

A NOVEL MEASURE FOR EVALUATING SEVERITY OF CODE BASED ON KNOWLEDGE BASE APPROACH (NMSC-KNOW)

GHADA MOHAMMAD TAHIR ALDABAGH^{1*}, SAFWAN OMAR HASOON²

Abstract: Code severity evaluation is a crucial part of software development, but traditional approaches frequently use narrow criteria to determine how common flaws are. This work presents a new metric and a cutting-edge model intended to offer a more accurate and insightful evaluation of severity levels. The suggested action greatly improves decision-making procedures where a precise severity assessment is necessary. The results show this innovative measure may be used to reliably evaluate severity by using a decision tree classifier to predict severity codes based on it. Additionally, the suggested model (NMSC_KNOW) predicts severity codes with greater accuracy, demonstrating its value as a reliable tool for software development.

INTRODUCTION

Severity assessment is essential In several fields, such as quality control, risk management, healthcare, and software development [12]. Conventional severity measurements, such as the Glasgow Coma Scale in medical settings or several event severity scales in risk management, and weighted average frequently have problems averaging out severity levels, which could result in errors. The shortcomings of these traditional approaches which take into account the frequency and distribution of varying levels of severity can lead not accurate results [2].

A knowledge base is an organized set of facts, guidelines, and data that can be applied to specific domains to support reasoning, problem-solving, and decision-making. It is frequently at the heart of expert systems, which use a wealth of domain-specific information to emulate human decision-making. Data is frequently arranged in a knowledge base into facts, rules (if-then statements), and relationships, which enables systems to deduce new information or conclusions [11]. The decision tree is a type of information representation used for regression or classification. To arrive at a conclusion or prediction, they divide data into branches according to feature values and employ a tree structure. Decision trees are a common tool for data-driven reasoning in many industries, including banking and healthcare, because they are easy to use and grasp visually.

Software developers can imitate human-like reasoning for decision-making and prediction by integrating decision trees with knowledge bases [1, 5]. Creating intelligent applications like recommendation engines, diagnostic tools, and automated decision-support systems is a great use for this combo. While a knowledge base improves the system by offering a thorough understanding of the

¹University of Mosul, College of Computer Sciences and Mathematics, Dept. Software, Mosul, Iraq.
Email: ghadaaldabagh@uomosul.edu.iq

² University of Mosul, College of Computer Sciences and Mathematics, Dept. Software, Mosul, Iraq.
Email: dr.safwan1971@uomosul.edu.iq

*Correspondence: ghadaaldabagh@uomosul.edu.iq

problem domain, increasing accuracy, and managing more complicated scenarios, a decision tree, for example, can swiftly classify data. By combining these methods, programmers can produce software that is more durable, intelligent, and adaptive and that can make deft decisions by drawing from a large body of knowledge [14]. This study presents a novel measure that uses the weight of severity codes to provide a more realistic portrayal of severity levels.

The goal of the new measure is to improve upon the severity evaluation instruments already in use by including a more advanced method that considers the relative importance and incidence of each severity level. With this new metric, the developer team can gain a more precise and consistent understanding of severity distributions, which will enhance the quality and effectiveness of their assessments. This study employs a new measure to produce a new model using a decision tree model a knowledge-based approach and a breadth-first search to predict the severity code using the new measure.

RELATED WORK

Severity has been measured using a variety of techniques, from straightforward scales to complex models. For instance, the Glasgow Coma Scale is frequently used in medical settings to evaluate patients' levels of consciousness. Weighted average and Macro average are used to evaluate the ratio of infected instances without evaluating the severity level. Incident severity scales are used in other disciplines to categorize the significance of incidents. However, these approaches frequently fail to capture the entire range of severity levels, especially when averaging severity codes are required. While other metrics have been investigated in the past, the lack of a reliable, average-based severity measure has not been adequately addressed. This work attempts to fill the vacuum in the literature by addressing the need for a new strategy. Some important studies that address new metrics for software development and fault prediction are as follows:

1. In "Improved software fault prediction using new code metrics and machine learning algorithms", (2024), Singh and Chhabra [13] suggest three new code measures to close the gap, based on cohesion, coupling, and class complexity. The suggested metrics replace the complexity, coupling, and cohesion measurements in the Promise repository metrics suite, and a new metric suite is generated. Tests reveal that with fewer code measures than the current metrics suite, the suggested metrics suite improves AUC and precision by more than 2%, f1-score, and recall by about 1.5%.
2. IN "The impact of correlated metrics on the interpretation of defect models." (2021) Jiarpakdee, et al. [3] investigates how correlated metrics can affect the interpretation of defect models, proposing techniques to mitigate these issues.
3. IN "AutoSpearman: Automatically mitigating correlated software metrics for interpreting defect models." (2018), Jiarpakdee, J., Tantithamthavorn, C., & Treude, C. Jiarpakdee, et al. [3] introduce AutoSpearman, a tool designed to address the challenges posed by correlated metrics in defect prediction models.
4. IN "Towards identifying software project clusters about defect prediction. (2010). " Jureczko and Madeyski [4] explore the clustering of software projects to improve defect prediction, suggesting that clustering can enhance the prediction accuracy.[9]
5. IN "Transfer defect learning.", (2013), Nam, et al. [6] presents transfer defect learning, a novel approach that leverages defect data from different projects to improve prediction models.
6. IN "Software defect detection based on selected complexity metrics using fuzzy association rule mining and defective module oversampling.", (2019), Naufal and

Kusuma [7] propose a defect detection method combining complexity metrics, fuzzy association rule mining, and oversampling techniques to enhance defect prediction in software modules.

RULE-BASED EXPERT SYSTEM STRUCTURE

Newell and Simon suggested a production system model at the beginning of the 1970's, the fundamentals of new rule-based expert systems [9]. The basic idea of the design of production was that the problems were solved by applying the knowledge of the person (expressed as rules of production Facts or information unique to a certain problem are stored in the short-term memory, whereas production rules are stored in the long-term memory. Figure 1 (a) Explain The production system model and the basic structure of a rule-based expert system are shown. The fundamental rule-based expert system elements are the knowledge base, the database, the inference engine, the explanation facilities, and the user interface [10].

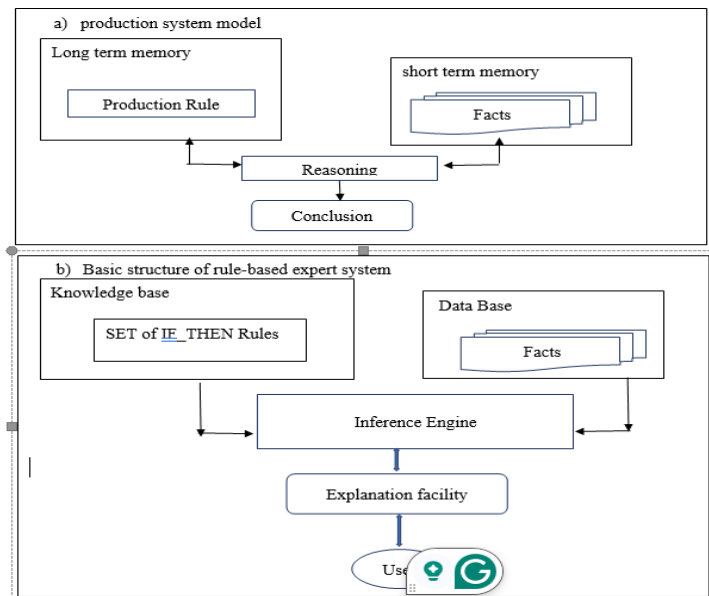


Figure 1: (a) Production System. (b) Basic Structure of a Rule-based Expert System.

The Knowledge Base

The domain knowledge necessary for problem-solving is contained in the knowledge base. A set of rules serves as the representation of knowledge in a rule-based expert system. Every rule follows the IF (condition) THEN (action) form and describes a relationship, suggestion, direction, strategy, or heuristic. A rule is said to fire and its action part is carried out when its condition part is met [10].

Facts

A set of facts is recorded in the database and is utilized to compare it with the IF (condition) portions of rules that are kept in the knowledge base [8].

Inference Engine

The expert system uses reasoning to arrive at a solution, which is carried out by the inference engine. It connects the facts in the database with the guidelines listed in the knowledge base [8].

Explanation Facilities

Using the explanation tools, the user can question the expert system on how a given conclusion is made or why a particular fact is required. An expert system needs to be able to defend its recommendations, analysis, and results.

User Interface

An expert system and a user looking for a solution to a problem communicate through the user interface. It is important to communicate in a nice and meaningful way [8]. Every rule-based expert system must include the above five elements. They make up its core, but there might be a few other parts as well as an external interface, developer interface, expert system shells, and knowledge engineer. Figure 1 (b) explains the complete structure of a rule-based expert system [8].

SEARCH TECHNIQUES

Search techniques are algorithms or approaches that are used to efficiently search and explore through vast datasets or problem spaces to locate answers, patterns, or particular information. Search strategies like Depth-First Search (DFS) and Breadth-First Search (BFS), depend only on the problem's initial state and potential solutions. These techniques, although methodically exploring potential solutions, can be computationally costly.

Depth-First Search

is a basic search technique that's used to explore and traverse tree and graph structures. It functions by exploring as far as it can along each branch before turning around, beginning at the root node (or any other random starting node in a graph). To keep track of nodes to investigate, DFS employs a stack-based method, either explicitly or by recursion. It thoroughly investigates every branch of a tree or graph until it arrives at a node that has no unexplored neighbors, at which point it turns around to look at other options. DFS's ability to examine deeper levels of the graph or tree structure first is one of its primary features. This makes it an excellent choice for situations in which a solution is located far from the root or when it is required to explore every path. DFS is frequently used for tasks like topological sorting, finding cycles in a graph, and resolving maze-like puzzles. DFS can occasionally become "stuck" in deep or endless pathways, even though it is efficient in terms of memory utilization (requiring memory proportionate to the depth of the tree), especially in graphs with numerous branches or loops. Although DFS is not as reliable as Breadth-First Search (BFS) in terms of locating the shortest path in an unweighted graph, its depth-orientedness makes it especially helpful for some

Breadth-First Search

When using breadth-first search, the search tree is examined from top to bottom using the method described until a solution is discovered. Each node in the node list is first examined to determine if it is a goal node; in the event that success, the software is terminated. If not, all of the node's successors are created. Next, a recursive search is performed on the list containing every newly created node. Until no more successors are produced, the entire process is repeated. This is a generic algorithm. That is, if the two application-specific functions "GoalReached" and "Successors" are available, it can be used for any kind of application. For a tree with constant branching factor b and depth d , the total computer time is thus given by equation (1)

$$c \cdot \sum_{i=0}^d b^i = \frac{b^{d+1} - 1}{b - 1} = O(b^d). \quad (1)$$

Although only the last level is saved in memory, the memory space requirement is also $O(bd)$. The speed of modern computers, which can create billions of nodes in a matter of minutes, means that the search quickly runs out of main memory [8].

Decision Tree Classifier

A supervised machine learning algorithm called a decision tree classifier is used for classification tasks. It operates by first dividing the dataset into subsets based on the value of input features, resulting in a three-like model of decisions. The tree construction process involves choosing the best feature to split the data at each node, usually using criteria like Gini impurity or information gain, to maximize the separation between classes. Decision tree classifiers are intuitive and simple to understand, as the decision-making process can be visualized and understood as a series of if-then statements.

Methodology of Evaluation Novel measure (NMSC)

Prediction severity code is a very important process, especially in software development. All previous measures are used to explain the ratio of infected instances without explaining the severity of infection of the instance. The new measure gives a full view of the severity ratio of each instance which helps the developer team to treat the most severity code for instances, and

reduce the time and effort of the developing team. Teams can effectively allocate resources and prioritize responses by using severity codes to categorize issues, bugs, or events in software systems according to their impact and immediacy. Usually, these codes have a low to high range, with each level denoting a distinct severity. The lowest severity level is trivial, and the highest level is blocking. These levels are explained below:

1. Trivial: denotes small problems that are usually non-critical or cosmetic and have little to no effect on functionality.
2. Normal: refers to normal severity like type mismatch
3. Minor: severity denotes problems that have a little more than effect but don't interfere with important functioning.
4. Major: severity refers to serious issues that compromise essential functions but may be remedied.

5. Blocking: the highest and most important severity that caused the failure of the system and completely stopped the system's operation. Severity codes help to identify and classify problems so that significant concerns are dealt with first and less important issues are handled according to urgency.

This paper applies the new measure (NMSC) and the (NMSC-KNOW) model on six datasets((eclipse, free desktop,gcc, gnome, Mozilla, and Winehq)), uses (NMSC) measure, and proposed model(NMSC-KNOW) to predict five types of severity (trivial, normal, minor, major, and, blocking) Figure 2 explains the framework of prediction severity code by using the new measure (NMSC)

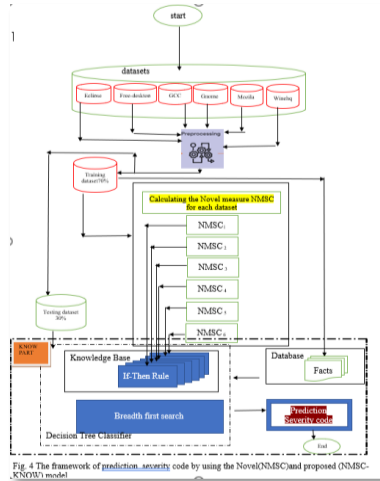


Figure 2: Framework of Prediction Using Novel (NMSC) and Proposed Model (NMSC-KNOW).

Evaluation of new measure (NMSC)

The new measure depends on the severity code and number of instances. The flowchart for calculating the new measure (NMSC) is explained in Figure 3.

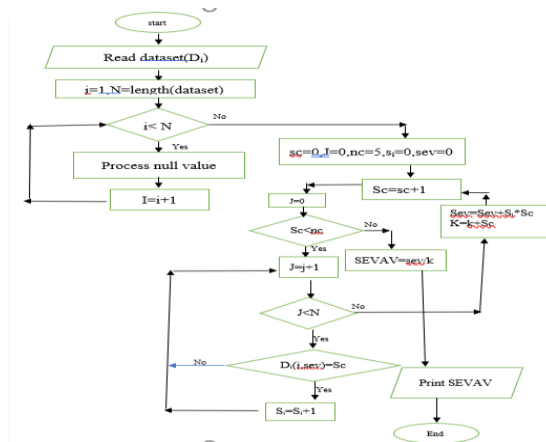


Figure 3: Flow Chart of Evaluation of the New Measure (NMSC).

The equations of evaluation the Novel measure (NMSC) explained as follows:

$$Sevm(i) = \sum_{i=1}^{sevc} \sum_{j=1}^{N_j} D(i, j) * i \dots \dots \dots (1)$$

$$WsevAv = \sum_{i=1}^{sevc} \frac{Sevm(i)}{\sum i} \dots \dots \dots (2)$$

Where

Sevm: The Severity Measure

Sevc: The Severity Code (Trivial, Normal, Minor, Major, Blocking)

N_j: Number of instances in each severity code

D: Dataset

WsevAv: Average of severity code

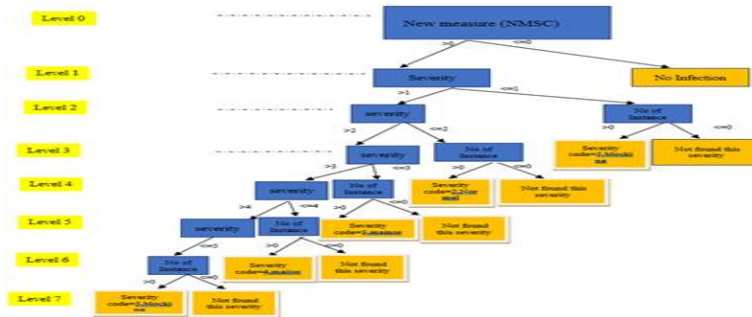
The average severity code measure value is explained below

$$WsevAv = \begin{cases} \leq 0 & \text{no infect} \\ > 0 & \text{infect found} \\ \geq 1 & \text{infected with high severity} \end{cases} \dots \dots \dots (3)$$

Proposed Model for Predicting the Severity Code Based on (NMSC)

The new measure (NMSC) provides a good idea for predicting the severity code.

Figure 4 explains the decision tree which uses (NMSC) to predict severity code



2. Data splitting was done using 80% training and 20% test datasets. This separation provided the model with an independent set of data for evaluation and ensured that it had enough data to learn from.
3. Model Training: The decision tree classifier was trained using the training set. Through training, the classifier acquired the ability to discriminate between samples according to their characteristics and the target variable.
4. Evaluation: To assess the trained model's effectiveness, it was put to the test on the test set. Several metrics were used to assess the model: accuracy (the percentage of correct predictions).

RESULTS AND ANALYSIS

This section explains the results of applying the severity measure by the decision tree classifier to predict the five severity codes in the case of no infection. Also, an analysis of the results is made to make clear observations about the performance of the model that uses a new measure (NMSC).

Experimental Results

The (NMSC-KNOW) achieved highest accuracy than the decision tree's accuracy classifier using the (NMSC) only is 0.8553540587. Table 1 explains the effectiveness of (NMSC) measure in predicting severity code in the Eclipse dataset

Table(1) Effectiveness of (NMSC) measure on predicting severity code in the Eclipse dataset

Severity found P/N	Severity code					NMSC	Accuracy of applying NMSC	Precision of applying NMSC	F1-score of applying NMSC	accuracy of (NMSC-KNOW)	precision of (NMSC-KNOW)	F1-score of (NMSC-KNOW)
	1	2	4	5	6	0.8553540587	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
F	F	F	F	F	F	2.31415	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
N	F	F	F	F	F	738	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
F	N	F	F	F	F	2387	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
F	N	F	F	F	F	0.60283	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
F	N	F	F	F	F	320376	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
F	F	N	F	F	F	1.98453	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
F	F	N	F	F	F	830871	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
F	F	F	N	F	F	2.15910	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
F	F	F	N	F	F	857735	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
F	F	F	F	N	F	2.10946	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
F	F	F	F	N	F	704673	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
N	N	F	F	F	F	0.58935	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
N	N	F	F	F	F	821024	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
N	N	N	F	F	F	0.25973	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
N	N	N	F	F	F	913794	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
N	N	N	N	F	F	0.10469	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
N	N	N	N	F	F	033428	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
N	N	N	N	N	F	96200345	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000
N	N	N	N	N	N	0.00000	0.8553540587	0.726577	0.78447	1.0000	1.0000	1.0000

Figure 6: Explains The NMSC's Effect on Predicting the Severity Code in Eclipse Dataset.

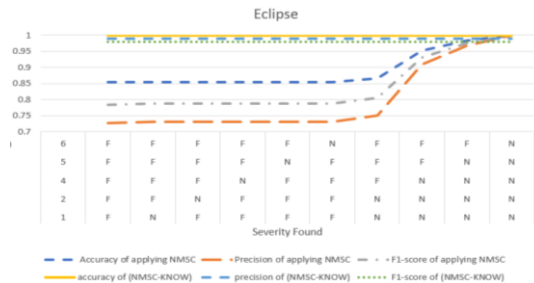
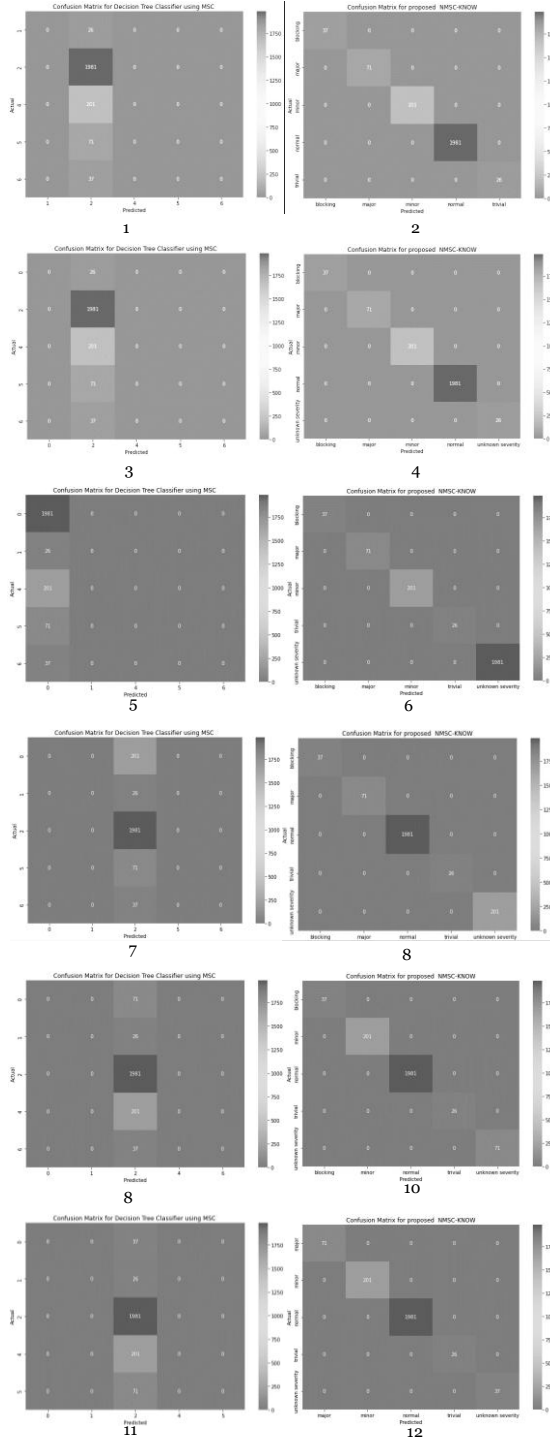


Figure 7: The NMSC's Effect on Predicting the Severity Code in Eclipse Dataset.

Figure 7 explains the confusion matrices of the classifier, indicating its performance

across different severity levels in Eclipse dataset where (1), (3), (5), (7), (9), (11), (13), (15), (17) classifier using NMSC measure and ,(2), (4), (6), (8), (10), (12), (14), (16), (18) of NMSC-KNOW model on eclipse dataset.



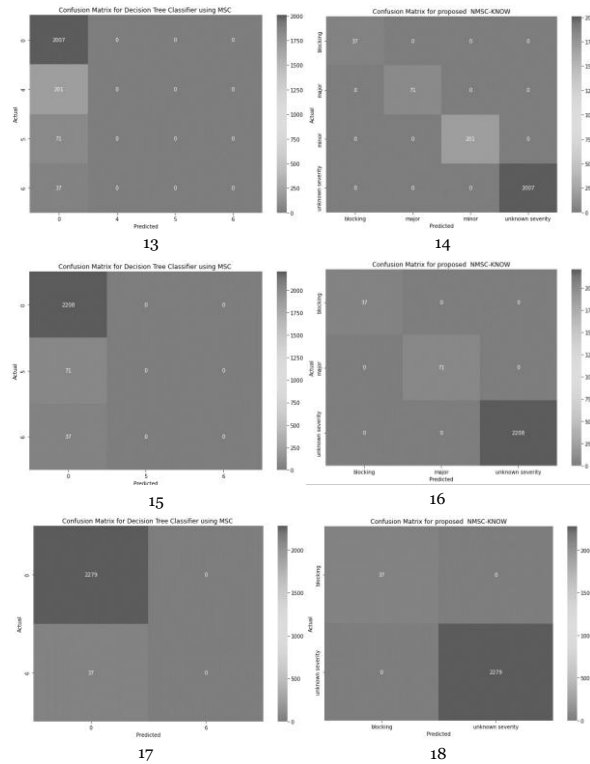


Figure 8: The Confusion Matrices of the Classifier Using Eclipse Dataset.

The Table 2 explains the effectiveness of (NMSC) measure in predicting severity code in Free-desktop dataset

Table(2) Effectiveness of (NMSC) measure on predicting severity code in the Free-desktop dataset

	Severity code					NMSC measure	Accuracy of applying NMSC	Precision of applying NMSC	F1-score of applying NMSC	accuracy of (NMSC-KNOW)	precision of (NMSC-KNOW)	F1-score of (NMSC-KNOW)
	1	2	4	5	6							
Severity found F/N	F	F	F	F	F	2.07595 526561	0.91770186 33540373	0.749912 5695868	0.80377 5613278	1.0000	1.0000	1.0000
	N	F	F	F	F	2.31344 058121	0.86597492 43406831	0.749912 5695868	0.80377 5613278	1.0000	1.0000	1.0000
	F	N	F	F	F	0.58709 565819	0.86597492 43406831	0.749912 5695868	0.80377 5613278	1.0000	1.0000	1.0000
	F	F	N	F	F	2.06192 700224	0.85535405 87219344	0.749912 5695868	0.80377 5613278	1.0000	1.0000	1.0000
	F	F	F	N	F	2.12160 525860	0.86597492 43406831	0.749912 5695868	0.80377 5613278	1.0000	1.0000	1.0000
	F	F	F	F	N	2.20152 222798	0.86597492 43406831	0.749912 5695868	0.80377 5613278	1.000	1.000	1.000
	N	N	F	F	F	0.57913 855734	0.87375702 55079983	0.763451 3396245	0.81488 8301131	1.0000	1.0000	1.0000
	N	N	N	F	F	0.31966 78775	0.95206658 01988759	0.887489 4415276	0.91396 3970727	1.0000	1.0000	1.0000
	N	N	N	N	F	0.11987 545407	0.98313878 08041504	0.966561 8623210	0.97477 9850686	1.0000	1.0000	1.0000
	N	N	N	N	N	0.00000 1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Figure 9: Explain the NMSC's Effect on Predicting the Severity Code in Free-Desktop Dataset.

Figure .9 explains the confusion matrices of the classifier, indicating its performance across different severity levels in Eclipse dataset where (1), (3), (5), (7), (9), (11), (13),

(15), (17) classifier using NMSC measure and ,(2), (4), (6), (8), (10), (12), (14), (16), (18) of NMSC-KNOW model on Free-desktop dataset

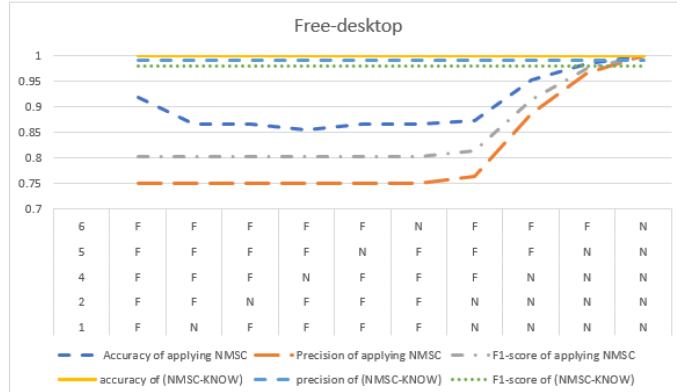
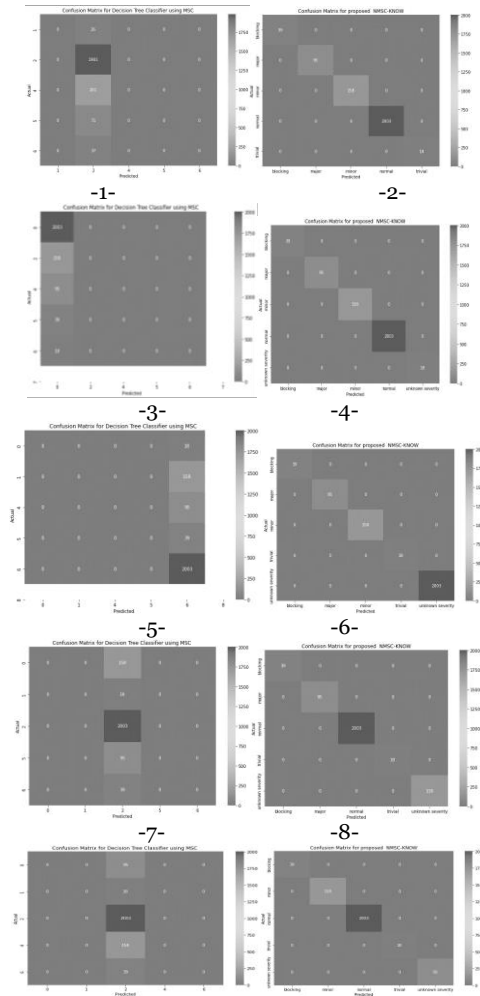


Figure 10: The NMSC's Effect on Predicting the Severity Code in Free-Desktop Dataset.



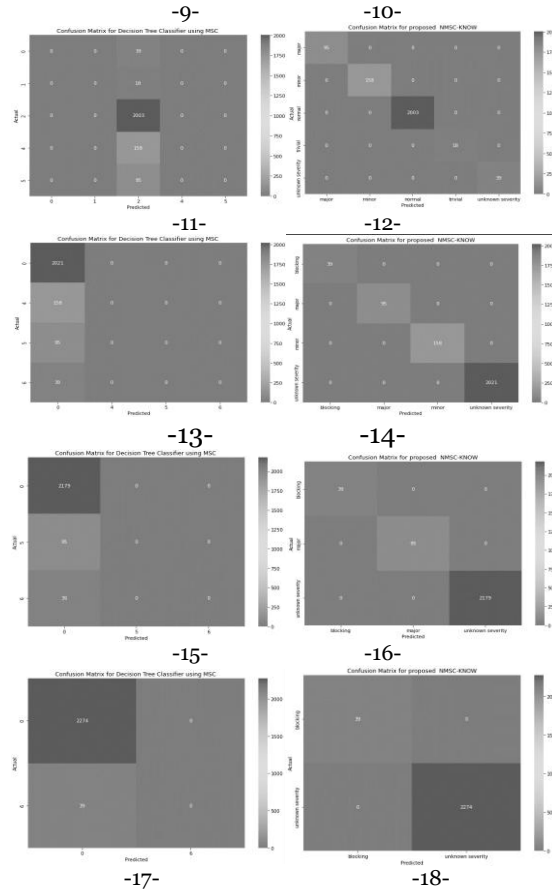


Figure 11: The Confusion Matrices of the Classifier Using Free-Desktop Dataset.

Table 3 explains effectiveness of (NMSC) measure on predicting severity code in on Gcc-bug-report dataset

Table3_effectiveness of (NMSC) measure on predicting severity code in on Gcc-bug-report dataset

Severity found F/N	Severity code					NMSC measure	Accuracy of applying NMSC	Precision of applying NMSC	F1-score of applying NMSC	accuracy of (NMSC-KNOW)	precision of (NMSC-KNOW)	F1-score of (NMSC-KNOW)
	1	2	4	5	6							
Severity found F/N	F	F	F	F	F	2.01694469579	0.993542399752523	0.9871265001060	0.990324058548	1.0000	1.0000	1.0000
	N	F	F	F	F	2.31344058121	0.993542399752523	0.9871265001060	0.990324058548	1.0000	1.0000	1.0000
	F	N	F	F	F	0.02931873197	0.999961331735045	0.9871265001060	0.990324058548	1.0000	1.0000	1.0000
	F	F	N	F	F	2.01276845858	0.993542399752523	0.9871265001060	0.990324058548	1.0000	1.0000	1.0000
	F	F	F	N	F	1.99590883428	0.993542399752523	0.9871265001060	0.990324058548	1.0000	1.0000	1.0000
	F	F	F	F	N	2.01313967966	0.993542399752523	0.9871265001060	0.990324058548	1.000	1.000	1.000
	N	N	F	F	F	0.02901711483	0.994741115966126	0.9874338725158	0.990324058548	1.0000	1.0000	1.0000
	N	N	N	F	F	0.02484087762	0.999961331735045	0.9895098877935	0.992118606142	1.0000	1.0000	1.0000
	N	N	N	N	F	0.00380501612	0.999497312555585	0.9989948778058	0.999246032022	1.0000	1.0000	1.0000
	N	N	N	N	N	0.00000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Figure 12: The NMSC's Effect on Predicting the Severity Code in Free-Desktop Dataset.

Figure 11 explains the confusion matrices of the classifier, indicating its performance across different severity levels in Eclipse dataset where (1), (3), (5), (7), (9), (11), (13), (15), (17) classifier using NMSC measure and ,(2), (4), (6), (8), (10), (12), (14), (16), (18) of NMSC-KNOW model on Free-desktop dataset.

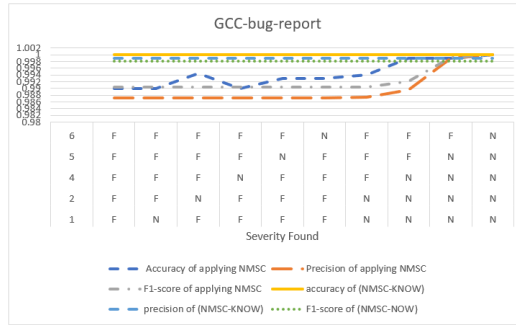
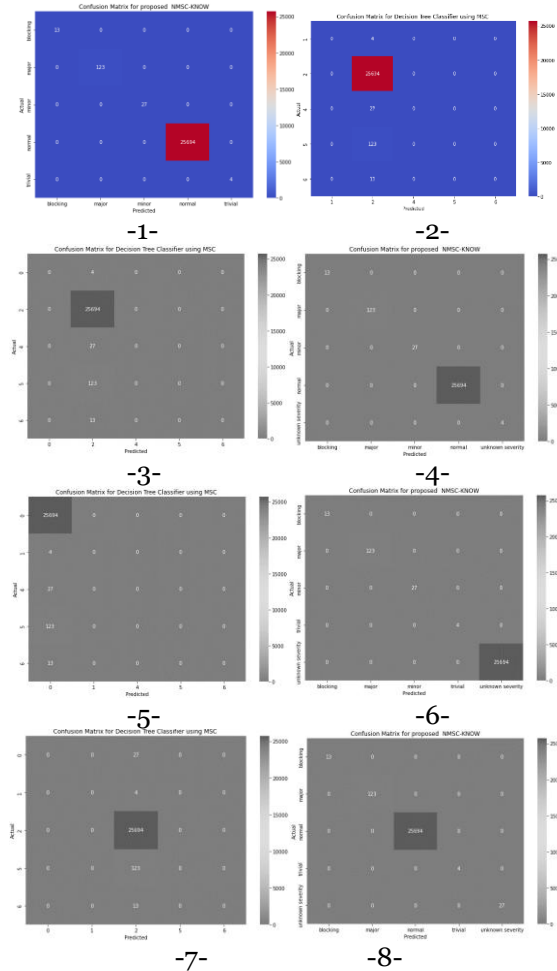


Figure 13: The NMSC's Effect on Predicting the Severity Code in GCC-Bug-Report Dataset.



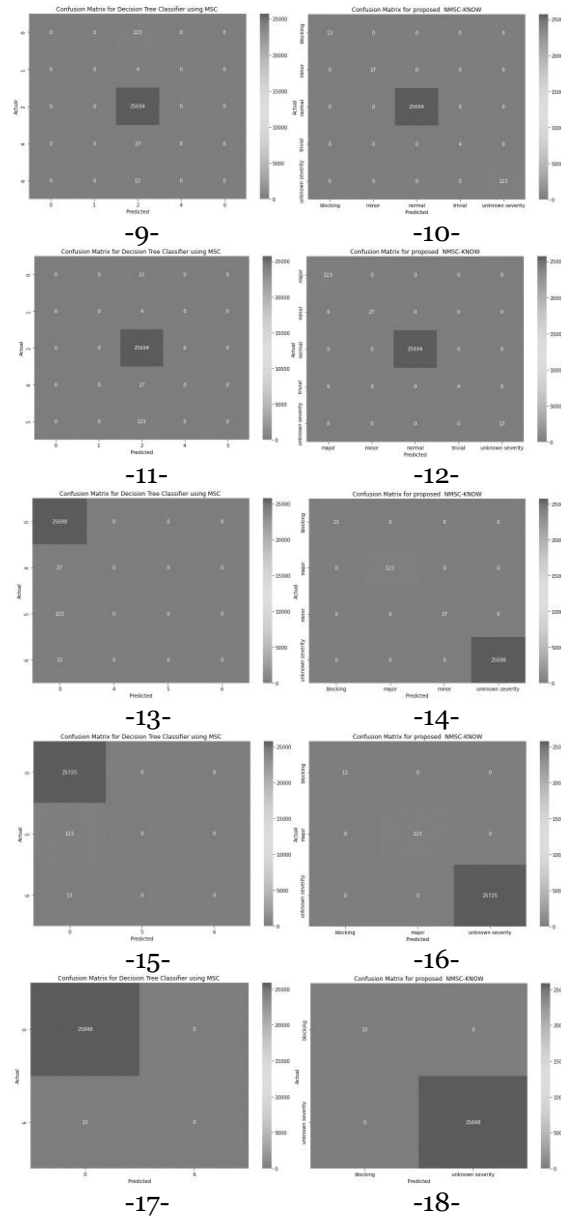


Figure 14: The Confusion Matrices of the Classifier Using GCC-Bug-Report Dataset

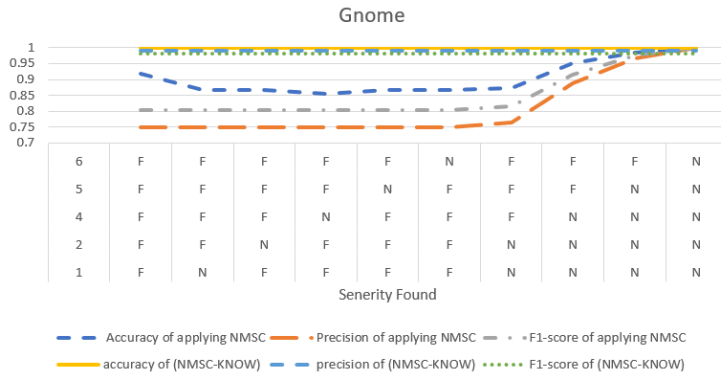
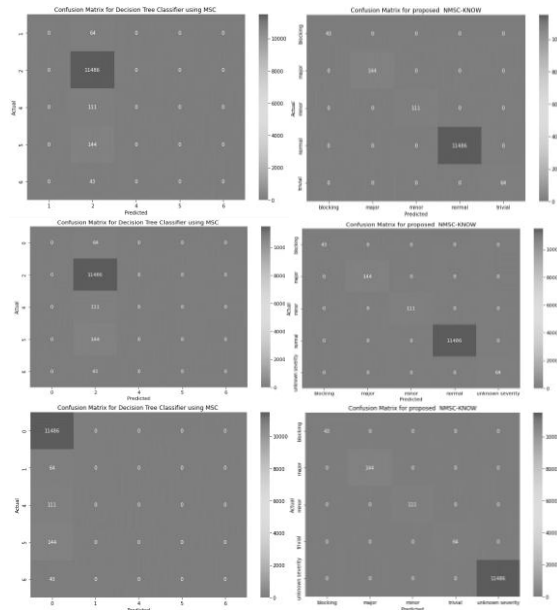
The Table 4 explains the effectiveness of (NMSC) measure in predicting severity code in the Genome dataset

Figure 12 explains the NMISC's effect on predicting the severity code in the Gnome dataset.

Figure 13 explains the confusion matrices of the classifier, indicating its performance across different severity levels where (1), (3), (5), (7), (9), (11), (13), (15), (17) classifier using NMSC measure and, (2), (4), (6), (8), (10), (12), (14), (16), (18) of NMISC-KNOW model on Gnome dataset

Table (4) Effectiveness of (NMSC) measure on predicting severity code in the Gnome dataset

Severity found F/N	Severity code					NMSC measure	Accuracy of applying NMSC	Precision of applying NMSC	F1-score of applying NMSC	accuracy of (NMSC-KNOW)	precision of (NMSC-KNOW)	F1-score of (NMSC-KNOW)
	1	2	4	5	6							
F	F	F	F	F	F	2.07216 407832	0.9694463 20054017	0.939826 1674662	0.95440 648256	1.0000	1.0000	1.0000
N	F	F	F	F	F	2.06713 369345	0.9694463 20054017	0.939826 1674662	0.95440 6482569	1.0000	1.0000	1.0000
F	N	F	F	F	F	0.13833 558406	0.9694463 20054017	0.939826 1674662	0.95440 6482569	1.0000	1.0000	1.0000
F	F	N	F	F	F	2.02678 933153 27483	0.9694463 20054017 5	0.939826 1674662	0.95440 6482569	1.0000	1.0000	1.0000
F	F	F	F	N	F	2.01046 590141 796	0.9694463 20054	0.939826 1674662	0.95440 6482569	1.0000	1.0000	1.0000
F	F	F	F	F	N	2.04593 180283 592	0.9694463 20054	0.939826 1674662	0.95440 6482569	1.000	1.000	1.000
N	N	F	F	F	F	0.13330 519918	0.9748480 75624578	0.957417 9834623	0.96783 3135793	1.0000	1.0000	1.0000
N	N	N	F	F	F	0.08793 045239	0.9842167 45442268	0.968682 6020089	0.97638 7891326	1.0000	1.0000	1.0000
N	N	N	N	F	F	0.02623 227548	0.9963706 95476029	0.992754 5628033	0.99455 9342163	1.0000	1.0000	1.0000
N	N	N	N	N	N	0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

**Figure 15:** The NMSC's Effect on Predicting the Severity Code in Gnome Dataset.

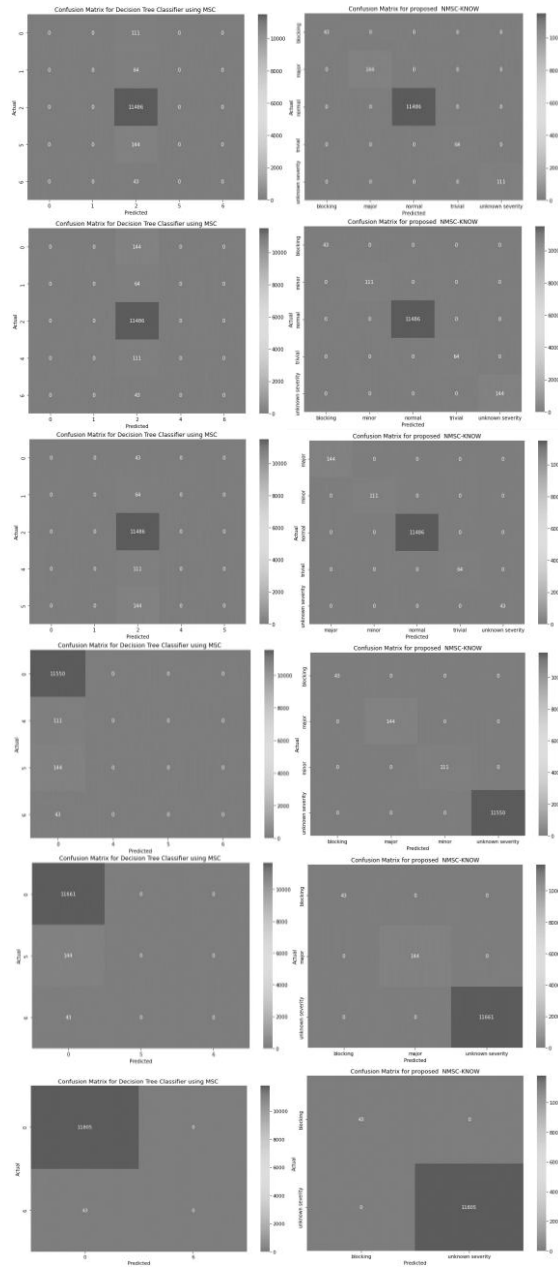
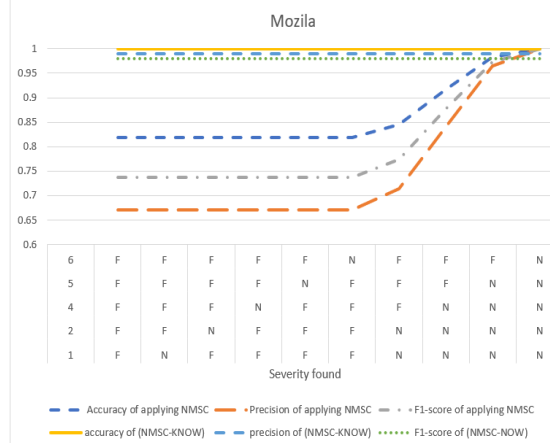
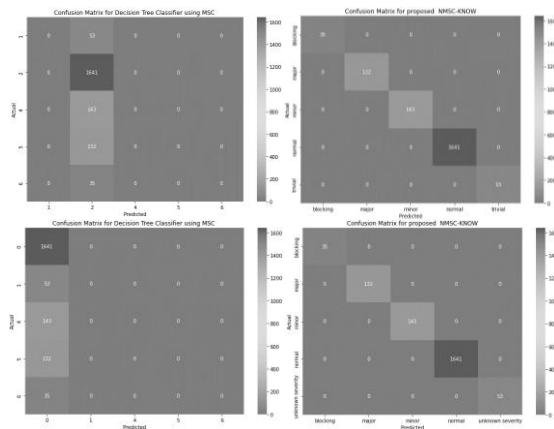


Figure 16: The Confusion Matrices of the Classifier Using Gnome Dataset.

Table 5 explains the effectiveness of (NMSC) measure in predicting severity code in the Mozilla dataset. Figure 14 explains the NMSC's effect on predicting the severity code in the Mozilla dataset. Figure 15 explains the confusion matrices of the classifier, indicating its performance across different severity levels where (1), (3), (5), (7), (9), (11), (13), (15), (17) classifier using NMSC measure and, (2), (4), (6), (8), (10), (12), (14), (16), (18) of NMSC-KNOW model Mozilla dataset.

Table(5) the effectiveness of (NMSC) measure in predicting severity code in the Mozilla dataset

Severity found F/N	Severity code				NMSC measure	Accuracy of applying NMSC	Precision of applying NMSC	F1-score of applying NMSC	accuracy of (NMSC-KNOW)	precision of (NMSC-KNOW)	F1-score of (NMSC-KNOW)
	1	2	4	5	6						
F	F	F	F	F	F	2.37987 823136	0.8188622 75449101	0.670535 4261536	0.73731 3028264	1.0000	1.0000
N	F	F	F	F	F	2.34943 607146	0.8188622 75449101	0.670535 4261536	0.73731 3028264	1.0000	1.0000
F	N	F	F	F	F	0.74977 542668	0.8188622 75449101	0.670535 4261536	0.73731 3028264	1.0000	1.0000
F	F	N	F	F	F	2.08523 804770	0.8188622 75449101	0.670535 4261536	0.73731 3028264	1.0000	1.0000
F	F	F	N	F	F	2.07795 189140	0.8188622 75449101	0.670535 4261536	0.73731 3028264	1.0000	1.0000
F	F	F	F	N	N	2.25711 148817	0.8188622 75449101	0.670535 4261536	0.73731 3028264	1.000	1.000
N	N	F	F	F	F	0.71933 326679	0.8453093 81237525	0.714547 9500081	0.77444 7859284	1.0000	1.0000
N	N	N	F	F	F	0.42469 308314	0.9166666 66666666	0.840277 77777777	0.87681 1594202	1.0000	1.0000
N	N	N	N	N	F	0.12276 674318	0.9825349 30139720	0.965374 8889446	0.97387 9324160	1.0000	1.0000
N	N	N	N	N	N	1.0000	1.000	1.0000	1.0000	1.0000	1.0000

**Figure 17:** Explain The Effthieve of Severity Average on the Accuracy of Predicting Severity Code in Mozilla Dataset.

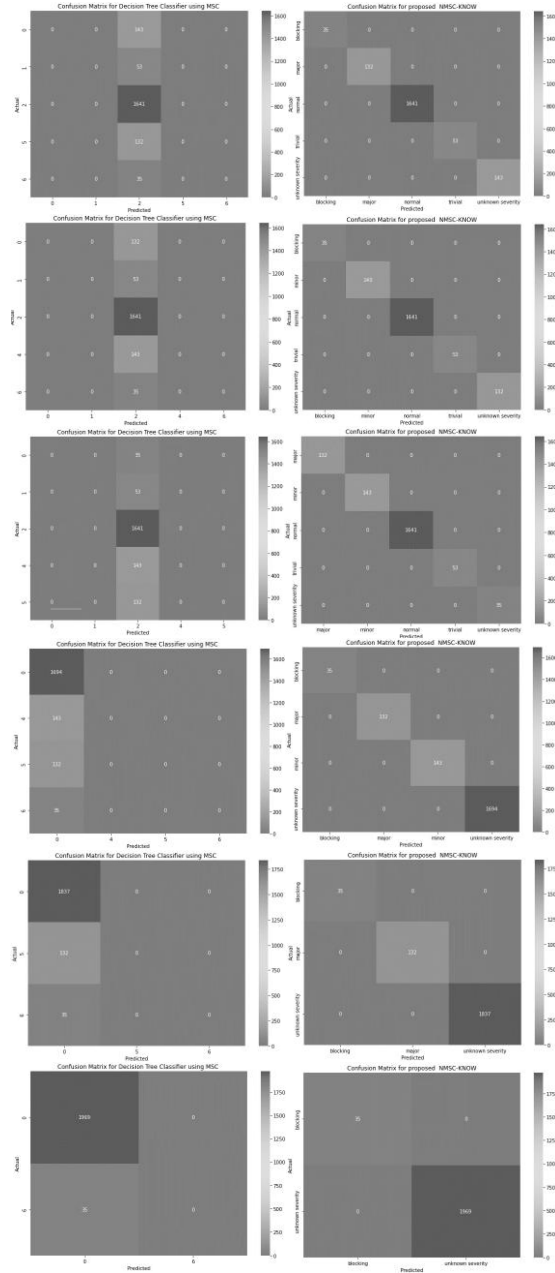


Figure 18: The Confusion Matrix for the Classifier, Indicating its Performance Across Different Severity Levels in Mozilla Dataset.

The Table 6 explains the effectiveness of (NMSC) measure in predicting severity code in the Winehq dataset

Figure 16 explains the NMISC's effect on predicting the severity code in the Winehq dataset.

Figure 17 explains the confusion matrices of the classifier, indicating its performance across different severity levels where (1), (3), (5), (7), (9), (11), (13),

(15), (17) classifier using NMSC measure and, (2), (4), (6), (8), (10), (12), (14), (16), (18) of NMSC-KNOW model Winehq dataset.

Table(6) the effectiveness of (NMSC) measure in predicting severity code in the Winehq dataset

Severity code	1	2	4	5	6	NMSC measure	Accuracy of applying NMSC	Precision of applying NMSC	F1-score of applying NMSC	accuracy of (NMSC-KNOW)	precision of (NMSC-KNOW)	F1-score of (NMSC-KNOW)
F	F	F	F	F	F	2.07595 526561	0.9177018 63354037	0.842176 7100034	0.87831 8706465	1.0000	1.0000	1.0000
N	F	F	F	F	F	2.04038 522799	0.9177018 63354037	0.842176 7100034	0.87831 8706465	1.0000	1.0000	1.0000
F	N	F	F	F	F	0.23625 349487	0.9177018 63354037	0.842176 7100034	0.87831 8706465	1.0000	1.0000	1.0000
F	F	N	F	F	F	1.95914 88039	0.9177018 63354037	0.842176 7100034	0.87831 8706465	1.0000	1.0000	1.0000
F	F	F	N	F	F	2.03401 677539	0.9177018 63354037	0.842176 7100034	0.87831 8706465	1.0000	1.0000	1.0000
F	F	F	F	N	N	2.03401 677539	0.9177018 63354037	0.842176 7100034	0.87831 8706465	1.000	1.000	1.000
N	N	F	F	F	F	0.15874 495184	0.9635093 16770186	0.910483 4159561	0.93182 5697061	1.0000	1.0000	1.0000
N	N	N	F	F	F	0.08387 698042	0.9860248 44720496	0.972244 9944060	0.97908 6436899	1.0000	1.0000	1.0000
N	N	N	N	F	F	0.04193 849021	0.9906832 29813664	0.981453 2618340	0.98604 6646834	1.0000	1.0000	1.0000
N	N	N	N	N	N	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

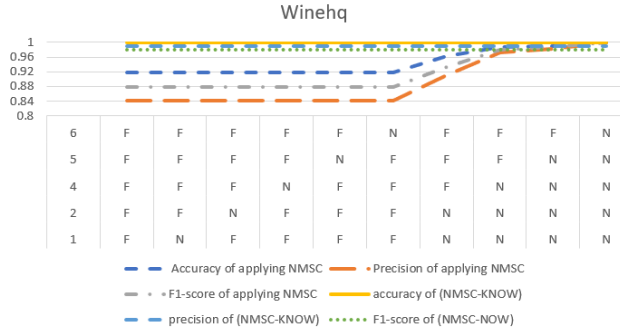
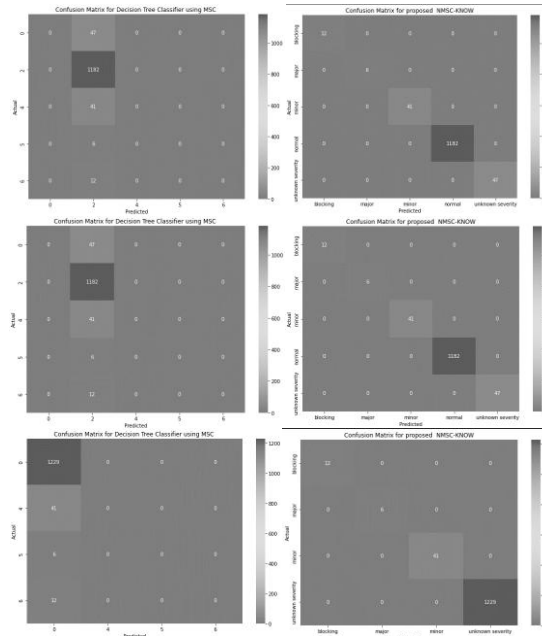


Figure 19: The Effective if Severity Average on the Accuracy of Predicting Severity Code in Winehq Dataset.



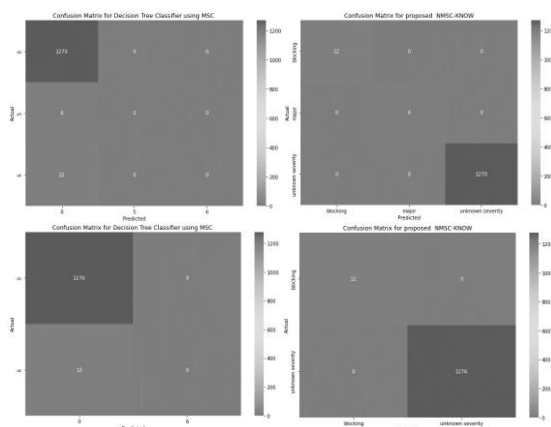


Figure 20: The Confusion Matrix for the Classifier, Indicating its Performance Across Different Severity Levels in Winehq Dataset.

CONCLUSION

This paper proposes novel measure called severity measure and proposes a model use this measure and approach of knowledge base to predict the severity code. The main purpose of this paper was to propose a new measure depending on the severity weight and propose a model that combines some algorithms of meta-learning (decision tree classifier) with a knowledge base approach and search methods (breadth-first search) to get a result most accurate model for predicting the severity code. The proposed measure (severity measure) and proposed knowledge base model have been applied on six datasets (Eclipse, Free desktop, GCC-bug-report, Gnome, Mozilla, and winehq). These datasets vary based on the number of projects(pattern), It is worth noting that the datasets contain five types of risk levels ranging from 1 to 6, with the absence of the number 3, as a dividing line indicating the transition of the risk level from the levels that have less impact on the operation of the code and the levels that actually affect the functions.

Each experiment used the proposed measure (NMISC) at first time, the second time used the proposed model (NMISC-KNOW) based on three evaluation metrics (accuracy, precision, and f1-score). The experimental results have shown that the proposed model (NSEV-KNOW) has produced the highest predicting accuracy and the minimum number of selected features for most of the datasets compared to the other method. Also, it was shown that the value of the proposed measure depends on the weight of severity found. It is inversely proportional to the degree of severity, increasing as the severity decreases and decreasing as the severity increases. In addition, the proposed model has the top accuracy, top precision and used the minimum features. As future works suggest hybridization of multilayer perceptron with expert system using proposed measure.

REFERENCES

[1] M. Al-Shamdeen, F. M. Ramo, *Performance Evaluation for Face Mask Detection Based on Mult Modification of Yolov8 Architecture*, Informatyka, Automatyka, Pomiar w Gospodarce i Ochronie Środowiska **14** (2024), 89-95.

- [2] E. Dunbar, K. Rumsey, J. Davis, P. O'Brien, A. M. Ritter, *Improving Glasgow Coma Scale Accuracy Among Emergency Medical Services Providers*, JEMS (2024).
- [3] J. Jiarpakdee, C. Tantithamthavorn, C. Treude, *Autospearman: Automatically Mitigating Correlated Software Metrics for Interpreting Defect Models*, in: Proceedings of the 34th International Conference on Software Maintenance and Evolution, Madrid, Spain, Institute of Electrical and Electronics Engineers Inc, 2018, 92-103.
- [4] M. Jureczko, L. Madeyski, *Towards Identifying Software Project Clusters with Regard to Defect Prediction*, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering, ACM Digital Library, 2010, 1-10.
- [5] I. D. Mienye, N. Jere, *A Survey of Decision Trees: Concepts, Algorithms, and Applications*, IEEE Access **12** (2024), 86716-86727.
- [6] J. Nam, S. J. Pan, S. Kim, *Transfer Defect Learning*, in: 2013 35th International Conference on Software Engineering (ICSE), IEEE, 2013, 382-391.
- [7] M. F. Naufal, S. F. Kusuma, *Software Defect Detection Based on Selected Complexity Metrics Using Fuzzy Association Rule Mining and Defective Module Oversampling*, in: 2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE), IEEE, 2019, 330-335.
- [8] M. Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*, Pearson Education, 2005.
- [9] A. Newell, H. A. Simon, *Human Problem Solving*, Upper Saddle River/Prentive Hall, 1972.
- [10] S. Ohlsson, *The Problems With Problem Solving: Reflections on the Rise, Current Status, and Possible Future of a Cognitive Research Paradigm*, The Journal of Problem Solving **5** (2012), 7.
- [11] A. L. Okute, P. Olom, A. Ewuru, *Artificial Intelligence and Expert Systems in Information Processing: An Imperative for Global Competitiveness*, African Journal of Vocational Education **6** (2016), 197-208.
- [12] G. Pascarella, M. Rossi, E. Montella, A. Capasso, G. De Feo, G. Botti Snr, A. Nardone, P. Montuori, M. Triassi, S. D'Auria, A. Morabito, *Risk Analysis in Healthcare Organizations: Methodological Framework and Critical Variables*, Risk Management and Healthcare Policy **14** (2021), 2897-2911.
- [13] M. Singh, J. K. Chhabra, *Improved Software Fault Prediction Using New Code Metrics and Machine Learning Algorithms*, Journal of Computer Languages **78** (2024), 101253.
- [14] M. Soori, F. K. G. Jough, R. Dastres, B. Arezoo, *AI-Based Decision Support Systems in Industry 4.0, A Review*, Journal of Economy and Technology (2024).