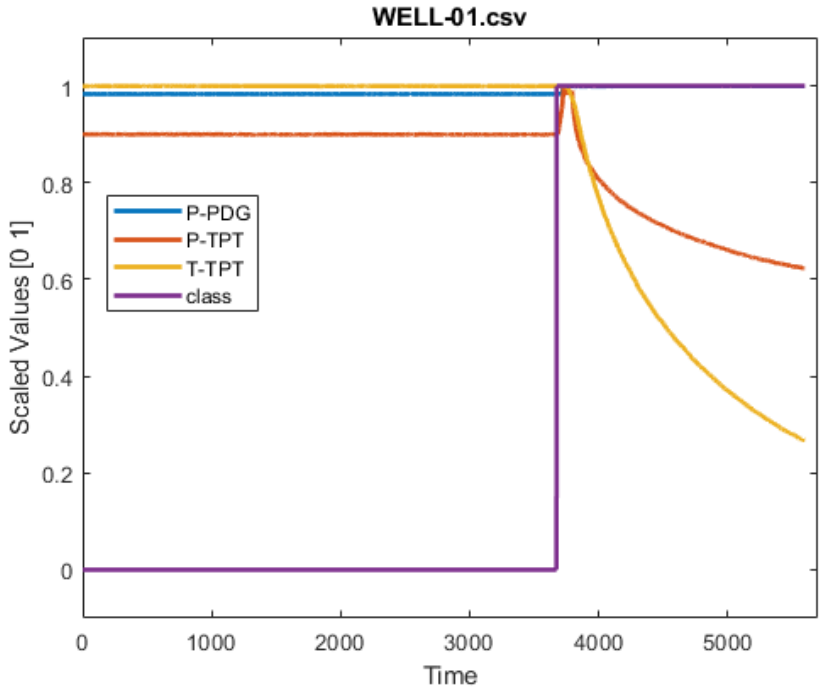
2nd List of Time Series – Part 2
2023-1
Deadline: 16/07/2023
Individual List

Do not use ready-made Matlab and Python functions, except when informed on the question. Program your functions and present them on the report. You can use pandas, numpy, and matplotlib in Python. Include the codes and obtained images in the report. Write the report in Portuguese or English. Send the report through the Google Classroom of the discipline.

3W Dataset is a public multivariate time series with rare undesirable real events in oil wells (https://github.com/ricardovvargas/3w_dataset). This dataset is composed of 1,984 CSV files, where each file represents one instance, which can be a real, simulated, or hand-drawn instance. There are one observation per line and one variable per column. Columns are separated by commas and decimals are separated by periods. The first column contains timestamps, the last one reveals the observations' labels, and the other columns are the Multivariate Time Series (MTS) (i.e. the instance itself).

For questions 4 and 5, there are 7 instances (5 simulated (SIM csv files) and 2 real (WELL csv files) instances) from the fault event “Spurious Closure of DHSV”. Each instance has the same number of columns: Timestamp, P-PDG, P-TPT, T-TPT, class, where Timestamp is the time of the observation (in seconds), P-PDG, P-TPT, and T-TPT are the variables of the process, and class is the label of each timestamp. Each instance begins in a normality condition (class 0) and, after some time, it evolves into a fault condition (class 2). The figure below shows the scaled values between 0 and 1 of P-PDG, P-TPT, T-TPT, and class of the WELL-01.csv instance.



4) For the real instances, WELL-01.csv and WELL-02.csv, compute for each variable (P-PDG, P-TPT, T-TPT) of each class (0 and 2) the Binary Pattern (BP). For this, for each instance, you will split the multivariate time series into two parts: the data labeled as class 0 and the data labeled as class 2. After, compute for each variable of each class the BP with n = 5 (each histogram will have 16 bins). Show the **normalized histogram** of each variable of each class of each instance (for each instance will have 3 histograms for each class (a total of 12 histograms). Use the subplot function to become the visualization easier. Analyze the obtained histograms and check if it is possible to distinguish the classes by the histograms and if the behavior of the classes is consistent in the two instances. Comment on your observations.

5) Use kNN (k-nearest neighbors) and the simulated instances to classify the real ones. For each simulated instance, use a sliding window of length equal to 100 (without overlapping) in each variable (P-PDG, P-TPT, T-TPT) to compute the features: mean, standard deviation, mean increase, mean decrease, standard deviation differences, and mean absolute differences (each data window will be represented by a vector of 6x3 = 18 features). For each vector, you must associate the most predominant class in the data window. These vectors will be the training set of the kNN classifier. Perform the same procedure in real instances to obtain their feature vectors. These vectors will be the test set. Standardize the data (training and test sets), that is, apply for each element of each vector the equation below, where the mean and standard deviation are obtained from the training set for each feature. Use kNN, with k = 1, to classify the test set. Evaluate the classification performance using the accuracy metric. You can use the sklearn.neighbors.KNeighborsClassifier function in Python, or fitcknn in Matlab.

$$\tilde{x} = \frac{x - \mu}{\sigma}$$