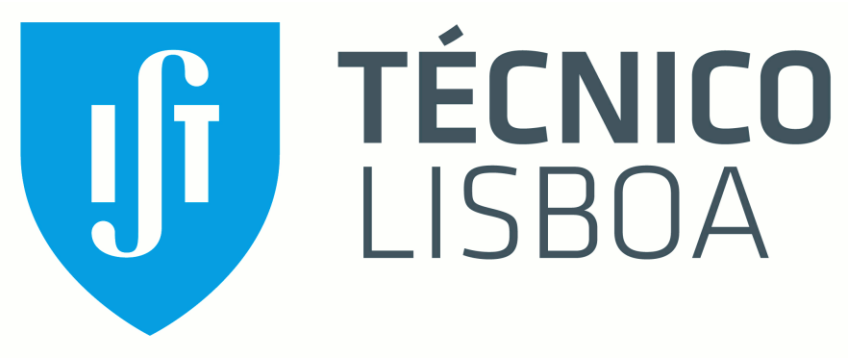


Applied Computational Intelligence

1st Semester 2022/2023

Group: 28

João Pedro Andrade Gonçalves nº 85211

jpedrogoncalves@tecnico.ulisboa.pt

Marco da Cruz da Mata nº 84297

marcocmata@tecnico.ulisboapt

Table of Contents

Part 1: Classification using NN

For the first part of this project, we were asked to develop a Neural Network to solve a classification problem. The task was to measure how many persons were inside an experimental lab, equipped with sensors during the Covid-19 pandemic, and was divided in two objectives:

a) Creating a binary model to detect if the number of people exceeds the 2 persons limitation.
b) Creating a multi-class model to detect exactly how many persons are inside the lab.

Data preparation

The dataset given had 9 features: a CO2 sensor (MH-Z14A), two digital infrared motion sensors (PIR), three light sensors (BH1750) and three temperature sensors (LMT84LP). It contained 10129 values and for each one the 9 features, time, date and number of persons on the room was given.

We started by removing obvious outliers. They consisted in temperatures that were below or equal to zero and values that were corrupted (not a number). For the light sensors the maximum value for each one was around 500, but we noticed a couple of values way beyond this mark. We decided to remove all the values above 1000.

Furthermore, sometimes during the day some light sensors malfunctioned and registered 0. In order to remove these outliers we had to convert the given times to seconds and created a condition to remove from the dataset rows where during the day (between 8am and 8pm) only one sensor didn't register a positive value.

Another commonly used technique for outlier detection in temporal datasets is the interquartile range (IQR). However, in this case our values are very concentrated on the extremes, mostly on the minimums (positively skewed) and using this technique we would lose a lot of valuable data. In fact, when testing the multi-class model after applying the IQR method, our model would only recognize 2 classes.

Construction of the models

In order to correctly train and test our model we had to separate our dataset, so our test set does not contain biased data (used to develop our model). We saved 10% of our dataset, chosen randomly, to later test our model. The remaining values were used to train and validate our model.

We decided to use k-fold cross-validation, with 4 folds. This means the training set will be divided in 4, with one rotating fold being used to test our data while the rest of the folds are training. This will result in less data being wasted when developing our model. After this process is finished, we tested our model on the test set, with data that wasn't previously seen by the model, avoiding overfitting.

The goal with our model was to achieve the best results with as less complexity as possible. We started by choosing the size and number of hidden layers. The size of the hidden layer should be bigger than the size of the input layer (9 in our case) and smaller than two times our input layer (18 in our case). We came to the conclusion that with size 10, our results were very satisfactory, with the improvements from making the hidden layer bigger than this being irrelevant. Despite our results being good with one hidden layer, they fluctuated a bit and some of the times would underperform. For this reason, we chose to have 2 hidden layers, making the performance of our model more stable. With 3 hidden layers our model started to overfit, performing very well in training but worse when testing.

To choose our hyper-parameters we did an average of multiple tests for each of them finding out which had the best results. The tested accuracy for the binary and multi-class models can be found in Table 1 and Table 2.

	binary	**multi-class**
lbfgs	0.988	0.980
sgd	0.940	0.880
adam	0.985	0.983

Table 1 – Solver for weight optimization

Despite the *lbfgs* solver having very good results it would not converge, even when raising significantly the number of iterations and the size of the hidden layers. For that reason, and to keep the complexity as low as possible, we opted for the *adam* solver.

	binary	**multi-class**
identity	0.940	0.927
logistic	0.982	0.956
tanh	0.980	0.971
relu	0.985	0.983

Table 2 – Activation function for hidden layers

Regarding the activation function, *relu* had by far the best and most consistent results.

Results

Our final model results, with the *ReLU* activation function and *adam* optimizer, can be found in Table 3 and Table 4.

	Precision	Recall	F1
0 persons	0.994	0.990	
1 person	0.882	0.918	
Macro	0.938	0.954	0.946

Table 3 – Results for the binary model

	Precision	Recall	F1
0 persons	0.990	1	
1 person	1	0.979	
2 persons	1	0.955	
3 persons	0.923	0.896	
Macro	0.978	0.957	0.968

Table 4 – Results for the multiclass model

Overall, our results were very satisfactory, with the Macro Precision, Recall and F1 being close or above 95% in both models. Since our dataset was highly unbalanced, it is natural that our precision and recall for 1 person in the binary model would drop slightly, when compared to 0 persons, the class where the majority of the data was concentrated. It is worth noting that for all classes the precision and recall are very similar, so almost all of the predicted positive values were indeed positive and almost all the truly positive values were predicted to be positive.

Part 2: Classification using Fuzzy Systems

For the second part of the project, using the same dataset, we were asked to develop a Fuzzy System to detect if the room's capacity had been exceeded.

Data preparation

Our approach when dealing with outliers for the second part was identical to the one used in the first one, since our given dataset was exactly the same.

To create our Fuzzy System, we had to reduce the number of features down to a manageable number. To achieve this, we focused on the features we considered most important to solving the problem and creating new features that would combine the data of comparable features, or present that data in a more useful manner. Therefore, we decided on the following features:

- **CO2 Level change** – The changes to the CO2 level inside the room are entirely dependent on how many people are in the room. The more people inside, the higher the change, and if there are no people inside the room, we expect the CO2 levels to decrease until they reach a point of stability. There are certain corner cases, such as there being 3 people in the room, and one leaving, causing the CO2 levels to decrease slightly even though there are 2 people in the room, but the other features should cover this imprecision, and we hoped to lessen this effect by using the average of the CO2 level change over the last 10 points
- **Average temperature** – The more people there are in the room, the higher the temperature inside will be. We averaged the readings from the 3 sensors and used that as our second feature
- **Average light** – The more people there are inside the room, the higher the readings on the light sensors. We noticed some unexpected behavior from the S1 light sensor, so we discarded it and used the average readings of the S2 and S3 light sensors as our third feature

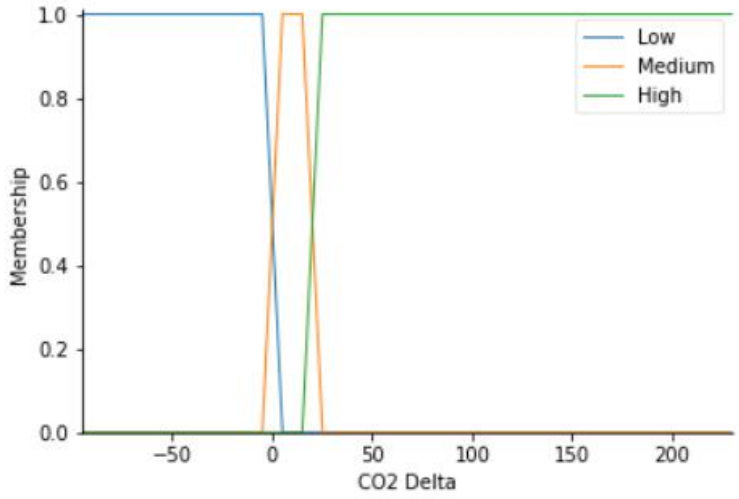
Construction of the model

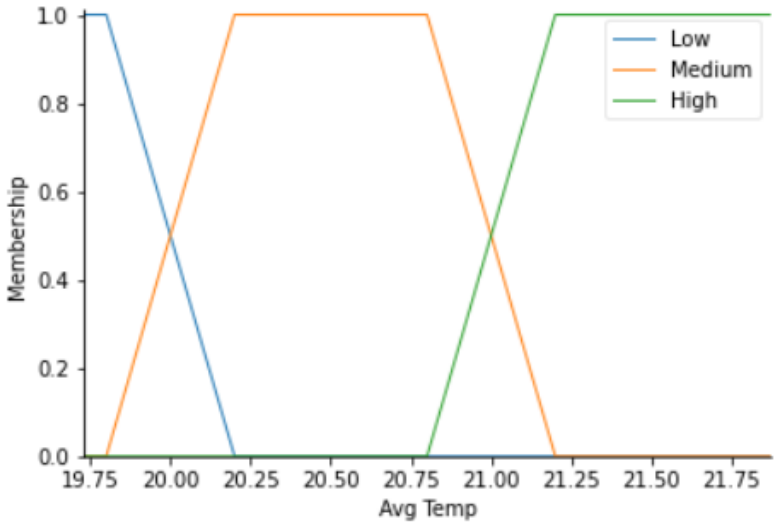
After gathering our important features, we divided each feature into a range of values that we would categorize as “Low”, “Medium”, or “High” levels to use in our Fuzzy System. The following table shows what we considered for each category:

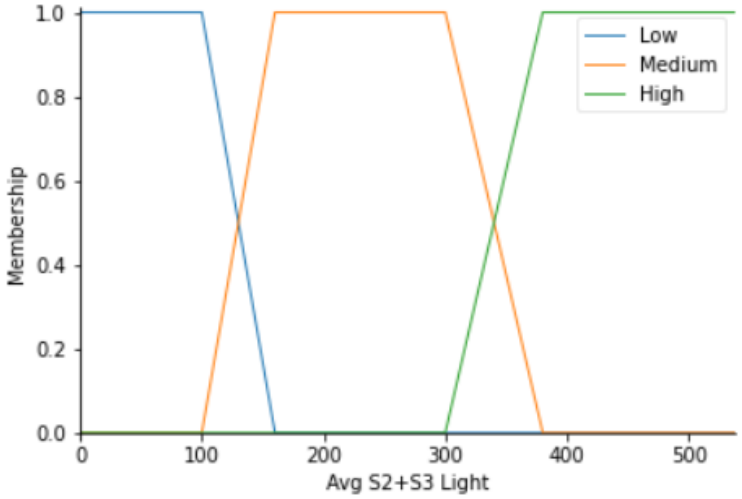
	Low	Medium	High
CO2 moving delta	< 0	0 - 20	> 20
Average Temperature	< 20	20 - 21	> 21
Average Light	0 - 130	130 - 340	> 340

Table 5 – Limits for the Fuzzy System’s features

Based on these conditions, we defined membership functions for the inputs, describing each range as a trapezoid, resulting in the membership functions graphed below:







Then, we described a set of 8 rules that would cover every possible combination of inputs and map them to an output. Those rules were:

Rule 1: IF CO2 moving delta is LOW, THEN the output is 2 or less

Rule 2: IF CO2 moving delta is MEDIUM AND the average temperature is LOW, THEN the output is 2 or less

Rule 3: IF CO2 moving delta is MEDIUM AND the average temperature is MEDIUM, THEN the output is 2 or less

Rule 4: IF CO2 moving delta is MEDIUM AND the average temperature is HIGH AND the average light reading is LOW, THEN the output is 2 or less

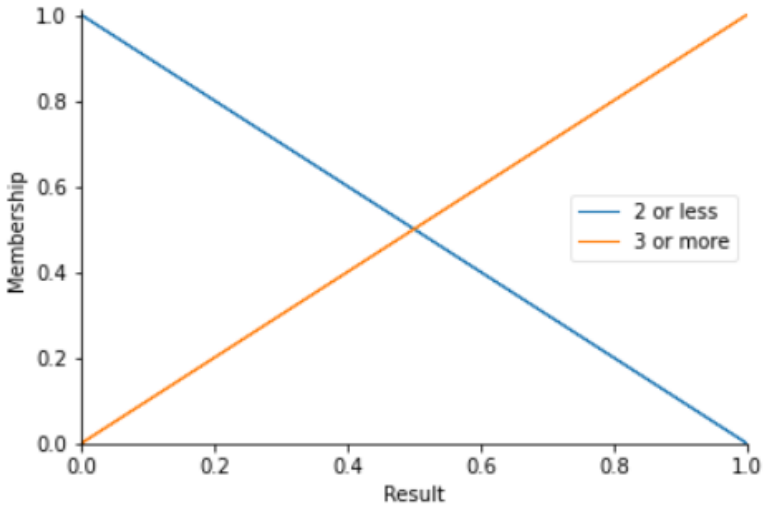
Rule 5: IF CO2 moving delta is MEDIUM AND the average temperature is HIGH AND the average light reading is MEDIUM OR HIGH, THEN the output is 3 or more

Rule 6: IF CO2 moving delta is HIGH AND the average temperature is MEDIUM OR, THEN the output is 3 or more

Rule 7: IF CO2 moving delta is HIGH AND the average temperature is LOW AND the average light reading is LOW OR MEDIUM, THEN the output is 2 or less

Rule 8: IF CO2 moving delta is HIGH AND the average temperature is LOW AND the average light reading is HIGH, THEN the output is 3 or more

Then, we defined the membership function for the output, which is graphed below:



Results

Unfortunately, we weren't successful at getting our Fuzzy System up and running without issues. Initially, we attempted to do it following the steps described previously. However, we encountered a particular error that was challenging to overcome:

```
Crisp output cannot be calculated, likely because the system is too sparse.
Check to make sure this set of input values will activate at least one connected
Term in each Antecedent via the current set of Rules.
```

Upon encountering this error, we did our best to track down its origin. At first, we triple checked if the rules we set in our code covered every possible input combination, and they do. As things stood, nothing seemed to be wrong with the code. The input values should have been mapped to their 'labels' correctly, but they weren't, due to an unknown problem (to us) regarding the definition of the membership functions. We tried different approaches, such as using the *automf* function to define the membership functions, but this would completely ignore the limits we set to define LOW, MEDIUM and HIGH levels for the inputs. That way, the model would never predict that there were 3 people inside the lab room. To avoid this problem, we tried to manipulate the data, such as setting the values for the average temperature to a certain value if it was over that value when it those values were on the

higher end of the spectrum, and doing the same for the lower values, in order to get more predictable results from the *automf* function, but it ended up not fixing the errors.

Afterwards, we built a Neural Network and trained it on the dataset using only the 3 features we used for the Fuzzy System. The results obtained were slightly worse than the ones from the binary model constructed with the original features and can be found in Table 6.

	Precision	**Recall**	**F1**
0 persons	0.981	0.993	
1 person	0.897	0.743	
Macro	0.939	0.868	0.899

Table 6 – Results for NN with same features as the Fuzzy System