TÉCNICO LISBOA

Applied Computational Intelligence

MEEC

First Project Report

Authors:

Diogo José Pereira Araújo (93906) diogoparaujo@tecnico.ulisboa.pt
Diogo Rafael Esteves Ribeiro (102277) diogo.esteves.ribeiro@tecnico.ulisboa.pt

Group 2

2022/2023 – First Semester, P1

Contents

1 Introduction

This project seeks to address the same problem using two different methodologies: NNs and fuzzy systems. The subject of the first part is neural network-based binary and multiclass classification, and in the second part it will be used a fuzzy system to solve the binary classification problem.

2 Part 1 - Classification using NN

In order to justify every decision the group made, it was chosen to put into practice a relatively simple multiclass model that gave the group the ability to make certain decisions regarding the preprocessing of the data. To implement certain model, the group used the MLP Classifier model from scikit-learn library [1]:

- Two hidden layers, each one with 12 neurons;
- Logistic activation functions in the hidden layers;
- Sgd as solver;
- $\alpha = 0.0001$;
- Constant learning rate of 0.05 throughout the NN;
- All remaining hyperparameters were the ones set as default by scikit-learn library.

All the above parameters were chosen from some (debatable) rules of thumb, e.g., $\alpha \in [0.01; 0.1]$, size of input layer $<$ size of hidden layer $>$ size of output layer, size of the hidden layer $> 2\times$ size of the input layer, etc.

2.1 Data preprocessing

The group started the preprocessing procedure by visualizing the data. Through the analysis of plots like those in figure 1, it is evident that the data had some anomalies. This section will contain a description of the techniques used by the group to treat such anomalies and clean the untreated data.

2.1.1 Missing values analysis

Through an analysis of the data, the following missing values were detected: (6461, 'S1Temp')[1], (279, 'S2Temp'), (4571, 'S2Temp'), (3683, 'CO2').

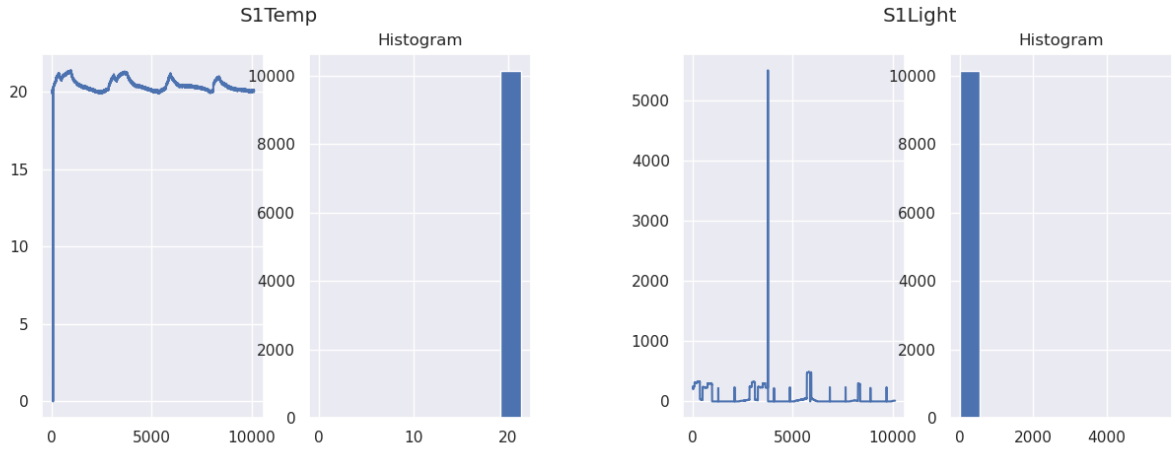
Given that the missing data is temporal, the group determined that computing the interpolation between the prior point and the following point was the most effective way to complete the missing values.

[1](Index, Feature it was detected)

2.1.2 Outlier analysis

In order to detect this anomalies, the group used the following method:

1. Compute the z-score ($z = \frac{x-\mu}{\sigma}$) for each of the input features;
2. Create a new data frame composed by the z-score values;
3. For all values in a "z-scored"[2] column, an outlier is one whose z-score value exceeds k times the mean of that "z-scored" column.



(a) (b)

Figure 1: (a) Plot of the feature S1Temp (left) and the respective histogram (right) without any data cleaning. (b) Plot of the feature S1Light (left) and the respective histogram (right) without any data cleaning.

k denotes an hyperparameter. In order to determine the ideal value of such hyperparameter several experiments were conducted. At the end, the group decided that 7.5 was the ideal value for k. With this value four outliers were detected. The detected anomalies were: (54, 'S1Temp'), (1186, 'S3Temp'), (3758, 'S1Light') and (2798, 'S3Light').

Because the data in question is temporal, the group decided to compute the interpolation between the prior point and the next one rather than removing outliers.

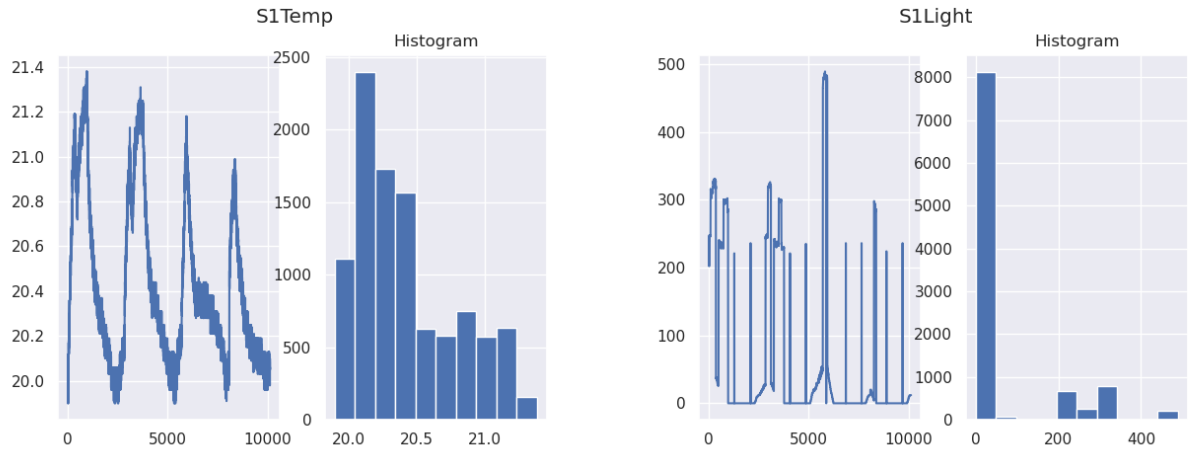
In figure 2 shows the plots of the features represented in figure 1 after handling outliers and missing values. The disparity between this two figures makes it clear that this data treatment was necessary for the group to move forward with the model implementation.

The table 1 shows how the simple multiclass model performed in two different scenarios: one without outlier treatment and the other with the treatment of outliers. The clear improvement in performance demonstrates the advantages of data cleaning, specifically, outlier treatment.

[2]"z-scored" value: z-score value of a given column"

Scores	With outliers	Without outliers
Macro-Precision	0.88245	0.94227
Macro-Recall	0.88885	0.94245
Macro-F1	0.87990	0.94124

Table 1: Simple multiclass model performance with untreated data and with the treatment of outliers.



(a) (b)

Figure 2: (a) Plot of the feature S1Temp (left) and the respective histogram (right) after handling outliers and missing data. (b) Plot of the feature S1Light (left) and the respective histogram (right) after handling outliers and missing data.

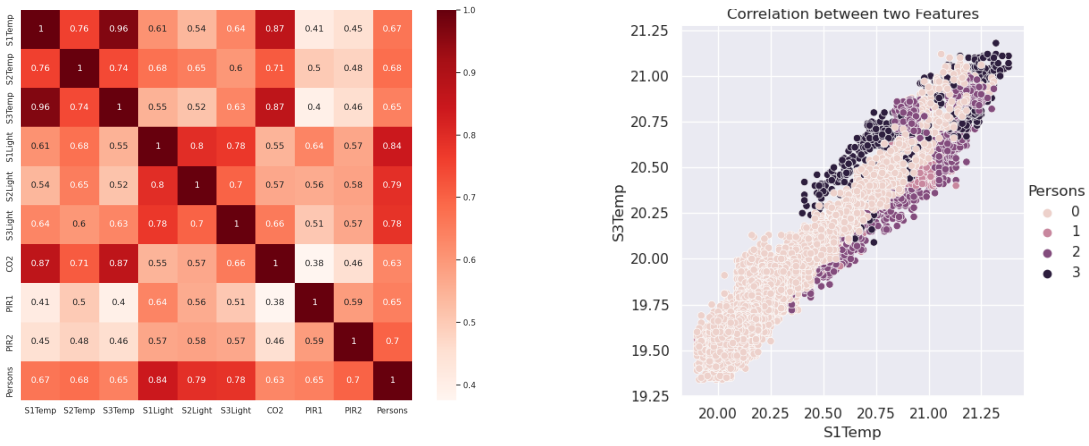
2.1.3 Noise cleaning

Figure 2 demonstrates that there is noise associated with the sensor data. To cope with this noisy data, the group used the moving average technique [2]. In order to make the model as unbias as possible, the moving average should be computed after the train-test split. However there was a dilemma. To create an unbiased score, it is wise to shuffle the data when doing the train-test split. However, in order to compute the moving average, the data must be in chronological sequence. In other words, the group had to choose between doing the train-test split with shuffle and having noisy data, or doing the train-test split without shuffle and having clean data.

The group decided to look at the problem as a pure classification one. Date and Time can be ignored using this approach. Therefore, the group felt that shuffling the data during train-test split was more relevant than cleaning the noisy data.

2.1.4 Feature selection

Looking at the correlation between the different features (figure 3), it is easy to conclude that there are highly correlated features. The group determined that if the Pearson correlation coefficient between two traits was more than 0.85, then one of the features was redundant and should be removed. With this approach 'S3Temp' and 'CO2' were the features removed.



(a) (b)

Figure 3: (a) Plot of the correlations table between all the features. (b) Plot of the correlation between the features 'S1Temp' and 'S3Temp'.

Scores	Without feature selection	With feature selection
Macro-Precision	0.94227	0.95326
Macro-Recall	0.94245	0.94512
Macro-F1	0.94124	0.94855

Table 2: Simple multiclass model performance without feature selection and with feature selection.

From table 2, it can be observed that the scores with and without feature selection are roughly the same. However, the group opted to remove both features since "S3Temp" and "CO2" were redundant.

2.2 Experimental setup

The first step of the experimental setup stage was to train-test split the data. As mentioned in the previous section the train-test split was performed with shuffle. 10% of the total data were used for testing and the remaining 90% for training and tuning the model.

2.2.1 Normalization

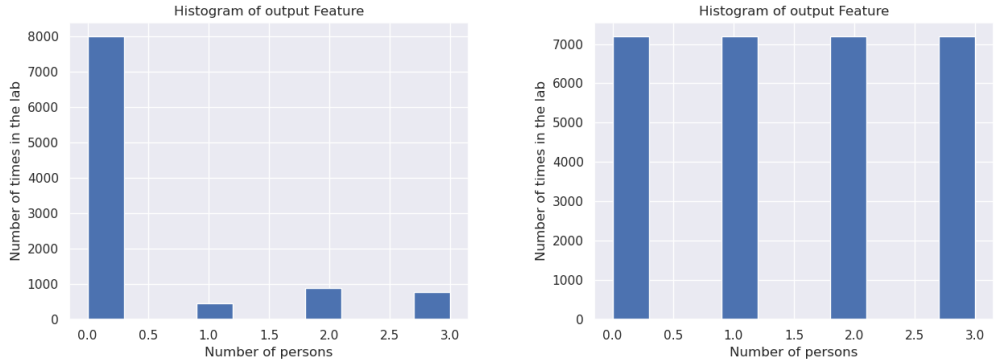
Since all of the outliers had been eliminated, the group chose to adopt min-max normalization. Data normalization was performed after the train-test split. It's also vital to note that the test set's normalization was done using the minimum and maximum values from the training set, for each feature separately.

2.2.2 Balance the data

Figure 4 (a) shows that this data set is clearly imbalanced. To cope with this issue the group decided to use the SMOTE method. The application of the SMOTE technique to the data is shown in figure 4 (b).

As shown in table 3, the simple multiclass model performance improved as a result of the data balancing.

However, the group had to be extremely cautious when using SMOTE, especially when fine-tuning the hyperparameters of our model. The group decided to use the imbalanced-learn library's pipeline method [3] for cross-validation in order to avoid the validation set including false data produced by the SMOTE technique.This topic will be explained in greater detail in the section 2.2.3.



(a) (b)

Figure 4: (a) Plot of the histogram for the feature 'Persons' with imbalanced data. (b) Plot of the histogram for the feature 'Persons' with balanced data.

Scores	With imbalanced data	With balanced data
Macro-Precision	0.95326	0.96237
Macro-Recall	0.94512	0.96329
Macro-F1	0.94855	0.96277

Table 3: Simple multiclass model performance with imbalanced data and with balanced data.

2.2.3 Hyperparameter tuning

As mentioned in the previous section, in order to fine-tune the hyperparameters of the model it was necessary to use the pipeline method from the imbalanced-learn python library [3].

In short, SMOTE transform and Min-Max normalization (mentioned in section 2.2.1) are the only two transforms included in the pipeline. The estimator is the MLP classifier [1].

The group then utilized the GridSearchCV method from scikit-learn library [5] with the pipeline as the estimator.

The range of hyperparameters tested in this procedure for both the binary model and the multiclass model are shown in the table 4. As mentioned in the section 2, the range of values of the hyperparameters follow some (debatable) rules of thumb.

Hyperparameters	Binary model	Multiclass model
Size and nº of the hidden layers	(10,), (12,), (9,9)	(10,10), (12,12), (14,), (12,)
Activation Function	logistic, tanh, relu	logistic, tanh, relu
Solver	sgd, adam	sgd, adam
Alpha	$\in [0.0001; 0.001]$	$\in [0.0001; 0.001]$
Learning rate	constant, adaptive	constant, adaptive
Learning rate init.	$\in [0.01; 0.1]$	$\in [0.01; 0.1]$

Table 4: Range of hyperparameters for the binary model and multiclass model.

Table 5 shows the optimal hyperparameters according to the cross-validation process explained above.

Optimal Hyperparameters	Binary model	Multiclass model
Size and nº of the hidden layers	(12,)	(12,12)
Activation Function	tanh	tanh
Solver	adam	adam
Alpha	0.0001	0.0001
Learning rate	adaptive	constant
Learning rate init.	0.01	0.01

Table 5: Optimal hyperparameters chosen by the cross-validation method described in section 2.2.3 for the binary model and multiclass model.

2.3 Results

The adjustment of the hyperparameters turned out to be significant in enhancing the performance of both models, as will be demonstrated in this section.

2.3.1 Binary classification results

Table 6 shows the scores obtain for the binary model defined at the end of section 2.2.3.

Scores	Binary model with the optimal hyperparameters
Precision	0.97619
Recall	0.89134
F1	0.93897

Table 6: The average of the optimal binary model's results across ten runs.

2.3.2 Multiclass classification results

Table 7 shows that the fine-tuning process of the model improved the models generalization performance. Highlighting the significance of a precise description of a neural network's hyperparameters.

Scores	Simple multiclass model	Optimal multiclass model
Macro-Precision	0.96237	0.98702
Macro-Recall	0.96329	0.97142
Macro-F1	0.96277	0.97418

Table 7: Performance comparison between the multiclass model with the optimal hyperparameters, and with the simple multiclass model.

3 Part 2 – Classification using Fuzzy Systems

3.1 Data preprocessing

Regarding data preprocessing, the group followed the same steps as in the previous chapter until the feature selection. Due to the flexibility of fuzzy systems, not many features (inputs) are needed to be able to represent the data. Indeed, this system becomes harder and harder to implement with the increase of features

3.1.1 Feature selection

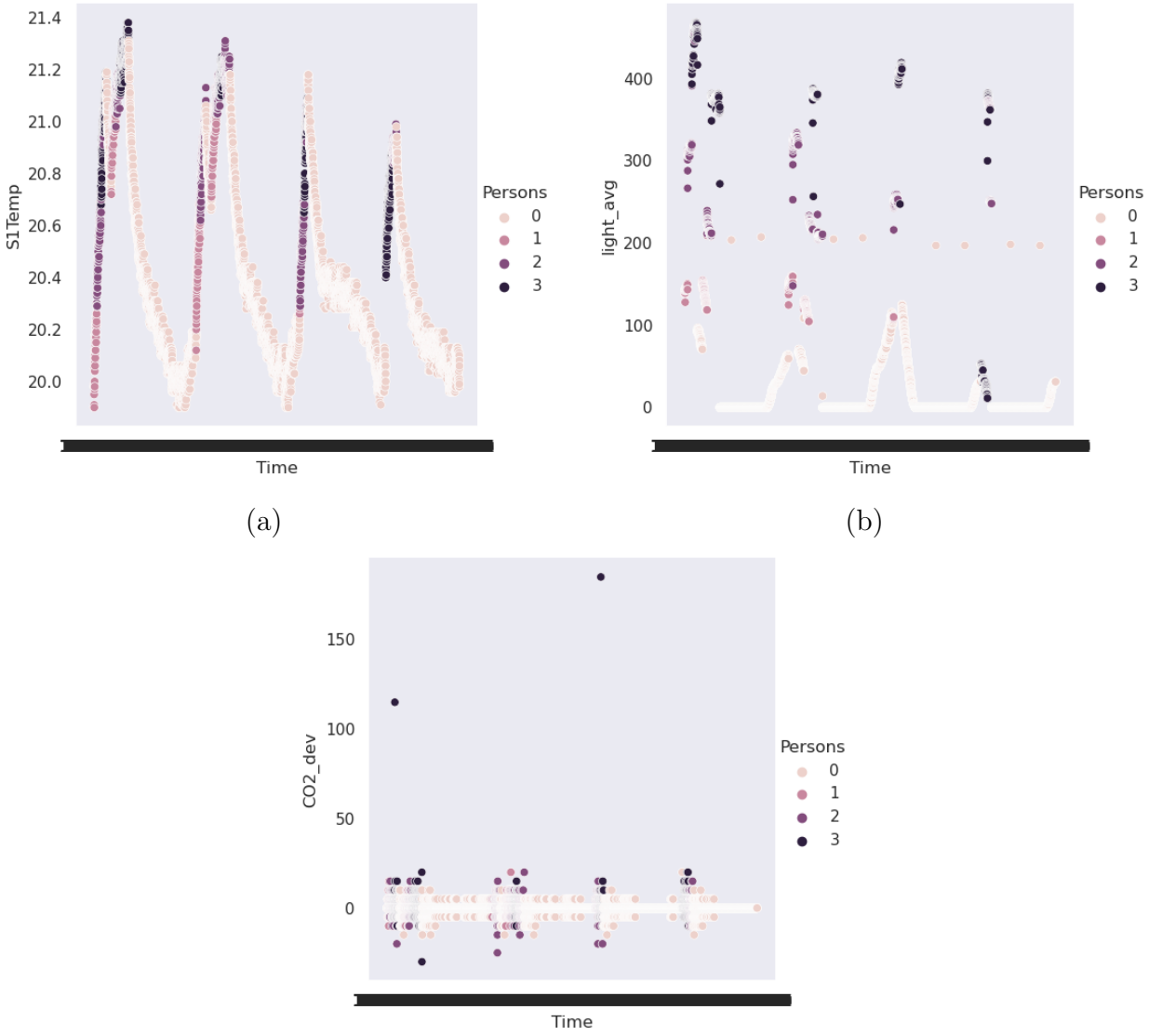
As mentioned earlier, the selection of the right features to outline our model is really important. The group started by eliminating features that were redundant. The group started by analysing the correlation matrix, presented in figure 3 (a), and tried to aggregate these features into new ones.

Since, the confusion matrix illustrates that the 'S1Temp' and 'S3Temp' are highly correlated (.96), we decided to drop the 'S3Temp'. Regarding 'S2Temp', we also eliminate it, because the group felt it did not add any relevant information about the data.

In relation to lights measurements, 'S1Light', 'S2Light', 'S3Light', we averaged it, to trying to generalize the amount of light inside the room. Since these sensors are in different places, they can be influenced by the light coming from the window, for example. In this way, we intend to reduce the external factors.

Regarding the movement sensors, the group tried two approaches. In the first one, we rearranged these inputs into the maximum element between both variables ("PIR1", "PIR2") in order to reduce the times, there are people in the room but since they don't move or are far away from the sensor, the sensor can't detect movement. In the second one, we engineer a feature that would correspond to the 'AND' operation between them. The objective was creating a boundary for more than 2 persons in the room and less than 2 persons, because if I have more people inside the room the probability of both sensors detect presence is higher. Although we tried both approaches, the results were not the best, so we have dropped both features.

In the end the group decided to use three features: S1Temp, the average of the three light sensors, and the variation of the of the CO2 along time. The three features used in the fuzzy system are illustrated in figure 5.



(a) (b)

(c)

Figure 5: The three features used in the fuzzy system. (a) S1Temp. (b) Average of the three light sensors. (c) CO2 variation.

3.2 Fuzzy System Implementation

The goal is to detect when there are more than 2 persons inside the lab to know if its capacity has been exceeded or not. In the table 8 are defined the 3 linguist variables inputs (Temperature, Light, CO2), the linguistic variable output (Persons) and the corresponding linguist terms and range used in our implementation.

Linguist Variables	Linguistic Terms	Range
Temperature	{"Cold", "Hot"}	[19.9:21.38]
Light	{"Low", "Medium", "High"}	[0:468]
CO2	{"Decrease", "Constant", "Increase"}	[-30:185]
Persons	{"Lower", "Equal"}	[0:3]

Table 8: Linguistic variables and respective linguistic terms.

After testing several times, we concluded that the combination between the triangular and trapezoidal membership functions would be the best for our problem. There is no method that will tells us exactly which are the boundaries of a fuzzy sets. Usually, like in the fuzzy rules the best metric is our common sense.

Table 9 and figure 6 shows the fuzzy sets defined and the associated membership functions, correspondingly.

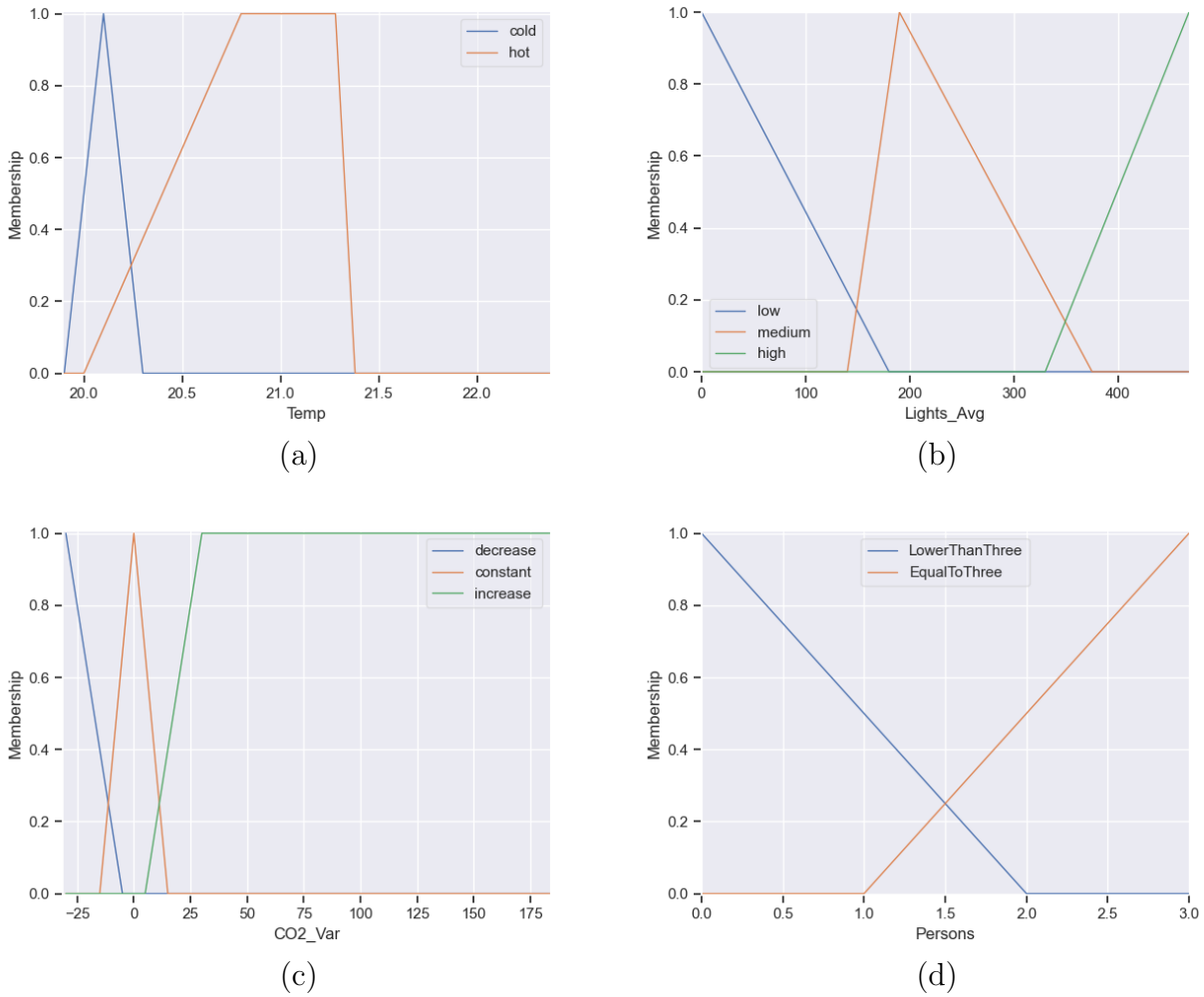


Figure 6: Membership functions of the fuzzy system. (a) Membership function of the 'S1Temp' feature. (b) Membership function of the average of the light sensors. (c) Membership function of the variation of CO2. (d) Memberships function of the output variable.

Temperature{"Cold"}: [19.9, 20.1, 20.3]	CO2{"Decrease"}: [-30, -30, -5]
Temperature{"Hot"}: [20.0, 20.8, 21.28, 21.38]	CO2{"Constant"}: [-15, 0, 15]
Light{"Low"}: [0, 0, 180]	CO2{"Increase"}: [5, 30, 185, 185]:
Light{"Medium"}: [140, 190, 375]	Persons{"Lower"}:[0, 0, 2]
Light{"High"}: [330, 468, 468]	Persons{"Equal"}: [1, 3, 2]

Table 9: Fuzzy Sets.

It should be noted that a very careful inspection was made of the plots of figure 5 to arrive at the border values shown in the table 9. The most relevant borders are the ones that allow us to distinguish between having three people in the room or less than three people. In figure 6 (a) the group decided that above 20.3ºC the temperature is "hot" and in figure 6 (b) we choose that above 375 LUX we only have 3 people in the room. About figure 6 (c) the variation of CO2 its faster when we have 3 people in the room. In the most cases this variation is positive, then CO2 is increasing. In this way, our most relevant border is on upper side on the map between CO2 "constant" and CO2 "high". Since in short variations of CO2 is kind of hard to distinguish between both of them, we decided to delimiter an intersection between 5 to 15 PPM promoting some fuzziness.

3.2.1 Rule set

Through analysis of figure 5, common sense and numerous experiments, the group arrived at the rules represented in figure 7.

```
rule1 = ctrl.Rule(mean_lights['low'] & CO2_deriv['decrease'], Persons['LowerThanThree'])
rule2 = ctrl.Rule(mean_lights['low'] & CO2_deriv['constant'], Persons['LowerThanThree'])
rule3 = ctrl.Rule(mean_lights['low'] & CO2_deriv['increase'], Persons['EqualToThree'])

rule4 = ctrl.Rule(mean_lights['medium'] & CO2_deriv['decrease'], Persons['LowerThanThree'])
rule5 = ctrl.Rule(mean_lights['medium'] & CO2_deriv['constant'], Persons['LowerThanThree'])
rule6 = ctrl.Rule(mean_lights['medium'] & CO2_deriv['increase'], Persons['EqualToThree'])

rule7 = ctrl.Rule(mean_lights['high'] & CO2_deriv['decrease'], Persons['LowerThanThree'])
rule8 = ctrl.Rule(mean_lights['high'] & CO2_deriv['constant'], Persons['EqualToThree'])
rule9 = ctrl.Rule(mean_lights['high'] & CO2_deriv['increase'], Persons['EqualToThree'])

rule10 = ctrl.Rule(temp['cold'] & CO2_deriv['decrease'], Persons['LowerThanThree'])
rule11 = ctrl.Rule(temp['cold'] & CO2_deriv['constant'], Persons['LowerThanThree'])
rule12 = ctrl.Rule(temp['cold'] & CO2_deriv['increase'], Persons['LowerThanThree'])

rule13 = ctrl.Rule(temp['hot'] & CO2_deriv['decrease'], Persons['LowerThanThree'])
rule14 = ctrl.Rule(temp['hot'] & CO2_deriv['constant'], Persons['EqualToThree'])
rule15 = ctrl.Rule(temp['hot'] & CO2_deriv['increase'], Persons['EqualToThree'])

rule16 = ctrl.Rule(temp['cold'] & mean_lights['low'], Persons['LowerThanThree'])
rule17 = ctrl.Rule(temp['cold'] & mean_lights['medium'], Persons['LowerThanThree'])
rule18 = ctrl.Rule(temp['cold'] & mean_lights['high'], Persons['LowerThanThree'])

rule19 = ctrl.Rule(temp['hot'] & mean_lights['low'], Persons['LowerThanThree'])
rule20 = ctrl.Rule(temp['hot'] & mean_lights['medium'], Persons['LowerThanThree'])
rule21 = ctrl.Rule(temp['hot'] & mean_lights['high'], Persons['EqualToThree'])
```

Figure 7: Rule base of the Fuzzy system.

Regarding to tunning, the ranges and rules presented before were the best obtained. We also tested the fuzzy system with fewer linguistic terms. Indeed, we have tried to use just the

mean of lights and the variton of CO2. The performance was actually very close to the obtained but metrics like precision fluctuated a bit depending on the training set it received.

3.3 Results

3.3.1 Fuzzy System vs Neural Network

In order to make the NN requested the group used the same method as the one described in section 2.2.3. The optimal hyperparameters given the set of features are illustrated in table 10.

Optimal Hyperparameters	**NN Binary model for part 2**
Size and nº of the hidden layers	(6,)
Activation Function	tanh
Solver	adam
Alpha	0.0001
Learning rate	constant
Learning rate init.	0.01225

Table 10: NN model for binary classification with "Fuzzy features".

In the table 11, it is possible to see the the average performance of each model (across ten runs). Although the NN performance was lower than the one presented in table 6, the NN obtained better results than the fuzzy classifier. The outcomes of the fuzzy classifier were, nevertheless, fairly good given the simplicity of the fuzzy system and the fact that it was implemented considerably more easily than the NN.

Scores	Simple multiclass model	Optimal multiclass model
Precision	0.75419	0.93670
Recall	0.87804	0.89156
1	0.80585	0.90358

Table 11: Performance comparison between the fuzzy classifier and NN described in table 10.

4 Conclusion

This project aimed to analyse two extremely different methods. In structural terms, the big difference between NNs and fuzzy systems is the way of solving problems. While NNs needs a lot of information, details and features, the fuzzy systems needs human logic, known as, common sense, which make the model easier to develop. Equally or more important than the algorithms the group used is data treatment, which should be made before applying any intelligent model. In this way, the group dealt with missing values, outliers, noise, feature selection, normalization and balance, before the model prediction.

The first algorithm implemented was the NN, more specifically the MLP. For both binary and multi-class problems, the values obtained by the metrics were very high. It should be noted that these values were obtained without the "S3Temp" and "CO2" features, as they were considered, by the group, redundant.

For the second algorithm implemented, the fuzzy system, the data treatment was practically the same, except for the feature selection. To escape from the combinatorial rule explosion and make the model more intuitive, the features to be used were carefully selected. We chose to use the average of the lights, the variation of CO2, and the temperature "S1Temp". With this model, the group obtain about 80% of F1-score. Then, we adapted the MLP with the features selected in the fuzzy system. Although the results have decreased a little, they are still better than the results obtained by the fuzzy system.

With this, we just want to point out that depending on the problem at hand, both methods should be considered.

References

[1] MLP Classifier using scikit-learn library, `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`

[2] Rooling average from pandas library, `https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rolling.html`

[3] Imbalaced Pipeline from imbalanced-learn library, `https://imbalanced-learn.org/stable/references/generated/imblearn.pipeline.Pipeline.html`

[4] The right way of using SMOTE with Cross-validation, `https://towardsdatascience.com/the-right-way-of-using-smote-with-cross-validation-92a8d09d00c7`

[5] GridSearchCV form scikit-learn library, `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`