TSC and eTSC Experiments Scope

Description

The framework allows for 2 types of experiments; Time Series Classification (TSC) and Early Time Series Classification (eTSC).

Components of the Framework:

1. Experiments: (2)
    - TSC: Classify new full length instances based on learning using full length training instances
    - eTSC: Classify new full length instances based on learning using split data
2. Algorithms: (10)
    - 1nn_ed: Euclidean Distance
    - 1nn_dtw: Dynamic Time Warping
    - 1nn_msm: Move-split-Merge (current implementation doesn't achieving same results as published)
    - pforest: Proximity Forest (Takes very long, I will try to reduce the number of trees)
    - tsf: Time Series Forest
    - ls: Learned Shapelets (current implementation doesn't achieving same results as published)
    - st: Contracted Shapelet Transformation (A simplified version of the original but close results)
    - weasel
    - cboss: Contracted Boss (A simplified version of the original but close results)
    - inception: Inception Time
3. Datasets (≈90)
    - UEA + UCR archive excluding (FINANCIAL, IMAGE, MOTION, SIMULATED)
    - External Medical Dataset
4. Cross-Validation:
    - False: Learn using the provided training dataset and measure performance on provided testing dataset
    - True: Learn using 5-fold cross validation on training dataset and a random search for hyper-parameters tuning

5. n_iterations: default is 50
    - the maximum number of iterations that will be used for hyper-parameters optimization
    - If the space of hyper-parameters (Total number of combinations for all possible values for all parameters together) is less than 50 a grid search is done
    - Example: classifier has 2 hyper-parameters A and B. A= {a1, a2, a3} and b= {b1, b2}. The total space is 6 [(a1, b1), (a2, b1)... (a3, b2)], then all 6 combinations will be tried.
    - If the space of hyper-parameters is larger than 50, only 50 iterations are carried out by random selection.
6. Scoring function: default is 'balanced accuracy'
    - The scoring function that will be used to calculate the classification score of a model
    - For TSC this is used to measure performance of a model
    - For eTSC the value is combined with earliness in a harmonic mean equation
7. Splits: default value is 20
    - The number of splits that will be done to the data in the eTSC experiment. Data is divided into equal chunks
    - 20 splits = chunks of 5%
    - In cases of Multivariate data, the splits are applied to all dimensions
    - For each Split, all models are trained on the split data and tested on the full length data.
    - Performance is calculated and stored for each split for each model
8. From Beginning:
    - True: split data is revealed to the classifier starting from the first chunk
    - False: split data is revealed to the classifier starting from the last chunk

TSC Setup:

- Always 10 algorithms will be used
- Choose between default split and cross validation
- Splits option is not applicable
- From Beginning option is not applicable

- Cost of running TSC with default split:
    - 10 Algorithms * 90 Datasets = 900 runs

- Cost of running TSC with cross validation:

- 10 Algorithms * 90 Datasets * 50 n_iterations = 45,000 runs (might be less because not all algorithms require 50 iterations)

eTSC Setup:

- Always 8 algorithms will be used, 1nn_ed and 1nn_dtw are excluded because they cannot handle comparing instances of different lengths
- Choose between default split and cross validation
- Choose number of splits
- Choose From Beginning option

- Cost of running eTSC with default split and 20 splits:
  - 8 Algorithms * 90 Datasets * 20 splits = 14,400 runs

- Cost of running eTSC with cross validation and 20 splits:
  - 8 Algorithms * 90 Datasets * 50 n_iterations * 20 splits = 720,000 runs (might be less because not all algorithms require 50 iterations)

Notes:

- Many papers depended on using the default split of data
- Splits and n_iterations are configurable so their values can be decreased to reduce needed runs
- Multivariate classification technique applied uses columnEnsemble which fits a classifier per dimension. This doesn't increase the number of runs but will consumes more resources.
- Number of Datasets and Algorithms can also decrease the number of needed runs.