Applied Computational Intelligence

1st Semester 2022/23

Project 1: Classification using neural networks and fussy systems

---

Authors:

Duarte Venâncio Leão Ribeiro da Silva, Nº 93243
Gonçalo Da Silva Dias Moura de Mesquita , Nº 94196

October 16, 2022

Contents

1 Introduction

In this project our main objective is to compare the results between neural networks classification and fuzzy systems classification. Consequently we applied both methods to the same problem. In this case, the problem was to predict the amount of people that were simultaneously inside a lab (multi classification problem) and detect when there were more than 2 people inside the lab (binary classification problem), the fuzzy systems only was applied to the binary problem. In order to solve this problem we had different types of data, most of them corresponding different sensors measurements, that would help us to predict the solution, the data available contained information about time, light sensors, temperature sensors, movement sensors and a CO2 sensor.

2 Part 1

2.1 Data Visualization

In order to be able to understand the data, we started by visualizing each feature in different types of plots. We observed that the data was very volatile, we were also able to distinguish some features with extreme outliers and with a few missing data, as we can notice in figure 2. We also visualized a map of the correlation between each feature, represented in figure 1. We can observe that 'S1Temp', 'S3Temp' and 'S1Temp', 'CO2' have big correlations. Consequently, taking into account feature selection, we removed 'S3Temp' and 'CO2' from the data set due to their big correlation with 'S1Temp' they would have very little impact in the final results. 'Time' and 'Date' were also removed since we were going to use cross validation.

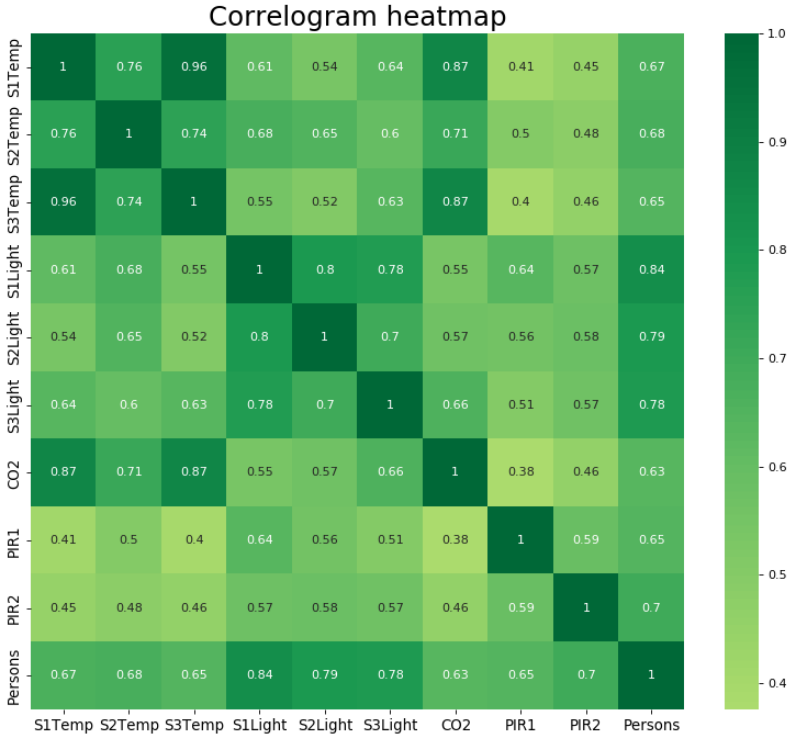

Figure 1: Correlation between features

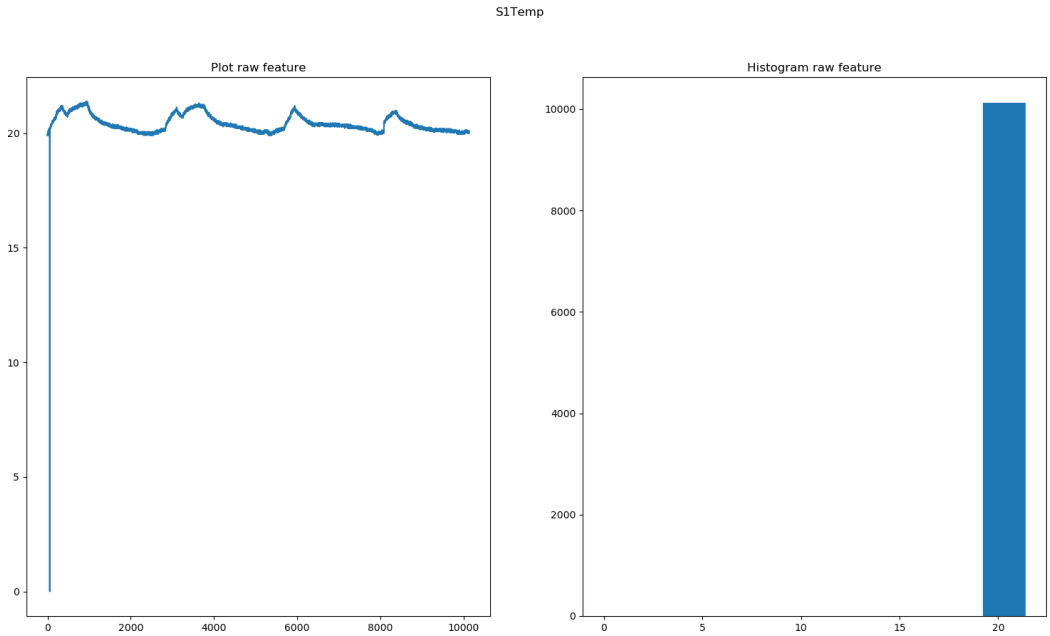

Figure 2: Plot of the feature 'S1Temp' with outliers

2.2 Data Preparation

After analysing the raw data, in data preparation we take a hard look into the detection of outliers and detection of missing data. We start by analysing the outliers. In order to detect them we create an upper and lower boundary, as in equation 1

$$upper_b = mean[feature] + k \times \sigma[feature] \qquad lower_b = mean[feature] + k \times \sigma[feature] \qquad (1)$$

Where $\sigma[feature]$ would be the standard deviation of each feature and k would be a constant chosen by us. Every value, in reference to his feature, which is greater than their $upper_b$ or lower than their $lower_b$ is replaced with the value of the previous index, due to the data being sequential (also used this method for replacement on the missing values). After observing the plots of each features we choose a k that would be able to take out extreme outliers, such as in figure 2. The reason we chose these method was because we could decide the upper and lower boundaries taking into account the types of data that we had. In this case as we are dealing with sensors data we have to be very careful in the removal of outliers. Consequently $k = 5.4$, to only take out extreme outliers. We notice that the data is much cleaner in figure 3 than in figure 2 therefore is much easier to understand the data.

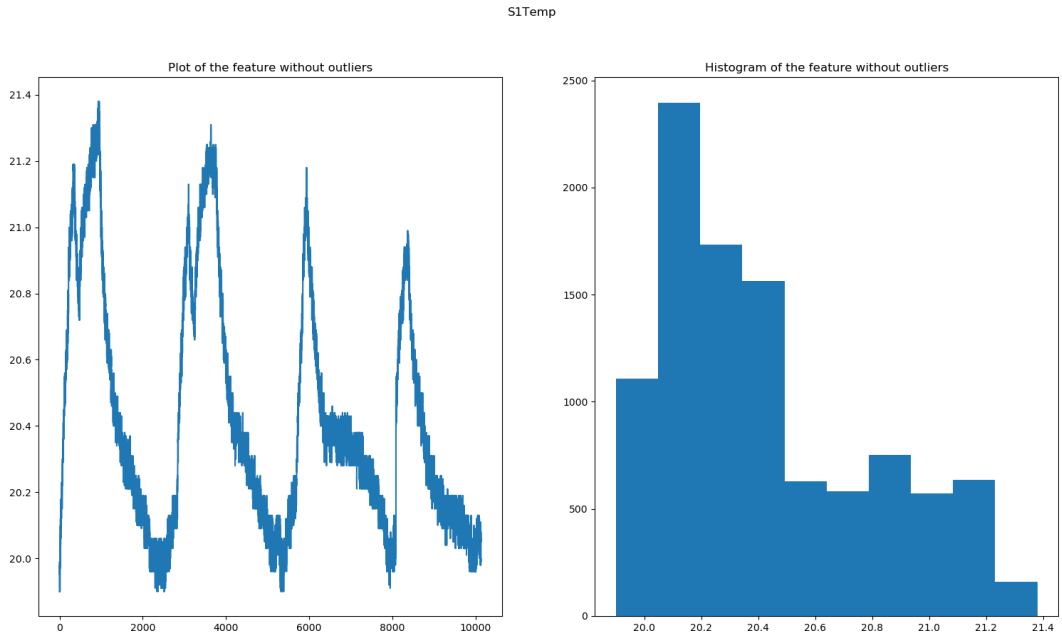

Figure 3: Plot of the feature 'S1Temp' without outliers

2.3 Data Setup

Data setup is where we split the data, normalize it, balance it and apply cross validation. In order to use cross validation we had to split the data set into 85% training and 15% testing. Cross validation was applied to the training set and the test set was used to test the model with unseen data.

In data visualization we were able to observe that the features were in very different scales. In order for the neural network have a better understanding of the trends between the data points of different features, the data had to be normalized. Consequently we used the z-score method.

It was very noticeable that the data was unbalanced in almost every feature, consequently we used the SMOTE method to create synthetic data to balance the data set. Unbalanced data sets usually perform badly, while at the same time giving the false sense of a highly accurate model. However it wasn't the case, as it will be shown ahead. We will show results with the balance and unbalanced data, and observe that they aren't too different in this case.

Our approach to solve this problem was to apply a multilayer perceptron. In order to have more accurate hyperparameters. We decided to use the function *GridShearchCV* from *Sklearn*, in which is implemented cross validation with the training set.

2.4 Results

After doing cross validation and tuning the hyperparameters we tested the model with the test data set, we immediately had very good results in both multi and binary classification. However the hyperparameters were the 'variable' that changed most the results.
The following results were obtained using cross-validation with 5 folds with different hyperparameters, so we can understand how important is to have the right ones.

The table 1 presents the results of the multi classification with unbalanced data. The best results of precision and recall of *GridShearchCV* function were obtained with the model with (14,) hidden layers as we can confirm by table 1. The table 2 presents the results of multi classification with the data balanced using the SMOTE method, we can notice that the results of recall are a

Hidden layers	Learning rate	alpha	Confusion matrix	Recall macro	Precision macro	Time (seconds)
(6,)	0.02	0.001	1194 0 1 4 0 67 0 0 0 0 123 9 0 0 1 121	0.9798	0.9717	0. 813
(10,)	0.003	0.0005	1198 1 0 0 0 67 0 0 0 0 127 5 0 0 8 114	0.9739	0.9710	2.48
(14,)	0.002	0.0005	1199 0 0 0 1 66 0 0 0 0 126 6 0 0 5 117	0.9746	0.9780	2.57
(6,6)	0.003	0.0005	1195 0 0 4 0 67 0 0 0 0 126 6 0 0 14 107	0.9570	0.9534	3.14

Table 1: Results of multi classification problem without SMOTE method

Hidden layers	Learning rate	alpha	Confusion matrix	Recall macro	Precision macro	Time (seconds)
(6,)	0.02	0.001	1169 1 0 3 0 82 0 0 0 1 138 6 0 0 4 116	0.9787	0.9690	3.29
(10,)	0.003	0.0005	1169 2 0 2 0 82 0 0 0 1 139 5 0 0 6 114	0.9763	0.9663	5.90
(14,)	0.002	0.0001	1168 2 0 3 0 82 0 0 0 1 140 4 0 0 5 115	0.9798	0.9682	7.83
(6,6)	0.003	0.0005	1166 2 0 5 0 82 0 0 0 2 135 8 0 0 5 115	0.97085	0.9540	4.89

Table 2: Results of multi classification problem with SMOTE method

Hidden layers	Learning rate	alpha	Confusion matrix	Recall	Precision	Time (seconds)
(6,2)	0.003	0.0005	1396 12 6 106	0.9464	0.8983	1.386
(10,)	0.003	0.0005	1396 12 7 105	0.9375	0.8974	2.145
(6,)	0.002	0.001	1400 8 8 104	0.9285	0.9285	3.12
(14,)	0.003	0.0005	1400 8 5 107	0.9553	0.9553	2.057

Table 3: Results of binary classification problem without SMOTE method

little bit better than in table 1, however the time it takes is much higher with a balanced data set since we have much more data and the precision decreases a little . Consequently, for the multi classification problem, we chose the model with unbalanced data and (14,) hidden layers to be the best.

Hidden layers	Learning rate	alpha	Confusion matrix	Recall	Precision	Time (seconds)
(6,2)	0.003	0.0005	1373 27 6 114	0.95	0.8085	2.234
(10,)	0.003	0.0005	1398 2 11 109	0.9083	0.9819	3.222
(6,)	0.002	0.001	1385 15 8 112	0.9333	0.8818	4.180
(14,)	0.003	0.0005	1393 7 1 119	0.9916	0.9444	3.470

Table 4: Results of binary classification problem with SMOTE method

The table 3 presents the results of the binary classification with unbalanced data. The best result of recall of *GridShearchCV* function was with the model with (14,) hidden layers as we can confirm by table 3. The table 4 presents the results of the binary classification with balanced data using the SMOTE method. We can notice the results are worse than with unbalanced data, table 4. Consequently, for the binary classification problem, we chose the model with unbalanced data and (14,) hidden layers to be the best. We can also understand by the result with more than one hidden layer that for a simple problem, more complex multi layer perceptrons works worst. In conclusion this neural network gives very good results with this problem, however we have to be very careful with the hyperparameters since it can change the results.

3 Part 2

3.1 Fuzzy Rule Based System

Here we implemented a Fuzzy rule based System where, after the outlier removal and missing values methods used in the first part were applied, the input features were reduced and transformed relatively to the features used in first half of the project.

This new features were created in order to lower the complexity of the system and to make it easier to understand. With that said the features chosen correspond to the overall level off light in the room which was obtained by averaging values of the three light sensors readings at each time step, the variation of the CO2 levels in the room which was calculated using the difference of between CO2 level at time step k and time step $k-1$ and the last feature used was the time of the day which was counted in seconds from 8 p.m. till 8 p.m. of the next day, i.e as if the day started at 8 p.m. instead of midnight.

After the feature selection, we had to define our membership functions for each feature. For the CO2 variation we decided to have three regions: decreasing, constant and increasing. Due to noise that might be present in the CO2 readings we chose a triangular membership function for the constant region, where we only are certain that it is really constant at 0, for the other regions we chose trapezoidal membership functions since that from a certain threshold we assume that the variable can be defined as a crisp variable. The membership functions of the light feature are all triangular since that the linguistic terms used do not really have direct translation to crisp values. As for the time input feature trapezoidal membership functions were chosen with the same reasoning that the trapezoidal functions were used in other inputs. In terms of the output, we divided it in two terms: low and high, which is analogous to a low and high number of people in the lab, in the the end of the system computation we chose the centroid method to defuzzify the output. In figures 4, 5, 6 and 7, it's possible to visualize the referred membership functions.

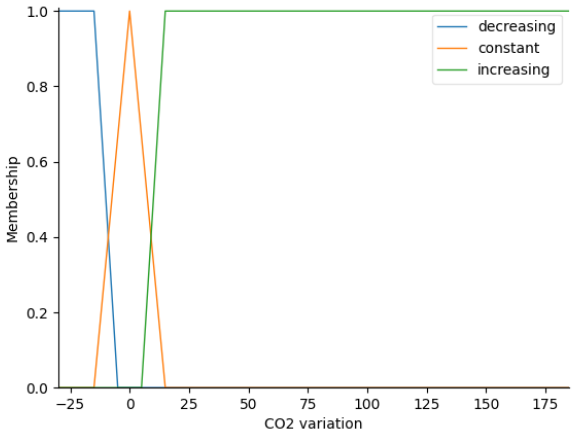

Figure 4: Membership functions of CO2 variation

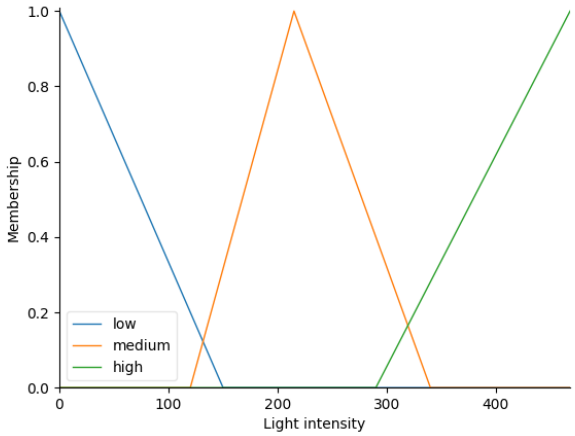

Figure 5: Membership functions of Light intensity

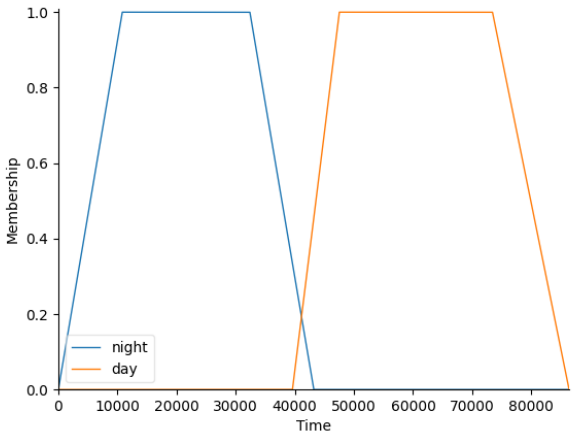

Figure 6: Membership functions of Time

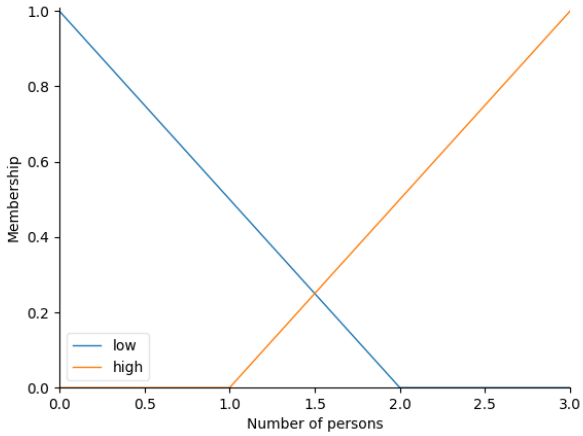

Figure 7: Membership functions of number of Persons

To determine the ranges of each membership function and the rules for this system, it was necessary to analyze the scatter plot composed of each feature, with the color determining the outcome where 1 represents a number of people higher than 2 and 0 lower than 3, this can be observed in Fig. 8. From this graphic representation, it is clear to see that the usually the greater than 2 data points occupy a certain region, which can be helpful to determine the interval within each feature that can provide this outcome. Furthermore, it's possible to establish that there are no more than a 2 people outcome, when the time value is lower than 60000. On another level, a big portion of the output positive values have a light intensity value higher than 250. As it can also be seen, there is a section of this light feature space that has some positive outcome data points, where there are also exists high number of the lower than 3 people data points, which indicates that this will be a tricky situation to apply a fuzzy rule based system, given that there is no significant separability of the two classes.

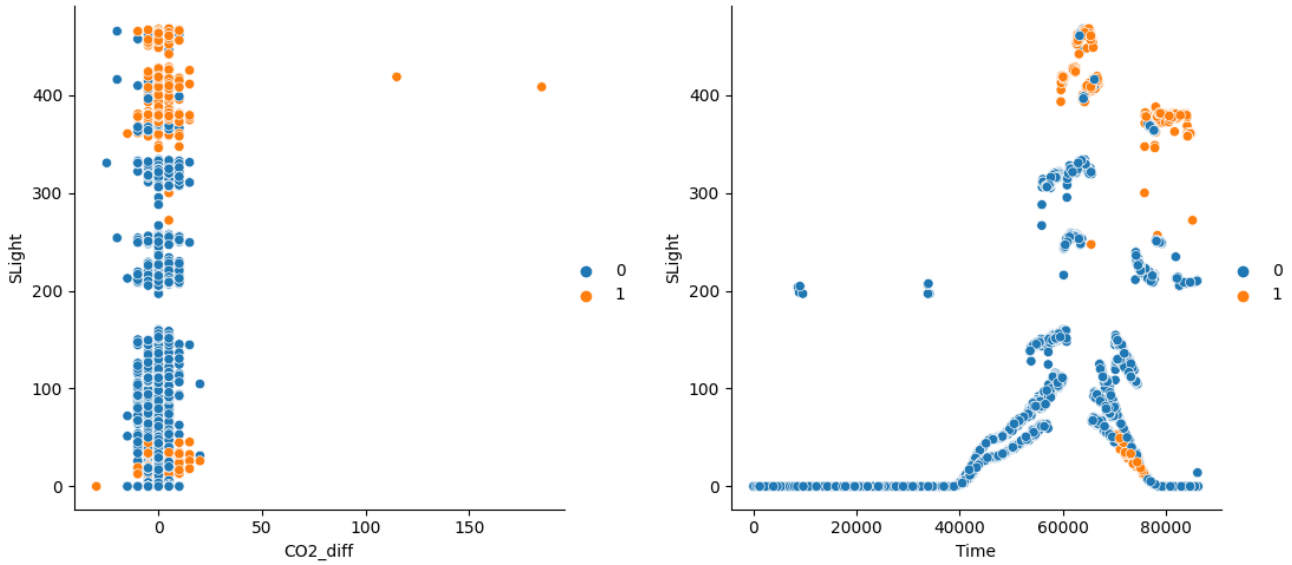

Figure 8: Scatter plot regarding the training set

Consequently, we can determine the rules that will take part in this fuzzy system, presented in table 5 where all combinations are AND operations. The rules in red were not used due to the overall worsening of the results for either of the outcomes and in some cases also due to the inclusiveness of the outcome given their respective antecedents, for example when the CO2 variation is constant during the day we have no knowledge of the real level of the CO2 thus being impossible to deduce the number of people in the room. These rules were selected by an iterative (some trial and error) and empirical process(i.e. data visualization).

Number of persons	Time: Night	Time: Day	Light: low	Light: medium	Light: high
CO2 variation: decreasing	Low	Low	Low	Low	High
CO2 variation: constant	Low	High	Low	Low	High
CO2 variation: increasing	Low	High	High	High	High
Light: low	Low	Low			
Light: medium	Low	Low			
Light: high	Low	High			

Table 5: Rules

With this set of rules, we trained and refined the ranges membership functions of our fuzzy system with a training set obtained with a train test split from our original data set now with the reduced and transformed features. Consequently the confusion matrix in table 6 was obtained, and also the precision, recall and F1 scores in table 7.

	Predicted 0	Predicted 1
Real 0	8316	87
Real 1	180	533

Table 6: Confusion matrix of the fuzzy rule based system's performance with the train set

Precision score	0.86
Recall score	0.75
F1 score	0.8

Table 7: Fuzzy rule based system's performance with the train set

The system had an overall successful performance distinguishing between positive and negative points, most of the positive points that it failed to classify reside in the low light region of the feature space, due to the difficulty to create a model that is robust to that region given the features. As it can be seen it was more successful at not mistaking real negative points as positive.

3.2 Fuzzy System Comparison to NN

After the implementation of the Fuzzy Rule Based System, we tested the model with the test data provided by the previously used train test split, where the test size corresponds to 10% of the whole data set, the results obtained are shown on table 8 and 9. From Fig. 9 it is clear to see why the system had better performance on the test set, as it can be seen the number of cases where the model usually fails to predict the right outcome (i.e. low light cases) is very reduced. These results also show that there was no overfit in the parameter tuning of fuzzy system.

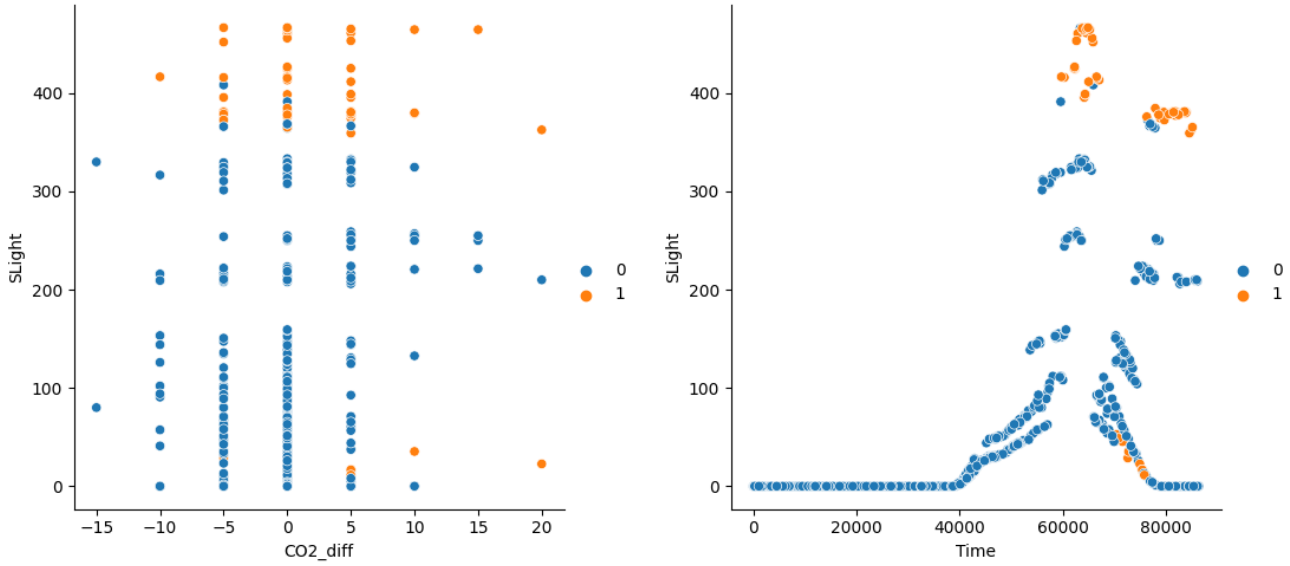

Figure 9: Scatter plot regarding the test set

	Predicted 0	Predicted 1
Real 0	931	14
Real 1	10	58

Table 8: Confusion matrix of the fuzzy rule based system's performance with the test set

Precision score	0.81
Recall score	0.85
F1 score	0.83

Table 9: Fuzzy rule based system's performance with the test set

In order to have a comparison to how other types of models would perform with this new features, relatively to the fuzzy system developed, a multilayer perceptron classifier was trained with the normalized features after a hyperparameter tuning with the help of *GridShearchCV* from *Sklearn*. After training, the results obtained on the previously mentioned test set are shown in tables 10 and 11.

	Predicted 0	Predicted 1
Real 0	932	13
Real 1	11	57

Table 10: Confusion matrix of the NN performance with the test set

Precision score	0.81
Recall score	0.84
F1 score	0.83

Table 11: NN performance with the test set

As it can be observed, the performance of this neural network was equivalent to the fuzzy system one, and worse than in the first half of the project due to the low number of features (i.e. the reduced available data leads in this case to less accurate predictions). From this it can be concluded that when the amount of data is reduced and the system is not as complex it might be useful to use a fuzzy rule based system because with this type of model the outcomes given certain inputs are explainable and are not black boxes as it is the case of neural networks, so we can have an intuition of what is going on.