TÉCNICO LISBOA

MEEC

Computational Intelligence

Neural Network and Fuzzy Systems Report

Authors:

Bruno Cebola (93030) bruno.m.cebola@tecnico.ulisboa.pt
Rui Abrantes (93176) rui.miguel.abrantes@tecnico.ulisboa.pt

Group 31

2022/2023 – 1st Semester, P1

Contents

1 Objective

During the COVID-19 pandemic, a restriction on the maximum amount of people that could be simultaneously inside a room was imposed by Técnico Lisboa. However, the students that needed to use these rooms had frequent deadlines and often ignored the imposed limits. In order to try and fink a way to track the occupancy of a space, it was created an experimental lab, equipped with different sensors, which had space for three simultaneous students, but was restricted to only two at a time.

The overall objective of this project is to classify the occupancy of this laboratory thought the use of Neural Networks, in order to state how maybe people where on the laboratory at the same time, and Fuzzy Systems, in order to detect if the space was overcrowded, i.e, if there were more than two students at a time.

2 Problem exposition

In the laboratory under analysis were placed 9 sensors: 3 light sensors and 3 temperature sensors at each desk, 2 motion detectors in opposite walls and a CO2 sensor at the center of the room. The data collected by these different sensors, or in some case part of it, correspond to the features used in our Neural Networks and Fuzzy Systems. Note that each sensor recording also had a timestamp, but due to the structure of our solution, which will present later on this report, it was deemed unnecessary for the classification algorithms.

Measurements were acquired over a period of several days and during this period each student manually annotated when entered or left the room. Therefore, the true occupancy was annotated during the measurement period. From these data it is necessary two obtain two outputs: the number of students in the laboratory at each time and if it is overcrowded.

In order to do this, two solutions, one for each problem, will be presented in this report: a Neural Network and Fuzzy System. For both, it will be explained in detail how their structure was defined, which tests were made and pros and cons of each solution.

3 Solution presentation

In order to achieve a final reliable solution capable of answering the questions at hand: **"How many people are in the laboratory?"** and **"Is the laboratory overcrowded?"**, it was necessary to perform several iteration of data analysis and model fine-tuning.

The tests, decisions and conclusions of this iterative process will be explained bellow, with respect to which macro topic they belong to: **Data Observation** on Subsection 3.1, **Neural Networks** on Subsection 3.2 and **Fuzzy Systems** on Subsection 3.3.

3.1 Data Observation

The measurements performed by the sensors to which we have access are considered raw data, i.e, they have never been processed to find errors, missing values or outliers, as it can be seen in Figure 1. This way, no matter which problem is going to be tackled, the pre-processing of the features is a mandatory first step to assure that the data will not affect the outcome negatively. This process was done in two stages: filtering and smoothing.

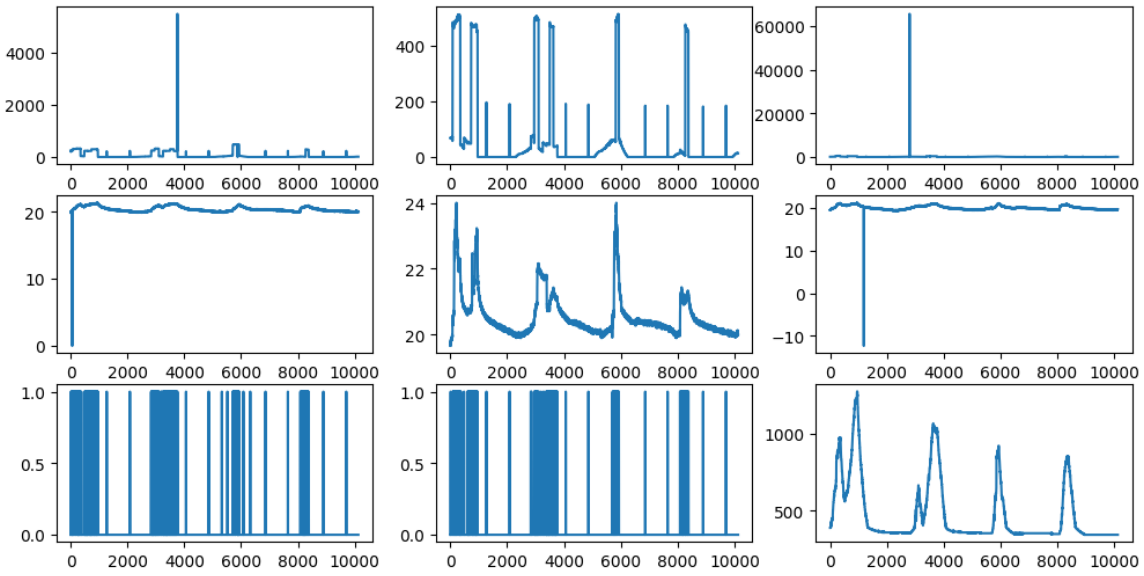


Figure 1: Raw data from the sensors

The chosen process of filtering the data was: removing lines with missing data, z-scoring the features that spawn across a range of values (which are all except the motion sensors) and remove the outliers based on equation (1), where *avg* corresponds to the average value of the feature, *std* to the standard deviation of the feature and the gain is a fine-tuned parameter, which in our case corresponds to 6. In Figure 2 it is possible to see the different features after this filtering process.

$$outliers = \begin{cases} feature <= (avg - std * gain) \\ feature >= (avg + std * gain) \end{cases} \tag{1}$$

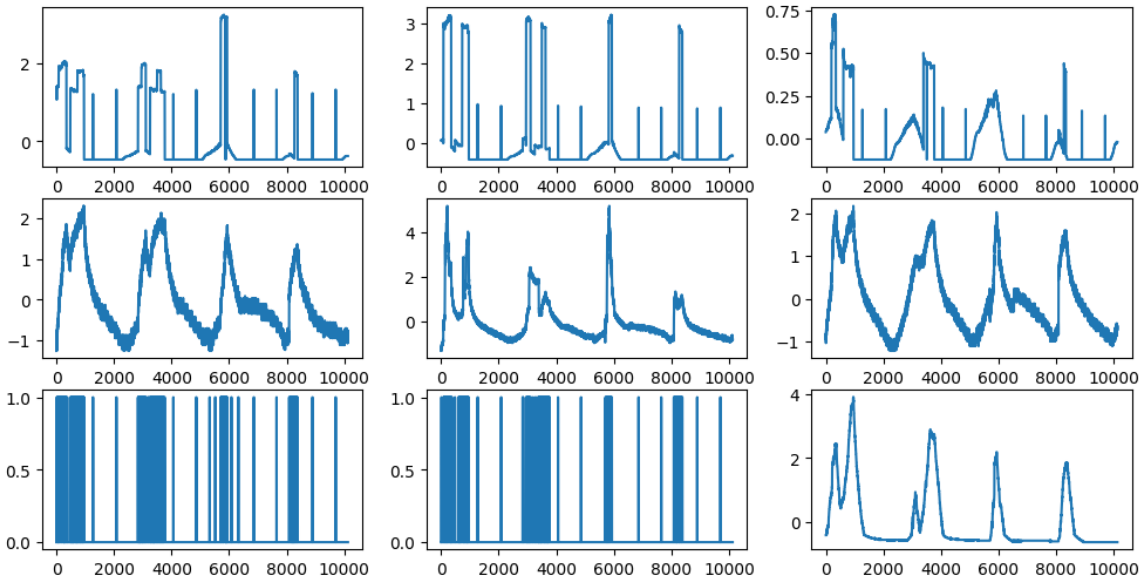


Figure 2: Features after z-scoring and outlier removal

Next, in order to smooth the data, we decided to compute the moving average (only for the classifiers part of the project) of the the features that spawn across a range of values, with a window of 75 values, in order to remove the Gaussian Noise and remove instantaneous peeks. Finally, we apply a min-max algorithm (2) to ensure all data would fall inside a fixed range. The final processed features are shown in Figure 3.

$$feature = \frac{feature - feature.min()}{feature.max() - feature.min()} \tag{2}$$

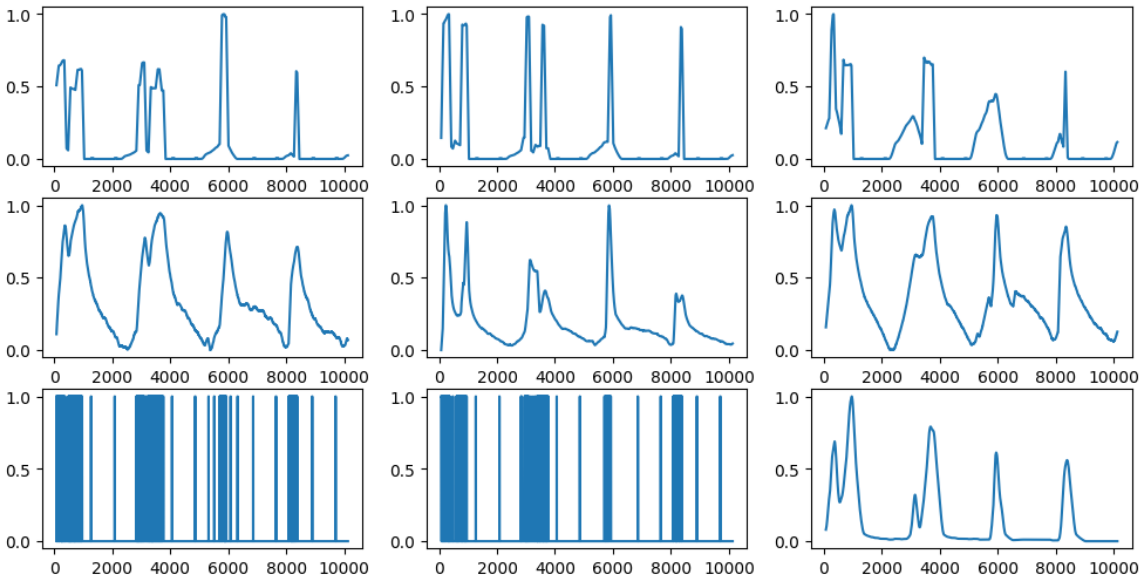


Figure 3: Final data after processing

It is important to notice that for the temperature, all three sensors have the same measurement's format over the time, which makes sense considering that, event though they are placed one at each desk, the temperature they measure is the temperature of the laboratory. This way, the only reason to have three sensors tracking the same variable is due to redundancy and reassurance, as with a ternary system if one of the sensors fails or collects corrupted data, then it is possible to use the other two to detect and correct this anomalies or even ignore the faulty sensor. In our case, since the shape of the measurements is similar, as previously stated, we decide to keep all three sensors and input features.

To finish the data pre-processing and analysis, we decided to plot an histogram to check how the data was distributed along the four classes: 0, 1, 2 and 3. As it can be seen in the Figure 4, the class that corresponding to an occupancy of 0 people clearly outbalances the other classes, which was to be expected if we take in account that students have a studying routine that is more intense during the day than at night and that the sensors were continuously capturing new data.

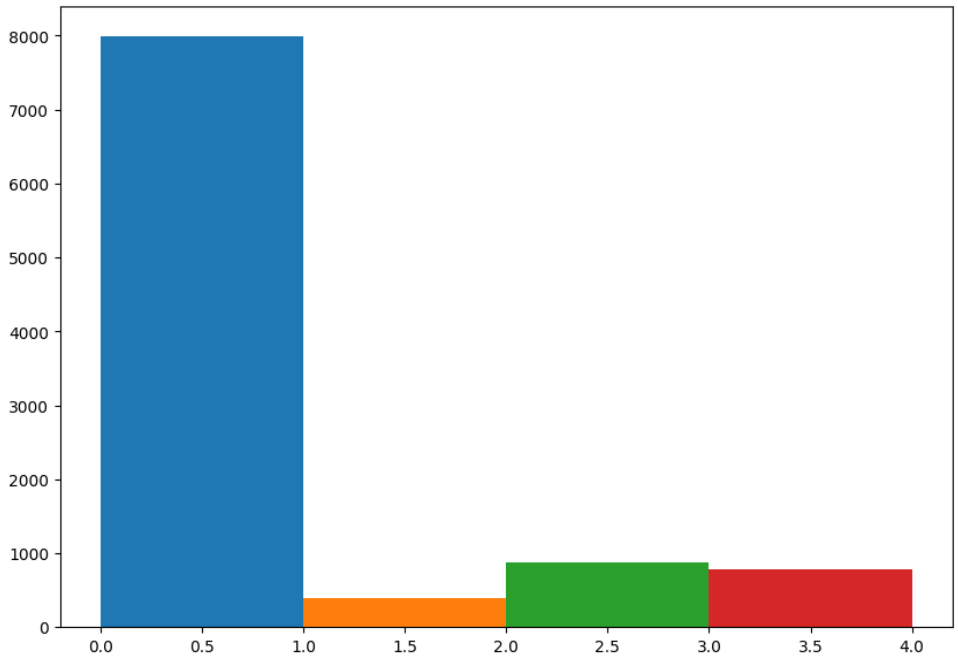


Figure 4: Laboratory occupancy distribution over class [0, 1, 2, 3]

Before moving on to start testing Neural Networks or Fuzzy Systems it was still necessary to decide which data would be used to train the models and which would be used to test it. This way a random 80/20 split was performed in order for the tests to have independent values from the trains and ensure that no bias would b inserted into the final metrics.

3.2 Neural Networks

To try to predict the number of people inside the laboratory at certain moment, the use of classifiers is highly recommended. However, choosing the right one to classify the data isn't an easy task because it depends intimately on the chosen parameters and the data itself.

This way, to start, our group decided to try various models with static parameters, in order to make visual direct comparisons between them. The models tried, with resource to *sci-kit learn*, were: the Nearest Neighbors, the Linear Support Vector Machine (also known as SVM), the RBF SVM, the Decision Tree, the Random Forest, the Naive Bayes and the MLP. In Figure 5 it is possible to observe the confusion matrices of the tested classifiers.

In Table 1 it is possible to see the evaluation metrics of all the tested models, sorted according to their recall. With this, it is possible to see that our worst recall was corresponds to the Linear SVM, with a value of 65%, which makes sense since considering that the problem at hand is not linear. On the other hand, the best recall, 96%, corresponds to the Nearest Neighbors algorithm, which suggests that the elements of each class group themselves in very distinctive clusters. This outcome is really important, because it indicates that although the Neural Net recall has been low (74%), if its parameters are chosen correctly than it will be able to correctly classify the data.

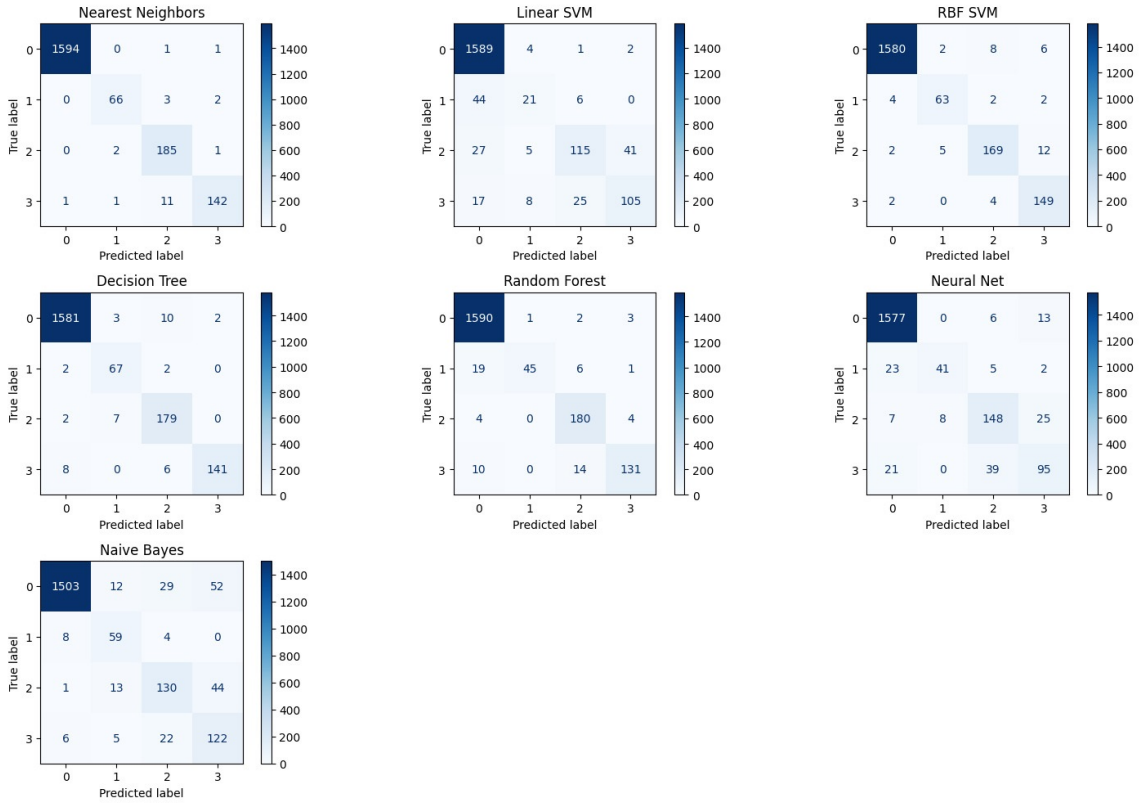


Figure 5: Confusion matrices of the tested classifiers

Model	Recall	Accuracy	Precision
Nearest Neighbors	0.96	0.99	0.96
Decision Tree	0.95	0.98	0.94
RBF SVM	0.93	0.98	0.93
Random Forest	0.86	0.97	0.95
Naive Bayes	0.81	0.90	0.73
Neural Net	0.74	0.93	0.81
Linear SVM	0.65	0.91	0.75

Table 1: Evaluation metrics for the tested models

In order to find the best parameters for our MLP classifier, we decided to do a grid search for the values listed bellow.

- **activations functions**: "relu", "tanh", "logistic"
- **hidden layer sizes**: (1), (18), (100)
- **random states**: 0, 2
- **max iterations**: 100, 200
- **alphas**: 0.0001, 0.01, 1

The best combination founded was {relu, (100), 2, 200, 0.0001}, where each entry corresponds to the variable in its position in list above, i.e, relu is the activation function, (100) is the hidden layer size, etc. Below, it is possible to see in Figure 6 the confusion matrix for this classifier and in Table 2 its metrics. It is evident that this method allow us to find a much better classifier, but it was still difficult to guarantee that we were not overfitting and that the the classifier was able to generalize well to unseen data, even thought we used the test set to obtain the metrics presented.

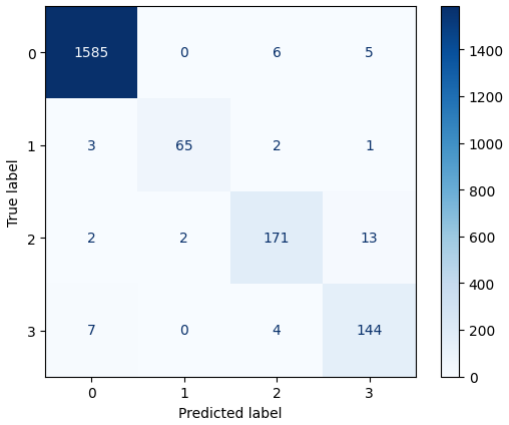


Figure 6: Confusion matrix of the MLP classifier with best parameters

Recall	Accuracy	Precision
0.94	0.98	0.95

Table 2: Evaluation metrics for the MLP classifier with best parameters

To tackle this problem, we decided to perform a k-fold cross-validation train on top of our best MLP. This allowed for the classifier to train using different data every iteration and do a final test with unseen values, guaranteeing that we would not over-fit the model, nor it would be too specific to the data presented to it. In Figure 7 it is possible to see the confusion matrix for this final classifier. Also in Table 3 is is possible to see the metrics.

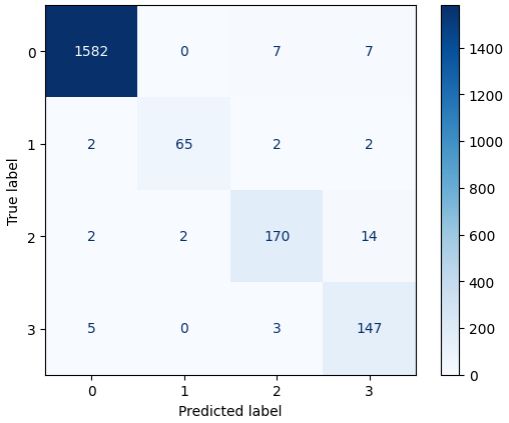


Figure 7: Confusion matrix of our MLP classifier after k-fold

Recall	Accuracy	Precision
0.94	0.98	0.94

Table 3: Evaluation metrics our MLP classifier after k-fold

3.3 Fuzzy System

In order to determine if the laboratory was overcrowded, we resorted to a Fuzzy System due to its capacity of representing intermediate values, such as 'High light', 'Medium light' and 'Low light'. However, a disadvantage of this type of systems is that they get to complicated to construct when there are many features to take into account.

This way, the first step was to determine which features were more important and which feature did not bearer valuable information. After looking at the plots in Figure 3, we came to the conclusion that only the light and motion sensors were needed to conclude if the laboratory was or not overcrowded. With these minimum amount of features we can reduce the amount of rules need, the probability of errors and expedite the creation of the fuzzy system, all while making possible to reach the right output.

To be clear, the CO_2 sensor was dismissed because the data presents a delay in time, due to the fact that CO_2 only starts to accumulate a while after someone enters the room and only starts to fade after that someone leaves. Regarding the temperature information, this not also follows the same pattern as the CO_2, but it is also highly impacted by the outside environmental conditions, making it an unpredictable measure to use with fuzzy logic.

Note that, for the light sensors, since they are placed at each desk and usually this sensors are highly directional, out opinion was that the external conditions would not be as impactful as for the temperature. This is why, even though they present similar problems, we decided to keep the light sensors as input for our fuzzy system.

Having our variables selected, it was now time to define the structure of our system. In Figure 8 it is possible to see how we decided to construct it. We decided to go along a "divide and conquer" strategy, because it would not only make it easier to create the rules, but would also allow to introduce or remove inputs, if needed.

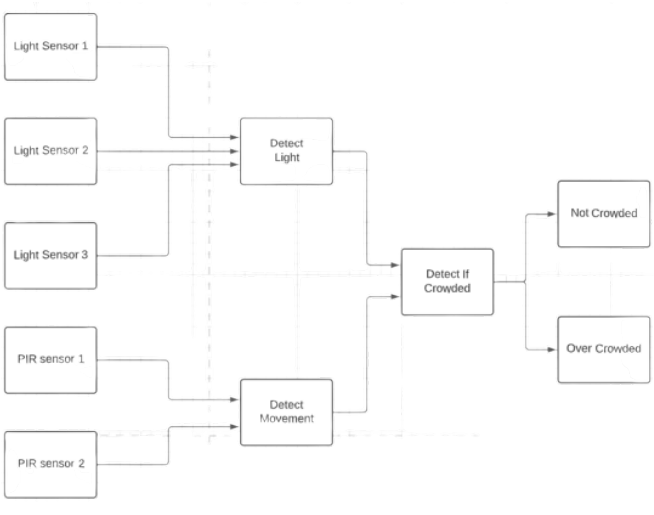


Figure 8: Fuzzy System Design

For the light input we used three linguistic terms: "High", "Medium" and "Low", and for the PIR sensors we used two: "Yes" and "No". Below it is possible to see the membership functions for both sensors. This functions are what allow to convert the input values into fuzzyfied expressions that be used in the rules.

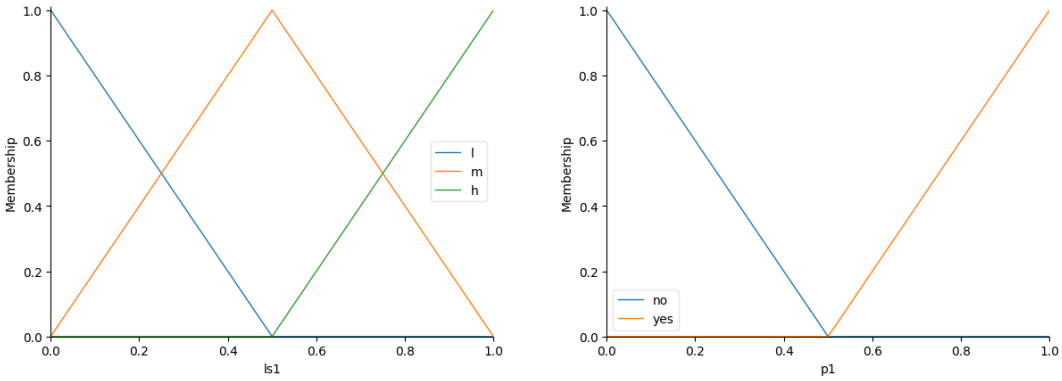


Figure 9: Membership functions: (a) light sensors and (b) motion sensors

After having our membership functions defined, we needed to define the output functions of the sub fuzzy systems, which in fuzzyfied terms have the same meaning, i.e, for light they are "High", "Medium" or "Low" and for the motion they are "Yes" or "No". The difference resides in how the final values get defuzzyfied. For both light and motion the output functions are represented below in Figure 10.

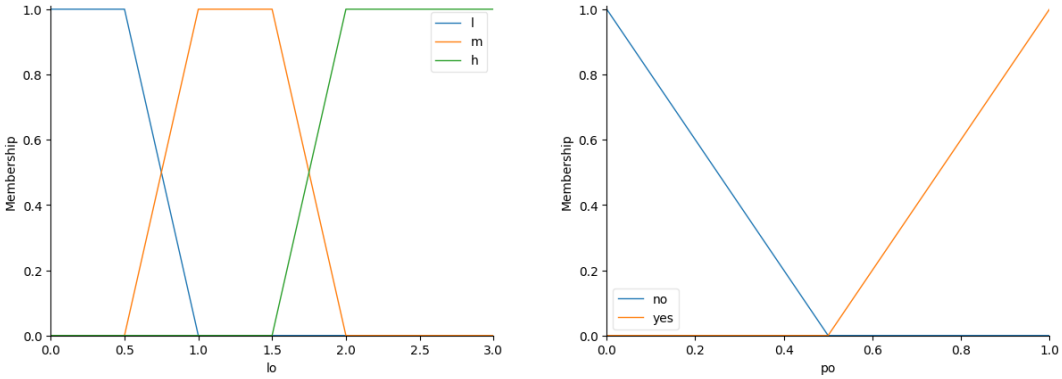
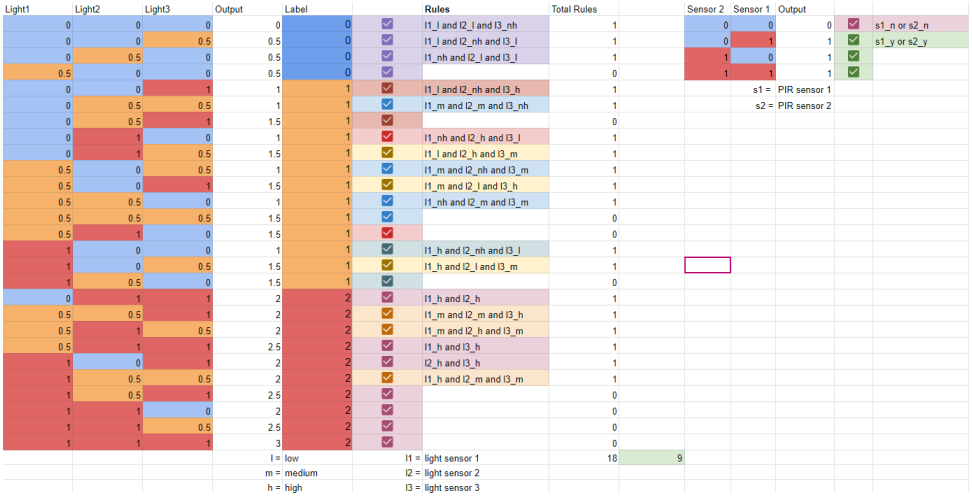


Figure 10: Output functions: (a) light sensors and (b) motion sensors

With both input and output defined, the only thing missing was the rules to connect both ends of the system. To do this, we created a truth table with all the values so that we could find the relations between them. We tried to simplify this rules and make them as comprehensive as possible in order to optimize the system. In Figure 11 it is possible to see the rules for both sub-fuzzy systems. Note that for he light, only 18 rules where needed (instead of the expected 27) and for the motion only 2 rules were needed (instead of the expected 4).



Light1	Light2	Light3	Output	Label		Rules	Total Rules
0	0	0	0	0	☑	l1_l and l2_l and l3_nh	1
0	0	0.5	0.5	0	☑	l1_l and l2_nh and l3_l	1
0	0.5	0	0.5	0	☑	l1_nh and l2_l and l3_l	1
0.5	0	0	0.5	0	☑		0
0	0	1	1	1	☑	l1_l and l2_nh and l3_h	1
0	0.5	0.5	1	1	☑	l1_m and l2_m and l3_nh	1
0	0.5	1	1.5	1	☑		0
0	1	0	1	1	☑	l1_nh and l2_h and l3_l	1
0	1	0.5	1.5	1	☑	l1_l and l2_h and l3_m	1
0.5	0	0.5	1	1	☑	l1_m and l2_nh and l3_m	1
0.5	0	1	1.5	1	☑	l1_m and l2_l and l3_h	1
0.5	0.5	0	1	1	☑	l1_nh and l2_m and l3_m	1
0.5	0.5	0.5	1.5	1	☑		0
0.5	1	0	1.5	1	☑		0
1	0	0	1	1	☑	l1_h and l2_nh and l3_l	1
1	0	0.5	1.5	1	☑	l1_h and l2_l and l3_m	1
1	0.5	0	1.5	1	☑		0
0	1	1	2	2	☑	l1_h and l2_h	1
0.5	0.5	1	2	2	☑	l1_m and l2_m and l3_h	1
0.5	1	0.5	2	2	☑	l1_m and l2_h and l3_m	1
0.5	1	1	2.5	2	☑	l1_h and l3_h	1
1	0	1	2	2	☑	l2_h and l3_h	1
1	0.5	0.5	2	2	☑	l1_h and l2_m and l3_m	1
1	0.5	1	2.5	2	☑		0
1	1	0	2	2	☑		0
1	1	0.5	2.5	2	☑		0
1	1	1	3	2	☑		0
			l = low			l1 = light sensor 1	18
			m = medium			l2 = light sensor 2	
			h = high			l3 = light sensor 3	

Sensor 2	Sensor 1	Output		
0	0	0	☑	s1_n or s2_n
0	1	1	☑	s1_y or s2_y
1	0	1	☑	
1	1	1	☑	
	s1 =	PIR sensor 1		
	s2 =	PIR sensor 2		

Figure 11: Rules for sub-fuzzy systems of light and motion

In order to say if the laboratory was indeed overcrowded at a specific point in time, it is necessary to combine both outputs from the previous sub-fuzzy systems. In respect to the mixed fuzzy system, the input membership functions correspond to the output function of the subsystems shown in Figure 10. Since this classification is binary, the final output should only be one of two possible categories and therefor the output function is as displayed in Figure 12.

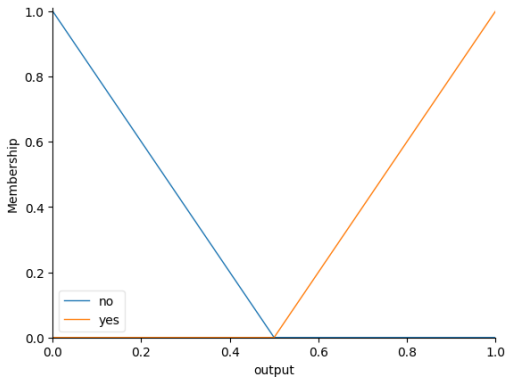


Figure 12: Output function for mixed fuzzy system

Again, the only thing missing are the rules to get to the output value from the input variables. In Figure 13 is its possible to see the table used and the respective rules, which could be reduced to an half of the expected total.

Output of the System				
Total light	Total mov	Output		
0	0	0	☑	tl_nh and tm_n
0	1	1	☑	tl_nh and tm_y
1	0	0	☑	
1	1	1	☑	
2	0	1	☑	tl_h
2	1	1	☑	
	n =	no		
	y =	yes		

Figure 13: Rules for mixed fuzzy system

The confusion matrix for the final predictions of this system can be seen in Figure 14 and the metrics are presented in Table 4. Although this results are not the best, we can see that most of the overcrowded situations were classified correctly, which due to the objective of this system is acceptable. It is important to mention, that during the diverse building iterations we were able to fine-tune the rules and improve in 6% the system's recall.

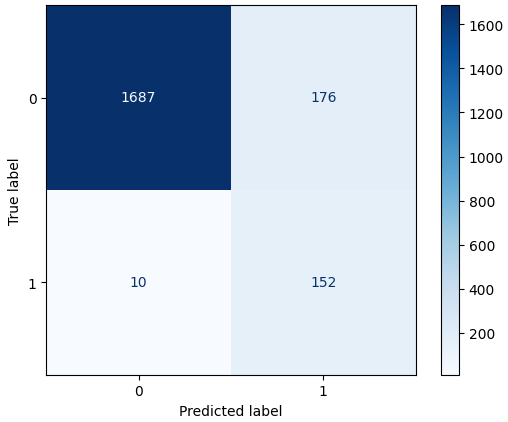


Figure 14: Confusion matrix of the mixed fuzzy system

Recall	Precision	F1
0.93	0.46	0.62

Table 4: Mixed fuzzy system metrics

Finally, in order to compare the fuzzy system with a neural network, we created a basic MLP classifier. This allowed us to understand which type of system would perform better. We can see the evaluation metrics of this MLP in the Table 5 and its confusion matrix in Figure 15. It is possible to observe that even though the metrics for the MLP are better, this has more overcrowded situations classified incorrectly, what for the objective of this project is something negative.

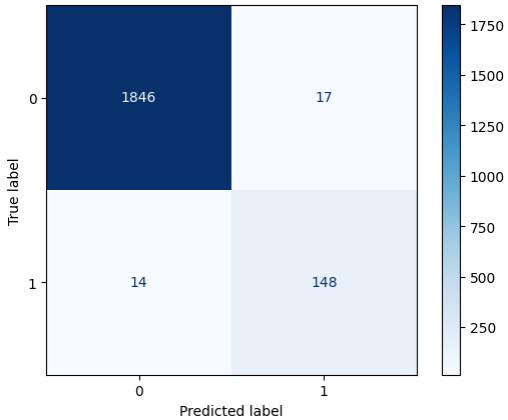


Figure 15: Confusion matrix of the comparison MLP

Recall	Precision	F1
0.91	0.89	0.90

Table 5: Comparison MLP metrics

4 Conclusion

In conclusion, the systems developed are quite robust and can generalize well to unseen data. However, there is still room for improvement, as the parameters are very difficult to fine-tune into the best solution, i.e, even if our system performs well it is impossible to know for sure if it is the best among all possibilities.

For the classification problem, where we need to indicate how many students are inside the lab at a specific time, the MLP trained using k-fold validation after grid search was the most promising classifier among those tested. Since the number of features is low and the relations between them are not know, this sort of classifier is appropriate.

For the binary problem, where we needed to indicate if the laboratory was or not crowded, the Fuzzy system achieved better results than the MLP, even though the metrics were poor. This is because it is more important to make a correct classification of the worst situations and to overestimate the good ones than the opposite, i.e, it is preferable to detect 'crowded' when its not, than to detect 'not crowded' when in fact it is.