Instituto Superior Técnico

MEEC

Applied Computational Intelligence

Project 1

Work done by:	*Number:*
Tiago Almeida	96328
Rodrigo Contreiras	90183

Group 35

16 of October 2022

Contents

Introduction

This first project involves estimating the number of people present in a specific room using information gathered from nine distinct types of sensors. To approach this problem, we will use a Neural Network and a Fuzzy System.

This project's first section is divided into two exercises. The first one there is a multi-class problem that requires figuring out how many people are in the room precisely, and the second is a binary problem that involves figuring out if there are more than two people in the room or not. Both of these goals will be to be accomplished by the use of a Neural Network-based classifier.

The binary problem will also be addressed in the second section, where we solve this problem by using a Fuzzy Rule based classifier.

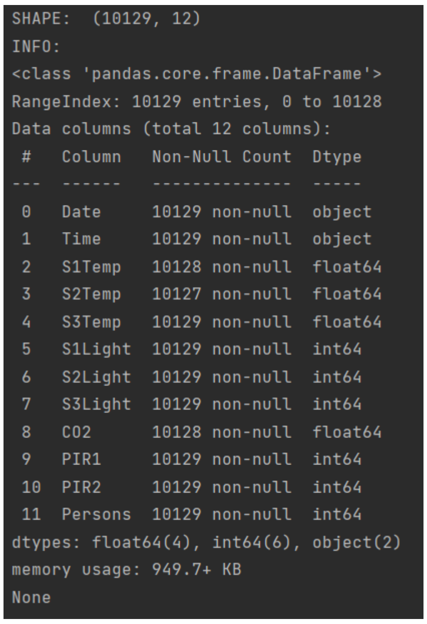
In the final section, we will compare the model's scores of both the Fuzzy Classifier and the Neural Network one, with the binary problem.

1 Preprocessing

1.1 Data upload and missing values - preprocessing

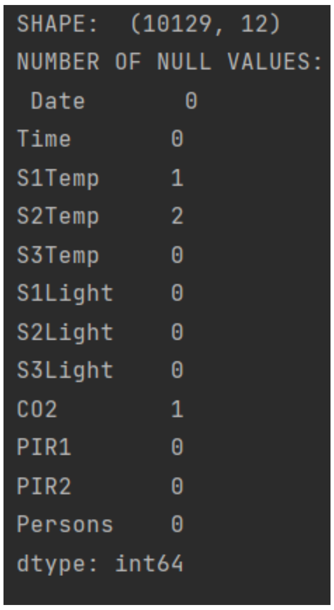
The file *Proj1_Dataset.csv* contains information on the different sensors and the true number of individuals in the room. The dataset was uploaded into a Pandas library data frame, through the function *upload_data()*.

To minimize errors from the systems, we search for duplicated data and dropped any that might exist using functions from the Pandas library. We used some functions - *.info()*, *.shape* and *.isnull().sum()* - to start understanding the data and to preprocess it.



```
SHAPE:  (10129, 12)
INFO:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10129 entries, 0 to 10128
Data columns (total 12 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Date     10129 non-null  object
 1   Time     10129 non-null  object
 2   S1Temp   10128 non-null  float64
 3   S2Temp   10127 non-null  float64
 4   S3Temp   10129 non-null  float64
 5   S1Light  10129 non-null  int64
 6   S2Light  10129 non-null  int64
 7   S3Light  10129 non-null  int64
 8   CO2      10128 non-null  float64
 9   PIR1     10129 non-null  int64
 10  PIR2     10129 non-null  int64
 11  Persons  10129 non-null  int64
dtypes: float64(4), int64(6), object(2)
memory usage: 949.7+ KB
None
```

Figure 1: Data overall information and shape.

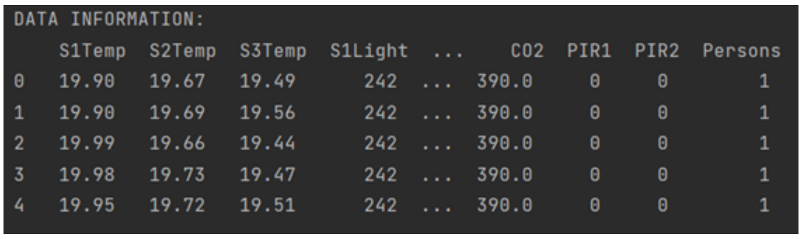


```
SHAPE:  (10129, 12)
NUMBER OF NULL VALUES:
 Date       0
Time       0
S1Temp     1
S2Temp     2
S3Temp     0
S1Light    0
S2Light    0
S3Light    0
CO2        1
PIR1       0
PIR2       0
Persons    0
dtype: int64
```

Figure 2: Number of null values.

We noticed that there were a few missing values that needed to be filled in, as shown in figure 2. This was solved with the function *create_subset()* - a data frame comprised of a subset of the nine initial features that we are going to employ in our models.

Afterwards the *.interpolate()* function was used to fill the missing values with the average of the closest ones. This choice was made after thorough deliberation and data analysis, which led to the conclusion that, because the values were obtained at close intervals, they may be regarded as linearly connected.



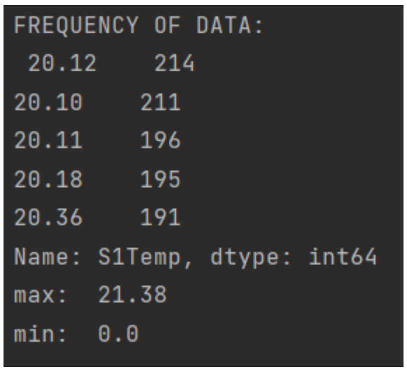
```
DATA INFORMATION:
    S1Temp  S2Temp  S3Temp  S1Light  ...    CO2  PIR1  PIR2  Persons
0   19.90   19.67   19.49      242  ...  390.0     0     0        1
1   19.90   19.69   19.56      242  ...  390.0     0     0        1
2   19.99   19.66   19.44      242  ...  390.0     0     0        1
3   19.98   19.73   19.47      242  ...  390.0     0     0        1
4   19.95   19.72   19.51      242  ...  390.0     0     0        1
```

Figure 3: Data information from a subset data frame.

1.2 Outliers detection

An identical formula to the one we used in the laboratory classes was employed to find outliers. For each parameter, the mean and standard deviation were calculated and the values that were *k*standard* deviations away from the mean were then substituted with the average of the values from the rows that were closest.

Trial and error were used to determine the value of k, and with the use of the Pandas library functions *.value_counts(),* .max() *and* .min(), we discovered that *k=6* was the value that was most indicative for all the features.



```
FREQUENCY OF DATA:
 20.12    214
20.10    211
20.11    196
20.18    195
20.36    191
Name: S1Temp, dtype: int64
max:  21.38
min:  0.0
```

Figure 4: S1Temp data frequency, minimum and maximum value.

1.3 Features Selection

Trial and error were used to determine which features would be employed, and the scatter plots of the data were examined to determine how they related to one another and whether they would be useful for future predictions.

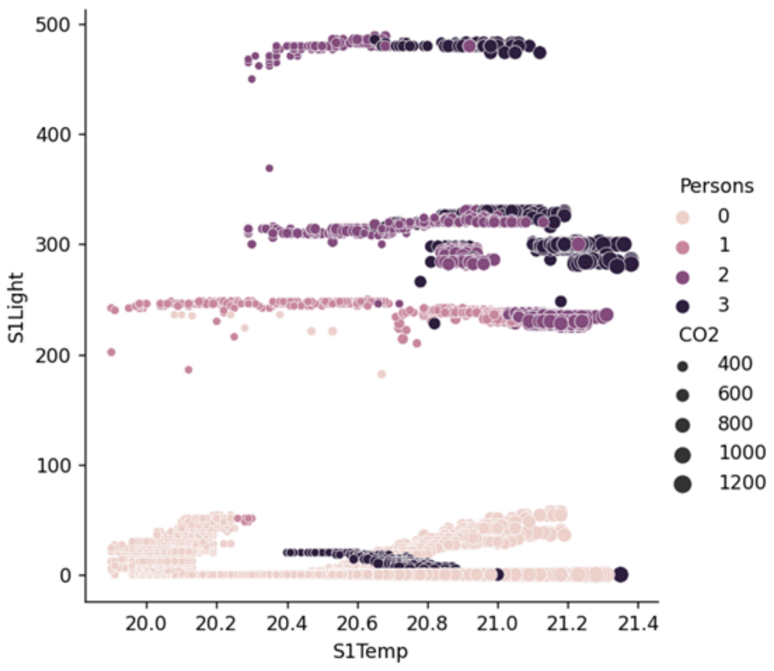


Figure 5: S1Temp, S1Light and CO2 features relation to number of people inside the room.

We came to the conclusion that just the Date and Time are not going to be used after performing many analyses to determine which features would be the best to base our models on. Given that people may have different schedules on other days, using both the date and the time may be biased and ultimately produce false results. With regard to the remaining features, we experimented with various combinations to determine which ones would produce better results and which ones are expendable. Although some of these combinations produced positive outcomes, in the end, this was the combination that was most effective.

1.4 Imbalanced data

We discovered that the data is unbalanced when we were analyzing it. The data frame contains 7996 records for no one in the room, 468 for one person, 884 for two people, and 781 for three people.

We attempted to implement a technique called SMOTE, which undersamples the majority classes and subsequently oversamples the minority classes, to address this issue. We did not employ it for the final findings and tests since, as compared to the results produced by using the data as it was retrieved from the data frame, we obtained the worst results. This was presumably due to the fact that, despite the data's imbalance, we had enough information from the minority classes to build useful models.

Another strategy we would have liked to try was to split the data from the majority class (when no one was in the room) into, say, five separate sets and use the methods we will discuss later, on each set to avoid the models from overfitting. Unfortunately we didn't have the time to implement this solution.

2 Part 1 - Proj1.1.py

2.1 Binary detection

First, a Binary column was made from the existing data, which took the value 1 for rooms with more than two occupants and the value 0 for rooms with two or fewer.

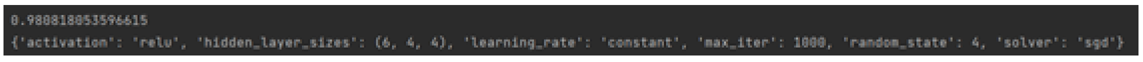
After the data is preprocessed, as explained in the previous section, the Neural Network-based classifier is going to predict if there are more than 2 people inside the room.

To start, a Y variable was generated that included the predicted binary results and an X variable that had all the features that would be used to forecast the binary result.

The function *train_test_split()* was used to split the data into a Train and a Test set. The Train set will be used to find the model that fits the data the best, and the Test set will be used to ensure that the supplied model accurately predicts yet-to-be-observed data.

The next step is to fit a *MinMaxScaler()* to the X_Train data, in order to transform it and the X_Test data. After this step is done it's time to train our model, taking advantage of our function *binary_problem()*.

The initial stage in training our model involved utilizing the built-in function *GridSearchCV()* to fine-tune the hyper-parameters of our Neural Network-based classifier, a *MLPClassifier()*. In order to determine which hyper-parameters matched our data the best without leading to overfitting, this built-in function did cross-validation on our training set.



```
0.988818053596615
{'activation': 'relu', 'hidden_layer_sizes': (6, 4, 4), 'learning_rate': 'constant', 'max_iter': 1000, 'random_state': 4, 'solver': 'sgd'}
```

Figure 6: Best hyper-parameters calculated after performing a Grid Search cross-validation.

The next step is to use the best hyper-parameters that have been derived to try and forecast the values for the Test set's unseen data. The scoring method used on the *GridSearchCV()* was the 'f1_micro', because it was the most indicated for unbalanced data. After many tests, we arrived at the best outcome shown in figure 7 and 8:

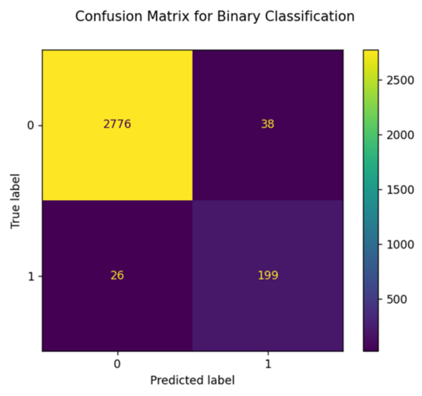
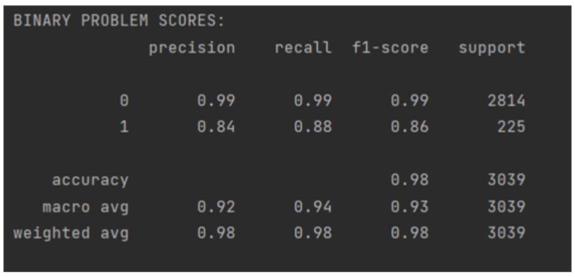


Figure 7: Binary confusion matrix.



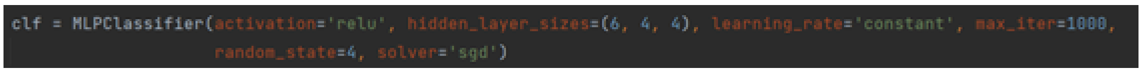
```
BINARY PROBLEM SCORES:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2814
           1       0.84      0.88      0.86       225

    accuracy                           0.98      3039
   macro avg       0.92      0.94      0.93      3039
weighted avg       0.98      0.98      0.98      3039
```

Figure 8: Binary classification report.

These results were achieved by using the hyper-parameters shown in figure 9:



```
clf = MLPClassifier(activation='relu', hidden_layer_sizes=(6, 4, 4), learning_rate='constant', max_iter=1000,
                    random_state=4, solver='sgd')
```

Figure 9: Neural Network-classifier used to predict binary data.

As we can see, our model presents decent scores in all categories: precision - with 0.84, meaning we get predict correctly 84% of the times when there are more than two people in the room; recall - the percentage of true predictions out of all predictions that were true, is also with a good value with 88%. In this case, the accuracy is not relevant since the data is unbalanced.

We are satisfied with the results of this model. All the times that there is no one in the room, we can predict it, and more than 80% of our predictions are correct when there is more than two people in the room.

2.2 Multi-class detection

This challenge and the preceding one were similar in many ways, at least in terms of how the data were prepared, how the hyper-parameters were tuned, and how the model was chosen. The X variable remained the same as on the Binary detection and the Y variable, which was now linked to information from the data in the people column of the data frame and provided the precise number of individuals in the room at that particular entry, altered.

The data went through the same steps as before in terms of the separation into a Training set and a Testing set as well as the *GridSearchCV()* process. After many tests, we arrived at the best outcome shown in figure 10 and 11:

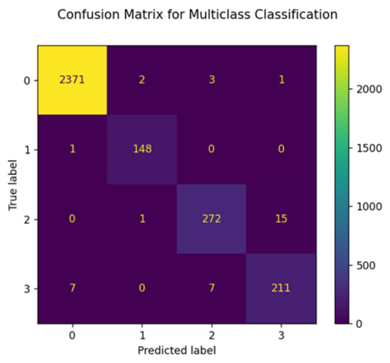
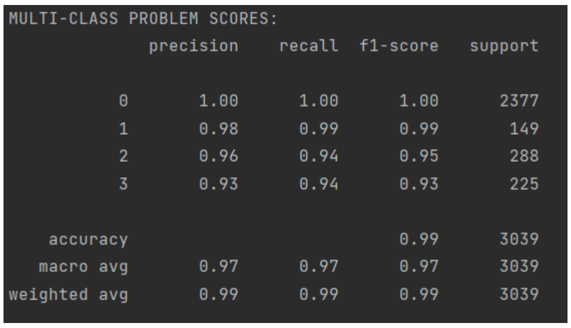


Figure 10: Multi-class confusion matrix.



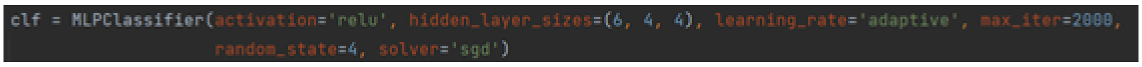
```
MULTI-CLASS PROBLEM SCORES:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2377
           1       0.98      0.99      0.99       149
           2       0.96      0.94      0.95       288
           3       0.93      0.94      0.93       225

    accuracy                           0.99      3039
   macro avg       0.97      0.97      0.97      3039
weighted avg       0.99      0.99      0.99      3039
```

Figure 11: Multi-class classification report.

These results were achieved by using the hyper-parameters shown in figure 12:



```
clf = MLPClassifier(activation='relu', hidden_layer_sizes=(6, 4, 4), learning_rate='adaptive', max_iter=2000,
                    random_state=4, solver='sgd')
```

Figure 12: Neural Network-classifier used to predict multi-class data.

Our model of Multi-class classification was very good. Our precision values are all above 93%, and the f1-score is also strong.

We are very satisfied with the results obtained with this model.

3 Part 1 - TestMe.py

Both the Neural Network model and the scaler built on the Proj1.1.py file are uploaded to this file. In turn, it also receives from the terminal the name of a data frame, which will be responsible to predict the Multi-class problem. Therefore, with specific unseen data provided as a terminal input, the TestMe.py will output the confusion matrix of the occupancy (0 people; 1 person; 2 people; and 3 people), the Precision and Recall of each class, as well as the overall Macro-Precision, Macro-Recall, and Macro-F1.

4 Part 2 - Proj1.2.py

4.1 Fuzzy Rule based classifier

We used the same preprocessing techniques as before to develop a fuzzy logic system because the data was still the same. However, since we needed to develop our model with the fewest possible rules, we had to cut back on the number of features we employed.

We ultimately decided to use only the Temperature, Light, and PIR capabilities after carefully weighing all the features, including how the CO2 was changing and the time of day. The PIR is helpful in that if it detected many people moving, both of them were more likely to be active. We put all the temperature values to one variable, the light to another, and lastly the PIR to another.

Lastly, to create our membership functions and consequent rules, we used the following plots to make our considerations:

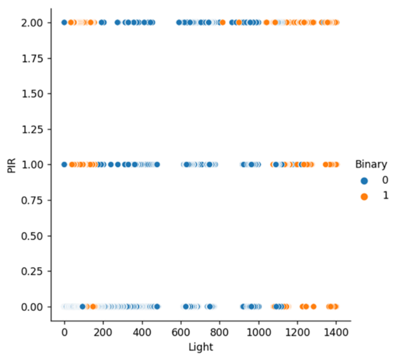


Figure 13: Scatter plot of PIR with Light.

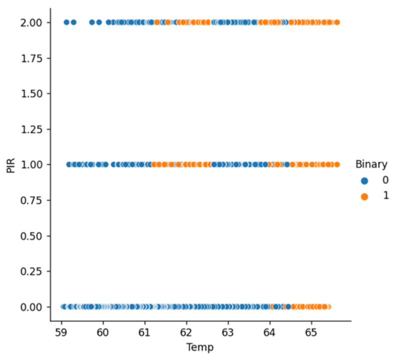


Figure 14: Scatter plot of PIR with Temp.

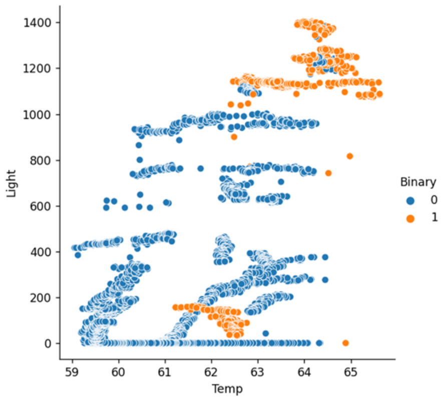


Figure 15: Scatter plot of Temp with Light.

We determined the following membership functions by examining the preceding plots:

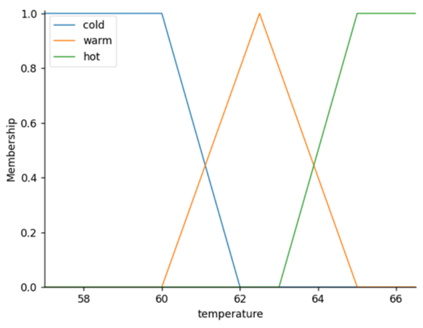


Figure 16: Temperature membership function.

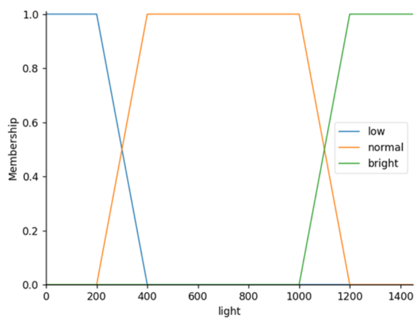


Figure 17: Light membership function.

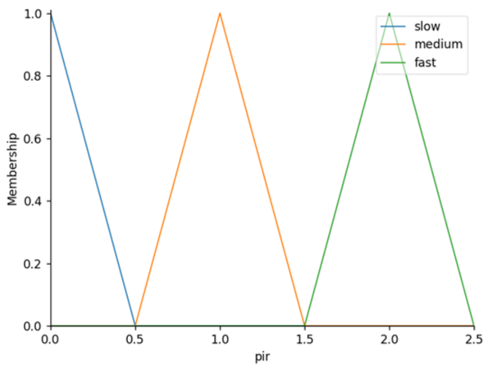
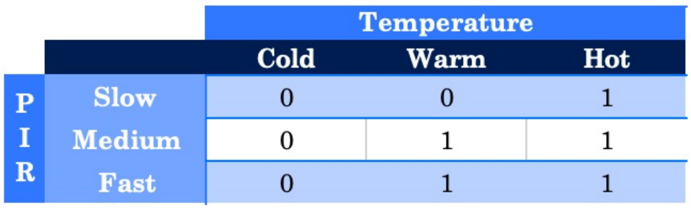


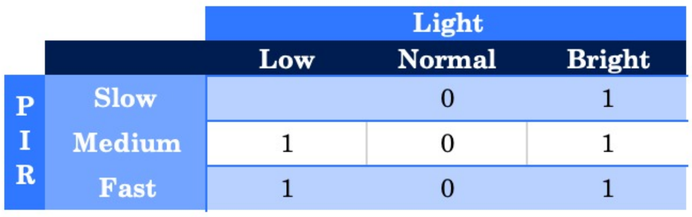
Figure 18: PIR membership function.

And derived the following rules:



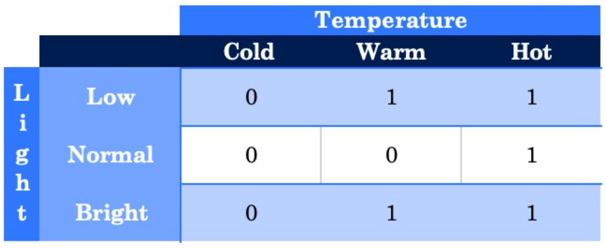
		Temperature		
		Cold	Warm	Hot
PIR	Slow	0	0	1
	Medium	0	1	1
	Fast	0	1	1

Figure 19: Rules that relate Temperature with PIR.



		Light		
		Low	Normal	Bright
PIR	Slow		0	1
	Medium	1	0	1
	Fast	1	0	1

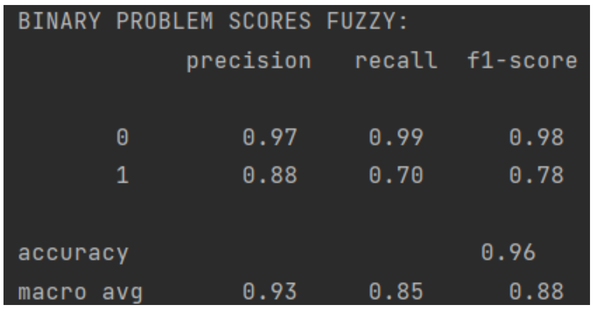
Figure 20: Rules that relate Light with PIR.



		Temperature		
		Cold	Warm	Hot
Light	Low	0	1	1
	Normal	0	0	1
	Bright	0	1	1

Figure 21: Rules that relate Temperature with Light.

Taking the membership functions into account and the analysis of the scatter plots, we were able to create the set of rules that were going to be used to predict the output, in this case, the response to the fact that there were more than two people on the room. Afterwards, we just took the information regarding the Temperature, Light and PIR of the data frame, and sent it to our Fuzzy Rule classifier. By comparing with the correct results, we obtained the following scores:



```
BINARY PROBLEM SCORES FUZZY:
              precision    recall  f1-score


           0       0.97      0.99      0.98
           1       0.88      0.70      0.78


accuracy                             0.96
macro avg          0.93      0.85      0.88
```

Figure 22: Fuzzy rule classification report.

In accordance with the project assignment and in order to compare results, we also developed a Neural Network classifier that makes use of the same features as our fuzzy classifier, trained the model, and made predictions while taking into account all the previously described techniques. The results were as follows:

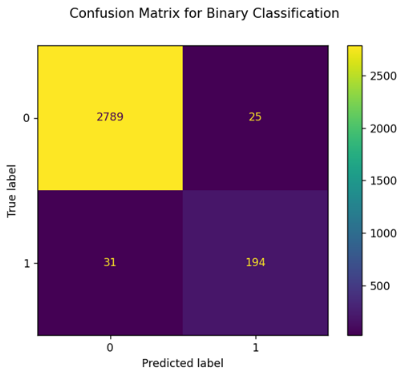
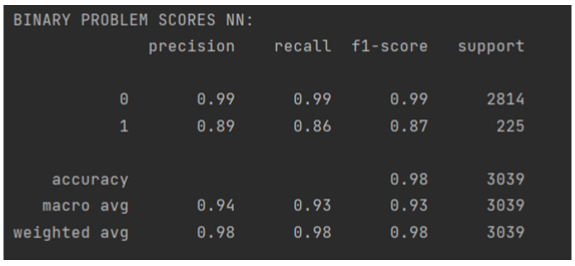


Figure 23: Neural Network binary confusion matrix.



```
BINARY PROBLEM SCORES NN:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2814
           1       0.89      0.86      0.87       225

    accuracy                           0.98      3039
   macro avg       0.94      0.93      0.93      3039
weighted avg       0.98      0.98      0.98      3039
```

Figure 24: Neural Network binary classification report.

We can clearly notice some differences in the recall and f1-score of both models.

The Binary solution had a superior score than the Fuzzy System. However, its precision was equivalent. The Fuzzy System, however, it's a cheaper and less heavy computationally model, that can obtain decent results.

5 Conclusions

This problem was a fun challenge. The preprocessing was a very important part of our project, so we dedicated some time to it. We think it compensated, since the values we obtained with the Neural Network classifier were all positive, having very good results in the Multi-Class problem. The Fuzzy Rule System was also a successful case, having good precision and overall score.

In the end, the outcomes produced by our Neural Network classifier were superior to those of our Fuzzy Rule System. There are many possible causes for this, but the most significant one is probably that the features we selected might not have been the best ones given that there was some output value overlap, as seen in Figure 14 when the PIR is 1 or 2 and the majority of the binary values between the temperatures of 63 and 64 are 0. Another possible cause is that the values of the Temperatures and Light membership functions could have been subjected to better tuning.

To finish, we believe that our models are decent and that our results are good.