Comparison of Mamdani Fuzzy Inference System Results Between MATLAB and Python Implementations

Amrul Hinung Prihamayu[1], Qiyqiy Fithri Awamirillah[2]
[1]Department of Islamic Economics, Faculty of Economics and Islamic Business, UIN Kiai Ageng Muhammad Besari Ponorogo, Ponorogo, 63492, Indonesia (amrul.hp@uinponorogo.ac.id)
[2]Department of Islamic Banking, Faculty of Economics and Islamic Business, UIN Kiai Ageng Muhammad Besari Ponorogo, Ponorogo, 63492, Indonesia (qiyqiyawamirillah@uinponorogo.ac.id)

Abstract

Planning in determining the amount of production decision-making for the next period depends on the remaining inventory from the previous period and also the estimated amount of demand in the next period. Total demand and supply are uncertain. Fuzzy logic is a technique to analyze uncertainty. The purpose of this study is to compare the results of a Mamdani fuzzy inference system for determining the optimal production quantity of chocolate products on two computing platforms: MATLAB and Python. Both implementations used an identical system design, consisting of the same membership functions, rule base, input-output ranges, aggregation method, and defuzzification method, and the Python implementation was executed with the Scikit-Fuzzy library on Google Colab. The test scenario used input values of production cost of Rp 800.00 per pack and product demand of 45,000 packs. MATLAB produced a crisp output of 64,100 packs, while Python resulted in 80,000 packs. This study shows an output difference of 24.8%, which is attributed to differences in numerical resolution of the universe of discourse and library-specific implementation of the aggregation and centroid defuzzification operations between the two platforms. These findings indicate that Python, using the open-source Scikit-Fuzzy library, offers an accessible and cost-effective alternative for fuzzy modeling, while careful verification of the implementation details remains necessary to ensure numerical consistency between platforms. As this comparison is based on a single test scenario, further validation using additional test cases is recommended.

Keyword: *fuzzy inference system, Mamdani, MATLAB, Python, scikit-fuzzy, production optimization, code generation, error correction, Google Colab*

I. INTRODUCTION

Human life cannot be separated from various problems, including those in the manufacturing industry. Various industries are carried out by humans to meet needs, both on a local and global scale. Along with the development of the era, the industry continues to change and become modern. This industry is carried out to obtain profits. One of the industrial activities that is very important is manufacturing, both goods and services (Azadegan *et al.*, 2011).

At present almost all companies engaged in the manufacturing sector are faced with a problem, namely the existence of a very high level of uncertainty. This requires the company to plan or determine the amount of production, so that it can meet market demand on time and in the right amount. Thus it is expected that the company's profits will be optimal.(Wawan, Zuniati and Setiawan, 2021)

Maximum profit is obtained from maximum sales. Maximum sales means being able to meet existing demands. If the number of products produced by the company is less than the amount of demand, the company will lose the opportunity to get maximum profit. Conversely, if the number of products produced is much more than the amount of demand, the company will suffer losses. Therefore, planning the number of products in a company is very important in order to meet market demand appropriately and in the right amount. Factors that need to be considered in determining the number of products, among others: production costs and the estimated amount of demand for the next period (Kalaiarasi and Swathi, 2024).

Traditional optimization techniques, which often require precise input data and rigid decision frameworks, often fail to capture the ambiguous and imprecise nature of real-world environments. In this context, fuzzy logic emerges as a powerful computational paradigm for modeling human reasoning and handling complex, uncertain inputs in production systems(Wang, 2025).

Among various fuzzy inference techniques, the Mamdani method is widely used because of its intuitive rule-based approach and its ability to emulate linguistic decision-making. The Mamdani fuzzy inference system allows the combination of expert qualitative knowledge and quantitative data, making it highly effective for production planning, demand forecasting, and resource allocation (Macioł, Macioł and Mrzygłód, 2020; Kusumadewi *et al.*, 2025). Various studies have shown its effectiveness in manufacturing by reducing operational risks, optimizing production levels, and increasing responsiveness to market variability (Roh, Hong and Min, 2014; Onukwulu *et al.*, 2023; Sharma, Ahmed and Saini, 2025).

Historically, software such as MATLAB has been the main platform for developing and simulating fuzzy logic systems, thanks to its comprehensive toolboxes, established reliability, and interactive visualizations. However, MATLAB's commercial license imposes a financial burden on researchers and institutions, thus encouraging the exploration of alternative solutions. The rapid growth of open-source technology, especially in the Python ecosystem with libraries such as Scikit-Fuzzy and SoftPy (Dyczkowski and others, 2024; Campagner and Ciucci, 2025), has democratized access to advanced computing tools, further reducing the barriers to entry for fuzzy logic research and applications (Zhang and Valeo, 2025).

Nevertheless, challenges remain when porting a fuzzy inference implementation from a commercial platform to an open-source one. Problems such as syntax errors, mismatches between algorithm logic and implementation, and the need for iterative debugging require careful attention from researchers to ensure model correctness and reproducibility.

Comparing MATLAB and Python outputs for the same Mamdani fuzzy inference system is scientifically important for several reasons. First, MATLAB remains the de facto reference platform for fuzzy logic research due to its mature, extensively validated Fuzzy Logic Toolbox, yet its commercial licensing restricts access for many researchers and institutions, particularly in developing countries; establishing whether an open-source alternative such as Python's Scikit-Fuzzy reproduces MATLAB's numerical results is therefore a prerequisite for recommending it as a credible substitute in academic and industrial practice. Second, cross-platform numerical validation is a recognized methodological concern in computational science: identical model specifications (membership functions, rule bases, and defuzzification formulas) can yield different crisp outputs depending on library-specific implementation details, such as universe-of-discourse discretization and numerical integration methods, and quantifying this discrepancy is essential before such tools are used interchangeably in decision-support systems. Third, from a production-planning standpoint, an output difference of the magnitude observed in this study can materially change a manufacturer's production recommendation, so understanding whether and why MATLAB and Python outputs diverge has direct practical implications for the reliability of fuzzy-logic-based decision support. This study therefore contributes empirical evidence on the numerical consistency between a commercial and an open-source implementation of the Mamdani method, informing both platform-selection decisions and future efforts toward reproducible fuzzy modeling.

Despite the growing body of literature applying Mamdani fuzzy inference to production planning under uncertainty, most of these studies report results from a single computational platform, typically MATLAB, without examining whether an equivalent open-source implementation reproduces the same numerical outcome. This represents a specific research gap: the numerical consistency of Mamdani fuzzy inference systems across commercial and open-source computing environments has rarely been empirically verified, even though such verification is a precondition for treating the two platforms as interchangeable in practice. To be clear about the scope of this contribution, the novelty of this study does not lie in the fuzzy production-planning application itself, which builds on established modeling approaches, but in the cross-platform numerical verification exercise: systematically checking whether MATLAB and Python, when configured with an identical membership function, rule base, and defuzzification specification, produce consistent crisp outputs for the same input scenario. This positions the study primarily as a software/platform comparison that uses a production-planning case as its test problem, rather than as a contribution to production-planning methodology per se.

The purpose of this study is to conduct a systematic comparison between the Mamdani fuzzy inference system on MATLAB and Python in determining the optimal production quantity of chocolate products. The main aspects evaluated include the accuracy and consistency of fuzzy outputs, the implementation differences

between the two platforms, and the practical implications for costs and accessibility in academic environments.

By discussing these topics, the research provides insight into the suitability of open-source computing platforms to advance fuzzy logic research and support informed decision-making in production optimization under uncertainty.

II. THEORY

A. Fuzzy Logic

Fuzzy logic is a form of multi-valued logic derived from fuzzy set theory that deals with reasoning involving approximate, uncertain, or imprecise information. Unlike classical (crisp) logic, which operates strictly on binary true/false values, fuzzy logic allows truth values to range continuously between 0 and 1. This characteristic makes fuzzy logic particularly suited for modeling real-world problems where data is inherently ambiguous or incomplete (Kusumadewi and Purnomo, 2010).

Fuzzy logic was first introduced by Lotfi A. Zadeh in 1965 as an extension of classical set theory. In fuzzy logic, each element possesses a membership degree valued between 0 and 1, in contrast to classical sets that recognize only 0 (non-member) or 1 (member). The primary advantage of fuzzy logic lies in its ability to represent human knowledge in the form of linguistic rules that are intuitive and easy to interpret (Kusumadewi and Purnomo, 2010).

B. Fuzzy Set and Membership Function

A fuzzy set is a set in which every element has a varying degree of membership. A fuzzy set A defined over universe U is expressed as the collection of ordered pairs $\{(x, \mu A(x)) \mid x \in U\}$, where $\mu A(x)$ is the membership function that maps each element x to a value in the interval [0, 1] (Kusumadewi and Purnomo, 2010).

Several common forms of membership function are used in fuzzy logic. First, the triangular membership function (trimf) is defined by three parameters [a, b, c], where a and c are the lower and upper boundaries and b is the peak point with a membership degree of 1. Second, the trapezoidal membership function (trapmf) is defined by four parameters [a, b, c, d] forming a trapezoid-shaped curve. Third, the Gaussian membership function produces a symmetrical bell-shaped curve. The choice of membership function depends on the nature of the problem and the availability of expert knowledge (Ross, 2009; Kusumadewi and Purnomo, 2010). In this study, the triangular and trapezoidal functions were selected for the input and output variables because they require only a small number of parameters, are computationally efficient, and can be directly derived from expert-defined production cost, demand, and production-level thresholds, whereas a Gaussian function would offer no additional benefit for a system based on linear, threshold-defined linguistic categories.

C. Mamdani Fuzzy Inference System

The Mamdani fuzzy inference system is one of the most widely used fuzzy inference system (FIS) methods, first developed by Ebrahim Mamdani in 1975. It is also known as the min-max method. According to (Kusumadewi and Purnomo, 2010), the Mamdani method employs IF-THEN rules to represent expert knowledge in linguistic form, making it intuitive and easily interpretable by human operators.

The Mamdani inference process consists of four main stages. The first stage is fuzzification, which converts crisp input values into fuzzy values using predefined membership functions. The second stage is rule evaluation, which evaluates each fuzzy rule based on the fuzzy values of input variables; the AND operator applies the minimum function while the OR operator applies the maximum function. The third stage is aggregation, which combines the outputs of all fuzzy rules into a single fuzzy set. The fourth and final stage is defuzzification, which converts the aggregated fuzzy set back into a single crisp value as the final system output (Kusumadewi and Purnomo, 2010).

D. Defuzzification Method

Defuzzification is the final stage of the Mamdani fuzzy inference system that produces a definite crisp value from a fuzzy set. Several defuzzification methods are commonly used. The Centroid of Area (CoA) or Center of Gravity (CoG) method is the most widely applied, computing the output as the centroid of the area under the aggregated fuzzy curve. The Bisector of Area (BoA) method divides the area under the curve into two equal parts. The Mean of Maximum (MOM) method takes the average of all points with the highest membership degree, while the Smallest of Maximum (SOM) and Largest of Maximum (LOM) methods take the smallest and largest values among those with the highest membership degree, respectively (Ross, 2009; Kusumadewi and Purnomo, 2010).

This study employs the Centroid of Area (CoA) method because it produces a smoother and more representative output across the entire

fuzzy distribution. The mathematical formulation of CoA is expressed as:

$$z^* = \frac{\int_U z \,.\, \mu(z) dz}{\int_U \mu(z) dz} \quad (1)$$

where z* is the crisp output value, μ(z) is the membership degree at value z, and U is the output universe of discourse (Castillo and Melin, 2008; Ross, 2009; Kusumadewi and Purnomo, 2010).

III. METHOD

A. Research Framework

This study aims to compare the results of the Mamdani fuzzy inference system (FIS) when implemented in two different computational environments: MATLAB and Python, with MATLAB serving as the reference baseline due to its well-established reliability in fuzzy logic computation. To ensure a valid comparison, both implementations were built using an identical system design: the same membership function types and parameters, the same rule base, the same input-output universes of discourse, the same min-max aggregation method, and the same centroid defuzzification method, so that any difference in the resulting output could be attributed to platform- and library-level implementation rather than to differences in model configuration.

The main steps of this research framework include the following stages: model design, MATLAB implementation, Python implementation, implementation testing and correction, and result evaluation. Google Colab was used as the execution platform for the Python-based model to ensure accessibility, reproducibility, and cost efficiency.

B. System Design

The fuzzy system was designed using the conventional Mamdani method, consisting of four major logical processes: fuzzification, rule evaluation, aggregation, and defuzzification. The model represents a decision-making process for determining optimal chocolate production based on two input variables production cost and product demand and one output variable production volume.

The system components are as follows:

1. Input Variable 1: Production Cost (Low, Standard, High)
2. Input Variable 2: Demand (Low, Normal, High)
3. Output Variable: Production Level (Decrease, Normal, Increase)

Each fuzzy set is defined with trapezoidal (trapmf) and triangular (trimf) membership functions distributed evenly over their universes of discourse:

1. Cost range: 500–1,500 (Rupiah per pack)
2. Demand range: 20,000–100,000 (packages)
3. Production range: 0–140,000 (packages)

The complete parameters of each membership function, identical in both the MATLAB and Python implementations, are presented in Table I.

TABLE I
Membership Function Parameters

Variable	Linguistic Term	Type	Parameters
Production Cost (Rp/pack)	Low	trapmf	[500, 500, 700, 900]
	Standard	trimf	[700, 1000, 1300]
	High	trapmf	[1100, 1300, 1500, 1500]
Demand (packages)	Low	trapmf	[20000, 20000, 40000, 60000]
	Normal	trimf	[40000, 60000, 80000]
	High	trapmf	[60000, 80000, 100000, 100000]
Production Level (packages)	Decrease	trapmf	[0, 0, 35000, 70000]
	Normal	trimf	[35000, 70000, 105000]
	Increase	trapmf	[70000, 105000, 140000, 140000]

The system uses three fuzzy inference rules:

1. If production cost is low and demand is high, then production increases.
2. If production cost is standard and demand is normal, then production is normal.
3. If production cost is high and demand is low, then production decreases.

The system uses three fuzzy inference rules, summarized in complete form in Table II.

TABLE II
Complete Rule Base of the Mamdani Fuzzy Inference System

Rule No.	IF Production Cost	AND Demand	THEN Production Level
R1	Low	High	Increase
R2	Standard	Normal	Normal
R3	High	Low	Decrease

C. MATLAB Implementation

The MATLAB implementation used the Fuzzy Logic Toolbox to define membership functions, rule bases, and inference structures.

The min–max method was applied for fuzzy intersection and union, while defuzzification employed the centroid technique to compute the crisp output.

The scenario tested involved a production cost of 800 Rupiah per pack and a demand of 45,000 packages. MATLAB's calculation produced a crisp output of 64,100 packages, representing an optimal mid-range production level.

MATLAB's surface and rule viewers were used to visualize the fuzzy relationships, confirming smooth nonlinear behavior consistent with Mamdani's formulation .

D. Python Implementation

The same system structure, membership functions, rule base, input-output ranges, and defuzzification method were replicated in Python using the Scikit-Fuzzy library, following the identical parameters used in the MATLAB model to ensure a like-for-like comparison.

However, the code was not executable on the first attempt. Several syntax and reference errors occurred due to:

1. Library import mismatches
2. Incorrect use of variable names between inputs and outputs
3. Formatting errors in rule declaration
4. Incomplete aggregation and defuzzification calls

Consequently, the researcher performed five iterative corrections (debugging cycles) before the Python program could run successfully. Once corrected and executed on Google Colab, the system yielded a crisp output of 80,000 packages.

This process highlighted the need for careful verification of library-specific syntax and function calls when porting a fuzzy inference model between computing platforms, to ensure logical consistency and functional validity.

E. Comparative Evaluation and Validation

A comparative analysis was conducted to evaluate the consistency of fuzzy logic results across both environments. The evaluation focused on:

1. Numerical Consistency examining the difference between defuzzified outputs (MATLAB vs. Python)
2. Model Integrity confirming identical definitions of fuzzy sets, rules, and domains in both systems
3. Implementation Effort evaluating the effort required to implement and validate the Python model relative to the MATLAB baseline

It should be noted that this comparison is based on a single test scenario (production cost of 800 Rupiah per pack and demand of 45,000 packages). This is a limitation of the present study, as a single scenario cannot fully characterize the behavior of either implementation across the entire input space; the consistency of the two platforms under a broader range of input combinations remains a direction for further validation.

IV. RESULTS AND DISCUSSION

A. MATLAB Calculation Results

The Mamdani fuzzy inference system was implemented using MATLAB's Fuzzy Logic Toolbox. For input values of production cost at 800 Rupiah per pack and demand of 45,000 packages, the inference engine, using the min–max method and centroid defuzzification, produced a crisp output of 64,100 packages as shown on Figure 3.

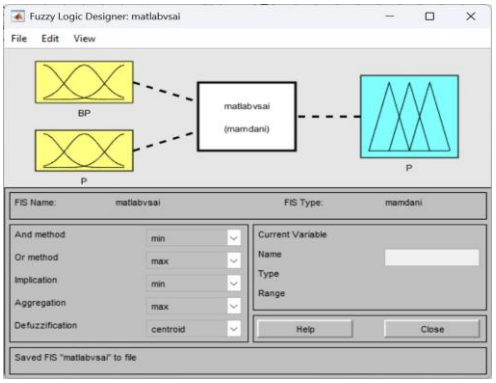

Figure. 1. Membership functions for input variables (production cost and demand) and output variables (production volume) in the MATLAB fuzzy system.

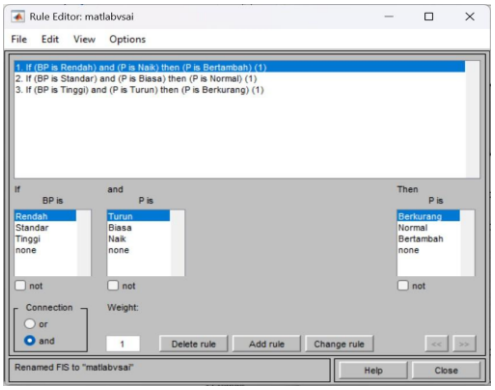

Figure. 2. Fuzzy rule base in the MATLAB implementation, visualizing the inference rules applied in the Mamdani system.

Figures 1, 2, and 3 provide a comprehensive visual overview of the MATLAB-based Mamdani fuzzy inference system developed in this study. Specifically, Figure 1 displays the membership functions for the two input variables production cost (ranging from 500 to 1,500 Rupiah per pack) and demand (ranging from 20,000 to 100,000 packages) along with the output variable, production volume (ranging from 0 to 140,000 packages).

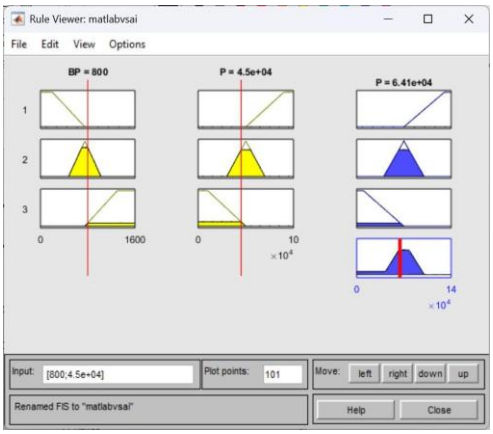

Figure. 3. Defuzzification output in MATLAB, demonstrating the conversion of fuzzy conclusions into a crisp production value.

Figure 2 presents the fuzzy rule base within MATLAB's Rule Editor, showing the three inference rules that govern the system: (1) low cost and high demand leads to increased production, (2) standard cost and normal demand results in normal production, and (3) high cost and low demand triggers decreased production. This visual representation confirms the correct logical structure and connectivity between inputs and output.

Finally, Figure 3 captures the defuzzification result using the centroid method for the test case (cost = 800 Rupiah, demand = 45,000 packages), yielding a crisp output of 64,100 packages. The surface and rule viewer outputs further validate the smooth, nonlinear decision-making behavior of the model.

Together, these figures offer a complete and transparent depiction of the MATLAB implementation, serving as a reliable benchmark for evaluating the accuracy and consistency of the Python implementation.

This value suggests the system recommends a moderate production level for the given scenario, balancing input cost and demand. MATLAB's graphical outputs confirmed smooth, expected fuzzy relationships among variables.

B. Python Calculation Results

The identical Mamdani system structure was implemented in Python using the Scikit-Fuzzy library. The initial Python script encountered multiple syntax and logic errors and required five iterative corrections before successful execution in Google Colab.

As shown in Figure 4, the initial Python script is presented, and Figure 5 shows the error messages encountered during the iterative correction process necessary to produce a functioning Python script.

After these corrections and using the same input values as MATLAB, Python calculated a crisp output of 80,000 packages, indicating a higher production suggestion under identical input conditions.

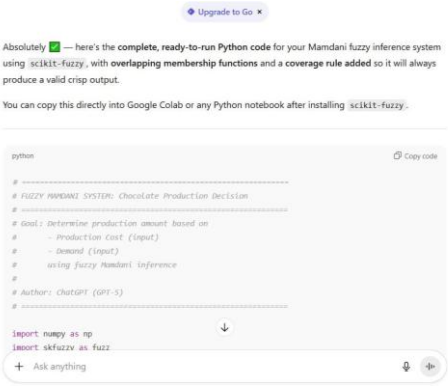

Figure. 4. Initial Python code developed for the Mamdani fuzzy computation.

Figure. 5. Screenshot of error messages encountered while running the initial Python code on Google Colab, highlighting the debugging efforts required for successful execution.

C. Comparative Analysis

The comparison between MATLAB and Python outputs is summarized in Table III.

TABLE III
Comparison of Matlab and Python Outputs

Platform	Output (Packages)	Diff. (%)
MATLAB	64,100	—
Python	80,000	+24.8

For the same inputs (production cost of 800 Rupiah per pack and demand of 45,000 packages), the MATLAB output of 64,100 packages results in a 24.8% difference compared to the Python output of 80,000 packages.

D. Error Correction and Code Reliability

The correction process for the Python script required five iterations to fix errors before the code was operational. This highlights the need for careful review and iterative testing when porting a fuzzy inference model to a new computing platform.

E. Discussion

Both implementations, MATLAB and Python, produce logically consistent fuzzy inference patterns but differ in absolute magnitude for production recommendations. MATLAB provides ready-to-use results with comprehensive built-in visualization, while Python with the open-source Scikit-Fuzzy library offers a more accessible and cost-effective alternative, requiring additional implementation effort to configure and verify.

Regarding the technical cause of the 24.8% difference, the two implementations used identical membership function parameters, rule base, and input-output ranges, so parameter or rule-base mismatch can be ruled out as a source of the discrepancy. Two more likely contributors remain. First, universe-of-discourse resolution: the numerical accuracy of the centroid calculation in a Mamdani system depends on how finely the output universe is discretized (the number of sample points used when performing the aggregation and centroid integral); MATLAB's Fuzzy Logic Toolbox and Scikit-Fuzzy may apply different default resolutions or sampling densities over the 0–140,000 output range, which shifts the computed centroid. Second, library-specific aggregation and defuzzification implementation: although both platforms nominally apply the max aggregation operator and centroid (center of area) defuzzification, the underlying numerical integration method (e.g., trapezoidal summation over discrete sample points versus a continuous closed-form calculation) can differ between MATLAB's toolbox and Scikit-Fuzzy, producing small but compounding differences in the final crisp output. Because this study used a single test scenario, it is not yet possible to isolate the precise contribution of each factor; systematically varying the output universe resolution and comparing intermediate aggregation results across additional test cases would be needed to fully attribute the source of the discrepancy, and is recommended for future work.

In this study, the use of two input variables, each using three linguistic variables, in the Mamdani fuzzy logic method to obtain output, requires four stages:

1. Formation of fuzzy sets
2. Application of implication function (rules)
3. Rule composition
4. Defuzzification

In this research, the writer uses two input variables and each input variable uses three linguistic variables. For further research, it can be done with more than two input variables and more than three linguistic variables.

V. CONCLUSION

The Mamdani fuzzy logic method for determining optimal chocolate production quantity requires four stages: fuzzy set formation, implication function application, rule composition, and defuzzification. Using two input variables (production cost and demand), each with three linguistic terms and an identical system configuration, MATLAB yields a crisp output of 64,100 packages, while the Python implementation, after five iterative corrections, produces 80,000 packages, a 24.8% difference. This difference is likely attributable to differences in universe-of-discourse resolution and library-specific aggregation and defuzzification implementation between the two platforms, rather than to any inherent unreliability of either platform. As this study is based on a single test scenario, these findings should not be generalized to conclude that either platform is more or less reliable than the other; further validation using multiple test scenarios is needed to draw firmer conclusions. Python, using the open-source Scikit-Fuzzy library, offers a more accessible and cost-effective alternative to MATLAB, though careful verification of implementation details remains necessary to achieve numerical consistency

across platforms. Future research may explore additional input variables, linguistic terms, and test scenarios to improve model robustness and further clarify the source of the observed output difference.

REFERENCE

Azadegan, A. *et al.* (2011) "Fuzzy logic in manufacturing: A review of literature and a specialized application," *IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews*, 41(2), pp. 258–270.

Campagner, A. and Ciucci, D. (2025) "softpy: A user-friendly Python library for soft computing," *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO 2025)*, pp. 115–118.

Castillo, O. and Melin, P. (2008) *Type-2 Fuzzy Logic: Theory and Applications*. Berlin, Heidelberg: Springer (Studies in Fuzziness and Soft Computing).

Dyczkowski, K. and others (2024) "Python library for interval-valued fuzzy inference," *SoftwareX*, 26, p. 101730.

Kalaiarasi, K. and Swathi, S. (2024) "Optimization of fuzzy inventory management in industrial processes using deep learning algorithms: A hybrid approach for enhancing demand forecasting and supply chain efficiency," *Advanced Engineering Letters*, 3(4), pp. 141–153.

Kusumadewi, S. *et al.* (2025) "Implementation of decision tree and Mamdani fuzzy inference system for Erythropoietin resistance prediction," *Biomedical Signal Processing and Control*, 104, p. 107496.

Kusumadewi, S. and Purnomo, H. (2010) "Aplikasi Logika Fuzzy untuk Pendukung Keputusan," *Yogyakarta : Graha Ilmu*, 2.

Macioł, A., Macioł, P. and Mrzygłód, B. (2020) "Prediction of forging dies wear with the modified Takagi–Sugeno fuzzy identification method," *Materials and Manufacturing Processes* [Preprint]. Available at: https://www.tandfonline.com/doi/abs/10.1080/10426914.2020.1747627 (Accessed: June 14, 2024).

Onukwulu, E.C. *et al.* (2023) "Transforming supply chain logistics in oil and gas: best practices for optimizing efficiency and reducing operational costs," *Journal of Advance Multidisciplinary Research*, 2(2), pp. 59–76.

Roh, J., Hong, P. and Min, H. (2014) "Implementation of a responsive supply chain strategy in global complexity: The case of manufacturing firms," *International Journal of Production Economics*, 147, pp. 198–210. Available at: https://doi.org/10.1016/j.ijpe.2013.04.013.

Ross, T.J. (2009) *Fuzzy Logic with Engineering Applications*. Hoboken, NJ: Wiley.

Sharma, Y.K., Ahmed, G. and Saini, D.K. (2025) "Comparative performance analysis of Mamdani and Sugeno fuzzy inference systems for sustainable cluster formation in WSNs," *Journal of Intelligent and Fuzzy Systems*, 49(5), pp. 1306–1332.

Wang, J. (2025) "Role of human collaboration in artificial intelligence with fuzzy based mathematical models and decision making problems," *Scientific Reports*, 15(1), p. 34029.

Wawan, W., Zuniati, M. and Setiawan, A. (2021) "Optimization of national rice production with fuzzy logic using Mamdani method," *Journal of Multidisciplinary Applied Natural Science*, 1(1), pp. 36–43.

Zhang, Z. and Valeo, C. (2025) "Fuzzy-based input method for uncertainty quantification in a deterministic model comparison with ChatGPT for peak flow prediction," *Journal of Hydrology X*, 28–29, p. 100208.