Enhancement of Recommendation Engine Technique for Bug System Fixes

Jalal Sadoon Hameed Al-Bayati [1], Mohammed Al-Shamma [2], and Furat Nidhal Tawfeeq [1,*]

[1] Department of Website, University of Baghdad, Baghdad, Iraq
[2] Department of Computer Engineering, University of Baghdad, Baghdad, Iraq
Email: jalal.hameed@uobaghdad.edu.iq (J.S.H.A.-B.); m.alshammaa@coeng.uobaghdad.edu.iq (M.A.-S.); furat@bccru.uobaghdad.edu.iq (F.N.T.)
*Corresponding author

***Abstract*—This study aims to develop a recommendation engine methodology to enhance the model's effectiveness and efficiency. The proposed model is commonly used to assign or propose a limited number of developers with the required skills and expertise to address and resolve a bug report. Managing collections within bug repositories is the responsibility of software engineers in addressing specific defects. Identifying the optimal allocation of personnel to activities is challenging when dealing with software defects, which necessitates a substantial workforce of developers. Analyzing new scientific methodologies to enhance comprehension of the results is the purpose of this analysis. Additionally, developer priorities were discussed, especially their utility in allocating a problem to a specific developer. An analysis was conducted on two key areas: first, the development of a model to represent developer prioritizing within the bug repository, and second, the use of hybrid machine learning techniques to select bug reports. Moreover, we use our model to facilitate developer assignment responsibilities. Moreover, we considered the developers' backgrounds and drew upon their established knowledge and experience when formulating the pertinent objectives. An examination of two individuals' experiences with software defects and how their actions impacted their rankings as developers in a software project is presented in this study. Researchers are implementing developer categorization techniques, assessing severity, and reopening bugs. A suitable number of bug reports is used to examine the model's output. A developer's bug assignment employee has been established, enabling the program to successfully address software maintenance issues with the highest accuracy of 78.38%. Best engine performance was achieved by optimizing and cleansing data, using relevant attributes, and processing it using deep learning.**

***Keywords*—bugs, fusion of intelligent optimization, artificial neural networks, machine and deep learning**

I. Introduction

Open-source bug reports play a significant role in software development cycles, accounting for up to 60% of reported issues within the software development system. The "10x Rule" posits that bugs that deviate significantly from the norm are typically more challenging to identify and rectify than other bugs. However, specific issues remain undetectable or unpatched based on our current understanding, bug identification, and resolution capabilities. In addition, there has been a substantial rise in software flaws. Furthermore, it is worth noting that 180 issues were detected and documented within Eclipse's bug-tracking system during software development. In addition, Debian has generated approximately 140 unresolved bug reports [1]. Software developers employ bug-reporting tools to streamline the process of detecting and resolving software defects. Furthermore, assigning problem reports in the software development business is a significant difficulty, requiring careful attention to ensure proper allocation [2].

Open-source project developers commonly utilize an open bug library to manage and track all reported software defects. The task of rectifying the bug necessitates the delegation of the error report to group members. Upon the arrival of a novel writing, a select cohort of developers is tasked with rectifying the software defect. Consequently, this assists bug triage personnel in determining the appropriate prioritization for bug resolution [3]. Bug repositories are problem-tracking systems encompassing a comprehensive database of software and programming codes. In contrast to proprietary business databases, open-source bug libraries are accessible to all users without any cost. The utilization of repositories is crucial to facilitating cooperation among programmers and supporting the functionality of the associated project [4].

The study utilizes datasets sourced from the same open-source project as Eclipse. A recommendation engine [5–7] is employed, incorporating optimization techniques through utilizing various machine learning algorithms, including C4.5, neural networks, deep learning architectures, and optimization techniques using the most recent evolutionary algorithm. Several researchers utilized support vector machines and explored the application of unsupervised learning techniques. Moreover, the researchers incorporated information derived from the Firefox browser, which is renowned for being an open-source software initiative. After a brief introduction, the article discusses the background and methodology process in Section III. A detailed description of the methodology

Manuscript received November 7, 2023; revised November 17, 2023; accepted January 10, 2024; published April 28, 2024.

used in this study is presented in Section IV of this work. In Section V, clustering is used to make the expected outcomes. The final section of this study discussed the study's conclusion and prospects.

II. Background

This section provides a concise overview of the diverse bug-fixing methodologies proposed by multiple scholars. Various methods have been proposed to identify the optimal bug framework that offers recommendations for the most effective bug-fixing applications. Xuan *et al.* [3] examined a social network-based methodology for prioritizing bug reports within the context of the Eclipse and Mozilla bug repositories. Shokripour *et al.* [8] employed a time-dependent bug identification technique in their study. The time metadata for each word in the database was examined. The statistical method known as Term Frequency-Inverse Document Frequency (TF-IDF, which is a metric utilized in machine learning and information retrieval) is employed to observe the syntactic variation of terms in documents. Jin *et al.* [9] employ an enhanced Linear Discriminant Analysis model to automate the classification of faults. The eligible bug reports were defined based on the distribution of bug problems, and the similarities between the bug fix creator and the distribution were examined. The proposed technique utilizes developer demographics to suggest a community of skilled developers who regularly participate in bug resolution and possess significant technical expertise.

The present study examined two distinct methodologies in executing these investigations: social network measurements and machine learning algorithms. Considering the given characteristics, we elucidate the methodologies employed in summarizing the comparisons, considering the number of programs, variables considered, and specific techniques utilized. This paper provides a comprehensive overview of the existing knowledge about problem triaging. Bug repositories are problem-tracking systems encompassing a comprehensive database of hardware, software, and programming concerns. Open-source bug repositories, in contrast to closed-source commercial repositories, are accessible to all individuals without any cost. These repositories play a crucial role in software development by facilitating progress exchange among programmers during the development process. Haruna *et al.* [10] utilized fine-tuned a pre-trained Bidirectional Encoder Representations from Transformers (BERT) triplet network that uses pairs. Duplicate-original bug reports made the positive class, and Original-Unrelated bug reports made the negative class. The original bug report was the same (anchor) in both positive and negative cases. In Lerch's study [11], an analysis of existing bug reports containing similar stack traces was proposed based on stack traces available from execution. Therefore, the user does not need to fill out the whole bug report only to discover that it is a duplicate of another one.

Wang *et al.* [12] utilized both the bug information, such as the summary and description, and the execution information to compute the similarities between the two bug reports. Utilizing execution information is a compelling technique. They attempt to duplicate the bug by following instructions to recreate it and monitor the functions invoked during the process. They use this data to construct a secondary collection of vectors and combine similarity ratings derived from bug report information and execution data. Their experiment attained a 90% recall rate for $k = 5$ on a significantly limited dataset consisting of only 232 bug reports, with a mere 42 pairs of duplicates. Kukkar *et al.* [13] used a Convolutional Neural Network (CNN) for feature extraction, a deep learning model that captures the syntactic and semantic meaning of features for similarity computations. The study utilizes several openly available bug data sources, for example, Eclipse, Open Office, Firefox, etc. The paper reports 89%,83%, and 78% recall rates for $k = 5$ for Eclipse, Open Office, and Firefox, respectively. The formula for the recall is based on the classification approach. It also accounts for the correct classification of non-duplicate reports comprising 80% to 90% of the data. Alipour *et al.* [14] incorporated more features based on bug reports and additional information already available. Contextual features improve the ability to identify duplicate bugs. Six contexts were created, each linked to a list of words used to define the context. The researchers used vectors from these new features and typical bug report details to compute the similarity between the two bugs. Kucuk and Tuzun [15] characterized Master, Duplicate, and Unique bug reports. Classification was also made of resolved master-duplicate bug reports and unresolved master-duplicate bug reports, along with measures that could be taken for each category. If the bug tracking system allows, the proposed actions suggest reopening the resolved bug report or creating a new scope. If there is still an open Master bug report, the new bug report cannot be submitted. Moreover, they identified severe differences between parent and child bug reports in both categories based on severity, the number of users involved, the bug surface time, and the lifecycle time. As part of the SiameseQAT, Rocha and Carvalho [16] considered information from individual bug reports and clusters of Parent-Child Reports in which each parent can have multiple children. As a result, the Parent Report becomes the collection's centroid. In addition to syntactical and semantic learning, SiameseQAT extracts topic information from structured and unstructured texts. Their report found that 85% of the recall was accounted for in datasets from Eclipse, Netbeans, and Open Office. Mahfoodh and Hammad [17] employed Word2Vec and Tensorflow to construct a neural network-driven machine learning model to detect duplicate bug reports. A word-matching technique is employed to ascertain the similarities between two phrases. In the study of Xiao *et al.* [18], a novel heterogeneous information network has been presented to integrate both structured and unstructured features of a bug report, such as version, priority, severity, etc. HINDBR, stands for Heterogeneous Information Network Based Duplicate Bug Report, an innovative Deep Neural Network (DNN) that detects duplicate defect reports that are semantically similar with high accuracy, utilizes representation learning to construct a complex neural network. Their findings indicate a notable 98% accuracy

rate in classifying a limited dataset consisting of pairs of bug reports that are either duplicates or non-duplicates.

III. METHODOLOGY

Fig. 1 presents the technique flowchart, which will be briefly clarified in the subsequent parts.

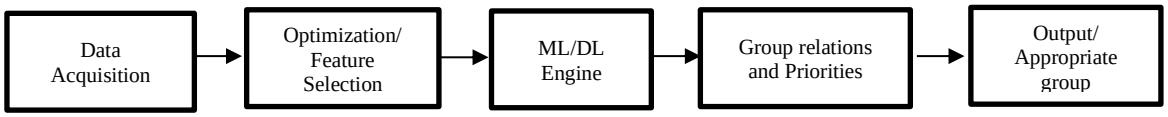

Fig. 1. Proposed methodology flowchart.

A. Data Acquisition

The dataset was acquired through a data error monitor on Eclipse's official website, eclipse.org. The kit comprises a dataset sample representing the results of bug measures in the Eclipse platform. The bug identification number is specified in the initial column of the table. Lastly, the provided columns contain 48 group labels ranging from one to seven. Each group inside the dataset corresponds to a specific group of developers that their respective firms have allocated to address and resolve defects in particular components of the software project. We conducted tests using MATLAB [19] and Weka [20] to evaluate the effectiveness of various machine-learning algorithms [20]. These tests involved classifying characteristics based on group labels and relying on such features. The accomplishment has been attained using a ten-fold cross-validation method. The data exhibits multiple dimensions, and the irrelevant columns should be removed. Certain features, also known as variables, have the potential to exhibit redundancy, noise, or a lack of relevance to the desired outcome. The inclusion of irrelevant features within machine learning models has the potential to reduce their accuracy, efficiency, and clarity [21]. The following figure illustrates the developers engaged in the project and reported software defects. An institutional framework delineates every cohort of software developers, and each group rectifies the quantity of software defects. Fig. 2 depicts the occurrence of bugs for each group of developers, The bug reports are classified into seven categories, with each category being a class that includes many bug reports and the individuals that resolved or marked the bugs. Our situation entails a collection of over 7,000 bug reports that include many developers. Fig. 2 displays the categories and the respective number of defects in each category.

The dataset comprises various groups representing distinct organizations engaged in application development and their respective total counts. The whole dataset containing Eclipse bugs is accessible at the following URL: https://data.mendeley.com/datasets/t6d9y7yt54/1 [22].

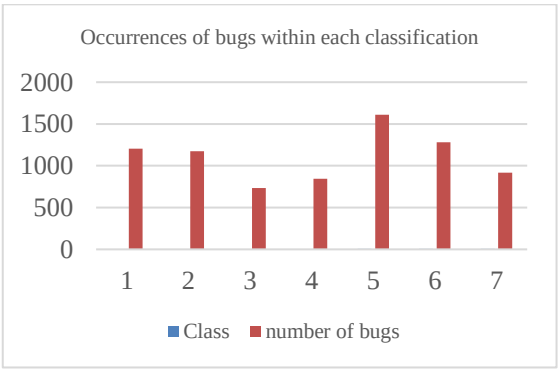

Fig. 2. Occurrences of bugs within each classification.

The following columns represent the features (columns) of the Eclipse bug report dataset. Each column in the dataset contains information about the bug, such as the platform from where the bug occurred. BugID; component; assignee Email; os; platform; milestone; nrKeywords; nrDependentBugs; peopleCC; openedhoursOpenedBeforeNextRelease; lastModified; priority; severity; resolution; firstFix; lastFix; hoursLastFixBeforeNextRelease; hoursLastFixAfter PreviousRelease; status;firstActivity; nrActivities; lastResolution; nrComments; hoursToLastFix; hoursToLastResolution; monthOpened; yearOpened; monthYearOpened; monthYearLastFixed.

The previously mentioned features include the specific information and components contained inside each bug report. Fig. 3 presents an example of a bug report document.

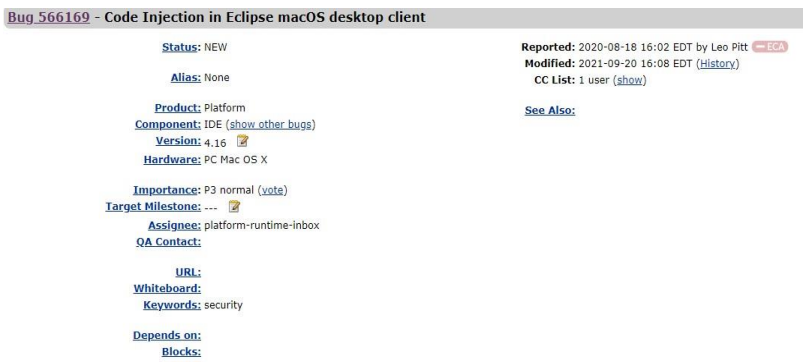

Fig. 3. Sample report of eclipse bugs.

B. Optimization/Feature Selection

Employing a dataset for training classifiers can result in inaccurate classifications and extend the duration of the training process. Including explanatory and clarifying elements inside the classification is considered excessive and contradictory, thus lacking meaningful contribution. To improve the precision of categorization, it is crucial to eliminate redundant and inconsistent attributes [23]. Feature selection approaches are employed to classify feature combinations that maximize the information acquired efficiently. To prevent any potential excessive influence on the performance of our dataset, the study opted to eliminate the last twenty attributes that exhibited the lowest correlation. Table I illustrates the effects of the correlations among the different factors. The algorithms have been constructed by leveraging the principles of natural selection and the mechanics of natural genetics. Genetic algorithms solve string structures resembling biological structures and undergo temporal development via survival filtering [24]. This objective is accomplished by utilizing a randomized yet synchronized exchange of information. As a result, a unique combination of genetic material is produced in each following generation, with only the most well-suited individuals being able to pass on their genes to future generations [25]. The essential attributes of the Genetic Algorithm (GA) and Harris Hawks Optimization (HHO) [26] are given below. The genetic algorithm employs the parameter set coding technique instead of directly altering individual parameters.

- Search from a population of points rather than a single point.
- Employ a reward as opposed to derivatives.
- Transformation laws that are probabilistic rather than deterministic.

A genetic algorithm is a stochastic algorithm that employs probabilistic principles. The determination of the search mechanism is achieved by implementing a random search strategy on a step-by-step superstructure model [27]. The optimal global solution can be achieved with a level of certainty of x%. The search process is initiated by selecting initial stochastic solutions called the "population". The structures in question are referred to as chromosomes. Chromosomes consist of genetic material known as genes [28]. The term "Gene" refers to the key parameters inside the network, such as rates of hot and cold canals. This study employed the Genetic Algorithm (GA) to perform feature reduction for classification purposes. Specifically, traits shown to have no impact on the outcome were eliminated from consideration [29, 30].

The concept of HHO draws inspiration from the predatory behavior of Harris hawks, namely their hunting techniques involving target exploration, surprise pouncing, and diverse attacking styles. HHO, also known as the Harris Hawks Optimization, is an optimization technique that operates on a population-based framework and does not rely on gradient information. It has the following phases [26]:

- **Exploration phase**

When examining the characteristics of Harris' hawks, it is evident that they possess the ability to locate and identify prey effectively through their formidable visual acuity. However, there are instances where the prey may prove challenging to spot. Therefore, the hawks patiently engage in the activities of waiting, observing, and monitoring the desert location to identify potential prey, which may occur after a considerable time. In the context of the Harris' Hawks Optimization (HHO) algorithm, the Harris' Hawks are seen as the potential solutions, with the most optimal solution being identified as the intended prey or approximate optimum in each iteration [26].

- **Exploitation phase**

Prey individuals consistently tend to evade and avoid circumstances that pose a potential threat to their well-being. Let us consider the probability, denoted as r, of prey either escaping ($r < 0.5$) or not successfully escaping ($r \geq 0.5$) before a surprise pounce occurs. Regardless of the actions taken by the prey, the hawks will engage in either a forceful or gentle pursuit to capture the victim. This behavior entails multiple individuals' coordinated encirclement of the prey, employing varying degrees of forcefulness or gentleness, contingent upon the prey's remaining energy levels. In practical scenarios, hawks gradually approach their chosen prey to enhance their likelihood of successfully collaborating in killing the rabbit by executing an unexpected pouncing maneuver. After a prolonged period, the prey that is attempting to escape will have a gradual depletion of energy. As a result, the hawks will increase their efforts in surrounding and capturing the prey that is now fatigued. The E parameter is employed to implement this technique and facilitate the HHO's transition between gentle and strong besiege procedures. In this context, soft besiege is observed when the absolute value of E is greater than or equal to 0.5. Conversely, the manifestation of a hard besiege is noted when the absolute value of E is less than 0.5 [26].

C. Feature Correlation

This study utilized the information gain technique to ascertain the traits that are most strongly correlated. The goal was to get valuable insights from these attributes, which align with the columns of the dataset. The primary objective of identifying correlations among the characteristics is to eliminate any redundant features that may impact the ultimate performance of our model. Given the presence of strong correlations between two features and the desire to streamline calculations, one of the features was eliminated. Table I presents the rankings of each attribute.

D. Engine

A wide variety of machine learning strategies are utilized by the recommendation engine. These strategies include Naive Bayes, Decision Trees, Support Vector Machines, and unsupervised machine learning algorithms such as Expectation Maximization. Initially, in the framework of recommendation systems, an investigation was carried out on several machine learning and deep learning algorithms. These algorithms included C4.5, Random Trees, and artificial neural networks. Through the employment of genetic algorithms in feature selection algorithms, the process of selecting ideal features has been

made easier, which has positively contributed to the training of the model. A bi-directional selection technique was utilized by the researchers to choose the most advantageous characteristics, and the Naive Bayes algorithm was included in the process [31]. MATLAB machine learning and deep learning techniques [32] and Weka [33]. were utilized in order to classify elements of the dataset. The machine learning algorithm and/or deep learning technique are the components that make up the ML/DL engine. These components are explicated in the subsequent subsections.

TABLE I. ATTRIBUTE CORRELATION RANKER

Rank	Attributes
0.64949	component3Days
0.47267	PredictedProbability_1
0.47267	PredictedProbability_2
0.36122	bugID
0.24057	os3Days
0.23763	PredictedProbability_1_1
0.22175	PredictedProbability_2_1
0.11869	status30Days
0.07545	nrActivities30Days
0.07222	nrActivities
0.07206	hoursLastFixAfterPreviousRelease
0.06964	nrActivities14Days
0.06582	nrActivities7Days
0.05873	status3Days
0.05732	hoursLastFixBeforeNextRelease
0.05713	nrActivities3Days
0.04461	nrActivities1Days
0.03219	hoursToLastFix
0.02681	monthOpened
0.02491	hoursToLastResolution
0.02247	priority30Days
0.02222	peopleCC
0.02163	filter_$
0.02082	nrPeopleCC30Days
0.02055	nrPeopleCC14Days
0.01949	hTLFix2Bins7Days
0.01898	hTLFix2Bins3Days
0.01892	nrPeopleCC7Days
0.01799	nrComments30Days
0.01752	nrPeopleCC3Days
0.01686	nrPeopleCC1Days
0.01621	platform3Days
0.01617	hTLFix2Bins14Days
0.01360	hTLFix2Bins1Days
0.01152	hTLFix2Bins30Days
0.00947	nrComments
0.00935	hTLFix2Bins0Days
0.00863	nrPeopleCC0Days
0.00863	initPeopleCC
0.00792	nrComments14Days
0.00658	severity3Days
0.00622	nrComments7Days
0	nrComments3Days
0	nrComments0Days
0	nrComments1Days
0	nrActivities0Days
0	nrDependentBugs
0	nrKeywords
0	esolution3Days

1) C4.5 algorithm with optimization

The C4.5 algorithm categorizes the dataset into seven distinct classes [34]. The main configuration of the C4.5 model is shown in Table II.

TABLE II. C4.5 PARAMETERS

Parameter	Value
Batch size	100
Confidence factor	0.25
Number of folds	3

2) Random trees with optimization

The novel occurrence of a particular entity is derived from the collective assemblage of trees cultivated inside the forested region. Every individual node produces a categorization for unknown cases, which are subsequently recorded as votes. The majority voting method involves aggregating the votes from many decision trees, and the class that obtains the most significant number of votes is designated as the predicted outcome for the new instance. According to Breiman [34, 35], the majority voting process in Random Forests serves as a categorization voting mechanism. Experiments are conducted on the voting behaviour of individuals.

The parameters employed in the RF algorithm are as follows:

- maxDepth = 0; The value assigned to the variable "maxDepth" represents the maximum depth of the trees. A value of 0 indicates that there is no limit on the depth.
- numFeatures = 0; The number of qualities to be employed in random selection; If the value is less than 1 (the default), the logarithm of M plus 1 is used, where M represents the number of inputs. Various numbers of characteristics, such as 10 or 20, were experimented with, but no significant impact on the accuracy results was observed.
- numTrees = 150; The number of trees to be generated throughout the duration of the experiment.

The random forest ensemble consists of thirty-five decision trees, each constructed using five features. The coefficient of variance for these features is measured to be 0.3347. After generating classifiers for each technique, iterate through each (Xi, Yi) pair in the original training set "T" and choose all "Tk" that do not contain the pair (Xi, Yi). A null set refers to a dataset subset containing no original records. The instances being referred to are the out-of-bag cases. The random forest ensemble comprises a total of thirty-five decision trees, each created based on a set of five attributes. The coefficient of variation for these features has been shown to be 0.3347. After producing classifiers for each approach, it is necessary to cycle through each pair (Xi, Yi) in the original training set "T" and choose all "Tk" that do not include the pair (Xi, Yi). A null set is defined as a subset of a dataset that does not include any elements from the original records. The instances being alluded to are the cases that are not included in the bootstrap sample.

3) Simple logistic with optimization

Logistic regression analysis is a suitable statistical method for analyzing a result that is dichotomous in nature. The logistic regression model is a statistical technique used for analyzing data. Logistic regression is a statistical technique that explains the association between a single

dependent variable and one or more independent variables [36].

4) CNN with optimization

The dynamics of intercellular communication and the underlying mechanisms of brain functionality shaped the development of this technique. The current methodology was established to attain the capability to do various jobs exclusively by analyzing training data samples. In image recognition, the employed approach involves training a model to distinguish between photos labeled as "keyboard" and those labeled as "not keyboard." This trained model is subsequently utilized to identify keyboards inside other photographs. No prerequisite knowledge or skills are necessary as the system automatically generates and differentiates features from the provided samples [37].

The development of this strategy was informed by the mechanisms of brain-cell interaction and the functioning of the brain. In the field of image recognition, the employed approach involves training a model to distinguish between photos labeled as "keyboard" and those labeled as "not keyboard". This trained model is subsequently utilized to identify keyboards inside other photographs. No prior knowledge or skills are necessary since the system automatically generates and differentiates features based on the inputted samples [38].

Typically, a Node is comprised of multiple levels. Each layer consists of several nodes that execute operations on diverse inputs. The transmission of signals progresses across the many levels of a neural network, starting with the input layer and concluding at the output layer. This process involves traversing the intermediate layers multiple times, with the number of iterations determined by the threshold and precision required to achieve optimal outcomes for the training model. The study also analyzed the neural network model as a training mechanism and the hybrid approach combining decision tree and naïve Bayes to determine the optimal outcomes for assigning bug reports from open-source systems to the appropriate developer [39]. Some of CNN architectures are mentioned below.

VGG16: VGG stands for the Visual Geometry Group; it is a multi-layered deep Convolutional Neural Network (CNN) standard architecture. The VGG16 architecture is a Convolutional Neural Network (CNN) known for its 16-layer depth. One can use a pre-trained neural network that has been trained on a dataset of more than one million photographs from the ImageNet database [40]. We employed a mini batch size of 32 and utilized max pooling with a stride of 2.

GoogLeNet: GoogLeNet is a convolutional neural network consisting of 22 layers. It is a modified iteration of the Inception Network. The Inception Network is a variant of deep convolutional neural network that was initially developed by academics associated with Google. The GoogLeNet architecture, which was presented at the ImageNet Large-Scale Visual Detection Challenge 2014 (ILSVRC14), effectively tackled computer vision tasks such as image classification and object detection [41]. We utilized a mini batch size of 32 and employed max pooling with a stride of 2.

5) Group relations

The necessary time for each programmer in both the training and testing sets has been computed, and the demand of each programmer in the bug's class has been determined using the following methodology:

- The objective is to ascertain the amount of time each programmer needs within the group.
- Identify the disparities in temporal measurements.
- The programmers should be arranged in ascending order of their speed, with the fastest programmers being assigned the highest rank.

IV. Findings and Analysis

Accuracy is a quantitative measure used to assess the performance of classification models. Informally, accuracy refers to the proportion of correct predictions made by our model. Accuracy is defined as the degree to which anything is correct or precise, according to established standards or criteria:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

For binary classification, accuracy can be computed by considering the number of true positives and true negatives.

Let TP represent the number of true positives, TN represent the number of true negatives, FP represent the number of false positives, and FN represent the number of false negatives.

Precision rates of 50% have been attained, and it has been posited that these precision rates are sufficient for bug triagers to discern the suitability of developers for assignment to specific bug reports [42]. In this study, we employed a 10-fold cross-validation technique and conducted attribute correlation analysis to select and evaluate features in the dataset. In formulating our bug measures, we employed various machine learning algorithms, including Naive Bayes, J48, Simplelogstic, random trees, artificial neural networks, and deep learning architectures. The utilization of an Intrinsic Network has yielded optimal results.

Additionally, our findings indicate the presence of certain features that are unsuitable for the classification task, resulting in a further deterioration of the results. Conversely, certain other features exhibited a greater classification accuracy rate. Accuracy optimization was achieved by selecting the top thirty characteristics with strong attribute correlation values [43]. The model output of different machine learning algorithms, namely Naïve Bayes, random trees, Simple Logistic, and Artificial Neural Networks, are presented in Table I. The algorithms were trained using a learning rate of 0.2, momentum of 0.5, batch size of 100, and not less than 500 iterations. With respect to the number of trees within a random forest, the findings indicate that frequently, an increased number of trees in a forest mostly escalates its computational burden without yielding any substantial benefits. This phenomenon was observed in the dataset under investigation. When a quantity exceeding 150 trees was employed. We conducted experiments with 400 and 600 trees; however, no statistically significant

differences were discovered. We employed optimization techniques to enhance the algorithms' classification accuracy and computational efficiency. The manipulation of class labels within the clustering algorithm resulted in improved outcomes. The demand for each programmer in the class is determined based on the consumption of the features from the previous hours. The corresponding findings are presented in Table III. The results in Table III demonstrate the accuracy outcomes and show the impact of combining the Naïve Bayes algorithm and Artificial Neural Networks (ANN) in the training set and utilizing C4.5 with Simplelogistic (50% training, 50% validation) in the dataset. The combined approach yields superior outcomes compared to using each method individually. Specifically, the optimization technique resulted in an approximate improvement of 7.4%, while the hybrid optimization approach achieved a gain of 8.2%. The HHO optimization also improves the results by approximately 2.9%.

TABLE III. ACCURACY OUTCOMES

Method	Classification accuracy (%) using GE	Classification accuracy (%) using HHO
C4.5	60.87	63.14
Random Trees	51.04	52.94
Simple Logistic	60.97	63.24
Naïve Bayes	48.76	50.57
Artificial Neural Networks (ANN)	62.66	64.99
VGG16	66.93	69.42
GoogLeNet	60.97	63.24
C4.5 optimization	65.14	67.57
Random Trees optimization	54.81	56.86
Simple Logistic optimization	65.44	67.88
Naïve Bayes optimization	52.33	54.28
Artificial Neural Networks with Optimization	67.33	69.83
Fusion Opt.C4.5 and Opt.SimpleLogistic	66.13	68.60
Fusion Opt.ANN and Opt.C4.5	67.13	69.63
Fusion Opt.Naïve Bayes and Opt.ANN	60.67	62.93
Fusion Opt.ANN and Opt.Random Trees	61.76	64.07
Fusion Opt. VGG16 and Opt. GoogLeNet	61.76	64.07

A. Clustering Findings

Through the utilization of a K-means technique, in which the value of K is set to 1000, it is possible to achieve the enhancement of system efficiency. This may be accomplished by modifying the classes or sets that the programmer has created. It was decided to construct a total of seven different groups, which are known as classes. Every programmer is placed in the cluster that is geographically closest to them, and this assignment is based on the mean value of their scores. The information presented in Table IV offers a comprehensive summary of programmers and the cluster links that represent them. Table V and Fig. 4 present the results of the accuracy tests performed using the new classes.

TABLE IV. NEW CLASSES AND THEIR DEVELOPERS

New class	Number of developers
1	1572
2	1223
3	1180
4	1089
5	949
6	902
7	855

TABLE V. ACCURACY OF CLUSTERED CLASSES

Method	Classification accuracy (%) using GE	Classification accuracy (%) using HHO
C4.5	67.03	69.53
Random Trees	58.49	60.67
Simple Logistic	66.73	69.22
Naïve Bayes	56.30	58.40
Artificial Neural Networks (ANN)	70.60	73.23
VGG16	75.57	78.38
GoogLeNet	68.81	71.38
C4.5 optimization	71.79	74.47
Random Trees optimization	62.76	65.10
Simple Logistic optimization	71.69	74.37
Naïve Bayes optimization	60.47	62.73
Artificial Neural Networks with Optimization	75.37	78.18
Fusion Opt.C4.5 and Opt.SimpleLogistic	74.28	77.04
Fusion Opt.ANN and Opt.C4.5	74.57	77.35
Fusion Opt.Naïve Bayes and Opt.ANN	68.02	70.56
Fusion Opt.ANN and Opt.Random Trees	61.76	64.07
Fusion Opt. VGG16 and Opt. GoogLeNet	71.79	74.47

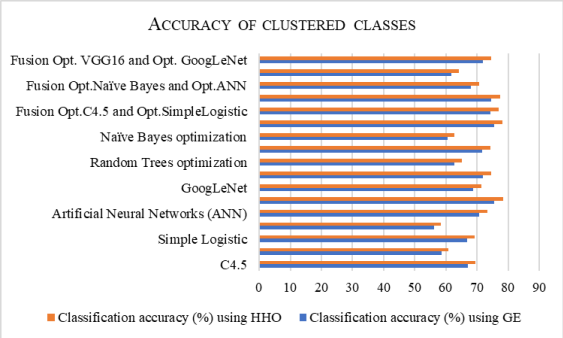

Fig. 4. Accuracy of clustered classes.

B. Analysis

Following the implementation of the new classes, we applied the same approaches in conjunction with a 10-fold cross-validation procedure, which resulted in significant gains of 11% being observed in the outcomes. This was after the new classes were introduced. During the duration of our experiments, we saw that there were limitations. The fact that the data are distributed in a random fashion makes it necessary to do substantial preprocessing processes before they can be employed. This is one of the limitations. Because of this, the utilization of real-time data would present difficulties, as it would necessitate a big amount of

hardware resources and a substantial amount of electricity due to the high level of processing complexity involved. In addition, it is a difficult task to choose the most effective method of artificial intelligence to use to power the system.

The assertion that only deep neural networks will do incredibly well with this kind of data is not something that can be considered accurate. One reason for this is because the complexity of the model, such as in deep neural networks, as well as its configurations that involve many running parameters, also influence the hardware resources that are required to implement these models. In the process of analyzing the results of different artificial intelligence models, it is essential to always take into consideration the selection of the best model and the fine-tuning of its hyperparameters. In terms of the performance of the model, ANN demonstrates positive results with a score of 73.23%; nevertheless, the architecture with the highest score was VGG16, which scored 78.38%. As a result of its ease of use in determining the classes among the dataset, Naïve Bayes demonstrated the least successful results, with a percentage of 58.4%.

V. Future Aspects

Subsequent investigations may employ exclusive selection filtering algorithms to focus on the salient terms utilized in bug reports [44]. Utilizing the chi-square selection method is appropriate for evaluating the computational criteria of the experiment. Developing a unique optimization algorithm is crucial for achieving accurate results that align with the specific dataset and methodology. It is advisable to explore alternative methodologies rather than solely relying on the current strategy [45]. The efficiency of the framework is measured in terms of time. Our methodology examines the correlation between the attributes of an exemplary software developer and the level of severity in bug reports. Current developers are not required to undergo retraining of prediction models when fresh data is received [46]. This approach reduces the duration required for updating the framework. However, leveraging the system's knowledge and understanding of each time is crucial to uphold its effectiveness [47]. Various potential enhancements and effective resolutions can be contemplated, given the imperative requirement for the system to operate autonomously in terms of evaluation, recognition, and suitable feature selection [48]. This includes activities such as code testing and time management during the execution of diverse tasks, enabling the system to automatically adapt and resolve incoming new tasks [37, 49].

VI. Conclusion

The framework technique helps bug triage programmers manage and choose solutions for issue reports. The open-source bugs repository is used for bug report assignments. Nave Bayes, J48, random forests, and simple logistic models were utilized. Information benefit values were used to identify decision-making features during feature collection. Therefore, we excluded the 20 least informative features. We used a method for each class using the 30 most important attributes.

Bug reports were identified by random forest and neural networks. Multiple scholarly papers propose algorithms that support the conclusion. Decision trees and SVMs are frequently inferior to the classification described above method. Two essential advantages come from choosing a random forest. One advantage of this strategy is its independence from interactive functions. Decision Trees do well on considerable datasets in tree ensembles. The random forest's features suggest 150 trees for this ensemble technique. Tree classifiers' effectiveness and dependence determine random forest accuracy. Top-voted tree ranking method. More random forest trees increase computing time but not per-second output. Machine learning techniques use hybrid and optimization methods to create a bug assignment framework that meets objectives.

Choosing the most suitable approach is a complex task requiring careful consideration of various factors, including available hardware resources, user experience, and the volume of the data needed to train the engine. Generally, Machine Learning (ML) approaches do not necessitate a large amount of data for training, whereas Deep Learning (DL) approaches require a substantial amount of data. Preprocessing significantly improves performance by acting as a filter for subsequent processes in the approach. Utilizing optimal and relevant characteristics, optimizing and cleansing the data, and processing it through a deep learning approach yielded superior engine performance. Nevertheless, it necessitated a substantial amount of time to execute and demanded high hardware specs.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

J.S.H. conducted the research and methodology, M. A. analyzed the data, and F.N.T. wrote the paper. All authors have approved the final version.

Acknowledgement

We express our gratitude to our colleagues for their valuable assistance in our study, and we extend our thanks to the University of Baghdad for their support during the duration of this research effort.

References

[1] T. Zimmermann and A. C. Artís, "Impact of switching bug trackers: a case study on a medium-sized open source project," in *Proc. International Conference on Software Maintenance and Evolution*, Cleveland, 2019.

[2] T. Zhang, J. Chen, G. Yang, B. Lee, and X. Luo, "Towards more accurate severity prediction and fixer recommendation of software bugs," *Journal of Systems and Software*, vol. 117, pp. 166–184, 2016. https://doi.org/10.1016/j.jss.2016.02.034

[3] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer prioritization in bug repositories," in *Proc. 34th International Conference on Software Engineering*, 2012.

[4] S. Singh, "Analysis of bug tracking tools," *International Journal of Scientific & Engineering Research*, vol. 4, no. 7, Jul. 2013.

[5] S. A. Qader and A. R. Abbas, "Dual-stage social friend recommendation system based on user interests," *Iraqi Journal of Science*, pp. 1759–1772, Jul. 2020. doi: 10.24996/ijs.2020.61.7.25

[6] A. L. Bakri *et al.*, "A study on the accuracy of prediction in recommendation system based on similarity measures," *Baghdad Science Journal*, vol. 16, no. 1, 0263, Mar. 2019. doi: 10.21123/bsj.2019.16.1(Suppl.).0263

[7] A. R. A. S. A. Mohammed and A. Kareem, "Design recommendation system in e-commerce site," *Iraqi Journal of Science*, vol. 57, no. 4A, pp. 2549–2556, 2022.

[8] R. Shokripour, J. Anvik, Z. M. Kasirun, and S. Zamani, "A time-based approach to automatic bug report assignment," *Journal of Systems and Software*, vol. 102, pp. 109–122, Apr. 2015. doi: 10.1016/j.jss.2014.12.049

[9] G. Jin, T. Wang, Y. Amirat, Z. Zhou, and T. Xie, "A layering linear discriminant analysis-based fault diagnosis method for grid-connected inverter," *J. Mar. Sci. Eng.*, vol. 10, no. 7, 939, Jul. 2022. doi: 10.3390/jmse10070939

[10] H. Isotani, H. Washizaki, Y. Fukazawa, T. Nomoto, S. Ouji, and S. Saito, "Duplicate bug report detection by using sentence embedding and fine-tuning," in *Proc. 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2021, pp. 535–544. doi: 10.1109/ICSME52107.2021.00054

[11] J. Lerch and M. Mezini, "Finding duplicates of your yet unwritten bug report," in *Proc. 2013 17th European Conference on Software Maintenance and Reengineering*, IEEE, Mar. 2013, pp. 69–78. doi: 10.1109/CSMR.2013.17

[12] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and *execution* information," in *Proc. 13th International Conference in Software Engineering*, 2008, pp. 461–470.

[13] A. Kukkar, R. Mohana, Y. Kumar, A. Nayyar, M. Bilal, and K.-S. Kwak, "Duplicate bug report detection and classification system based on deep learning technique," *IEEE Access*, vol. 8, pp. 200749–200763, 2020. doi: 10.1109/ACCESS.2020.3033045

[14] A. Alipour, A. Hindle, and E. Stroulia, "A contextual approach towards more accurate duplicate bug report detection," in *Proc. 2013 10th Working Conference on Mining Software Repositories (MSR)*, *IEEE*, May 2013, pp. 183–192. doi: 10.1109/MSR.2013.6624026

[15] B. Kucuk and E. Tuzun, "Characterizing duplicate bugs: An empirical analysis," in *Proc. 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Mar. 2021, pp. 661–668. doi: 10.1109/SANER50967.2021.00084

[16] T. M. Rocha and A. L. D. C. Carvalho, "SiameseQAT: A semantic context-based duplicate bug report detection using replicated cluster information," *IEEE Access*, vol. 9, pp. 44610–44630, 2021. doi: 10.1109/ACCESS.2021.3066283

[17] H. Mahfoodh and M. Hammad, "Word2Vec duplicate bug records identification prediction using tensorflow," in *Proc. 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, *IEEE*, Dec. 2020, pp. 1–6. doi: 10.1109/3ICT51146.2020.9311954

[18] G. Xiao, X. Du, Y. Sui, and T. Yue, "HINDBR: Heterogeneous information network based duplicate bug report prediction," in *Proc. 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, Oct. 2020, pp. 195–206. doi: 10.1109/ISSRE5003.2020.00027

[19] MATLAB. (January 2021). [Online]. Available: https://www.mathworks.com/products/matlab.html

[20] Weka 3: Machine Learning Software in Java. (May 2022). [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/

[21] M. Afshar and H. Usefi, "Optimizing feature selection methods by removing irrelevant features using sparse least squares," *Expert Syst. Appl.*, vol. 200, 116928, Aug. 2022. doi: 10.1016/j.eswa.2022.116928

[22] L. Çarkacıoğlu. (March 2023). Dataset on eclipse bug records on bugzilla. *Mendeley Data* [Online]. Available: https://data.mendeley.com/datasets/t6d9y7yt54/1

[23] E. Lughofer and M. Pratama, "Evolving multi-user fuzzy classifier system with advanced explainability and interpretability aspects," *Information Fusion*, vol. 91, pp. 458–476, Mar. 2023. doi: 10.1016/j.inffus.2022.10.027

[24] M. A. Albadr, S. Tiun, M. Ayob, and F. Al-Dhief, "Genetic algorithm based on natural selection theory for optimization problems," *Symmetry (Basel)*, vol. 12, no. 11, 1758, Oct. 2020. doi: 10.3390/sym12111758

[25] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimed Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, 2021. doi: 10.1007/s11042-020-10139-6

[26] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, Aug. 2019. doi: 10.1016/j.future.2019.02.028

[27] A. Shafiee, M. Nomvar, Z. Liu, and A. Abbas, "A new genetic algorithm based on Prenatal Genetic Screening (PGS-GA) and its application in an automated process flowsheet synthesis problem for a membrane based carbon capture case-study," *Chemical Engineering Research and Design*, vol. 128, pp. 265–289, Dec. 2017. doi: 10.1016/j.cherd.2017.10.009

[28] K. Park, D. Shin, and S. Chi, "Variable chromosome genetic algorithm for structure learning in neural networks to imitate human brain," *Applied Sciences*, vol. 9, no. 15, 3176, Aug. 2019. doi: 10.3390/app9153176

[29] Y. Masoudi-Sobhanzadeh, H. Motieghader, Y. Omidi, and A. Masoudi-Nejad, "A machine learning method based on the genetic and world competitive contests algorithms for selecting genes or features in biological applications," *Sci. Rep.*, vol. 11, no. 1, 3349, Feb. 2021. doi: 10.1038/s41598-021-82796-y

[30] S. Y. Hera and M. Amjad, "Prediction of explicit features for recommendation system using user reviews," *Iraqi Journal of Science*, pp. 5015–5023, Nov. 2022. doi: 10.24996/ijs.2022.63.11.36

[31] H. Chen, S. Hu, R. Hua, and X. Zhao, "Improved naive Bayes classification algorithm for traffic risk management," *EURASIP J. Adv. Signal Process*, vol. 2021, no. 1, 30, Dec. 2021. doi: 10.1186/s13634-021-00742-6

[32] C. Warren, "MATLAB for engineers: Development of an online, interactive, self-study course," *Engineering Education*, vol. 9, no. 1, pp. 86–93, Jul. 2014. doi: 10.11120/ened.2014.00026

[33] Y. Dou and W. Meng, "Comparative analysis of weka-based classification algorithms on medical diagnosis datasets," *Technology and Health Care*, vol. 31, pp. 397–408, Apr. 2023. doi: 10.3233/THC-236034

[34] N. Khanna. (September 2023). J48 classification (C4.5 algorithm) in a nutshell. *Medium* [Online]. 3. Available: https://medium.com/@*nilimakhanna1*/j48-classification-c4-5-algorithm-in-a-nutshell-24c50d20658e

[35] L. Breiman, "Random forests*," Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001. doi: 10.1023/A:1010933404324

[36] J. K. Harris, "Primer on binary logistic regression," *Fam. Med. Community Health*, vol. 9, no. 1, 001290, Dec. 2021. doi: 10.1136/fmch-2021-001290

[37] O. Adebayo, A. Patel, and J. Summers, "ANN crowds in early-stage design: An investigation of influence of small training sets on prediction," *Procedia CIRP*, vol. 119, pp. 589–595, 2023. doi: 10.1016/j.procir.2023.02.153

[38] O. A. M. López, A. M. López, and J. Crossa, "Fundamentals of artificial neural networks and deep learning," *Multivariate Statistical Machine Learning Methods for Genomic Prediction*, 2022, pp. 379–425. doi: 10.1007/978-3-030-89010-0_10

[39] M. Gallo, G. D. Luca, L. D. Acierno, and M. Botte, "Artificial neural networks for forecasting passenger flows on metro lines," *Sensors*, vol. 19, no. 15, 3424, Aug. 2019. doi: 10.3390/s19153424

[40] J. Tao, Y. Gu, J. Sun, Y. Bie, and H. Wang, "Research on vgg16 convolutional neural network feature classification algorithm based on transfer learning," in *Proc. 2021 2nd China International SAR Symposium (CISS)*, IEEE, Nov. 2021. pp. 1–3. doi: 10.23919/CISS51089.2021.9652277

[41] X. Wang *et al.*, "SpikeGoogle: Spiking neural networks with GoogLeNet-like inception module," *CAAI Trans. Intell. Technol.*, vol. 7, no. 3, pp. 492–502, Sep. 2022. doi: 10.1049/cit2.12082

[42] M. Samir, N. Sherief, and W. Abdelmoez, "Improving bug assignment and developer allocation in software engineering through interpretable machine learning models," *Computers*, vol. 12, no. 7, 128, Jun. 2023. doi: 10.3390/computers12070128

[43] S. Kumar and I. Chong, "Correlation analysis to identify the effective data in machine learning: prediction of depressive disorder and emotion states," *Int. J. Environ Res. Public Health*, vol. 15, no. 12, 2907, Dec. 2018. doi: 10.3390/ijerph15122907

[44] H. Mahmud, A. K. M. N. Islam, S. I. Ahmed, and K. Smolander, "What influences algorithmic decision-making? A systematic literature review on algorithm aversion," *Technol. Forecast Soc. Change*, vol. 175, 121390, Feb. 2022. doi: 10.1016/j.techfore.2021.121390

[45] D. P. M. Abellana and D. M. Lao, "A new univariate feature selection algorithm based on the best–worst multi-attribute decision-making method," *Decision Analytics Journal*, vol. 7, 100240, Jun. 2023. doi: 10.1016/j.dajour.2023.100240

[46] M. M. Taye, "Understanding of machine learning with deep learning: Architectures, workflow, applications and future directions," *Computers*, vol. 12, no. 5, 91, Apr. 2023. doi: 10.3390/computers12050091

[47] A. M. Abubakar, H. Elrehail, M. A. Alatailat, and A. Elçi, "Knowledge management, decision-making style and organizational performance," *Journal of Innovation & Knowledge*, vol. 4, no. 2, pp. 104–114, Apr. 2019. doi: 10.1016/j.jik.2017.07.003

[48] N. Pudjihartono, T. Fadason, A. W. Kempa-Liehr, and J. M. O. Sullivan, "A review of feature selection methods for machine learning-based disease risk prediction," *Frontiers in Bioinformatics*, vol. 2, Jun. 2022. doi: 10.3389/fbinf.2022.927312

[49] Y. Xu *et al.*, "Artificial intelligence: A powerful paradigm for scientific research," *The Innovation*, vol. 2, no. 4, 100179, Nov. 2021. doi: 10.1016/j.xinn.2021.100179