Ex-Fuzzy: A library for symbolic Explainable AI through Fuzzy Logic Programming

Ex-Fuzzy: A library for symbolic Explainable AI through Fuzzy Logic Programming

Javier Fumanal-Idocin[a], Javier Andreu-Perez[a]

[a] *Centre for Computational Intelligence, School of Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom*

Abstract

Understanding the decisions taken by machine learning systems is instrumental in their deployment in real-world systems, as it enables responsible decision-making, fosters trust, and facilitates debugging and improvement. The research field devoted to studying the techniques that explain and illustrate those decisions is called explainable AI. Fuzzy logic, with its interpretable fuzzy rule-based inference, has emerged as a popular tool for Explainable AI because of these interpretable classifiers. However, current fuzzy logic libraries provide limited inference capabilities and integration to machine learning or are only available in the Java or R language, which makes their integration with the standard machine libraries in Python challenging. This paper describes a software library that contains a Python implementation to perform fuzzy inference using different kinds of fuzzy sets, with a special focus on result visualization. This library follows the scikit-learn programming interface, enabling researchers to utilize it with minimum fuzzy logic background seamlessly. This toolkit unveils novel tools for programming fuzzy systems that are learnable using machine learning methods, leading to data-powered systems that maintain full transparency and accountability, accessible to virtually anyone without specialized AI training. The interpretability of these systems makes them highly valuable in industries like healthcare, law, and security.

Keywords: Fuzzy Logic, Python Software, Fuzzy Rules Classifier, Explainable AI

1. Introduction

Fuzzy logic is one popular tool used in the field of AI to solve classification and regression problems [1]. The basis of fuzzy logic consists of giving real numbers as degrees of truth instead of the classic true or false values used in classical logic. This formalization naturally models concepts whose boundaries or definitions are not always exact and can lead to ambiguities [2]. One of the most important concepts in fuzzy logic is the idea of a linguistic variable, which consists of mapping an imprecise term that appears in natural language to a fuzzy membership [3]. For example, when we say "the submission date was late", we are implying that the "submission date" variable took a linguistic value as "late". Using linguistic variables, we can model the imprecise reasoning that people use. They can also be used to build explainable classifiers, which usually take the form of Fuzzy Rule-Based Classifiers (FRBC) because they detect patterns that take the form of a series of antecedents and one consequent:

$$\text{IF } \mathbf{x}_1 \text{ is } \mathbf{a}_{j1} \ldots \mathbf{x}_n \text{ is } \mathbf{a}_{jn} \text{ THEN class } j \text{ for } j = \{1, \ldots, c\} \tag{1}$$

where $\mathbf{x}$ is a multidimensional vector, j is the consequent class, $\mathbf{a}_j$ is an antecedent linguistic value for class j, and c is the number of different classes to discriminate.

However, besides the popularity of fuzzy logic, a consensus library for a widespread implementation does not exist. Previous research in FRBCs has led to a wide range of algorithms implemented in Java, which is not easily integrated with other tools used in scientific programming, such as numpy or sci-kit learn [4, 5, 6]. In the same way, fuzzy libraries written in Python lack important functionalities, like missing a proper FRBC implementation [7]. MATLAB fuzzy toolbox is also a popular library used in fuzzy developments, but it only implements the Adaptive-Network-based Fuzzy Inference Systems (ANFIS), which is very different from the evolutionary algorithms that are used in the state-of-art FRBCs up to this day. Although there is some software to implement them [8], they lack a coherent interface with other standard scientific libraries.

To solve these problems, we present the "ex-Fuzzy" library in this paper. This is a library written in Python that follows a similar interface to other machine learning libraries, which should ease its use for users who are non-familiar with fuzzy logic. It provides the necessary tools to implement an FRBC using a genetic algorithm to optimize its parameters. It also provides

different tools to visualize the rules and linguistic variables obtained in order to help explain the decisions taken by the FRBC computed. It also supports the use of different kinds of fuzzy sets: classical, interval-valued, general type 2 and temporal fuzzy sets.

The rest of the paper is as follows. In Section 3, we describe the software and show various examples of use cases. In Section 4, we compare our software module with other fuzzy implementations publicly available. Finally, in Section 5, we give our conclusions for this work.

2. Motivation of an XAI toolbox based on Fuzzy Logic

Explainable Artificial Intelligence is a hot topic nowadays in artificial intelligence as many intelligent systems aim to be transparent and accountable. There are toolboxes out there that present to distil or posthoc interpretation of the output of deep neural networks (purely black model models) [9, 10, 11] or analyzing gradients [12]; however, there's an important area of research which is purely explicit machine learning methods, such as those powered by human-understandable nested structures like graphs of threes, and of course simple propositional rules that link symbolic linguistic representations. Fuzzy sets are capable of encapsulating symbolic conceptual information about a domain, while fuzzy logic provides a significant methodological approach for artificial intelligence and machine learning, requiring total transparency in their reasoning process. Fuzzy logic has been successfully employed with competitive performance in areas such as neuroscience [13], human-computer interaction [14, 15], intelligent environments [16]. There are a few compelling libraries that can help the development of fuzzy systems [17, 18, 19, 20]. However, this 'ex-Fuzzy' fills a specific niche that the initial library missed, allowing for the development of fuzzy systems that are learned from data where explainability is the core focus. This library was crafted in Python to seamlessly integrate with machine and deep learning libraries, as well as data science stacks.

3. Software description

3.1. Software architecture

The ex-fuzzy library is formed of a series of modules containing Python classes and functions to implement the core ideas of approximate reasoning. An overview of the different functions and modules implemented can be seen

in the online documentation available. The three key modules implemented are:

- fuzzy_sets.py: this module contains the classes to implement fuzzy sets and fuzzy variables. An object of the FS() (fuzzy set) class consists of a fuzzy membership, a name and a domain. This library supports both Type 1 (T1), Interval-valued (IV), and General Type 2 (GT2). Fuzzy variables are objects that contain a list of fuzzy sets that represent the distinct linguistic variables present in that fuzzy variable.

- rules.py: this module implements fuzzy rules and fuzzy rule bases. Each Rule() object consists of a series of antecedents, which are fuzzy sets and a consequent. This consequent can be another fuzzy set or a class. Another form to represent rules is the RuleSimple() class, which is a list of integers that corresponds to the linguistic variables in a fuzzy variable. RuleSimple() objects are used within RuleBase() objects, in order to avoid redundant computations. When dealing with a classification problem, we also use MasterRuleBase() which consists of a list of RuleBase() objects, one per each consequent class. Depending on the type of fuzzy set used, we have implemented a different class, RuleBaseT1(), RuleBaseT2() and RuleBaseGT2(), which inherit from the RuleBase() class.

- evolutionary_fit.py: this module implements the BaseFuzzyRulesClassifier() class, which is an FRBC classifier that uses a genetic algorithm to optimize the rules obtained for the classification problem. Some constrains like the number of rules or the maximum number of antecedents can be specified. In order to compute the genetic optimization, we use the Pymoo library [21]. The class FitRuleBase() implements an object of the class Problem() in the Pymoo library, which enables the use of this library to solve this problem.

- utils.py: contains a series of utilities to compute fuzzy partitions using quantiles extracted from empirical data for all the different fuzzy sets supported. It also contains a series of functions to operate with temporal data, like discretizing a continuous temporal variable and assigning each sample to a different time moment.

- classifiers.py: this module implements different types of fuzzy rule classifiers based on the BaseFuzzyRulesClassifier class. There are three of

these. RuleMineClassifier, which first computes all the possible rules with a minimum of support, confidence and lift measures, and then looks for the optimal subset [22, 23]. FuzzyRulesClassifier performs a two step genetic optimization, first it computes a large set of good rules, and the optimizes them within the number of rules and antecedents constraints. RuleFineTuneClassifier combines the previous approaches. First, it computes the rules with minimum support, confidence, and lift and then performs the two-step optimization described in FuzzyRulesClassifier.

- temporal.py: shows an example of how to extend the ex-fuzzy library. In this case, to support temporal fuzzy sets [23], which also requires the addition of temporal fuzzy variables and temporal fuzzy rule bases.

The rest of the modules contain methods to compute the centroid of IV datasets (centroid.py); evaluate and FRBCs metrics like support, confidence and dominance score (eval_rules.py); evaluate classification in general (eval_tools.py); persistence of the rules obtained (persistence.py); mine rules with minimum values of support, confidence and lift (rule_mining.py) and visualize the results and patterns discovered (vis_rules.py).

3.2. Software functionalities

Ex-Fuzzy implements a wide set of tools to analyze fuzzy rule systems and visualize its results. Additionally, it also provides with functions to generate and optimize fuzzy partitions from the data and to evaluate the results obtained using the classifiers (Figure 1). The library includes support for different types of fuzzy sets: T1, T2, GT2 and temporal fuzzy sets.

Fuzzy classification is supported using a genetic algorithm to search for the rules obtained for each dataset (Figure 2). This optimization process is implemented using the Pymoo library [21]. The fitness function is the Matthew Correlation coefficient:

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2}$$

where TP is true positive, TN means true negative, FP is false positive, and FN is false negative. We also add two conditions to regularize the size of the rule bases. One for minimizing the number of rules, and another one to minimize the number of antecedents per rule:

$$l_1 = \frac{1}{R}\sum_{i=1}^{R}\frac{nAnts(r_i)}{\text{maxAnts}}, \tag{3}$$

$$l_2 = \frac{1}{R}\sum_{i=1}^{R} ds(r_i) > h, \tag{4}$$

where $ds(r)$ is the dominance score of the rule r, h is the minimum dominance score threshold, and R is the number of rules in the rule base.

We combine them using a convex combination between the MCC and the sum of these two additional losses, so that the final fitness function f is:

$$f = 0.95 * MCC + 0.05 * (l_1 + l_2). \tag{5}$$

The library also supports optimizing the linguistic variables in the optimization process and using a custom fitness loss.

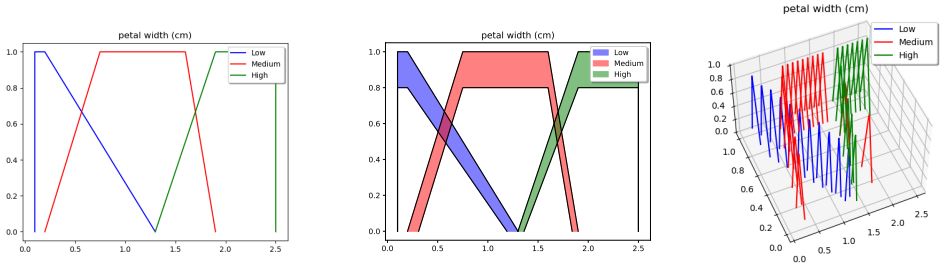
Once an FRBC has been trained, we can visualize the rules obtained by printing them (Figure 3) or exporting them to latex tabular format (Figure 4). We can also show the importance of each rule (For more details about this, see reference [23]) and plot the antecedents in a network structure (Figure 5). This network structure can also be exported in the gexp format used in the Gephi library [24].

Ex-Fuzzy can also easily export and import rules from plain text using the format displayed in Figure 3. This can be especially useful to reuse rules without using binary formats. Besides, it can also be used to create rule bases manually.

The library comes with a series of demos so that users can easily familiarize themselves with these functionalities.

4. Comparison with available modules

Most of the previous software regarding fuzzy rules has been developed using Java. One of the most notable libraries in this sense is Juzzy [18]. Juzzy contains the tools to model T1 and T2 fuzzy sets, focused on regression and control problems. However, no classification algorithm is provided. It was originally designed to be used in cloud/web applications, which is why it was implemented in Java.



(a) T1 (b) IV (c) GT2

Figure 1: **Fuzzy variable visualization** for three linguistic variables using three different kinds of fuzzy sets.

One of the most popular pieces of software used for the development of FRBC is Keel, which focuses on fuzzy evolutionary algorithms and its correspondent online dataset repository [26]. A fair amount of research in fuzzy evolutionary algorithms has used this tool as a basis for further developments [4, 27, 22], but its Java implementation has made it difficult to add in the same workflow as other scientific libraries, like Scipy. This has limited its impact outside the fuzzy community. Besides, Java implementation presents other problems, like the lack of a native interface for vectorized computing, which is extremely important for scientific computation [28].

The most important fuzzy library in Python is scikit-fuzzy [7], which consists of a series of tools to perform different kinds of fuzzy data mining. It also supports different kinds of membership and fuzzy sets. Besides, one of its most remarkable features is the implementation of the c-means algorithm to perform fuzzy clustering [29]. Scikit-fuzzy is a popular library. However, it is focused only on control and regression problems, which are less popular problems in the literature than classification. So, its penetration outside the fuzzy community has also been limited.

5. Conclusions

In this paper, we have presented the ex-Fuzzy library for Python. This library contains the necessary functions to perform fuzzy inference using different kinds of fuzzy sets and supports both regression and classification problems using genetic optimization. It also provides an interface comparable to other scientific libraries (i.e. scikit-learn), so that users that are not

```python
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split

import ex_fuzzy.fuzzy_sets as fs
import ex_fuzzy.evolutionary_fit as GA
import ex_fuzzy.utils as  utils
import ex_fuzzy.eval_tools as eval_tools

# Set hyperparameters
n_gen = 50
n_pop = 30
nRules = 15
nAnts = 4
vl = 3
tolerance = 0.01
fz_type_studied = fs.FUZZY_SETS.t1

# Load the iris dataset
iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.33, random_state=0)

# Train the FRBC
precompute_partitions = utils.construct_partitions(X,
    fz_type_studied)
fl_classifier = GA.BaseFuzzyRulesClassifier(nRules,
    precompute_partitions, nAnts,
vl, fz_type_studied, tolerance)
fl_classifier.fit(X_train, y_train, n_gen=n_gen, pop_size=
    n_pop)

eval_tools.eval_fuzzy_model(fl_classifier, X_train, y_train,
    X_test, y_test,
plot_rules=True, print_rules=True, plot_partitions=True)
```

Figure 2: **Classification example**. Code example required to run a FRBC in the Iris dataset.

```
Rules for consequent: 0
----------------
IF sepal width (cm) IS High WITH DS 0.06609379048467275
IF petal length (cm) IS Low WITH DS 0.33312911495032443
IF petal length (cm) IS Low WITH DS 0.33312911495032443

Rules for consequent: 1
----------------
IF petal length (cm) IS Medium AND petal width (cm) IS Medium
    WITH DS 0.8615857305145669

Rules for consequent: 2
----------------
IF sepal width (cm) IS Medium AND petal length (cm) IS High
   WITH DS 0.39182329323731413
IF sepal width (cm) IS Low WITH DS 0.10615268772642006
IF petal length (cm) IS Medium WITH DS 0.018046949473796342
IF sepal width (cm) IS High AND petal length (cm) IS High
   WITH DS 0.19932835296119167
```

Figure 3: **Rules output example**. Some of the rules obtained as a result of executing the code in Figure 2.

Consequent	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	DS	Acc
1	●	●	●	●	0.23	0.86
	●	●	●	●	0.45	1.00
2	●	●	●	●	0.23	0.94
3	●	●	●	●	0.11	0.67
	●	●	●	●	0.25	0.67

Figure 4: **An example of a rulebase exported for tabular format in latex.** ● → low, ● → medium, ● → high, ● → irrelevant. DS stands for dominance score, and Acc. for the accuracy obtained by each rule in the samples where it fired. Colors are only supported when using three linguistic labels.

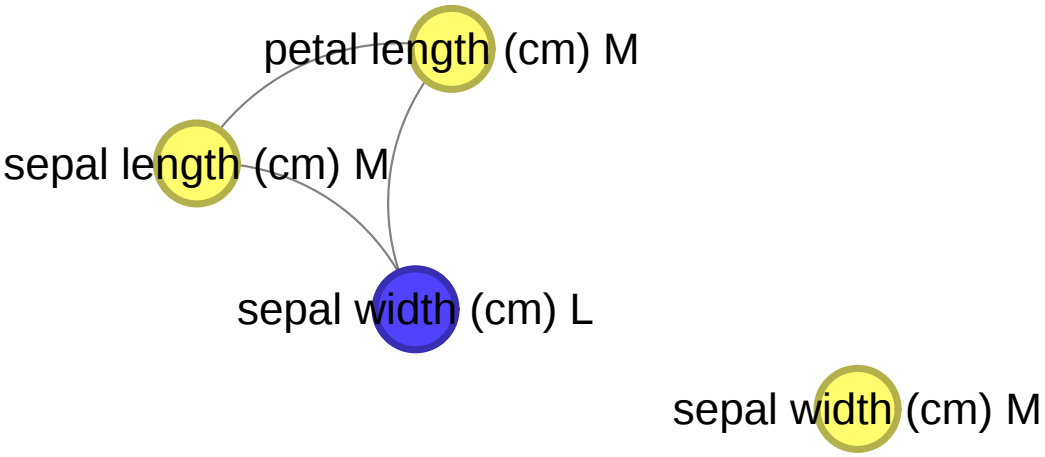


Figure 5: **Network visualization** of the antecedents of the rules obtained in Figure 3 using the Gephi tool [24]. It is also possible to use the networkx library to visualize the network, but the creators of this library do not recommend using it for visualization [25]

Table 1: Code metadata

Nr.	Code metadata description	Please fill in this column
C1	Current code version	1.0.4
C2	Permanent link to code/repository used of this code version	`https://github.com/Fuminides/ex-fuzzy`
C3	Legal Code License	GPL
C4	Code versioning system used	Git
C5	Software code languages, tools, and services used	Python
C6	Compilation requirements, operating environments & dependencies	Python 3
C7	If available Link to developer documentation/manual	https://fuminides.github.io/ex-fuzzy/
C8	Support email for questions	j.fumanal-idocin@essex.ac.uk

familiar with fuzzy logic can easily train a FRBC just as any other kind of machine learning classifiers. Besides, we also provide a series of tools to easily compute fuzzy partitions, save and visualize results and rules, and export them to other tools such as network or Gephi.

We believe that ex-Fuzzy can be particularly useful to fuzzy researchers interested in classification problems and non-fuzzy researchers interested in the explainable properties of FRBCs.

6. Required metadata

See Table 1.

7. Acknowledgments

Javier Fumanal-Idocin research has been supported by the European Union under a Marie Sklodowska-Curie YUFE4 postdoc action.

The authors would also like to thank for their feedback and appraisal to: Iosu Rodriguez-Martinez, Asier Urio-Larrea, Mehrin Kiani, Hani Hagras and Humberto Bustince.

References

[1] J. M. Mendel, Uncertain rule-based fuzzy systems, Introduction and new directions (2017) 684.

[2] N. P. Archer, S. Wang, Fuzzy set representation of neural network classification boundaries, IEEE transactions on systems, man, and cybernetics 21 (4) (1991) 735–742.

[3] S. Porebski, Evaluation of fuzzy membership functions for linguistic rule-based classifier focused on explainability, interpretability and reliability, Expert Systems with Applications 199 (2022) 117116.

[4] J. A. Sanz, A. Fernández, H. Bustince, F. Herrera, Ivturs: A linguistic fuzzy rule-based classification system based on a new interval-valued fuzzy reasoning method with tuning and rule selection, IEEE Transactions on Fuzzy Systems 21 (3) (2013) 399–411.

[5] J. Hühn, E. Hüllermeier, Furia: an algorithm for unordered fuzzy rule induction, Data Mining and Knowledge Discovery 19 (2009) 293–319.

[6] J. Alcalá-Fdez, R. Alcala, F. Herrera, A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning, IEEE Transactions on Fuzzy systems 19 (5) (2011) 857–872.

[7] J. Warner, J. Sexauer, A. Unnikrishnan, G. Castelão, F. A. Pontes, T. Uelwer, laurazh, F. Batista, alexbuy, W. V. den Broeck, W. Song, T. G. Badger, R. A. M. Pérez, J. F. Power, H. Mishra, G. O. Trullols, A. Hörteborn, Jdwarner/scikit-fuzzy: Scikit-fuzzy version 0.4.2 (Nov. 2019). doi:10.5281/zenodo.3541386.
URL https://doi.org/10.5281/zenodo.3541386

[8] S. Mkhitaryan, P. Giabbanelli, M. K. Wozniak, G. Nápoles, N. De Vries, R. Crutzen, Fcmpy: a python module for constructing and analyzing fuzzy cognitive maps, PeerJ Computer Science 8 (2022) e1078.

[9] H. T. T. Nguyen, H. Q. Cao, K. V. T. Nguyen, N. D. K. Pham, Evaluation of explainable artificial intelligence: Shap, lime, and cam, in: Proceedings of the FPT AI Conference, 2021, pp. 1–6.

[10] M. Al-Zeyadi, J. Andreu-Perez, H. Hagras, C. Royce, D. Smith, P. Rzonsowski, A. Malik, Deep learning towards intelligent vehicle fault diagnosis, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–7.

[11] B. Gutiérrez-Serafín, J. Andreu-Perez, H. Pérez-Espinosa, S. Paulmann, W. Ding, Toward assessment of human voice biomarkers of brain lesions through explainable deep learning, Biomedical Signal Processing and Control 87 (2024) 105457.

[12] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, et al., Captum: A unified and generic model interpretability library for pytorch, arXiv preprint arXiv:2009.07896 (2020).

[13] J. Andreu-Perez, L. L. Emberson, M. Kiani, M. L. Filippetti, H. Hagras, S. Rigato, Explainable artificial intelligence based analysis for interpreting infant fnirs data in developmental cognitive neuroscience, Communications biology 4 (1) (2021) 1077.

[14] C. F. Vega, J. Quevedo, E. Escandón, M. Kiani, W. Ding, J. Andreu-Perez, Fuzzy temporal convolutional neural networks in p300-based brain–computer interface for smart home interaction, Applied Soft Computing 117 (2022) 108359.

[15] A. R. Andreu-Perez, M. Kiani, J. Andreu-Perez, P. Reddy, J. Andreu-Abela, M. Pinto, K. Izzetoglu, Single-trial recognition of video gamer's expertise from brain haemodynamic and facial emotion responses, Brain Sciences 11 (1) (2021) 106.

[16] S. A. Cortez, C. Flores, J. Andreu-Perez, A smart home control prototype using a p300-based brain–computer interface for post-stroke patients, in: Proceedings of the 5th Brazilian Technology Symposium:

Emerging Trends, Issues, and Challenges in the Brazilian Technology, Volume 2, Springer, 2020, pp. 131–139.

[17] T. R. Razak, C. Chen, J. M. Garibaldi, C. Wagner, Designing the hierarchical fuzzy systems via fuzzyr toolbox, in: 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, 2021, pp. 1–6.

[18] C. Wagner, Juzzy-a java based toolkit for type-2 fuzzy logic, in: 2013 IEEE Symposium on Advances in Type-2 Fuzzy Logic Systems (T2FUZZ), IEEE, 2013, pp. 45–52.

[19] P. D'Alterio, J. M. Garibaldi, R. I. John, C. Wagner, Juzzy constrained: Software for constrained interval type-2 fuzzy sets and systems in java, in: 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, 2020, pp. 1–8.

[20] D. Sharma, P. K. Gupta, J. Andreu-Perez, J. M. Mendel, L. M. López, A python software library for computing with words and perceptions, in: 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, 2021, pp. 1–6.

[21] J. Blank, K. Deb, pymoo: Multi-objective optimization in python, IEEE Access 8 (2020) 89497–89509.

[22] J. A. Sanz, D. Bernardo, F. Herrera, H. Bustince, H. Hagras, A compact evolutionary interval-valued fuzzy rule-based classification system for the modeling and prediction of real-world financial applications with imbalanced data, IEEE Transactions on Fuzzy Systems 23 (4) (2014) 973–990.

[23] M. Kiani, J. Andreu-Perez, H. Hagras, A temporal type-2 fuzzy system for time-dependent explainable artificial intelligence, IEEE Transactions on Artificial Intelligence (2022).

[24] M. Bastian, S. Heymann, M. Jacomy, Gephi: an open source software for exploring and manipulating networks, in: Proceedings of the international AAAI conference on web and social media, Vol. 3, 2009, pp. 361–362.

[25] A. Hagberg, P. Swart, D. S Chult, Exploring network structure, dynamics, and function using networkx, Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008).

[26] I. Triguero, S. González, J. M. Moyano, S. García López, J. Alcalá Fernández, J. Luengo Martín, A. L. Fernández Hilario, M. J. d. Jesús Díaz, L. Sánchez, F. Herrera Triguero, et al., Keel 3.0: an open source software for multi-stage analysis in data mining (2017).

[27] D. P. Pancho, J. M. Alonso, J. Alcalá-Fdez, L. Magdalena, Analyzing fuzzy association rules with fingrams in keel, in: 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, 2014, pp. 2352–2359.

[28] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al., Array programming with numpy, Nature 585 (7825) (2020) 357–362.

[29] J. Nayak, B. Naik, H. Behera, Fuzzy c-means (fcm) clustering algorithm: a decade review from 2000 to 2014, in: Computational Intelligence in Data Mining-Volume 2: Proceedings of the International Conference on CIDM, 20-21 December 2014, Springer, 2015, pp. 133–149.