IPTEK The Journal of Technology and Science, 35(3), 2024 (e/pISSN:2088-2033, 0853-4098)

DOI: 10.12962/j20882033.v35i3.21992 | Received 9 Nov, 2024; Revised 2 Dec, 2024; Accepted 23 Dec, 2024

ORIGINAL RESEARCH

COMPARISON OF KNN, RANDOM FOREST, AND F-PSO ALGORITHMS ON SIMPLE FEATURE SCALING FOR AGILITY LEVEL CLASSIFICATION

Tri Yulianto Nugroho | Umi Laili Yuhana* | Daniel Siahaan

Departement of Informatics Engineering, Institut Teknologi Sepuluh Nopember, Surabaya 60111, Indonesia

Correspondence

*Umi Laili Yuhana, Dept of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. Email: yuhana@its.ac.id

Present Address

Gedung Teknik Informatika, Jl. Teknik Kimia, Surabaya 60111, Indonesia

Abstract

Classifying agility levels presents challenges due to variations in team members' personalities, roles, and undesirable behaviors. This study aims to enhance classification accuracy by comparing the performance of three algorithms: K-Nearest Neighbors (KNN), Random Forest, and Fuzzy-Particle Swarm Optimization (F-PSO) in classifying agility levels using simple feature scaling as part of the data preprocessing. Simple feature scaling is employed to ensure that all parameters are on the same scale, thereby improving the model's effectiveness in learning classification patterns. F-PSO was selected for its ability to perform adaptive global search optimization within a fuzzy environment, while KNN and Random Forest serve as benchmarks. The study involved 160 participants from various Scrum teams to evaluate the effectiveness of these algorithms. The parameters considered included team members' personalities (based on the Keirsey model), roles within the team, and the identification of negative behavior patterns (antipatterns). The results indicated that the F-PSO algorithm significantly outperformed KNN and Random Forest in terms of accuracy, improving from an average accuracy of 25% before optimization to 93.75% after applying F-PSO. This approach enables Scrum teams to identify and address obstacles affecting agility, facilitating earlier problem prediction and resolution, leading to more adaptive and effective teams.

KEYWORDS:

Agility Level, Anti-patternAntipattern KNN, Random Forest, Scrum, Simple Feature Scaling

1 | INTRODUCTION

The digital revolution, Industry 4.0, has necessitated substantial changes in work practices across various sectors[1]. Businesses must adapt swiftly to remain competitive in this rapidly evolving digital landscape[2]. Agility, a concept that originated in software development, has emerged as a critical focus for organizations across all domains[3].

Agility extends beyond a mere methodology; it embodies a mindset that empowers individuals and teams to operate more effectively[4]. The Agile Manifesto underscores the significance of human interaction and collaboration in software development, a process fundamentally driven by human efforts[5]. Consequently, effective teamwork is essential for the success of software projects[6].

Nevertheless, ineffective project management strategies, termed antipatterns, can inhibit development teams, resulting in increased costs, delays, and diminished product quality[7]. Identifying and addressing these antipatterns can improve team productivity and efficiency[8].

Prior research has investigated various methods for classifying agility levels, including K-nearest neighbors (KNN) and Random Forest algorithms. While these classification techniques have demonstrated promising outcomes, they often struggle with accuracy and adaptability when applied to complex and dynamic team environments. These limitations highlight the need for more advanced approaches to accommodate the nuances of team dynamics and individual personalities[9].

This research focuses on a comparative analysis of the KNN, Random Forest, and Fuzzy-Particle Swarm Optimization (F- PSO) algorithms for classifying agility levels, incorporating simple feature scaling during the data preprocessing phase. Simple feature scaling normalizes the data, ensuring each feature contributes equally to the model's performance. The primary contribution of this study lies in the evaluation and comparison of these algorithms regarding their effectiveness when simple feature scaling is applied in the preprocessing stage.

The proposed approach integrates personality classification based on the Keirsey Temperament Sorter[10] and team roles to predict potential occurrences of anti-patterns[11]. Fuzzy Logic and Particle Swarm Optimization mitigate ambiguity in agility classification, enabling early identification and minimizing the risk of anti-patterns[12].

Software development teams comprise various antipatterns with distinct responsibilities. System analysts are intermediaries between business needs and technical specifications[13]. Designers concentrate on the aesthetics and functionality of user interfaces[14]. Programmers write, test, and maintain software code[15]. Software testers ensure that applications are bug-free and meet quality standards[16].

Agile methodologies facilitate flexibility, allowing teams to respond swiftly to changing project requirements[3]. By segmenting work into short cycles and consistently gathering feedback, teams can make rapid adjustments and improve client collaboration[17]. Early identification of problems during development produces a higher-quality final product[18]. Understanding team members' strengths, weaknesses, and personalities is crucial for achieving optimal outcomes[19] and effectively managing conflicts[20].

This study contributes by developing a classification method utilizing advanced techniques (Fuzzy Logic for addressing data uncertainty and Particle Swarm Optimization for process optimization) and emphasizing the importance of personality and team interactions[21–25]. Assigning individuals to roles that align with their personalities and skills can improve efficiency, productivity, adaptability, and overall project effectiveness[21, 24, 26–32]. This research validates the proposed method through case studies and real-world applications, including implementing optimization algorithms in industry settings[28, 33].

2 | PREVIOUS RESEARCHES

Various studies have focused on the classification of agility levels in software development teams due to their impact on project efficiency and success. Agility emphasizes flexibility, collaboration, and adaptability in responding to changing project requirements. This approach relies on effective team roles, including system analysts, designers, programmers, and testers, each with specific responsibilities within the software development process[13, 14].

Various algorithms have been utilized to classify team agility levels. Previous research has employed KNN and Random Forest algorithms for this classification task. KNN is known for its simplicity and ability to handle non-linear data, while Random Forest provides advantages in overcoming overfitting and enhancing accuracy through an ensemble approach[12]. However, studies have also indicated that these algorithms have limitations when dealing with high complexity and dynamic variations in team data[11].

Research by Neill[16] and Settas and Stamelos[11] has shown that errors in team management are often influenced by antipatterns, which are recurring practices that negatively impact project effectiveness. Early detection of these antipatterns is crucial for boosting productivity and reducing the risk of project failure. Algorithms that address these challenges are needed to provide more accurate and reliable classification outcomes.

Fuzzy-Particle Swarm Optimization (F-PSO) has been proposed to enhance accuracy in classifying agility levels. F-PSO combines the global optimization strengths of Particle Swarm Optimization with the ability of fuzzy logic to handle data uncertainty (Picha & Brada, 2019). Using F-PSO in classification allows for better adaptive adjustments to data variations compared to classical methods such as KNN and Random Forest.

This research also highlights the importance of data preprocessing, where simple feature scaling is employed to normalize data so that each feature contributes equally to the model's performance. Feature scaling helps improve algorithm accuracy by ensuring that data is not distorted by differences in feature scales. This is highly relevant in comparative algorithm studies, ensuring that each algorithm is tested under comparable conditions.

By employing simple feature scaling in the data preprocessing stage, this study evaluates and compares the performance of KNN, Random Forest, and F-PSO algorithms in classifying agility levels. This evaluation aims to provide deeper insights into which algorithm is most effective for identifying potential antipatterns in software development teams, enabling early detection and mitigation to enhance overall project productivity and quality.

3 | METHOD

This study uses data from two sources: a previous research project involving students[2] and industry data on hard skills provided by PT Maulidan Teknologi Kreatif (accessed through the Rasyid Institute platform, www.rasyidinstitute.com). To ensure consistency, the data undergoes reprocessing using a simple feature scaling method, resulting in all values being scaled to a maximum of 100. This study compares the KNN, Random Forest, and Combined Fuzzy-PSO methods to determine agility levels.

KNN method starts with data pre-processing. The processed data is converted into a format suitable for the KNN algorithm. The preprocessing must determine parameter K. This parameter represents the number of nearest neighbors to be considered. It is determined based on testing or trial and error to achieve optimal results. KNN bases its decision on distance calculation. The distance between the data to be classified and the training data is calculated using a specific metric, such as the Euclidean distance. For classification (scoring), the new data is classified based on the majority class of its K nearest neighbors, or a score is calculated based on the average values of the nearest neighbors.

The Random Forest method uses training data to construct multiple decision trees in the ensemble. Each tree is trained using different subsets of the data (with bagging technique). Random Forest uses a random feature selection mechanism, where features are selected randomly at each split in the tree to minimize overfitting and increase tree diversity. Random forest also utilizes an aggregate prediction process. For each new input, all trees in the forest generate predictions, and the final result is determined through majority voting or averaging the predictions. The trained model is evaluated to ensure prediction accuracy and consistency in determining agility scores.

The combined Fuzzy-PSO method begins with input data bounds definition. The process identifies the minimum and maximum values in the processed data. The next process is fuzzification and fuzzy set definition, where data is converted into fuzzy values using predefined fuzzy sets. As for the rule formation and application, the method applies fuzzy logic rules to the data to produce uncertain or approximate outputs. The fuzzification results are transformed into exact numerical values in the defuzzification and control process. Finally, the agility scoring utilizes the Particle Swarm Optimization (PSO) algorithm. PSO is used to maximize

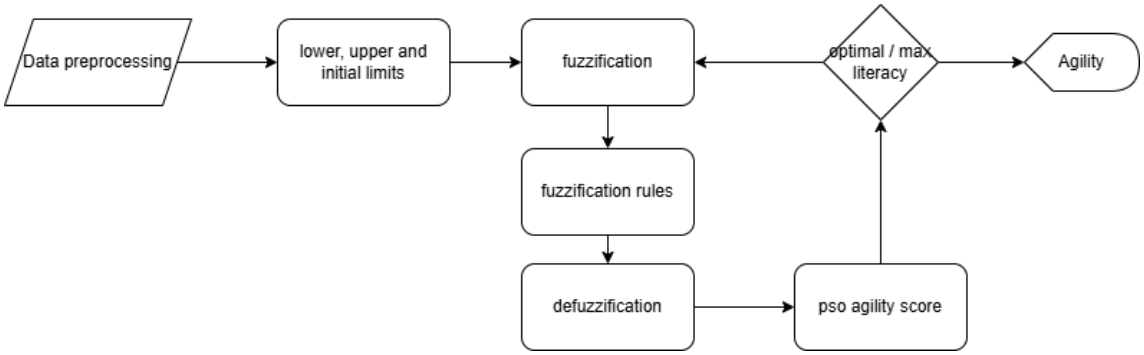

FIGURE 1 Illustrating the Fuzzy-PSO agility process.

TABLE 1 The distribution of datasets according to participant role and gender.

Gender	Role		Total
	Developer	Product Owner	
Man	95	21	116
Woman	37	7	44
Total	132	28	160

TABLE 2 The participant data according to the Keirsey Temperament Sorter (KTS) classification system.

Role	#Person	Role	#Person	Role	#Person	Role	#Person
Artisan	43	Guardian	42	Rational	22	Idealist	53
Promotor	7	Supervisor	6	FieldMarshals	9	Teacher	15
Crafter	4	Inspector	6	Masterminds	9	Counselor	14
Performer	16	Provider	17	Inventor	1	Champion	13
Composer	16	Protector	13	Architect	3	Healer	11

TABLE 3 The number of participants according to the anti-pattern types identified.

AntiPattern Type	#Participants	AntiPattern Type	#Participants
Warm Bodies	73	Ultimate Weapon	22
Doppelganger	48	Road to Nowhere	8
Technology Bigot	7	Fear of Success	21
Divergent Goals	134	Boiling Frog	35
Sidelining	1	Mediocracy	7
Buzzword Mania	19	Kiosk City	21
Institutional Mistrust	15	Ant Colony	70

the outcome of the complex mathematical model, with the final agility score calculated by optimizing the output from the fuzzy system. These processes are further explained in Figure 1 .

3.1 | Size of Dataset

Table 1 presents data collected from partners in 160 participants. This data encompasses information from 116 male participants and 44 female participants. Two Scrum roles are represented within the data set: 132 Developers and 28 Product Owners.

Table 2 provides a breakdown of the 160 participants from Table 1 based on the Keirsey Temperament Theory. The data shows that the largest group (53 people) identifies as Idealists, followed by Artisans (43), Guardians (42), and Rationals (22).

Table 3 further reveals that participants identified various recurring negative patterns known as antipatterns. These included Warm Bodies, Ultimate Weapon, Doppelganger, and others. In this identification of antipatterns, many participants exhibited multiple antipatterns. In total, 481 antipatterns were identified from the 160 participants.

TABLE 4 The example of data before and after pre-processing.

Team	ID	Data Type	Before		After	
			X_{max}	X_{old}	$X_{max\text{-}new}$	X_{new}
Project 1	2	Keirsey	400	230	100	57.50
		Antipattern	140	52	100	37.14
		Role	11	11	100	100.00

Drawing upon existing data, 160 students were categorized into 31 projects, with an average of 5 members per project. Subsequently, Keirsey Temperament Sorter data, Scrum data, and anti-pattern data for each team member will be collected.

3.2 | Data Processing

Each of these datasets undergoes pre-processing using the simple feature scaling method. This method transforms the values of variables or data into smaller ranges, typically between 0 and 1 or -1 and 1. By performing feature scaling, we can reduce the scale differences between variables, making data analysis easier and improving the performance of machine learning algorithms.

Example of Preprocessing, As illustrated in Table 4 , for the Keirsey data type, a student with ID "2" classifiPreprocessingject team "Project 1" has a maximum value (X_{max}) of 400 calculated by adding together the highest scores for each part of the Keirsey personality type, a minimum value (X_{min}) of zero, and an original value (X_{old}) of 230 from the Keirsey data. The standardized scale ($X_{max\text{-}new}$) is 100, which will then be normalized using Equation 1 to produce the normalized value (X_{new}).

$$X_{new} = \frac{X_{old} - X_{min}}{X_{max} - X_{min}} \times X_{max\text{-}new} \tag{1}$$

3.3 | Determining Agility Level

3.3.1 | KNN

Determining the level of agility using the K-Nearest Neighbors (KNN) algorithm is done through several stages. The first stage is data pre-processing, where the raw data is converted into a format suitable for processing by the KNN algorithm. This ppreprocessings data normalization, removing outliers, or converting categorical data to numeric so that it can be compared with existing training data. The second stage is determining the parameter (K), which is the number of closest neighbors used to determine the class or score of new data. The value (K) is usually determined through testing or trial and error methods to obtain optimal results.

After the parameter (K) is determined, the third stage calculates the distance between the data to be classified and the training data. This distance is important for determining the nearest neighbors of the new data. In the fourth stage, new data is classified or assessed based on the majority class of (K) its nearest neighbors or by calculating the average score of those neighbors. For example, if most of the nearest neighbors of the new data are in a certain class, then the new data will be classified into the same class.

Based on the results shown in Figure 2 , the data is divided into three intervals, namely 1 to 26 (low level), 33 to 63 (medium level), and 66 to 100 (high level), which indicate different groups or levels of agility. The values above each bar graph (71, 47, and 42) represent the results or scores calculated using KNN for each interval. This graph indicates a decrease in agility scores as the interval increases, so it can be used as a basis for determining a certain agility level based on the results of the KNN algorithm calculation.

3.3.2 | Random Forest

Determining the agility level using the Random Forest algorithm involves several stages. The first stage is constructing a decision tree, where the training data is used to build several decision trees. Each tree is trained using different data through the bagging technique, which helps increase the diversity of the trees and thus improves the overall prediction accuracy. The second stage involves random feature selection at each branch in the tree, where several features are randomly selected to reduce the risk of

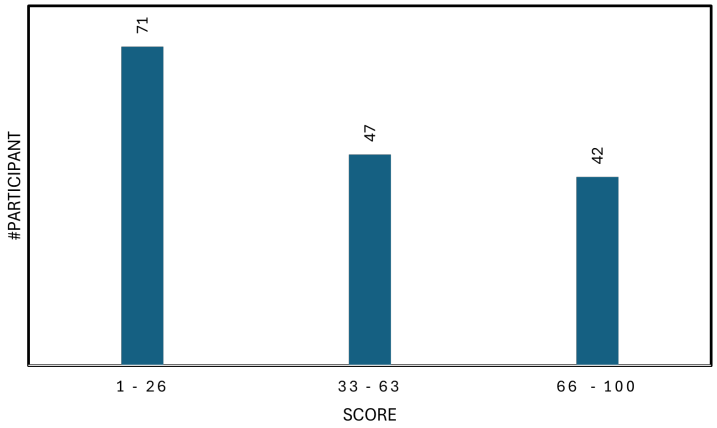

FIGURE 2 Agility score of KNN.

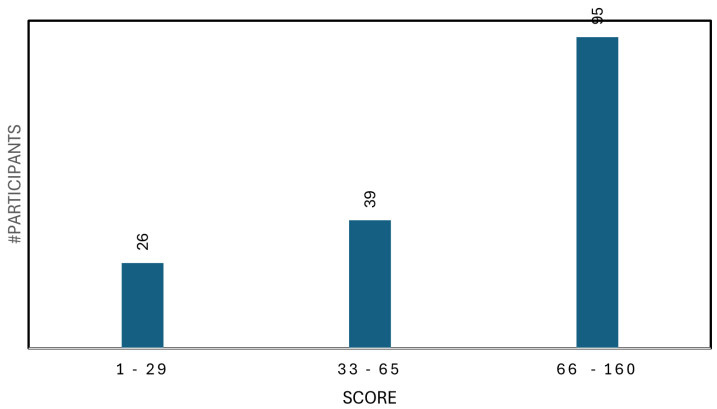

FIGURE 3 Agility score of Random Forest.

overfitting and increase the forest's diversity. This strategy allows the model to handle the data more effectively and prevents the model from relying on a particular set of features.

In the third stage, prediction aggregation is performed. For each new input, all trees in the random forest produce individual predictions, and the final result is determined through majority voting (for classification) or the average of the predictions (for regression). This approach provides a final score based on the combined predictions from all three trees when determining the agility level. Finally, in the evaluation and assessment stage, the accuracy and consistency of the model are evaluated to ensure reliable agility score predictions. This evaluation is important to verify the model's ability to produce accurate predictions based on real data.

Based on Figure 3 , the data is divided into three intervals: 1–29 (low level), 33–65 (middle level), and 66–160 (high level), which may represent different groups or levels of agility. The values displayed above each bar (26, 39, and 95) indicate the scores generated by the Random Forest algorithm for each interval. This graph shows the increase in agility scores as the interval increases, which provides a basis for categorizing certain levels of agility based on the Random Forest results.

3.3.3 | Fuzzy-PSO

To utilize the fuzzy method and Particle Swarm Optimization (PSO) in calculating agility scores, the process began by defining the lower and upper bounds for the input data. A value of 1 was selected as the lower limit, representing the minimum non-zero

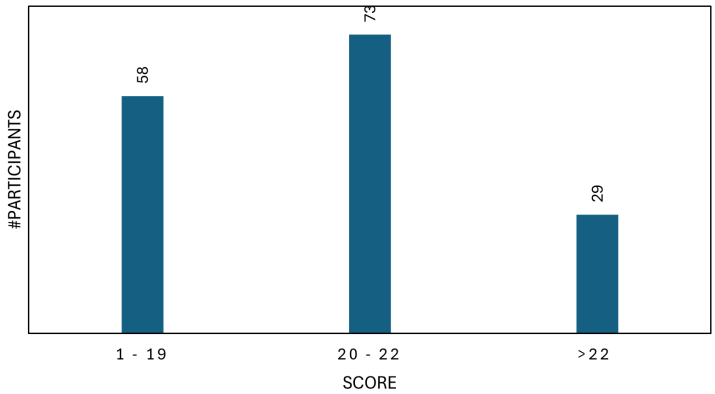

FIGURE 4 Agility score of F-PSO.

TABLE 5 The confusion matrix of KNN results.

Target	Matrix Value	Accuracy (%)	Training Time (sec)	Amount of Data
Individual	TP: [0 25 2]	31.25	0.260	160
	TN: [29 0 8]	(ρ= 0.828)		
Team	FP: [35 7 49]	53.12		
	FN : [0 32 5]	(ρ= 0.614)		

value found in the Keirsey data, role data, and anti-pattern data. Conversely, the upper limit was set at 100, corresponding to the highest observed value within these datasets.

Once these limits were established, fuzzification was performed. This step involved converting the data into fuzzy values and categorizing them into appropriate sets: low (1-50), medium (51-100), and high (the set difference between 1 and 100). Fuzzy rules were then defined to connect fuzzy inputs to fuzzy outputs, simulating human decision-making through linguistic logic. This study applied nine rules, covering combinations of low, medium, and high values for each data point.

Following the rule definition, defuzzification was conducted, resulting in a range of values from 17.3 to 81. Subsequently, PSO optimization was carried out using predetermined parameters: a maximum score of 100, 100 iterations, a minimum step of 0.000001, and a minimum value of 0.000001. This optimization process yielded clusters representing three distinct agility levels (see Figure 4 for visualization).

4 | RESULT AND DISCUSSION

The computational results were evaluated using Anaconda Navigator software, which operates in Python. The Flask framework was used for the user interface (UI), and various Python libraries were used to facilitate the programming process.

This study compares different ways to improve agility. This study looks at how well agility improves without special methods. Then, it compares it with the results of classification methods (Random Forest and K-Nearest Neighbor methods), fuzzy logic combined with particle swarm optimization (Fuzzy-PSO). All of these methods were tested using a standard data splitting process where 80% of the data is used for training and 20% for evaluation.

4.1 | KNN Classification Results

Table 5 and Figure 5 present the confusion matrix for the KNN classification method. The accuracy for the agility level category ranges from only 31.25% to 53.12%, with standard deviations between 0.61 and 0.83. However, the fastest computation time is achieved at 0.260 seconds.

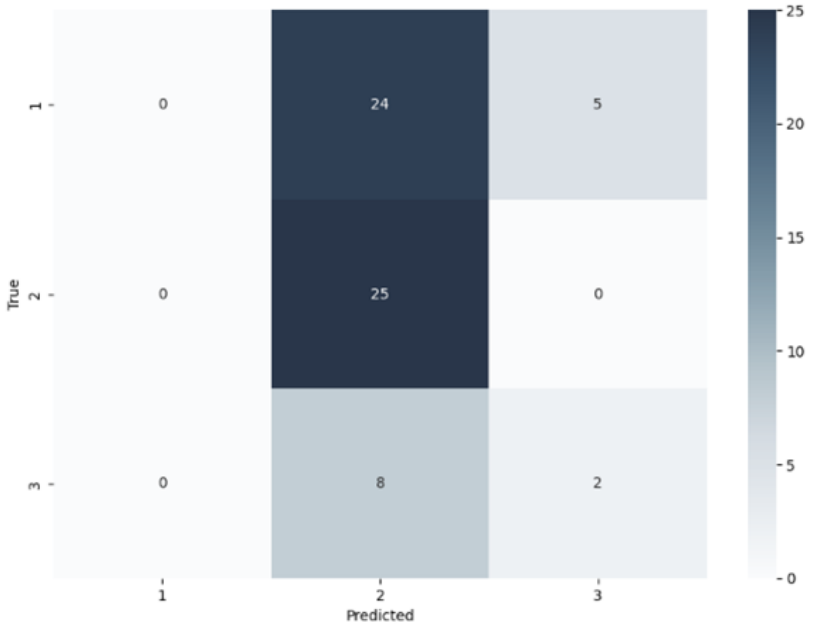

FIGURE 5 The confusion matrix of KNN.

TABLE 6 The confusion matrix of Random Forest results.

Target	Matrix Value	Accuracy (%)	Training Time (sec)	Amount of Data
Individual	TP: [1 15 2]	37.50	0.265	160
	TN: [3 1 42]	(ρ= 0.758)		
Team	FP: [60 4 18]	18.75		
	FN : [0 44 2]	(ρ= 0.708)		

TABLE 7 The confusion matrix of F-PSO results.

Target	Matrix Value	Accuracy (%)	Training Time (sec)	Amount of Data
Individual	TP: [1 59 0]	87.50	201.61	160
	TN: [0 4 0]	(ρ= 0.758)		
Team	FP: [63 1 60]	100.00		
	FN : [0 0 4]	(ρ= 0.175)		

4.2 | Random Forest Classification

Results

Table 6 and Figure 6 present the confusion matrix for the classification performance of the Random Forest method. The accuracy for the agility level categories ranges from a mere 18.75% to 37.5%, accompanied by standard deviations between 0.70 and 0.75. Despite these low accuracies, the Random Forest method offers the fastest computation time, reaching a mere 0.265 seconds.

4.3 | Fuzzy-PSO Classification Results

The results of applying the fuzzy-PSO method to categorize the level of agility are presented in Table 7 and Figure 7 . This approach shows high accuracy, with success rates ranging from 87.5% to 100%. However, the processing time is relatively long, with the fastest calculation taking quite a long time, namely 201.61 seconds for 160 data points.

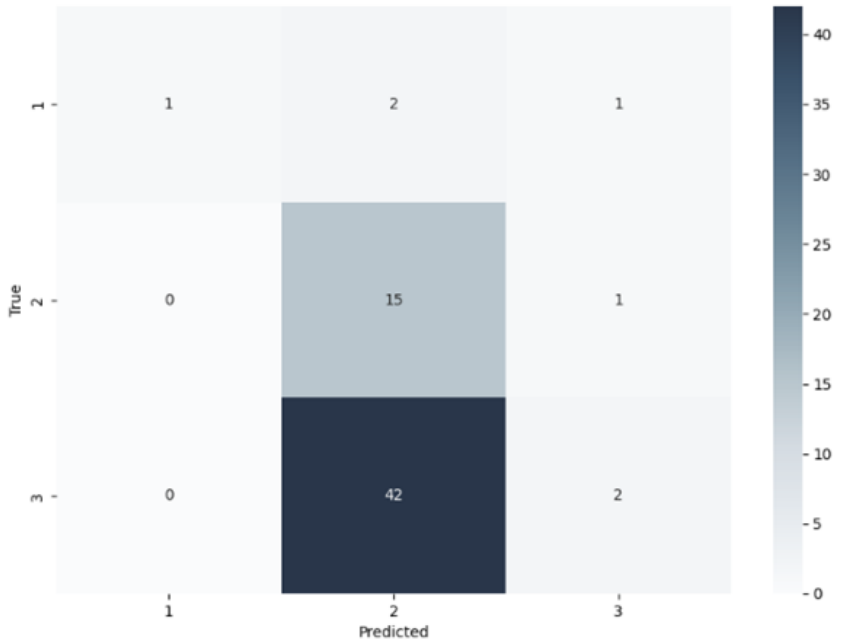

FIGURE 6 The confusion matrix of Random Forest.

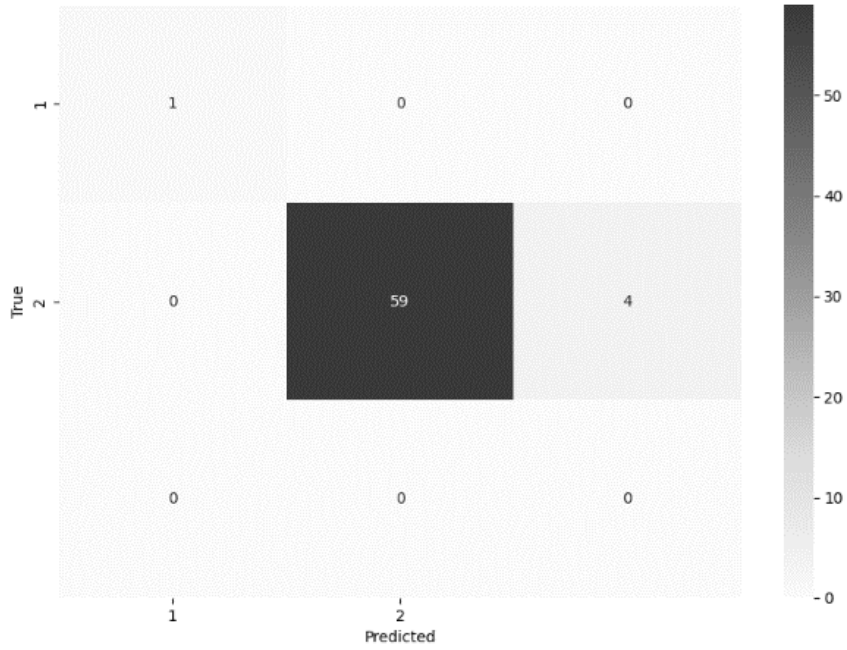

FIGURE 7 The confusion matrix of F-PSO.

4.4 | Analysis

During the testing phase, researchers successfully analyzed the classification process to determine the level of agility for both teams and individuals. This analysis employed the early stopping and train_test_split methods from the sci-kit-learn library, using an 80:20 training-to-testing data ratio.

TABLE 8 The early stopping and train test split test results.

Method	Target	Test Matrix Parameter (%)					
		Acc	Prec	Recall	F1-Score	ρ	Time
Random Forest	Personal	37.50	37.50	37.50	37.50	0.758	0.265
	Teams	18.75	18.75	18.75	18.75	0.709	0.265
KNN	Personal	31.25	31.25	31.25	31.25	0.828	0.260
	Teams	53.12	53.12	53.12	53.12	0.614	0.260
Fuzzy-PSO	Personal	87.50	87.50	87.50	87.50	0.111	201.610
	Teams	100.00	100.00.00	100.00	100.00	0.175	201.610

Three different computer methods, i.e., KNN, Random Forest, and Fuzzy-PSO, were compared using data presented in Table 5 , Table 6 , and 7 . The results showed that Fuzzy-PSO was the most accurate and reliable method, outperforming the other two in overall performance. However, KNN and Random Forest were quicker to set up. A comparison summary can be found in Table 8 .

5 | CONCLUSION

The findings show that the combined Fuzzy-PSO method outperforms the Random Forest and KNN methods in classifying the agility level. The Fuzzy-PSO method can increase the accuracy rate to an average of 68.75%, from an average of 25% before optimization to an average of 93.75% after optimization on a dataset of 160 participants.

CREDIT

Tri Yulianto Nugroho: Conceptualization, Methodology, Validation, Resources, Data Curation, Writing - Original Draft, Software, and Visualization. **Umi Laili Yuhana:** Conceptualization, Methodology, Validation, Writing - Review & Editing, and Supervision. **Daniel Siahaan:** Software, Writing - Review & Editing, Supervision, and Formal analysis.

References

1. Csalódi R, Süle Z, Jaskó S, Holczinger T, Abonyi J. Industry 4.0-Driven Development of Optimization Algorithms: A Systematic Overview. Hindawi Limited 2021;p. 6621235.

2. Arshanty RA, Sa'adah U, Sari DIP, Rasyid MBA. Pengembangan Aplikasi People Management dalam Software Development dengan Pendekatan Project Management Antipatterns. Jurnal Nasional Teknik Elektro dan Teknologi Informasi 2020 December;9(4):343–353.

3. Gilal AR, Jaafar J, Omar M, Basri S, Waqas A. A rule-based model for software development team composition: Team leader role with personality types and gender classification. Inf Softw Technol 2016 June;74:105–113.

4. Eilers K, Peters C, Leimeister JM. Why the agile mindset matters. Technol Forecast Soc Change 2022 June;179.

5. Xie X, Wang B, Yang X. SoftRec: Multi-relationship fused software developer recommendation. Applied Sciences (Switzerland) 2020 June;10(12):1–20.

6. Settas D, Meditskos G, Bassiliades N, Stamelos IG. Detecting antipatterns using a web-based collaborative antipattern ontology knowledge base. In: Lecture Notes in Business Information Processing, Verlag Springer; 2011.p. 478–488.

7. Russo D, Stol KJ. Gender Differences in Personality Traits of Software Engineers. IEEE Transactions on Software Engineering 2022 March;48(3):819–834.

8. Purwaningsih I, Widodo S, Harini E, Kusumaningrum B, Putrianti G, Muanifah M. Adaptation Measuring Instrument Keyrsey Temperament Sorter. In: The 1st International Conference on Science and Technology for an Internet of Things Yogyakarta, Indonesia; 2019. p. 1–6.

9. Sotarjua LM, Santoso DB, Waluyo UHSKJR, Timur KT, Karawang K, Barat J. Perbandingan Algoritma Knn, Decision Tree,*Dan Random*Forest Pada Data Imbalanced Class Untuk Klasifikasi Promosi Karyawan (Comparison of Knn, Decision Tree, *And Random*Forest Algorithms on Imbalanced Class Data for Employee Promotion Classification). Jurnal INSTEK (Informatika Sains dan Teknologi) 2022;7(2).

10. Yilmaz M, O'Connor RV. Understanding personality differences in software organizations using Keirsey temperament sorter. IET Software 2015;9(5):129–134.

11. Settas D, Stamelos I. 8. In: Lee R, editor. Resolving Complexity and Interdependence in Software Project Management Antipatterns Using the Dependency Structure Matrix Berlin, Heidelberg: Springer Berlin Heidelberg; 2008. p. 205–217. https://doi.org/10.1007/978-3-540-70561-1_15.

12. Picha P, Brada P. Software process anti-pattern detection in project data. In: ACM International Conference Proceeding Series, Association for Computing Machinery New York, NY, USA; 2019. p. 1–12.

13. Zhu H, Zhou MC, Seguin P. Supporting software development with roles. IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans 2006 November;36(6):1110–1123.

14. Ahmed F, Capretz LF, Bouktif S, Campbell P. Soft skills and software development: A reflection from software industry. International Journal of Information Processing and Management 2013;4(3):171–191.

15. Akarsu Z, Orgun P, Dinc H, Gunyel B, Yilmaz M. Assessing Personality Traits in a Large Scale Software Development Company: Exploratory Industrial Case Study. In: Systems, Software and Services Process Improvement Cham: Springer International Publishing; 2019. p. 192–206.

16. Neill CJ. Antipatterns in Systems Engineering: An Opening Trio. INCOSE International Symposium 2012;22(1):1233–1245.

17. Gilal R, Omar M, Gilal AR, Rejab MM, Waqas A, Sharif KIM. Can time pressure and personality make any sense together in software engineering? International Journal of Innovative Technology and Exploring Engineering 2019 November;9(1):1071–1075.

18. Lima ACES, Castro LND. Tecla: A temperament and psychological type prediction framework from Twitter data. PLoS One 2019 March;14(3).

19. Putra N, Saputra ID. Metode Fuzzy untuk Mengidentifikasi Kepribadian Siswa. Jurnal Sistim Informasi dan Teknologi 2022 September;p. 136–142.

20. Trujillo-Dalbey F. Managing intercultural conflict effectively: Stella Ting-Toomey and John G. Oetzel, Sage Publications, Thousand Oaks, CA, 2001, 233 pages. International Journal of Intercultural Relations 2002;26(3):334–337. https://www.sciencedirect.com/science/article/pii/S0147176702000093.

21. Dorigo M, Gambardella LM. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1997;1(1):53–66.

22. Arcelli D, Cortellessa V, Trubiani C. Performance-based software model refactoring in fuzzy contexts. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Verlag Springer; 2015.p. 149–164.

23. R Chi Y. xin Su. D hong Zhang, X xin Chi, and H jun Zhang, "A hybridization of cuckoo search and particle swarm optimization for solving optimization problems, " Neural Comput Appl 2019 January;31:653–670.

24. Putra I. Optimasi Proses Etl Dengan Metode Heuristik Untuk Membangun Data Warehouse (ETL Process Optimization Using Heuristic Methods to Build a Data Warehouse). JST (Jurnal Sains dan Teknologi) 2019 July;8:35.

25. Mamur H, Üstüner MA, Bhuiyan MRA. Future perspective and current situation of maximum power point tracking methods in thermoelectric generators. Sustainable Energy Technologies and Assessments 2022;50:101824.

26. Hidalgo ES. Adapting the scrum framework for agile project management in science: case study of a distributed research initiative. Heliyon 2019 3;5:1–31.

27. Hazzan O, Dubinsky Y. The Agile Manifesto. Cham: Springer International Publishing; 2014. https://doi.org/10.1007/978-3-319-10157-6_3.

28. Wang K. Computational Intelligence in Agile Manufacturing Engineering. pp 2001;p. 297–315.

29. Hill GM. The complete project management office handbook. 2 ed. New York, USA: Auerbach Publications; 2004.

30. Kazançoğlu Y, Sağnak M. Integrated Fuzzy Analytic Network Process And 0-1 Goal Programming Technique For Enterprise Resource Planning (Erp) Software Selection. Ege Akademik Bakis (Ege Academic Review) 2019 January;19(1).

31. Vaishnavi V, Suresh M. Assessing the Readiness Level of Healthcare for Implementing Agility Using Fuzzy Logic Approach. Global Journal of Flexible Systems Management 2020 June;21(2):163–189.

32. Kakar A. A Rhetorical Analysis of the Agile Manifesto on its 20th Anniversary. Journal of the Southern Association for Information Systems 2023 February;10(1):20–29.

33. Alsalibi B, Mirjalili S, L Abualigah R. I. yahya, and A. "A Comprehensive Survey on the Recent Variants and Applications of Membrane-Inspired Evolutionary Algorithms, " Aug. 01, Springer Science and Business Media B. V: H. Gandomi; 2022.

How to cite this article: Nugroho T.Y., Yuhana U.L., Siahaan D. (2024), Comparison Of KNN, Random Forest, And F-PSO Algorithms On Simple Feature Scaling For Agility Level Classification, *35(3): 163-174*.