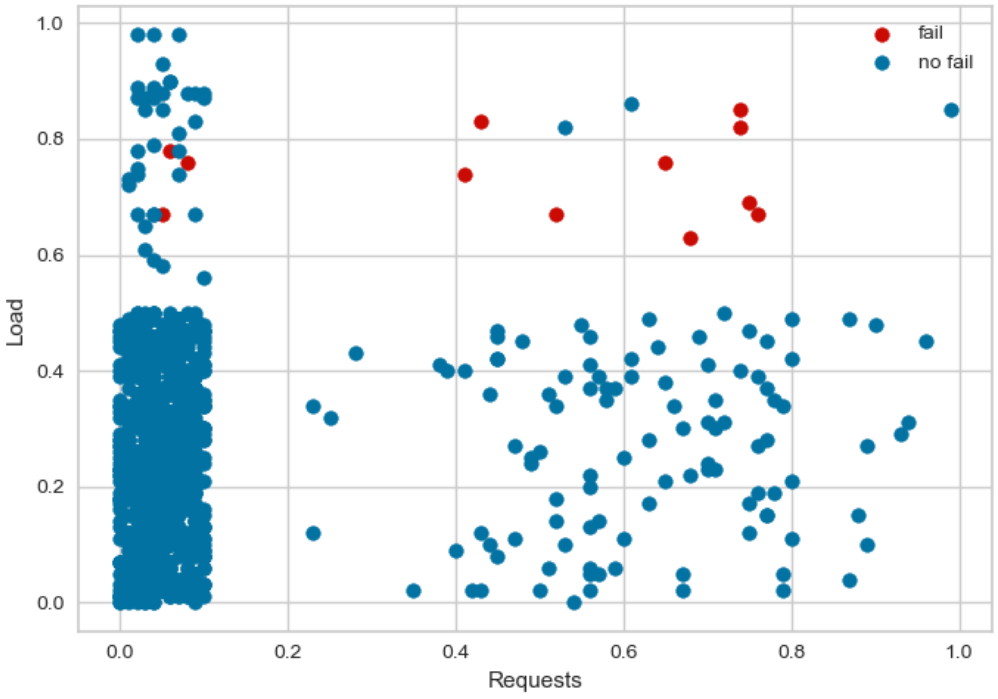
Project 1 – Prediction using NN and Fuzzy Systems

This project's objective is to use previously learnt methods and explore new techniques to make predictions on a dataset. This dataset represents the workings of an IoT gateway device and its failures. The dataset features data on the gateway's requests and processor load, and if it failed or not.

Each record corresponds to a 20-minute period of operation of the device that contains two inputs and one output:

- Input 1: The normalized number of requests received by the device during the 20 min period.
- Input 2: The processor average load during that period.
- Output: Normal Operation (0) or Crash (1).

Firstly, we can plot the data in a scatter plot, separating the events that lead to a fail and the events that did not:



As we can see, the dataset is highly unbalanced since there are very few fail instances. As the guide states, the data is sequential and keeping its sequence is important.

As a first attempt, the data was split into train, test and validation datasets, and it was given to a MLP NN very similar to the last lab. This yielded the following results:

- Training set:
  - Precision: 0.75
  - Recall: 0.75
  - F-measure: [0.9979798 0.75]
  - Confusion Matrix:

    [494 , 1]
    [1 , 3]
- Testing set:
  - Precision: 0
  - Recall: 0
  - F-measure: [0.99193548 0.]
  - Confusion Matrix:

    [246 , 2]
    [2 , 0]

The next test was performed with an LSTM from Keras library. For this each set was split into multiple ordered sequences. The number of lines per sequence was adjusted until the best result was achieved (n=10).

- Training set:
  - Precision: 1
  - Recall: 0.5
  - F-measure: [0.99 0.67]
  - Confusion Matrix:

    [482 , 0]
    [1 , 4]
- Testing set:
  - Precision: 0
  - Recall: 0
  - F-measure: [0.997921 0.]
  - Confusion Matrix:

    [240 , 0]
    [1 , 0]

We can see a little improvement from the last test. But not very significant.

Next, we asked an expert for some guidance on how to approach this problem. The first suggestion was to balance the training set. For this, the oversampling library was used with the oversampling parameter set as “minority”.

This did not result in better figures with Keras. But with scikitlearn’s MLP with the same parameters as before, the results were as follows:

- Training set:
  - Precision: 0.98
  - Recall: 1
  - F-measure: [0.99 0.99]
  - Confusion Matrix:

    [484 , 7]
    [0 , 491]
- Testing set:
  - Precision: 0.2
  - Recall: 1
  - F-measure: [0.99 0.33]
  - Confusion Matrix:

    [245 , 4]
    [0 , 1]

It is noticeable that the results for the testing set are getting better. Changing the sizes of the splits of the sets, we can get more fails on the testing set. This way we can get a better understanding of the performance of the MLP NN.

- Training set:
  - Precision: 0.99
  - Recall: 1
  - F-measure: [0.99 0.99]
  - Confusion Matrix:

    [484 , 6]
    [0 , 491]

- Testing set:
  - Precision: 0.4
  - Recall: 1
  - F-measure: [0.99 0.57]
  - Confusion Matrix:

    [365 , 6]
    [0 , 4]

Overall, this resulted in better results on the training. But there are still improvements to be done.

The next suggestion from the expert was to maintain the order of the sets, but this was already considered from the beginning, so no tests were made in this regard. The next suggestion was to add the previous requests as features for the NN ($Number\_of\_Requests_{t-1}$; $Number\_of\_Requests_{t-2}$). For this, two new columns were created based on the Requests column but shifted one and two positions. The first two lines of the dataset were removed due to the NaNs created by the shifting operation. On this test, the same balancing operation was made to the training set:

- Training set:
  - Precision: 0.89
  - Recall: 1
  - F-measure: [1 0.94]
  - Confusion Matrix:

    [489 , 1]
    [0 , 490]

- Testing set:
  - Precision: 1
  - Recall: 1
  - F-measure: [1 1]
  - Confusion Matrix:

    [248 , 0]
    [0 , 1]

Improvements are seen on the results of the training set. Changing again the sizes of de datasets:

- Training set:
  - Precision: 1
  - Recall: 1
  - F-measure: [1 1]
  - Confusion Matrix:

    [489 , 1]
    [0 , 490]

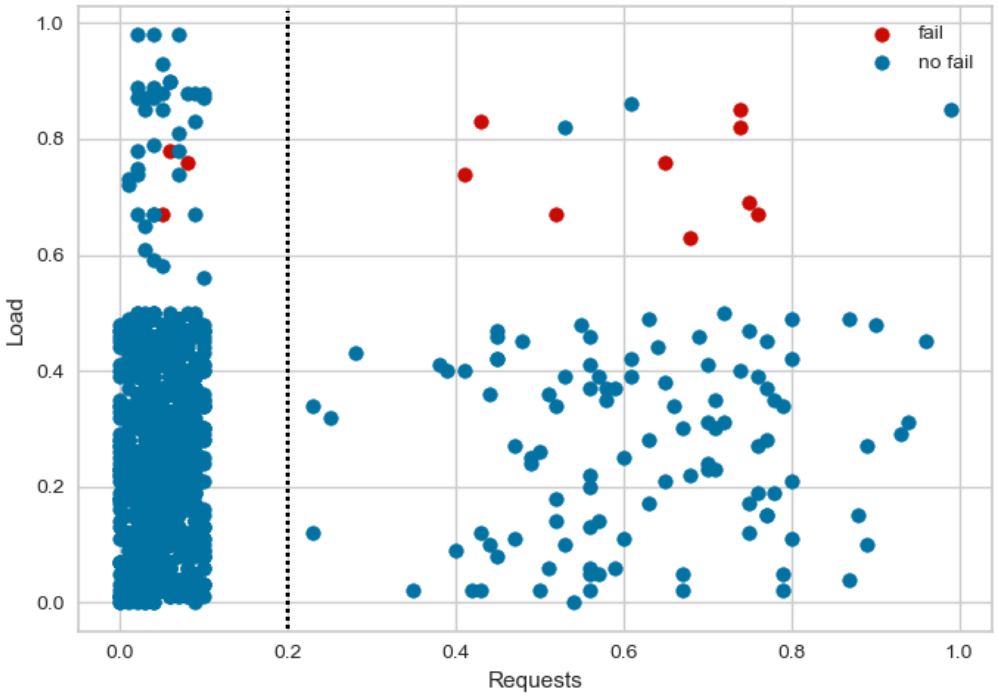
- Testing set:
  - Precision: 0.66
  - Recall: 1
  - F-measure: [1 0.8]
  - Confusion Matrix:

    [368 , 2]
    [0 , 4]

With this we can confirm that the results of the testing set predictions are getting better.

The next expert advised to look into cases where the number of requests is high in periods before the processor load gets noticeably high.

Looking at the scatter plot from the dataset again we can see that there Is a clear distinction from high and low request values. One option would be to create a feature that would indicate if the current event has a high request value or not. We can still see that there are fail occurrences mixed with the non-fail ones, but we cannot forget that there is also sequential information that we cannot see on this plot.



So, for this test, a new feature was created with a binary value indicating that if the current request value is above 0.2.

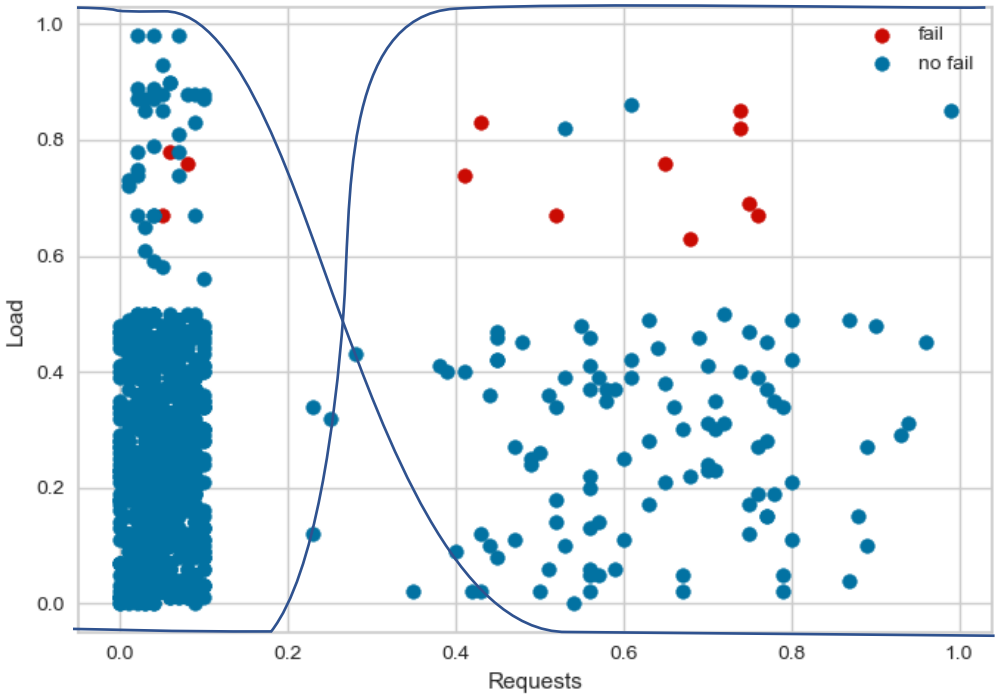
- Training set:
  - Precision: 1
  - Recall: 1
  - F-measure: [1 1]
  - Confusion Matrix:

    [489 , 1]
    [0 , 490]

- Testing set:
  - Precision: 0.8
  - Recall: 1
  - F-measure: [1 0.89]
  - Confusion Matrix:

    [369 , 1]
    [0 , 4]

This greatly improved the results as can be seen for the precision score on the testing set. Keeping with this same methodology, another feature was added to indicate high load, for values higher than 0.55. But this approach did not yield better results. For this dataset it is easy to set a concrete threshold for the request parameter, but a better approach would be to define this with a fuzzy classifier, since the transition is not smooth and the preset value might not work for other datasets from the gateway.



Conclusions

On this project, the analysis of the problem had the help from some experts that gave some suggestion on how to approach the problem and had inside knowledge on the behavior of the gateways. With this the approach was more methodical and structured and the results were getting better as more advices were considered.

Although the approach that seemed more applicable for this problem, the LSTM, was used it did not present very significant results and using an MLP with features that were constructed from previous values of the request values was proven produce better results.

Adding a feature that indicated the instances where the requests were too high, improved the model. The idea of adding a fuzzy classifier to choose whether the requests were too high was not implemented due to time restraints but would make the model more robust and would work better on other datasets.