Control Engineering
Hof University
Alfons Goppel Platz 1
95028 Hof

Hof University
University of Applied Sciences

Lab Sessions Condition Monitoring and Predictive Maintenance

Lab PMCM02
Regression, Classification

Prof. Dr.-Ing. V.Plenk

Version 0.3

1 Production Plant Data for Condition Monitoring

[1] provides data for 8 run-to-failure experiments.

The data set contains a time stamp and 25 signals from 8 different components of the device under test. For example, L1 could be the current of a stepper motor and L2 the speed of that stepper motor. The data provides several characteristics per component. Ax, Bx and Cx represent one component each. Lx represents the other 5 components: L1+L2, L3+L6, L4+L5, L7+L8 and L9+L10.

There are several runs in the data set. Each letter describes one run. The runs for C7 and C13 are split into two files because these runs had a short break. There is no information about why. It could just be a short check. The break is about 100 timestamps long.

The 8 runs show a progression from the starting state (new components) to one or more components being changed (component not working properly anymore), i.e. from “young” / fully functional to “old” / broken. The exact timestamp of the failure is not known, but it must be in the last observations. You can take the last 500 observations if you want to be sure that you have the failure.

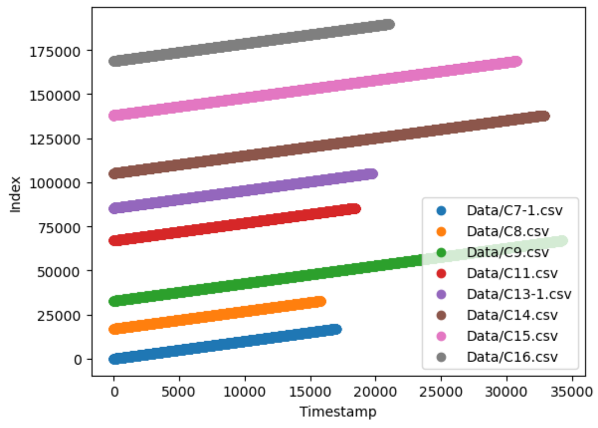

Figure 1.1: Timestamps in the CSV-files

2 Tasks

Task 1: Upload the CSV-files
Upload the CSV-Files to your instance of Jupyter-Labs on the iisys-server!

Task 2: Read the CSV-files
Create a Jupyter notebook to load the CSV-Files.
Look at the column-names and check whether the features were loaded correctly!

Task 3: Eliminate NAN-Values
Check the loaded data for NAN-Values and print a list of the `Timestamp` and the feature names for each occurence of a NAN-Value!
Drop rows with NAN-Values!

Task 4: Calculate your own target vectors
Assume that the `Timestamp` in each CSV file is proportional to the wear of the component. Based on this assumption, calculate your own target vectors!
One that contains the "lifetime" of the device from 0 % ("young" / fully functional) to 100 % ("old" / broken) as a continuous variable to be used as a regression target.
One with for discrete age classes from "new" through "low-med", "high-med" to "old" – in 25 % intervals, i.e. 0 intervals, i.e. 0 . . . 25 %, 25 . . . 50 %, 50 . . . 75 % and 75 to 100 % – as the classification target.
Note that each CSV file covers a different range of timestamps. This means that each run of the device will have a different total lifetime in units of `Timestamp`.
Create a plot to show the progression of the new target variables over time as a curve!

Task 5: Scoring metrics
Define *one* appropriate numerical scoring metric for classification and *one* for regression models! (e.g. recall, RMS of error)
Define a threshold for the minimum acceptable result and justify the threshold!
Define *one* appropriate graphical scoring metric for classification and *one* for regression models! (e.g.: Confusion Matrix, Plot Predicted vs. True Value)
Define what the graph should look like for the minimum acceptable result!

Task 6: Split the data into training and test data
Discuss different alternatives for dividing the data into training and test data! (e.g: Random k-fold split, stratified split, split based on `timestamp`, split based on target value)
Compare these alternatives in terms of “difficulty” for your method and “relevance” for the for the intended application!

Task 7: Train and evaluate two models
Train at least one model for classification and one for regression! (E.g: `KKNeighborsClassifier` and `linear_model.Ridge()` ...)
Try at least two approaches to split the data into training and test data!
Score and rank your models according to your scoring metric!

Bibliography

[1] inIT, Artificial I.: *Production Plant Data for Condition Monitoring*. https://www.kaggle.com/datasets/inIT-OWL/production-plant-data-for-condition-monitoring. Version: 08 2024