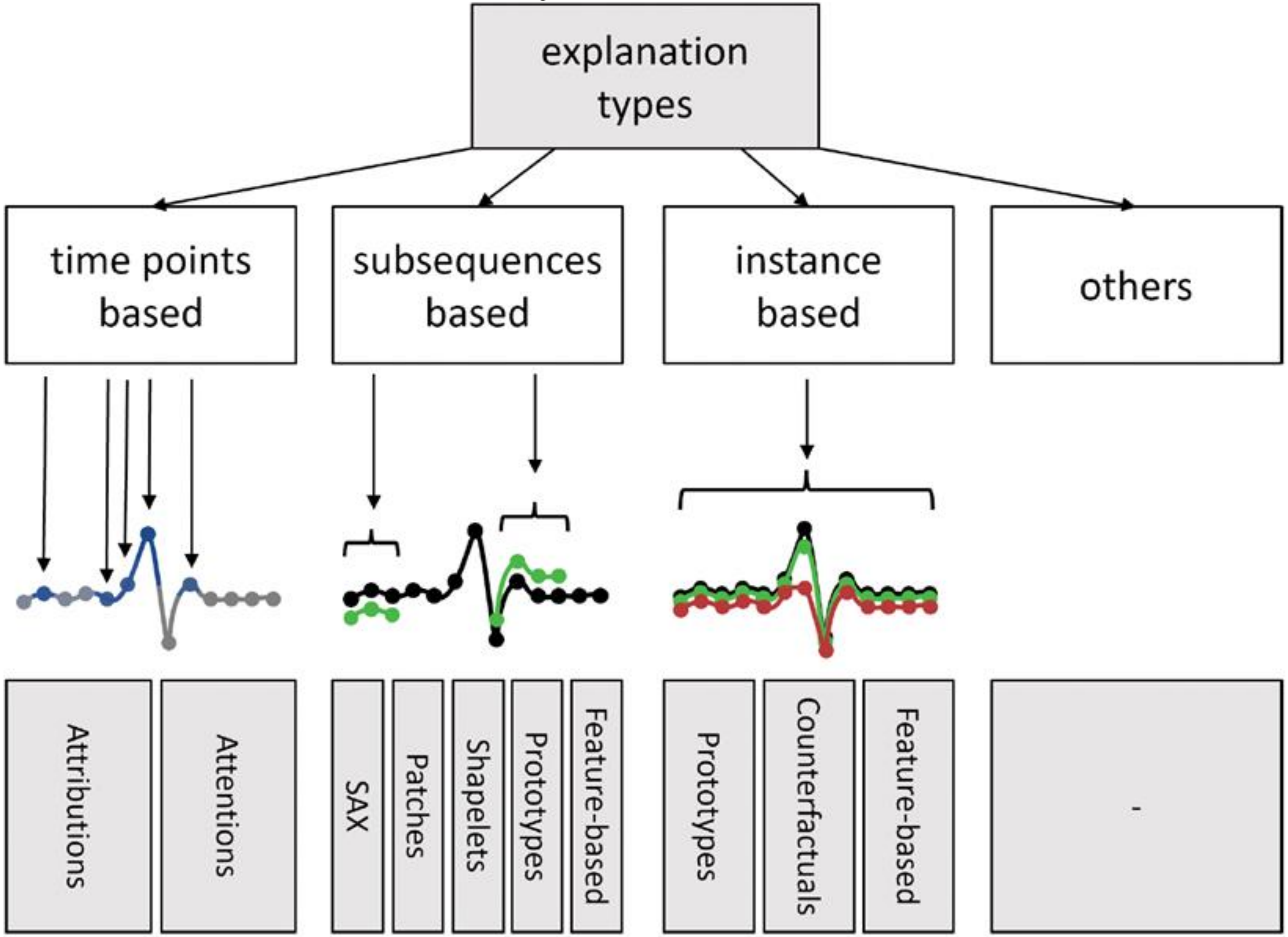
- IF Outlook = Sunny AND Humidity = Normal THEN Play Tennis = Yes

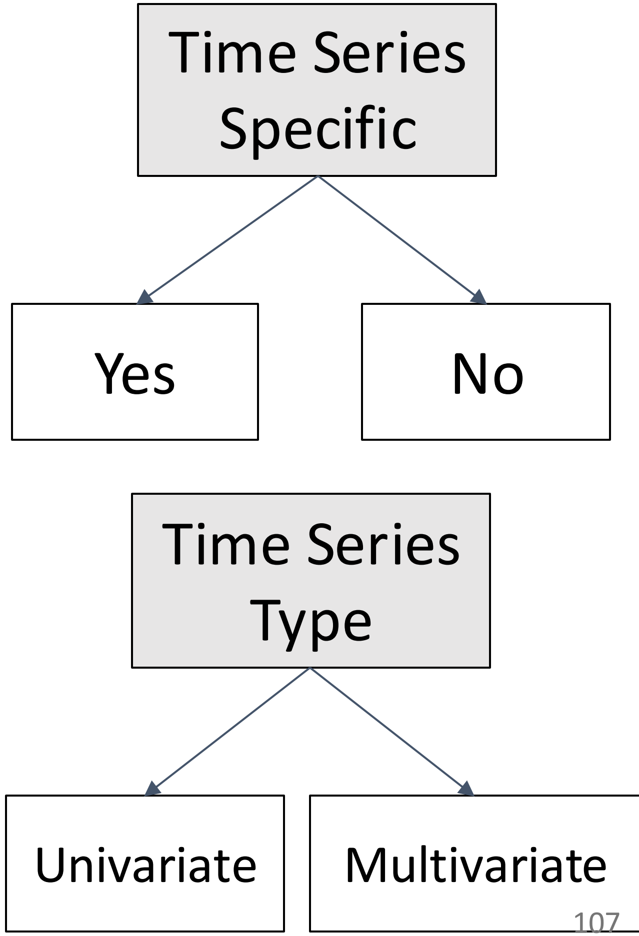
- Outlook: 0.7
- Temp: 0.0
- Humidity: -0.4
- Wind: 0.0

- Outlook = Sunny
- Temp = Hot
- **Humidity = Hight**
- Wind = Weak

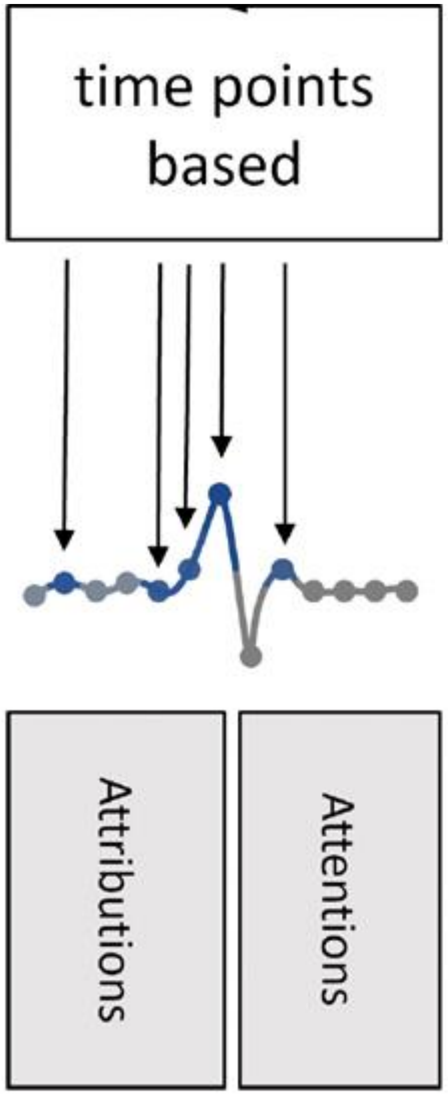
- Black Box Prediction:
- Play Tennis = No

XAI-TS Taxonomy



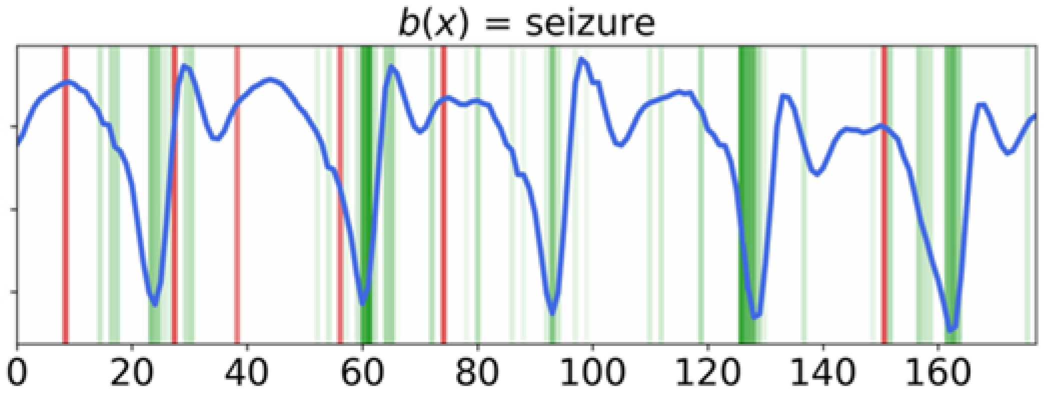


Time Points-based Explanations



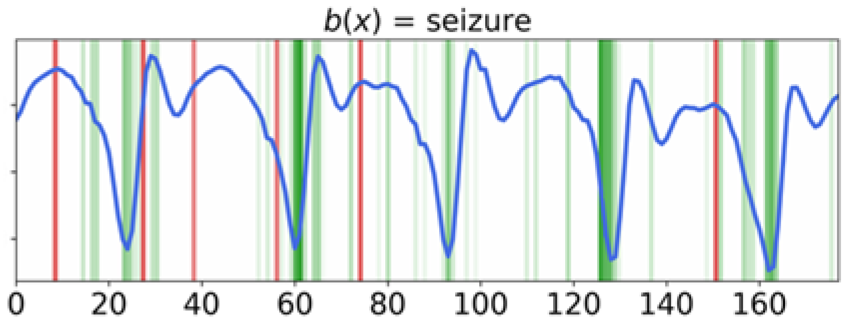
Score-based Explanations

- Score-based explanation methods attribute a local importance to each input variable w.r.t. their contribution towards the predicted.
- The higher is the value in absolute term, the higher is the importance, the closer is to zero the small is the importance for the returned outcome.
- If the score is positive the feature-value has a positive contribution towards the outcome, while if the score is negative the feature-value has a negative contribution.
- Issue: these approaches can require a “default” value to be used as baseline or to simulate the “removal” of a point/subsequence.
- Explanation depends on the value used to replace real values.

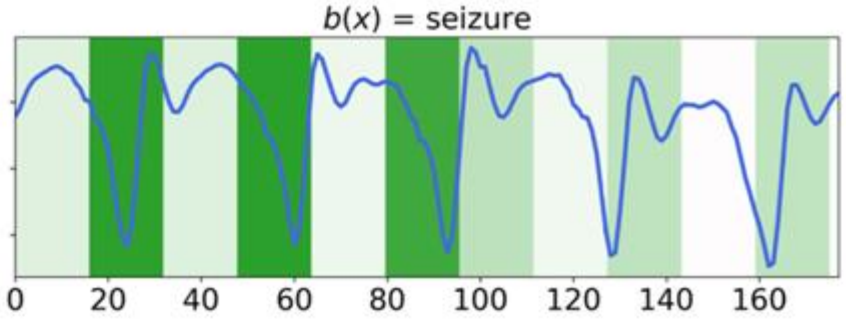


Score-based Explanation Strategies

- **Consider each timestep as a feature:**
  - Works fine for time-independent TS transformation such as DTF, SFA, BOSS, Kernels
  - Assume time independent values if used on raw TS, so minimal misalignments can cause problem, close time stamps can have opposite contribution



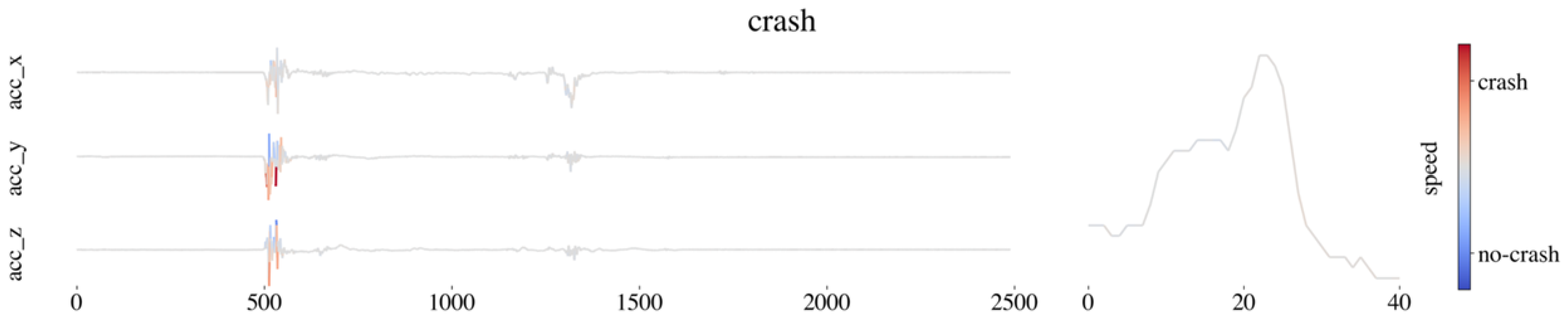
- **Split the TS in subsequences and consider them as features**:
  - Explanation depends on the time-dependent transformation: PAA, SAX, BOP, Shapelets



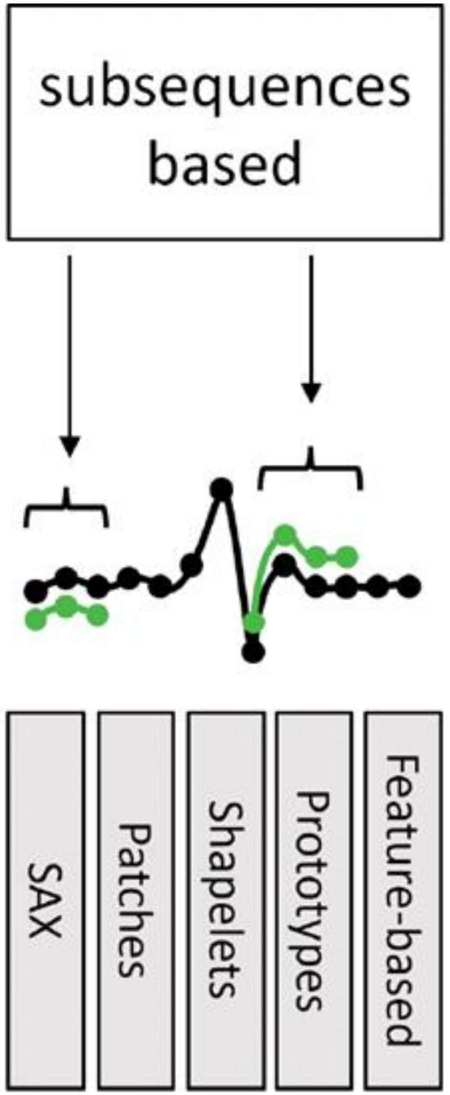
Most Common Score-based TS Explainers

Attribution methods can be model-agnostic or model-specific.

- **SHAP** is available in an inefficient agnostic version, and in multiple efficient model specific version (TreeShap, GradienShap, LinearShap).
- **GradientShap** is a modified version of Integrated Gradients, optimized for neural networks.

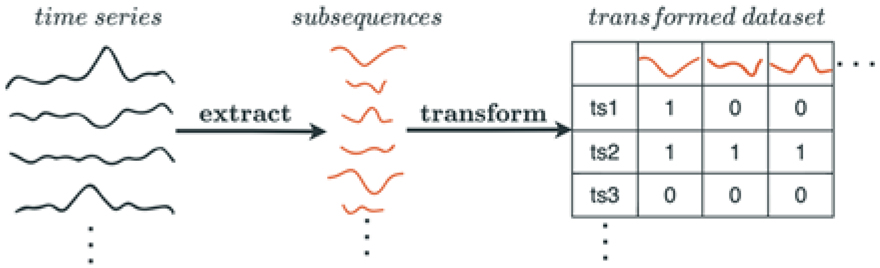


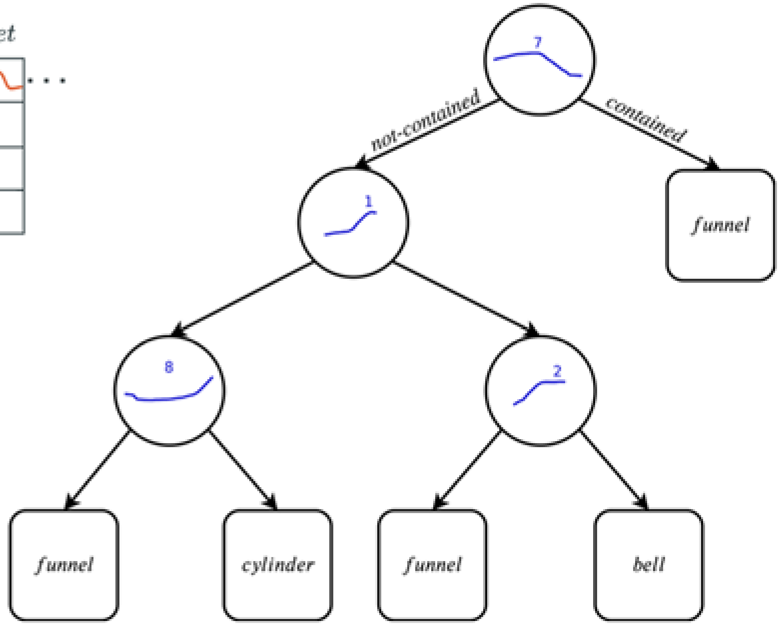
Subsequences-based Explanations

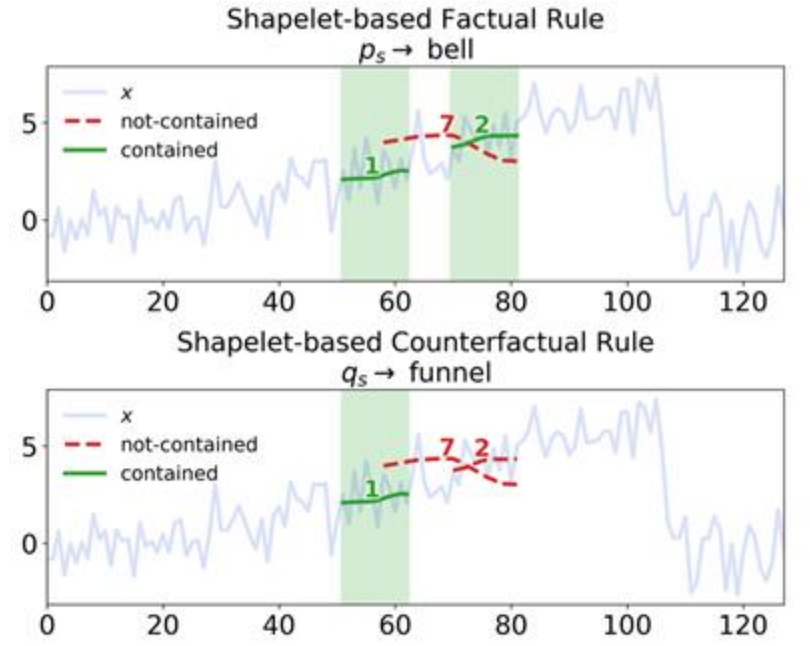


Subsequences-based Explanations

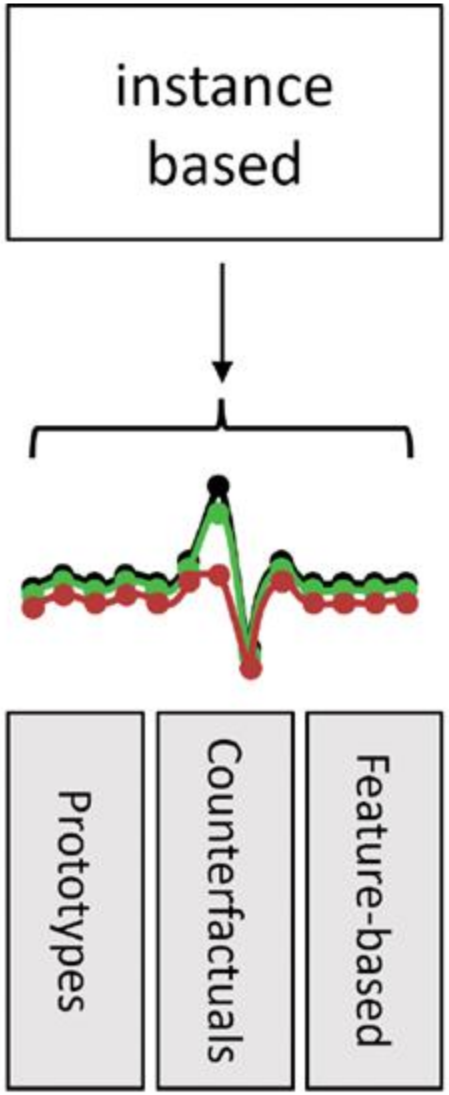
- After having transformed the TS into time-dependent humanly understandable feature describing subsequence that can be referred into the input TS, any interpretable ML approach can be used as it is or as a surrogate to explain another model.





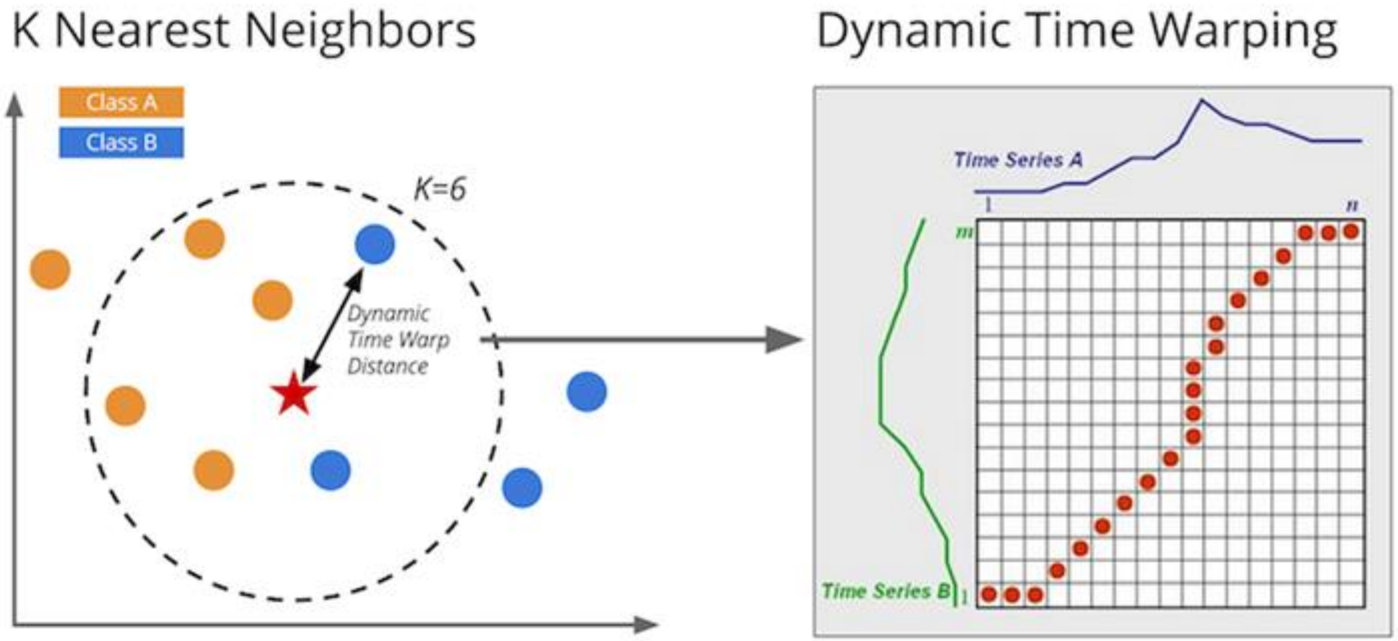


Instance-based Explanations



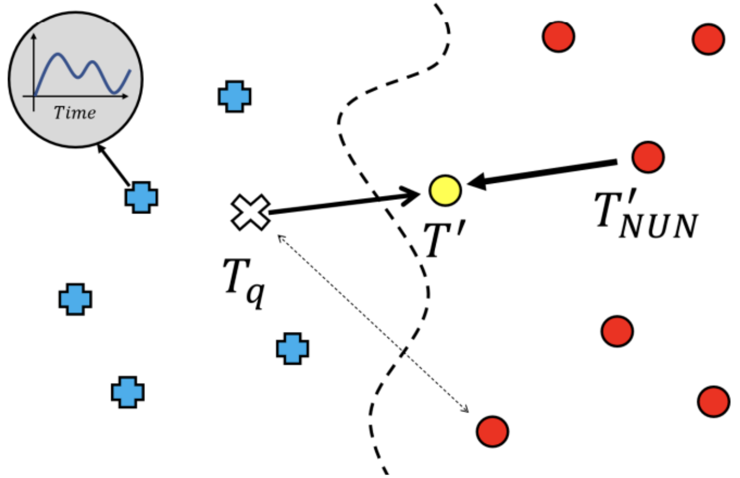
Counterfactuals

- Counterfactual time series show the minimal changes in the input data that lead to a different decision outcome.

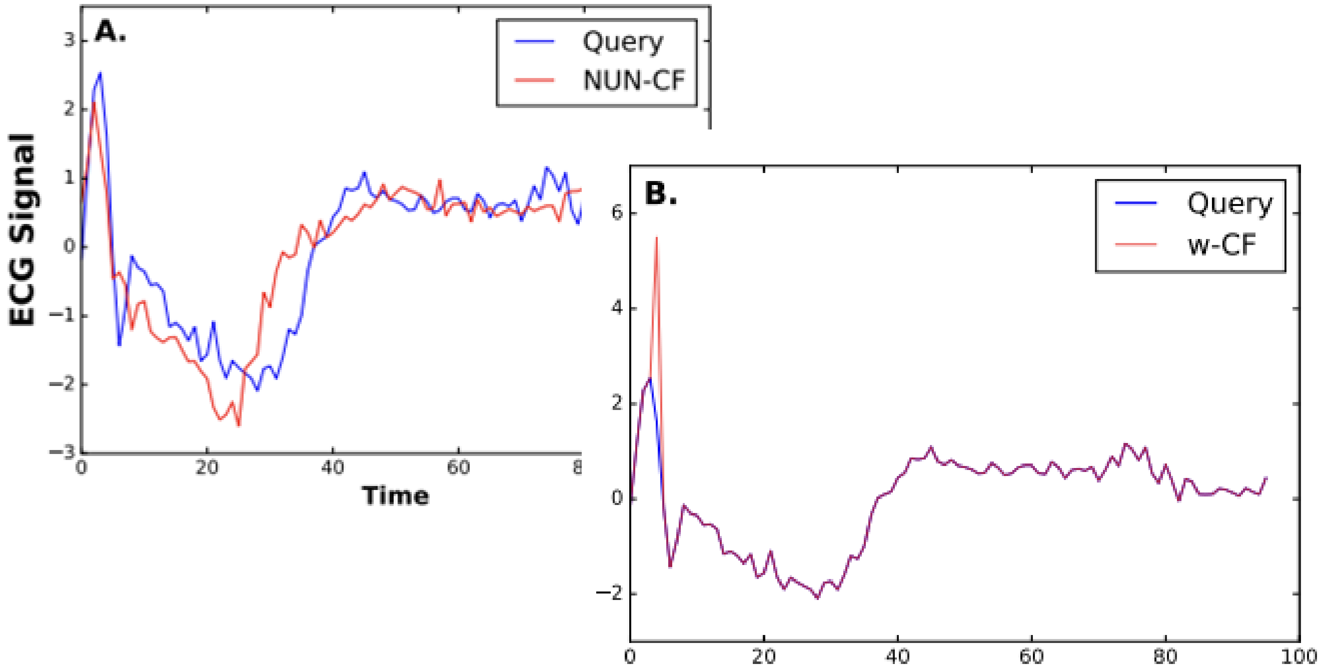


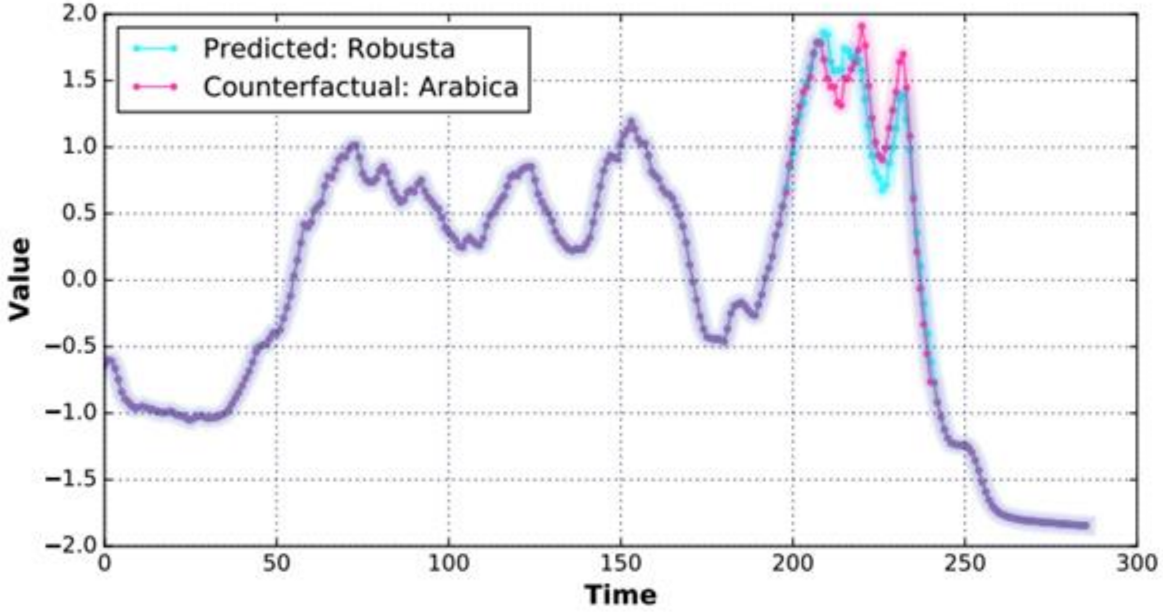
- KNN can be paired with the Euclidean or DTW distance to classify time series.
- To find a counterfactual, it searchers for the closest instance having a different class.

Counterfactuals



- Native Guides builds upon the KNN approach, generating novel counterfactuals, following four identified key properties: proximity, sparsity, plausibility, and diversity.





References

- Convolutional neural networks for time series classification. Bendong Zhao et al. 2017
- Deep Residual Learning for Image Recognition. Kaiming He et al. 2015
- Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. Zhiguang Wang et al 2016
- InceptionTime: Finding AlexNet for Time Series Classification Hassan Ismail Fawaz et al. 2019
- Multivariate LSTM-FCNs for Time Series Classification. Karim et al. 2019
- Tapnet: Multivariate time series classification with attentional prototypical network. Zhang et al. 2020
- ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. Angus Dempster et al. 2019
- MINIROCKET: A Very Fast (Almost) Deterministic Transform for Time Series Classification. Angus Dempster et al. 2021
- MultiRocket: Multiple pooling operators and transformations for fast and effective time series classification. Chang Wei Tan et al. 2021
- Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles. Jason Lines et al. 2018.
- TS-CHIEF: A Scalable and Accurate Forest Algorithm for Time Series Classification. Ahmed Shifaz et al. 2019
- HIVE-COTE 2.0: a new meta ensemble for time series classification. Matthew Middlehurst et al. 2021
- Bake off redux: a review and experimental evaluation of recent time series classification algorithm. Matthew Middlehurst et al. 2024
- A survey of methods for explaining black box models. Riccardo Guidotti et al. 2018.
- Explainable AI for Time Series Classification: A Review, Taxonomy and Research Directions. Andreas Theissler et al. 2022