International Journal of Computer and Communication Technology

Volume 8 | Issue 3 Article 3

January 2022

Predicting Faultiness of Program Modules Using Mamdani Model by Fuzzy Profile Development of Software Metrics

Mohammed Ali
Birzeit University, mdali.mail21@gmail.com

Ahmed Abusnaina
Birzeit University, aabusnaina@birzeit.edu

Recommended Citation

Ali, Mohammed and Abusnaina, Ahmed (2022) "Predicting Faultiness of Program Modules Using Mamdani Model by Fuzzy Profile Development of Software Metrics," *International Journal of Computer and Communication Technology*: Vol. 8 : Iss. 3 , Article 3.
DOI: 10.47893/IJCCT.2022.1422
Available at: https://www.interscience.in/ijcct/vol8/iss3/3

1. INTRODUCTION

Measuring the reliability of software systems has become a key concern in software industry. This is because humans are extremely depending on software systems these days (Pandey & Goyal, 2010). This increasing demand on systems leads to keen need for rapid development approaches and accurate tools to measure the overall system quality. Especially, software failures lead to several serious drawbacks in business supported by these systems. Hence, there is an increasing need to resolve the software faults in the early phases of Software Development Life Cycle (SDLC) (Rizvi et al., 2016). Furthermore, if early identification of software faults and errors is practiced, the more it will be cost effective to fix them. Thus, fault prediction and estimation models are needed to assess software reliability in early phases of SDLC utilizing data collected during those phases (Kumar & Ranjan, 2017). Importantly, it has been found by literature that the absence of failure data during early SDLC phases can be compensated by software metrics that may be collected statically from software components and artifacts (Pandey & Goyal, 2010). Importantly, software metrics have a fuzzy nature. Accordingly, this makes fuzzy prediction models more candidate to deal with these problems (Pandey & Goyal, 2010).

For this end, a software module, represented by software metrics of fuzzy nature, will be modelled as fuzzy sets (Pandey & Goyal, 2010). In addition, fuzzy inference models require fuzzy profiles development and fuzzy rules (Yadav, 2015). Those two requirements are often provided from domain experts. However, it will be infeasible to get consensus from experts about them. Instead, data mining techniques are alternative and recommended solution to satisfy these requirements (Pandey & Goyal, 2010)(Subhashis & Bappa, 2016) (Rizvi et al., 2016)(Kumar & Ranjan, 2017) (Singh et al., 2016).

Most of fault prediction models rely on the failure data of a software system collected from the testing and usage phases. However, there will be no available failure data just after the software module has been developed. Thus, if it is possible to tag a software module as fault-prone in this time, software engineers can well manage the consequent activities in terms of time, effort, and resources, and finally producing reliable systems (Pandey & Goyal, 2010)(Pandey & Goyal, 2010) (Yadav, 2015)(Kumar & Ranjan, 2017). More important, the unavailability of failure data in early phases makes software metrics a better choice to be used in fault prediction models. These metrics are measures that are collected statically from the source code of a software module. Also, because they are associated with some level of vagueness, fuzzy logic will be the most suitable approach for software defect prediction (Yadav, 2015).

The main contribution of this research is to build a Mamdani fuzzy inference system without the assistance of domain expert or using any other literature

assumptions. In particular, it used the data itself to satisfy the requirements of building such models. The employed data represented software metrics which are of a fuzzy-nature data. For this end, the proposed model tested and validated a published technique used to develop fuzzy profiles for inputs and outputs from data. In addition, it used k-mean clustering algorithm to generate the fuzzy rules needed to the inferencing process in the proposed model. Finally, the new system was able to estimate and predict faults in the developed software modules with a significant accuracy, and then, it was benefitable in supporting the testing efforts and decisions.

In sum, problem definition in this research is as follows:
Given a just developed software module, estimate the fault-proneness degree of that new module and predict its class with respect to either fault-prone (FP) or non-fault-prone (NFP). So that, this could support the testing phase in several aspects.

This work will address the above research problem by trying to answer the following research question:

- **RQ1**: Do software metrics, that are related to the implementation phase, enable building of fault prediction models to support testing phase efforts of SDLC?
- **RQ2**: What will be the effect of developing fuzzy profiles from data, instead of expert opinions, on fault prediction by Mamdani fuzzy inference systems?
- **RQ3**: Can k-mean clustering algorithm be used to generate "if-then" fuzzy rules for effective functioning of Mamdani fuzzy inference systems?

2. BACKGROUND

Fuzzy logic is a type of multi-valued logic, it deals with fuzzy sets which were proposed by Lotfi Zadeh in 1965 (Zadeh, 2015). Elements of a fuzzy set have a multi-valued membership relation toward that set, on the contrary of the elements of classical sets which have only a binary membership value [0,1] (Subhashis & Bappa, 2016).

Figure 1 shows a Mamdani fuzzy inference system for fault prediction. It consists of four parts: first, fuzzification, which is the conversion of the crisp input to a fuzzy linguistic variable such as low, medium, and high using a predefined input membership function (Subhashis & Bappa, 2016) (Wang, 2015). Second, fuzzy rule base, which is a set of rules used by inferencing process to convert fuzzy input to fuzzy output (Subhashis & Bappa, 2016) (Wang, 2015). Third, inferencing process, which evaluates the fuzzy rules on inputs and then aggregates the results of applying those rules to finally producing one fuzzy output (Subhashis & Bappa, 2016) (Wang, 2015). Actually, given the fuzzy sets A and B. In addition, μA, and μB are the membership functions that represent A and

B in the universe R, the fuzzy operators union (OR), intersection (AND), and complement (NOT) are represented by the equations 1,2, and 3, respectively.

$$\mu(A \cup B)(R) = Max\big(\mu A(R), \mu B(R)\big) \quad (1)$$

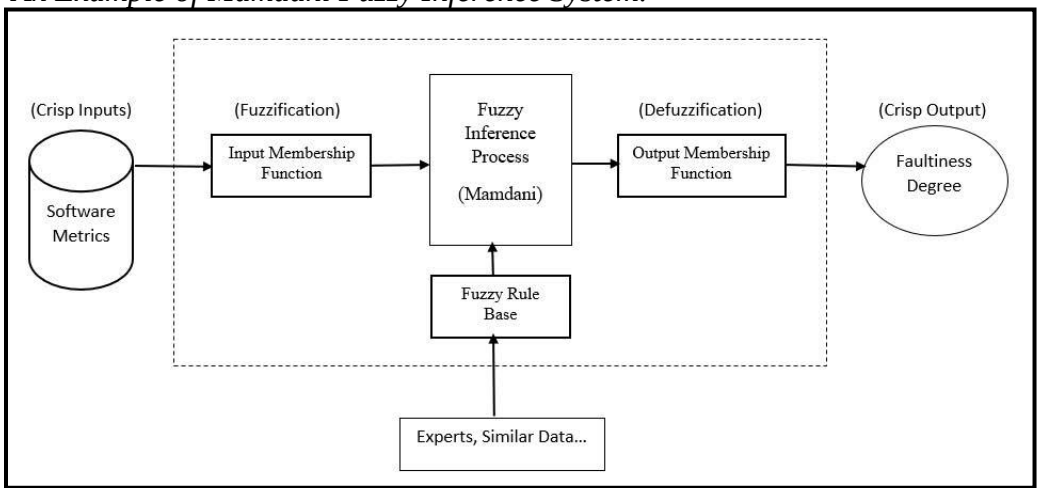
$$\mu(A \cap B)(R) = Min\big(\mu A(r), \mu B(R)\big) \quad (2)$$

$$\mu\tilde{A} = 1 - \mu A(R) \quad (3)$$

Forth, defuzzification, which is the process of converting the fuzzy output to crisp using the outputs membership function. Whereas, A membership function is a graphical representation that represents how the transition from one fuzzy set to another happens.

Figure 1

An Example of Mamdani Fuzzy Inference System.



Software reliability is the probability of a software system to perform defined tasks in a specified environment with failure-free (Rizvi et al., 2016). Moreover, software reliability is affected by failures which are caused by software faults. The main sources for faults are user inputs and program internal state. Therefore, the knowledge represented by software metrics plays a vital role in early prediction and early estimation of those faults (Malhotra, 2016). In fact, the most widespread categories of software metrics are McCabe and Halstead metrics (Yadav, 2015).

3. RELATED WORK

Pandey and Goyal (2010) built a Mamdani fuzzy system to predict faults using KC2 dataset. Moreover, fuzzy profile development for inputs and output utilized opinions of domain experts. Whereas, fuzzy rules were generated using ID3 decision tree algorithm. Accordingly, they achieved 87% overall average

accuracy for their model which had outperformed other models. In addition, same authors developed a fuzzy system to predict number of faults using software metric in KC2. However, this time, fuzzy rules were generated by consulting the experts in the field of software engineering.

Farahbod and Eftekhari (2013) proposed an approach based on clustering-based methods for obtaining the fuzzy classification rules from numeric data to create a model mapping the input output relationship of the system. The results proved that their proposed approach produced significant performance.

Sadeghian et al. (2014) proposed k-mean clustering-based algorithm to extract fuzzy rules from numeric data of customer ratings in banks. This method may be used to support software engineering tasks. In particular, the main concept of this method is that by grouping similar data into one cluster, then it could be represented as a rule.

Yadav (2015) built fuzzy inference system to predict faults using software metrics by mainly using k-mean clustering algorithm in order to develop fuzzy profiles. In another work, they completely utilizing experts' knowledge and consulting the domain experts to build the same fuzzy model.

Subhashis and Bappa (2016) proposed a Mamdani fuzzy model that helps software engineers and stakeholders in predicting software quality in the requirement engineering phase. They proposed a novel approach to extract fuzzy rules using an approach based on calculating the relative importance of each software metric, and later taking the decision to include or exclude it from the fuzzy rule structure.

Singh et al. (2016) proposed an approach to identify software faults before they actually become run time failures. Actually, their approach utilized k-mean clustering algorithm to implement metrics selection method to form effective fuzzy classification rules.

Erturk and Sezer (2016) used Mamdani fuzzy inference system to develop predictive model for fault prediction. More importantly, the process of fuzzy profile development and fuzzy inference rules extraction were created by the help of domain experts and their knowledge in software systems.

Rizvi et al. (2016) presented a study for software reliability prediction using the fuzzy logic just before the implementation phase of SDLC. Particularly, the proposed effective model that utilized expert opinions to estimate the software reliability using four requirement metrics and four design metrics to be all as inputs for the fuzzy inference system. Finally, the results of this work were validated statistically.

AI-Jamimi (2016) proposed a framework using Mamdani type fuzzy inference system that utilized 21 historical software metrics, aimed to predict the fault proneness of a given software module. Fuzzy profile development was achieved by utilizing expert knowledge. Importantly, correlation-based technique to select

the most discriminating metrics to be used in forming the rules.

Kumar and Ranjan (2017) proposed three fuzzy models to predict faults in all phases of SDLC by using software metrics. The first two approaches used Mamdani fuzzy inference system. The options of building these systems were provided by domain experts. Their results were validated using MMRE, and BMMRE techniques.

Golnoush et al. (2018) designed an unsupervised methodology for prediction fault-proneness software modules from software metrics. They used expert knowledge to implement two types of fuzzy inference systems: Mamdani and Assilian, and Takagi and Sugeno. In addition, genetic algorithm was employed in order to optimize the rulebase of the fuzzy inference systems. Consequently, the results show significant improvement in terms of the predefined performance measures by using the GA for rule-base optimization.

Diwaker et al. (2019) proposed a model for estimating and predicting of software reliability based on Mamdani fuzzy inference system. This model supported software engineering using an approach called component-based SE. the fuzzy profile development besides 243 fuzzy rules were provided by the assistance of domain experts. the model was implemented using MATLAB on publicity available dataset called QWS. The results of the proposed model show that the fuzzy inference system produces better performance and less error rate as compared to another soft computing technique called PSO.

Ali and Abusnaina (2021) Proposed a hybrid approach used fuzzy set theory to build effective fuzzy classifier model that is able to classify the reported software bugs using Jira bug tracking system to either bug or not-bug. This approach offered an important feature for software engineers as it enhanced their ability to well manage software maintenance activities toward reaching to the overall reliability in the supported software system.

By reviewing the related literature, it is found that there was very limited number of research papers that utilized the fuzzy inference methods to estimate and predict the faultiness degree of a software module to support testing phase. Importantly, software metrics have a vagueness nature making fuzzy logic the most suitable technique to model such these problems. In addition, only few of fuzzy works used Mamdani fuzzy systems. Moreover, most of reviewed papers used binary classification machine learning algorithms regarding this research problem. Furthermore, regarding building the fuzzy inference systems, the most of papers utilized the knowledge of domain experts in order to satisfy the requirements of building these systems such as fuzzy profile development for inputs and outputs besides to the fuzzy rules generation. However, very few papers used the data itself to build inference systems and satisfy its requirements by implementing data mining techniques on that data. Finally, there were very few papers addressed the current research problem by utilizing the metrics of the

developed software modules to early predict faulty modules to support testing phase.

4. RESEARCH METHODOLOGY

This research made use of a software metrics dataset called KC2. This dataset consists of 21 software metrics. It shows a crisp value against each metric, and it is available publicity from Promise repository (Shirabad, & Menzies, 2005). It is originated from NASA Data Metric Program (DMP). More importantly, authors, such as Pandey and Goyal (2010) Yadav (2015), proved that 13 out of 21 software metrics from this dataset offer a significant role in software fault prediction. These selected 13-software metrics are clustered below by four categories:

- Halstead metrics category: total number of operators (N1), total number of operands (N2), number of unique operators (NN1), and number of unique operands (NN2).
- Line of code metrics category: total line of code (LOC), LOC blank (BL), LOC code and comments (CCL), LOC comments (CL), and executable LOC (EL).
- McCabe metrics category: cyclomatic complexity (CC), essential complexity (EC), and design complexity (DC).
- Branch Count Metric Category: branch count metric (BC).

The main goal of this research is to develop a Mamdani fuzzy inference model that will be used to estimate and predict the degree of faultiness of a software module before it is tested. Mainly, the prerequisites and requirements for building the model will be satisfied using the data itself rather than just consulting the domain experts. Figure 2 shows a high-level diagram of the proposed approach which is composed of the following four main parts:

First, developing fuzzy profile for inputs and outputs, it includes defining the shape of the membership function, the fuzzy sets of linguistic variables used in this function, and the range of the possible values for that sets.

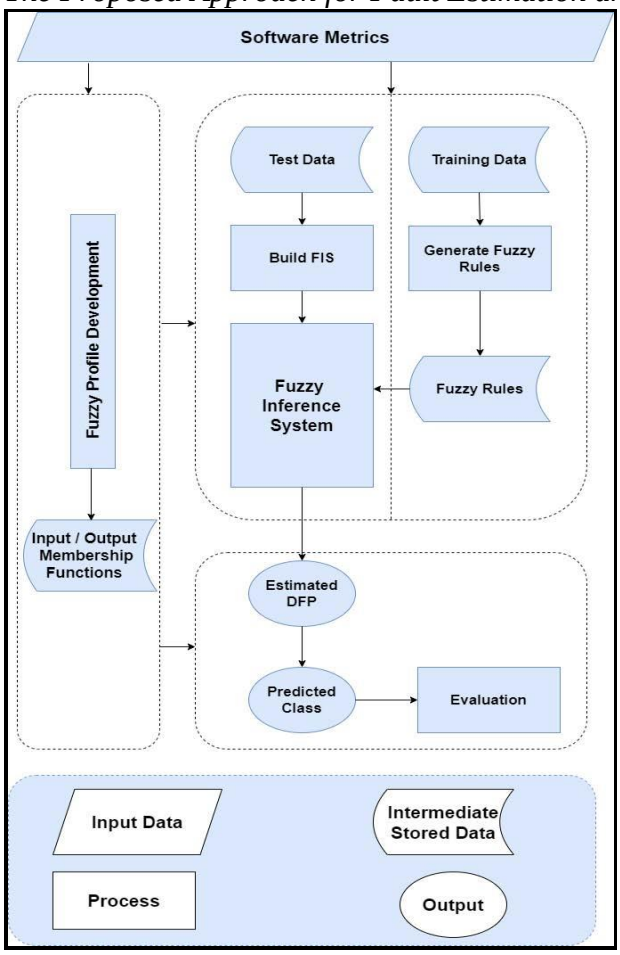
Second, by employing trainingdata of software metrics, fuzzy inference rules will be generated. In using k-mean clustering algorithm from machine learning domain. More importantly, a new approach will be used to select the antecedent portion for rules. In particular, the selection strategy depends on a controlled threshold for membership values constructed from the previous component.

Third, building Mamdani fuzzy inference system (FIS). Test data will be the inputs. Membership functions for inputs and output besides input fuzzification are all provided by the first component. In addition, the fuzzy inference rules are extracted from the second component. Furthermore, other configurations such defuzzification method and rules evaluation methods, the default parameters were used since they were wildly experimented by literature.

Finally, the first output of the built Mamdani model will be the estimated degree of faultiness. Then, this output will be checked against the output membership function to obtain the membership value in each class. Consequently, the class of the larger membership value will be considered as the final predicted class of the system. After that, given the desired and predicted output classes, an evaluation model will be conducted in order to measure system performance.

Figure 2

The Proposed Approach for Fault Estimation and Prediction



5. EXPERIMENTAL DESIGN

This section explains the followed methodology to develop fuzzy profiles for inputs and outputs. Moreover, it illustrates the methodology applied to generate the fuzzy inferencing rules. Then, it shows details of implementation of Mamdani fuzzy system. Besides, it shows results of each process.

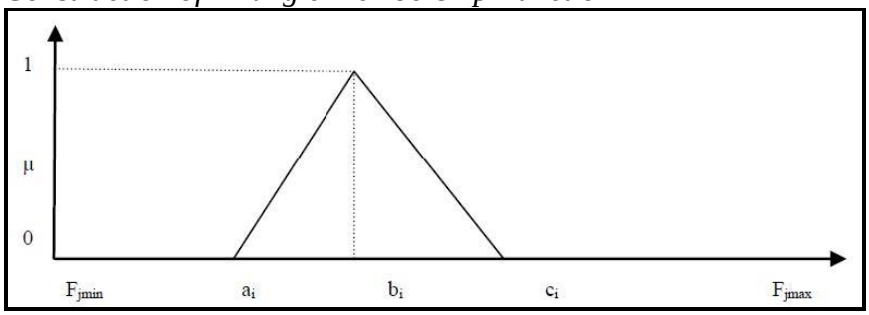
5.1.Inputs Fuzzy Profile Development

In this study, triangle membership function was used to represent the linguistic variables of inputs (software metrics). Triangle membership function is widely utilized in literature because it can better represent the knowledge of domain experts and it is computationally effective (Pandey & Goyal, 2010) (Yadav, 2015). Moreover, each input will be represented by five linguistic variables (fuzzy sets): very low, low, medium, high, and very high. Thus, development of a fuzzy profile for an input means to find fuzzy ranges and overlap area of adjacent membership functions for that input.

This task is often provided by experts, but this research will use and validate the methodology proposed by Yadav (2015) to construct membership functions for software metrics. Figure 3 shows a triangle membership function, where the horizontal axis represents the inputs. The vertical axis represents the corresponding membership value of each input variable. The points (a_i, bi, ci) represent the fuzzy range of the input variable which is called "support" (Wang, 2015). In particular, construction a triangle membership function means to calculate the values of (a_i, bi, ci).

Figure 3

Construction of Triangle Membership Function



Given an input software metric (feature) which is represented by F_j, it has n values (v_{1j} - v_{nj}). F_{jmin} represents the minimum value of F_j. F_{jmax} represents the maximum value of F_j. The membership functions are generated using the following steps:

1. Sort F_j values in ascending order.
2. Cluster F_j values using k-mean algorithm into five clusters (y_1-y_5), where y_{imin} represents the minimum value of a cluster. y_{imax} represents the maximum value of a cluster.
3. Find out the prototypes (centroids) for each cluster (b1-b5).
4. Regarding F_i, calculate the difference between adjacent data ($diff_i$ = v_{i+1} - v_i).
5. Calculate the similarity value between adjacent data according to formula 4:

$$S_m = \begin{cases} 1 - \dfrac{diff_i}{C + \sigma_s} & for\ diff_i \le C * \sigma_s \\ 0 & Otherwise \end{cases} \quad (4)$$

where,

S_m = donates the similarity value between adjacent data of input variable.

σ = the standard deviation of $diff_i$

C = a fixed parameter with value of 4, representing the triangle shape.

6. The membership value for the two boundaries y_{imin}, and F_{imax} will be the minimum similarity value of each cluster.
7. Calculate left vertex point a_i according formulas 5 and 6:

$$a_i' = b_i - \frac{b_i - y_{i\,min}}{1 - \mu(y_{i\,min})} \quad (5)$$

$$a_i = \begin{cases} 0 & for\ a_i' \le 0 \\ b_{i-1} & for\ 0 < a_i' \le b_{i-1} \\ a_i' & for\ a_i' > b_{i-1} \end{cases} \quad (6)$$

8. Calculate right vertex point c_i according formulas 7 and 8:

$$c_i' = b_i + \frac{y_{i\,max} - b_i}{1 - \mu(y_{i\,max})} \quad (7)$$

$$c_i = \begin{cases} c_i' & for\ c_i' \le b_{i+1} \\ b_{i+1} & for\ c_i' > b_{i+1} \end{cases} \quad (8)$$

9. Calculate the membership value for each crisp value of feature F_i according to formula 9:

$$\mu(v) = \begin{cases} \dfrac{b_i - V}{b_i - a_i} & for\ v < b_i \\ \dfrac{c_i - V}{c_i - b_i} & for\ b_i \le v < c_i \\ 0 & otherwise \end{cases} \quad (9)$$

Figures 4, 5, and 6 show the constructed input triangle membership function for 3 out of 13 software metrics as sample results for applying the previous methodology.

Figure 4

Fuzzy Profile for "Number of unique operators" Input Software Metric

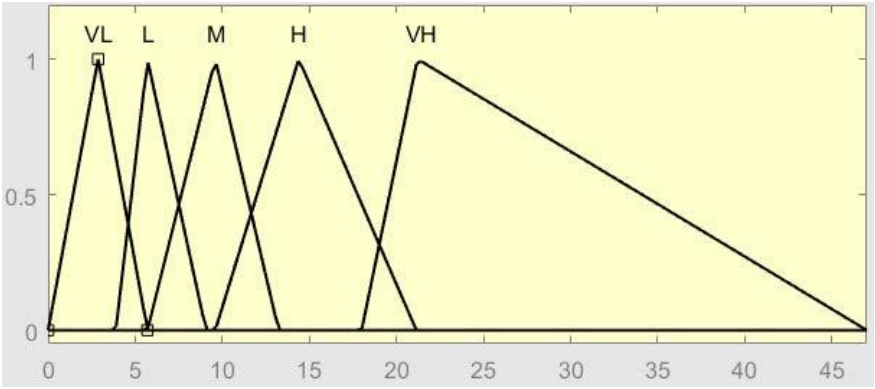


Figure 5

Fuzzy Profile for "Total line of code" Input Software Metric

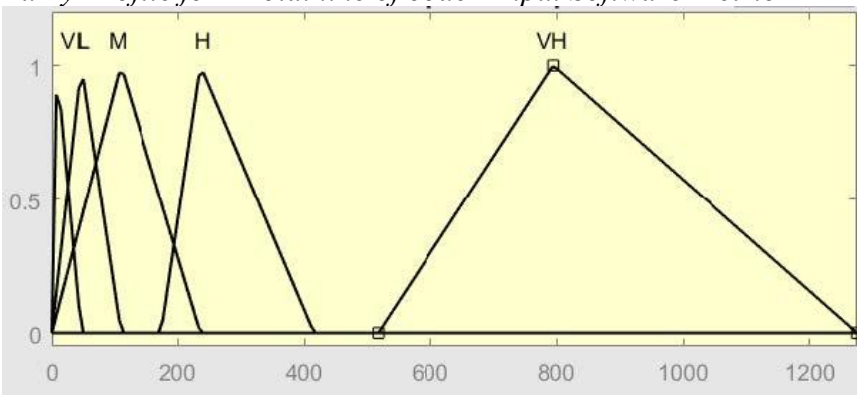
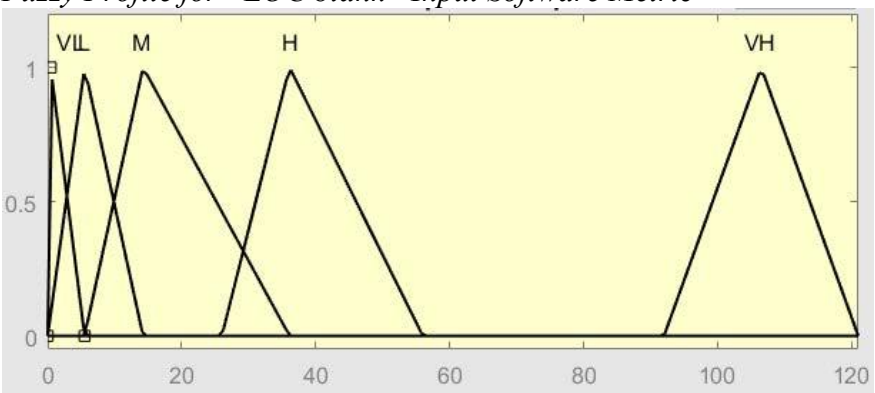


Figure 6

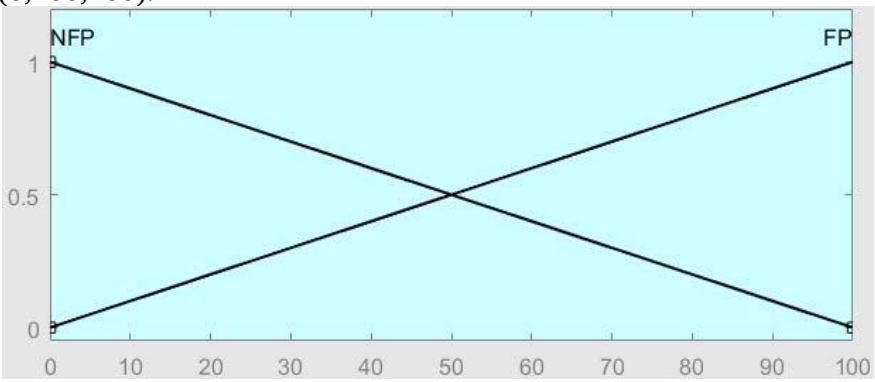
Fuzzy Profile for "LOC blank" Input Software Metric



5.2.Output Fuzzy Profile Development

Degree of fault-prone (FPD) is the output of the proposed model. It is categorized into two fuzzy sets not-fault-prone (NFP), and fault-prone (FP). According to scale used by Pandey and Goyal (2010) it is assumed that this output variable is of linear nature. Thus, it will be represented by triangular membership

function. The fuzzy profile ranges will take the following values: NFP (0,0,100), FP (0,100,100).



5.3.Fuzzy Rules Generation

These below steps represent the employed methodology to extract the inference rules for the proposed model. To the best of our knowledge, this will be the first work that follows such this methodology for fuzzy rule generation. More importantly, this method extracted fuzzy rules using the training portion of the dataset.

1. The data of each class (FP, NFP) from the dataset was divided into two parts, training and testing, by 20%, 40%, 60%, and 80% ratios for training portions.
2. k-mean clustering algorithm was applied to each subset of training data independently. Many runs were executed in order to optimize the value of number of clusters (k). In particular, each run was validated against three conditions: high inter-class separation, high intra-class homogeneity, and special condition was introduced in order to ensure that every cluster must have at least one data instance (module) assigned to it. This will help in generating rules with acceptable inference power. Accordingly, the results are k number of data records (clusters), for each, 13 feature prototypes (centroid) are produced. Each prototype is a representative to k values of the respective feature (software metric).
3. For each centroid value of a feature (software metric), within a cluster, the centroid value was converted into the corresponding linguistic fuzzy set (VL, L, M, H, or VH). Then, formula 9 was used to compute the membership value of that crisp value (centroid) to the corresponding fuzzy set.
4. Threshold based mechanism was introduced to decide whether to include a software metric in a rule formulation or not. This was performed by checking the membership value of a software metric

against a list of threshold values: 10%, 30%, 50%, and 70%. This ensured generating good rules with bias free.

Table 1 shows results of applying the above methodology using several ratios of training data and 10% threshold value as an exclusion parameter. Table 2 shows membership values of the antecedents for a generated fuzzy rule and their corresponding fuzzy sets.

Table 1
Number of Generated Fuzzy rules

Data (%)	NFP class	FP class
20	5	4
40	8	4
60	8	5
80	9	8

Table 2
A Generated Fuzzy Rule Example

Software metric	Membership value	Corresponding fuzzy set
LOC	0.1165	M
CC	0.061	M
EC	0.6257	L
…..	…..	…..
N1	0.637	L
N2	0.604	L
BC	0609	M
The generated fuzzy rule: **IF** (*LOC is M*) **AND** (*CC is M*) **AND** (*EC is L*) **AND** ……. **AND** (*N1 is L*) **AND** (*N2 is L*) **AND** (*BC is M*) **THEN***FP*		

6. MAMDANI FUZZY INFERENCE SYSTEM

According to figure 1, Mamdani fuzzy inference system (FIS) was built in order to estimate faults, in particular, four steps were applied in order to build Mamdani fuzzy inference system (FIS) that will be able to map linguistic input variables (software metrics in this case) to a crisp output (degree of faultiness in this case) in a way approximates humans in their reasoning about things using

fuzzy rules (Wang, 2015):

1. Fuzzify the software metrics:
   This means to convert the crisp numerical values of software metrics into the corresponding fuzzy sets (VL, L, M, H, and VH) using triangular membership function which was constructed in the previous section.
2. Evaluation fuzzy rules.
   "AND" operator in the generated rules will be applied using a fuzzy operator called "MIN". In particular, it applies a rule of two inputs by finding the minimum membership value of that inputs to their corresponding fuzzy sets. Actually, it functions as exactly as Boolean logic in such $(1\ AND\ 0\ \Rightarrow MIN(1,0) \Rightarrow 0)$. After that, the result of rule evaluation will be mapped to the two output fuzzy sets (FP and NFP) using the membership function for outputs, discussed in the previous section. Specifically, this operation is named in fuzzy logic as implication.
3. Apply the aggregation:
   This process aimed at combining the fuzzy sets resulted from evaluation and implication of each rule into one and only one fuzzy set that contains in some point the desired crisp output. For this, an aggregation technique called "MAX" was used.
4. Defuzzification:
   This step was used to extract a crisp output (degree of fault proneness) from the aggregated fuzzy sets in the previous step. A fuzzy technique called CENTROID was used for this step.

Table 3 shows the selected options and hyperparameters of the developed fuzzy inference systems.

Table 3

Mamdani FIS hyperparameters

Option	Value	Source
Inputs membership function	Triangle	**constructed**
Output membership function	Triangle	literature
Fuzzy inference rules		**Generated**
Implication operator	Min	literature
Aggregation method	Max	literature
Defuzzification method	Centroid	literature
AND method	Min	literature

OR method	Max	literature

Furthermore, according to the previous mentioned steps, One Mamdani FIS was built for each set of training and testing data (20%, 40%, 60%, and 80%). 10 runs were carried out and the average performance was recorded. After that, the model that showed the best performance was revalidated a gain using four experiments testing the effect of threshold values (0.1, 0.3, 0.5, and 0.7). Also, 10 runs were carried out and the average performance was recorded. Finally, the best performant model was selected.

Finally, in order to binary classify software modules into either fault-prone or not fault-prone, the membership value for the system output (degree of fault proneness) was calculated using formula 10 which is documented in MATLAB online pages.Then, according to formula 11, the predicted class was given the name of the output fuzzy set thathas the larger computed membership value for that estimated output. Accordingto the following formula:

$$\begin{cases} \dfrac{x-a}{b-a} & a \le x \le b \\ \dfrac{c-x}{c-b} & b \le x \le c \\ 0 & c \le x \; or \; x \le a \end{cases} \tag{10}$$

$$If\ \mu(FIS\ output) \geq 50\ Then\ FP\ else\ NFP \tag{11}$$

Evaluation

This paper used accuracy performance metric to evaluate the performance of the built models. It is an estimation to how these models recognize the correct classes FP and NFP from the test data (Han & Kamber, 2011). This accuracy can be derived using the confusion which is a tool that provides useful information about the behaviour of the models in terms of the following:

1. Ture Positives (TP): the number of fault-prone data instances that are actually fault-prone.
2. True Negative (TN): the number of non-fault-prone data instances that are actually non-fault-prone.
3. False Negative (FN): the number of non-fault-prone data instances that are actually fault-prone.
4. False Positive (FP): the number of fault-prone data instances that are actually, non-fault-prone.

Accordingly, the accuracy is computed using the formula 12:

$$\frac{TP + TN}{TP + TN + FP + FN} * 100 \quad (12)$$

7. RESULTS AND DISCUSSION

The proposed approach, with the optimal configuration, achieves 83% accuracy in classifying software modules to either fault-prone or not fault-prone using certain metrics for these modules before beginning the testing phase. To the best of our knowledge, this will be the first approach that implements such this methodology and achieves significant performanceregarding current research problem.Moreover, Table 4 shows model performance according to the percentage of data used for training/test. In particular, test data is used to evaluate the built model, and training data is used to extract the inferencing rules. In addition, Table 5 shows the performance of the model that was build using 20:80 partitioning of the dataset according to the threshold values used to optimize the fuzzy rules.

Table 4
Model Performance According to The Data Ratio Used to Create Rules Base

Data (%)	Accuracy (%)
20	83
40	77.3
60	77
80	77.9

Table 5
Model Performance According to threshold values and 20:80 Data Partitioning

Threshold (%)	Accuracy (%)
10	80.6
30	80.4
50	79.4
70	83

Importantly, by comparing with baseline approach proposed by Pandey and Goyal (2010), the proposed approach generated fuzzy inference rules for both classes. FP and NFP. However, baseline approach only extracted rules for fault-prone class. In addition, fuzzy profile development in the baseline approach

utilized expert knowledge. Whereas, the proposed approach assumes that data is the most accurate source to be consulted regarding developing fuzzy profiles.

Moreover, Finally, according to performance data, it will be valid to conclude that software metrics, that are related to software modules, have predictive power that supports testing of these modules. Additionally, this paper validated a methodology for constructing fuzzy profiles from data and proves that data of software metrics can be used to build Mamdani FIS without the assistance of domain experts. As well, k-mean clustering algorithm, as an unsupervised machine learning technique, proves its capabilities in obtaining accurate and representative inference rules.

8. THREATS TO VALIDITY

Prediction models are often sensitive to values set for hyperparameters or even exposed to the researcher bias. This paper contained many of them. For example, number of clusters hyperparameter (k) used with k-mean clustering algorithm, the value of threshold used to create the fuzzy rules, and several parameters for Mamdani fuzzy inference system. Many techniques were applied in order to mitigate these threats. For instance, a well-known technique for cluster validity in k-mean algorithm was used, randomness was employed, and literature was considered regarding some decisions.Moreover, this paper made use of KC2 dataset, one potential is that it is not known much about the circumstances of obtaining this dataset besides to the tools used in this process. Furthermore, the used dataset is relatively small and dedicated to only one developing language, C++.

9. CONCLUSION AND FUTURE WORK

This paper applies a hybrid approach combining machine learning and Fuzzy logic to estimate and predict software faults using software metrics for the sake of supporting testing phase. The study starts by critically analyzing related works and providing research gaps. Then, Mamdani fuzzy inference system was built. First, this work applied and validated a methodology to create fuzzy profiles. Second, machine learning was introduced through a proposed methodology by using k-mean clustering algorithm to extract fuzzy inference rules. Finally, the proposed model achieves a significant and competitive predictive accuracy. In addition, it shows that only the given 13 software metrics can be used to build fuzzy systems. Moreover, it is concluded that data can eliminate the need for domain expert assistance as it is effective source for building fuzzy inference systems.

REFERENCES

Yadav, H. B., & Yadav, D. K. (2015). A fuzzy logic-based approach for phase-wise software defects prediction using software metrics. Information and Software Technology, 63, 44-57.

Abaei, G., Selamat, A., & Al Dallal, J. (2020). A fuzzy logic expert system to predict module fault proneness using unlabeled data. Journal of King Saud University-Computer and Information Sciences, 32(6), 684-699.

Farahbod, F., & Eftekhari, M. (2013). A new clustering-based approach for modeling fuzzy rule-based classification systems. Iranian Journal of Science and Technology. Transactions of Electrical Engineering, 37(E1), 67.

Chatterjee, S., & Maji, B. (2016). A new fuzzy rule-based algorithm for estimating software faults in early phase of development. Soft Computing, 20(10), 4023-4035.

Diwaker, C., Tomar, P., Solanki, A., Nayyar, A., Jhanjhi, N. Z., Abdullah, A., & Supramaniam, M. (2019). A new model for predicting component-based software reliability using soft computing. IEEE Access, 7, 147191-147203.

Kumar, S., & Ranjan, P. (2017). A Proposed Methodology for Phase Wise Software Testing Using Soft Computing. International Journal of Applied Engineering Research, 12(24), 15855-15875.

Wang, C. (2015). A study of membership functions on mamdani-type fuzzy inference system for industrial decision-making. Lehigh University.

Novák, V. (2005). Are fuzzy sets a reasonable tool for modeling vague phenomena? Fuzzy Sets and Systems, 156(3), 341-348.

Yadav, H. B., & Yadav, D. K. (2015). Construction of membership function for software metrics. Procedia Computer Science, 46, 933-940.

Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

Yadav, D. K., & Yadav, H. B. (2015, June). Developing Membership Functions and Fuzzy Rules from Numerical Data for Decision Making. In IFSA-EUSFLAT.

Malhotra, R. (2019). Empirical research in software engineering: concepts, analysis, and applications. Chapman and Hall/CRC.

Pandey, A. K., & Goyal, N. K. (2010). Fault prediction model by fuzzy profile development of reliability relevant software metrics. International Journal of Computer Applications, 11(6), 34-41.

Singh, P., Pal, N. R., Verma, S., & Vyas, O. P. (2016). Fuzzy rule-based approach for software fault prediction. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(5), 826-837.

Sadeghian, R., Gholamaliei, B., & Peyman, L. P. (2014). Fuzzy Rules Extraction Based on Deterministic Data (Case Study: Bank's Customers Rating). Jordan Journal of Mechanical & Industrial Engineering, 8(6).

Zadeh, L. A. (1996). Fuzzy sets. In Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh (pp. 394-432).

Pandey, A. K., & Goyal, N. K. (2010). Predicting fault-prone software module using data mining technique and fuzzy logic. International Journal of Computer and Communication Technology, 2(2), 56-63.

Erturk, E., & Sezer, E. A. (2016). Software fault prediction using Mamdani type fuzzy inference system. International Journal of Data Analysis Techniques and Strategies, 8(1), 14-28.

Ramani, S., Gokhale, S. S., & Trivedi, K. S. (2000). SREPT: software reliability estimation and prediction tool. Performance evaluation, 39(1-4), 37-60.

Rizvi, S. W. A., Khan, R. A., & Singh, V. K. (2016). Software reliability prediction using fuzzy inference system: early-stage perspective. International Journal of Computer Applications, 145(10), 16-23.

Shirabad, & Menzies, (2005). The {PROMISE} Repository of Software Engineering Databases, http://promise.site.uottawa.ca/SERepository.

Al-Jamimi, H. A. (2016, August). Toward comprehensible software defect prediction models using fuzzy logic. In 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS) (pp. 127-130). IEEE.

Ali MD, Abusnaina AA. Classifying bug reports to bugs and other requests: an approach using topic modelling and fuzzy set theory. International Journal of Advanced Computer Research. 2021; 11(56):103-115. DOI:10.19101/IJACR.2021.1152031